

---

# **buildpy Documentation**

***Release 1.8.12***

**Blake Huber**

**Jun 05, 2021**



---

# README

---

<b>1</b>	<b>Summary</b>	<b>1</b>
<b>2</b>	<b>Getting Started</b>	<b>3</b>
<b>3</b>	<b>Documentation</b>	<b>5</b>
<b>4</b>	<b>Supported Linux Distributions</b>	<b>7</b>
<b>5</b>	<b>Help</b>	<b>9</b>
<b>6</b>	<b>Author &amp; Copyright</b>	<b>13</b>
<b>7</b>	<b>Disclaimer</b>	<b>15</b>
<b>8</b>	<b>License</b>	<b>17</b>
<b>9</b>	<b>Dependencies</b>	<b>27</b>
<b>10</b>	<b>Installation</b>	<b>29</b>
10.1	Debian, Ubuntu, Linux Mint, Ubuntu Variants . . . . .	29
10.2	Redhat, CentOS, Fedora RPM-based Distributions . . . . .	31
<b>11</b>	<b>Upgrading</b>	<b>35</b>
11.1	Debian, Ubuntu Variants . . . . .	35
11.2	Upgrading Redhat-based Distributions . . . . .	36
<b>12</b>	<b>Uninstall</b>	<b>39</b>
12.1	Debian and Ubuntu Variants . . . . .	39
12.2	Redhat, CentOS, Fedora RPM Variants . . . . .	39
<b>13</b>	<b>Use Cases</b>	<b>41</b>
13.1	Verify Operating System Dependencies . . . . .	41
13.2	Compile Python 3 - Basic Installation . . . . .	42
13.3	Compile Python 3 - Advanced Installation . . . . .	42
13.4	Compile Python 3 - Specific Version . . . . .	43
13.5	Unattended Install . . . . .	43
13.6	Uninstalling Python 3 Versions . . . . .	44

<b>14</b>	<b>Frequently Asked Questions</b>	<b>45</b>
14.1	General Questions . . . . .	46
14.2	Required User Permissions . . . . .	46
14.3	Python Version Support . . . . .	47
14.4	Supported Linux Operating System Distributions . . . . .	47
14.5	Uninstall Python3 . . . . .	48
14.6	Logging FAQ . . . . .	48
14.7	Miscellaneous Questions . . . . .	48
<b>15</b>	<b>Screenshot Index</b>	<b>51</b>
<b>16</b>	<b>Operating System Detection</b>	<b>53</b>
<b>17</b>	<b>Python Binary Version Status</b>	<b>55</b>
<b>18</b>	<b>OS Packages Installed Status</b>	<b>57</b>
<b>19</b>	<b>Download Python Source (Default)</b>	<b>59</b>
<b>20</b>	<b>Download Python Source, Named Version</b>	<b>61</b>
<b>21</b>	<b>Remove Build Artifacts</b>	<b>63</b>
<b>22</b>	<b>Program Information</b>	<b>65</b>
<b>23</b>	<b>Log Files</b>	<b>69</b>
<b>24</b>	<b>buildpy Source Code</b>	<b>71</b>
24.1	Bash Module Index . . . . .	71
24.2	buildpy . . . . .	71
24.3	colors.sh . . . . .	126
24.4	exitcodes.sh . . . . .	131
24.5	version.py . . . . .	132
24.6	os_distro.sh . . . . .	132
24.7	std_functions.sh . . . . .	150
<b>25</b>	<b>Debian Package Creation</b>	<b>167</b>
25.1	Debian Package Creation Start . . . . .	167
25.2	Debian Package Creation Build . . . . .	168
25.3	Debian Package Final Contents . . . . .	169
<b>26</b>	<b>RPM Package Creation</b>	<b>171</b>
26.1	RPM Package Assembly . . . . .	172
26.2	RPM Package Build . . . . .	173
26.3	Docker RPM Package Build . . . . .	174
26.4	RPM Package Final Contents . . . . .	175
<b>27</b>	<b>Current Release</b>	<b>177</b>
27.1	v1.8.12   Release Notes . . . . .	177
<b>28</b>	<b>Release History</b>	<b>179</b>
<b>29</b>	<b>Release Note Index</b>	<b>181</b>
29.1	v1.8.11   Release Notes . . . . .	181
29.2	v1.8.10   Release Notes . . . . .	182
29.3	v1.8.9   Release Notes . . . . .	182



29.4	v1.8.8   Release Notes . . . . .	183
29.5	v1.8.5   Release Notes . . . . .	183
29.6	v1.8.3   Release Notes . . . . .	184
29.7	v1.8.1   Release Notes . . . . .	184
29.8	v1.7.20   Release Notes . . . . .	185
29.9	v1.7.19   Release Notes . . . . .	185
29.10	v1.7.18   Release Notes . . . . .	186
29.11	v1.7.17   Release Notes . . . . .	186
29.12	v1.7.16   Release Notes . . . . .	186
29.13	v1.7.15   Release Notes . . . . .	187
<b>30</b>	<b>Module Index</b>	<b>189</b>
<b>31</b>	<b>Site Index</b>	<b>191</b>
<b>32</b>	<b>Search Project</b>	<b>193</b>



# CHAPTER 1

---

## Summary

---

- [buildpy](#) is a utility for compiling and installing on Linux.
- Automatically downloads and compiles Python 2.7 to 3.7+ from [python.org](#) source.
- Manages the installation of Linux OS dependencies to enable python3 advanced runtime features.
- Optional `quiet` mode for fully automated, unattended installs.

---

### System-level Python3

- [buildpy](#) compiles, installs, and configures Python binaries for the entire system.
- If you wish to build Python for a single user, consider [pipenv](#)

---

[Back to Table Of Contents](#)

---



## CHAPTER 2

---

### Getting Started

---

Before starting, we recommended reviewing the following:

- Read the [Frequently Asked Questions](#).
- Reviewing the [buildpy feature overview](#) (below) to learn about buildpy.



# Developer Tools

**buildpy** | Python Source Compiler for Linux

[Back to Table Of Contents](#)

---



## CHAPTER 3

---

### Documentation

---

#### Online

- Complete html documentation available at <http://buildpy.readthedocs.io>.

**Download:** Available via download in the formats below

- [pdf format](#)
- [Amazon Kindle \(epub\) format](#)

#### Source Code

- [buildpython3](#) git repository

[Back to Table Of Contents](#)

---





---

## Supported Linux Distributions

---

### Ubuntu, Ubuntu-based Variants

- : Ubuntu variants based on 14.04+
- : Ubuntu variants based on 16.04, 16.10+
- : Ubuntu variants based on 18.04, 18.10+
- , ,

### Redhat, Redhat-based Variants

- 
- 
- 
- (2017+)
- (2018+)

---

**Note:** Older versions than listed above may be compatible, but not have not been tested

---

Back to [Table Of Contents](#)

---



## CHAPTER 5

---

Help

---

To display the help menu:

```
$ buildpy --help
```

## Build Python3 from Source | Linux

## DESCRIPTION

Utility to compile and install the latest versions of python3  
official source binaries download from <https://www.python.org>

## SYNOPSIS

```
$ buildpy < OPTION >

-I | --install <value>
[-d | --download <value> ]
[-c | --clean <value> ]
[-o | --optimization ]
[-s | --show <value> ]
[-i | --info ]
[-q | --quiet ]
[-t | --os-detect ]
[-b | --backup-pip ]
[-V | --version ]
[-U | --uninstall <value> ]
```

**Note:** Requires sudo or root privileges

## OPTIONS

- b, --backup-pip:** Create local Backup copy of installed python packages (single operation only)
- c, --clean:** Remove all build and installation artifacts. Can optionally be invoked with a Python major revision number provided as a parameter to clean specific build artifacts
- d, --download:** Download Python build artifacts, but take no other action. Optionally can be invoked with Python major revision number to download a specific Python binary set:

```
$ buildpy --download 3.4
```

- I, --install <value>:** Build, compile, and install Python binaries on local machine for all system users. Value indicates the latest Python minor version for the release specified by <value> (DEFAULT: 3.6)
- o, --optimization:** Compile with Optimizations (Optional)  
Compiling with optimization drastically lengthens compile time; but results in increased speed of the CPython compiler (~ 10%). Recommended only if abundant cpu resources
- i, --info:** Display detailed information on the buildpy program and functions it contains
- q, --quiet:** (Optional) Suppress output to stdout. Option is invoked for scripted or unattended compile and install.
- s, --show:** (Optional) Show latest Python minor version available for named Python major version (DEFAULT: 3.6)
- t, --os-detect:** Detect Operating System type and exit (single operation only)
- U, --uninstall:** Uninstall the Linux system Python3 binaries and related libraries and support artifacts from the local file system for Python major revision given as a parameter
- V, --version:** Display buildpy version and license information

## EXAMPLES

**Compile and install latest Python 3.6 source binary**  
\$ buildpy --install 3.6

**Compile with Optimizations, then install latest Python 3.7**  
\$ buildpy --install 3.7 --optimizations

**Show most recent Python 3.4 source binary available**  
\$ buildpy --show 3.4

[Back to Table Of Contents](#)

---



## CHAPTER 6

---

### Author & Copyright

---

All works contained herein copyrighted via below author unless work is explicitly noted by an alternate author.

- Copyright 2017-2021 Blake Huber, All Rights Reserved.

#### **Software License**

- Software is licensed and protected under the [GNU General Public License Agreement v3](#).

Back to [Table Of Contents](#)

---





## CHAPTER 7

---

### Disclaimer

---

*Code is provided “as is”. No liability is assumed by either the code’s originating author nor this repo’s owner for their use at AWS or any other facility. Furthermore, running function code at AWS may incur monetary charges; in some cases, charges may be substantial. Charges are the sole responsibility of the account holder executing code obtained from this library.*

Additional terms may be found in the complete [license agreement](#).

---

[Table Of Contents](#)

---



## CHAPTER 8

---

### License

---

GNU GENERAL PUBLIC LICENSE  
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

#### Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS
----------------------

## 0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

## 1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

## 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

## 3. Protecting Users’ Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.

## 4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

## 5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, **and** giving a relevant date.
- b) The work must carry prominent notices stating that it **is** released under this License **and** any conditions added under section

(continues on next page)

(continued from previous page)

7. This requirement modifies the requirement **in** section 4 to "keep intact all notices".

c) You must license the entire work, **as** a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along **with any** applicable section 7 additional terms, to the whole of the work, **and all** its parts, regardless of how they are packaged. This License gives no permission to license the work **in any** other way, but it does **not** invalidate such permission **if** you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, **if** the Program has interactive interfaces that do **not** display Appropriate Legal Notices, your work need **not** make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

## 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the **object** code **in, or** embodied **in**, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used **for** software interchange.

b) Convey the **object** code **in, or** embodied **in**, a physical product (including a physical distribution medium), accompanied by a written offer, valid **for** at least three years **and** valid **for as** long **as** you offer spare parts **or** customer support **for** that product model, to give anyone who possesses the **object** code either (1) a copy of the Corresponding Source **for all** the software **in** the product that **is** covered by this License, on a durable physical medium customarily used **for** software interchange, **for** a price no more than your reasonable cost of physically performing this conveying of source, **or** (2) access to copy the Corresponding Source **from a** network server at no charge.

c) Convey individual copies of the **object** code **with** a copy of the written offer to provide the Corresponding Source. This alternative **is** allowed only occasionally **and** noncommercially, **and** only **if** you received the **object** code **with** such an offer, **in** accord **with** subsection 6b.

d) Convey the **object** code by offering access **from a** designated place (gratis **or for** a charge), **and** offer equivalent access to the Corresponding Source **in** the same way through the same place at no further charge. You need **not** require recipients to copy the Corresponding Source along **with** the **object** code. If the place to copy the **object** code **is** a network server, the Corresponding Source may be on a different server (operated by you **or** a third party)

(continues on next page)

(continued from previous page)

that supports equivalent copying facilities, provided you maintain clear directions `next` to the `object` code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it `is` available `for as long as` needed to satisfy these requirements.

e) Convey the `object` code using peer-to-peer transmission, provided you inform other peers where the `object` code `and` Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

## 7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty **or** limiting liability differently **from the** terms of sections **15 and 16** of this License; **or**
- b) Requiring preservation of specified reasonable legal notices **or** author attributions **in** that material **or in** the Appropriate Legal Notices displayed by works containing it; **or**
- c) Prohibiting misrepresentation of the origin of that material, **or** requiring that modified versions of such material be marked **in** reasonable ways **as** different **from the** original version; **or**
- d) Limiting the use **for** publicity purposes of names of licensors **or** authors of the material; **or**
- e) Declining to grant rights under trademark law **for** use of some trade names, trademarks, **or** service marks; **or**
- f) Requiring indemnification of licensors **and** authors of that material by anyone who conveys the material (**or** modified versions of it) **with** contractual assumptions of liability to the recipient, **for** **any** liability that these contractual assumptions directly impose on those licensors **and** authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

## 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

## 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise



does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

#### 11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work

from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

#### 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

#### 14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

#### 15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

#### 16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE

WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

#### 17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
{one line to give the program's name and a brief idea of what it does.}
Copyright (C) {year} {name of author}
```

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License
along with this program. If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
{project} Copyright (C) {year} {fullname}
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program’s commands might be different; for a GUI interface, you would use an “about box”.

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.



## CHAPTER 9

---

### Dependencies

---

- *Root Privileges*. Ability to assume user with root privileges or access such privileges via **sudo**
- **Python3-support via Operating System Packages**. Dependencies on various operating system packages available from official Linux distribution's repository. You can view os package dependencies with the following command once **buildpy** is installed:
- **Bash 4.1 or higher**. No preexisting python implementation is required. **buildpy** is written in [bash](#).

Back to [Table Of Contents](#)

---



### 10.1 Debian, Ubuntu, Linux Mint, Ubuntu Variants

**Method 1:** Install the **debian-tools** repository.

The easiest way to install **buildpy** is via the Debian-tools repository:

1. Download the public key:

```
$ wget -qO - http://awscloud.center/keys/public.key | sudo apt-key add -
```

2. Install the repository:

```
$ string='deb [arch=amd64] http://deb.awscloud.center <distribution> main' ; \  
sudo echo $string > /etc/apt/sources.list.d/debian-tools.list
```

Where **<distribution>** is one of the following:

- **trusty:** Ubuntu 14.04, Ubuntu 14.04 based Linux distributions
- **xenial:** Ubuntu 16.04, 16.04 based Linux distributions
- **bionic:** Ubuntu 18.04, 18.04 based Linux distributions

3. Verify package repository installation

```
$ apt list buildpy -a
```

4. Update and install the package:

```
$ sudo apt update && sudo apt install buildpy
```

5. Verify Installation. To verify a Debian (.deb) package installation:

```
$ apt show buildpy
```

```
blake@ubuntu1-desktop:~$ echo -e "\n"; apt list buildpy -a
```

```
Listing... Done
buildpy/bionic 1.7.5 amd64
buildpy/bionic 1.7.3 amd64
buildpy/bionic 1.7.2 amd64
buildpy/bionic 1.7.1 amd64
buildpy/bionic 1.7.0 amd64
buildpy/bionic 1.6.17 amd64
buildpy/bionic 1.6.16 amd64
buildpy/bionic 1.6.15 amd64
buildpy/bionic 1.6.14 amd64
buildpy/bionic 1.6.13 amd64
```

```
Package: buildpy
New: yes
State: installed
Automatically installed: no
Version: 1.6.15
Priority: optional
Section: devel
Maintainer: Blake Huber <blakeca00@gmail.com>
Architecture: amd64
Uncompressed Size: 118 k
Depends: bash (>= 4.2), curl (>= 7.0), bc (>= 1.0), debianutils, sudo,
        util-linux
Description: Compile and Build Any Python3 Version on Linux

  buildpy is a Utility for building python3 from source binaries on Linux.

  Compile and install Python 3.0 to 3.7+ major versions. Python 2.x can be
  compiled and installed; however, this has not been tested extensively. When a
  Python major version number is specified (Python 3.6 for example), buildpy
  automatically finds and installs the latest minor version available for
  download from https://www.python.org/ftp/python. buildpy has a "quiet mode" for
  scripted configuration management installs.

  The buildpy utility builds Python binaries for the entire system. If you wish
  to build Python for a single user, consider pipenv
Homepage: https://bitbucket.org/blakeca00/buildpython3
```



[Back to Table Of Contents](#)

---

**Method 2:** Install directly via `.deb` package.

If you do not want to install an additional repository on your local machine, you may instead download and install the `.deb` package using the `apt` or `apt-get` package managers.

1. **Download:** the `.deb` installation package (v1.6.3 shown):

```
$ wget https://bitbucket.org/blakeca00/buildpython3/downloads/buildpy-1.6.3_amd64.  
→deb
```

2. **Install** via `apt` package manager:

```
$ sudo apt install ./buildpy-1.6.3_amd64.deb
```

**Note:**

- Installation of the `.deb` package will also install the **debian-tools** repository on your local machine. If you do not want this, please comment out the single line in `/etc/apt/sources.list.d/debian-tools.list`.

[Back to Table Of Contents](#)

---

## 10.2 Redhat, CentOS, Fedora RPM-based Distributions

The easiest way to install **buildpy** on redhat-based Linux distributions is via the developer-tools package repository:

(**Fedora** Users: use `dnf` in place of `yum` below)

1. Download and install the repo definition file

```
$ sudo yum install wget
```

```
$ wget https://bitbucket.org/blakeca00/buildpython3/downloads/developer-tools.repo
```

```
$ sudo mv developer-tools.repo /etc/yum.repos.d/ && sudo chown 0:0 developer-  
→tools.repo
```

2. Update local repository cache

```
$ sudo yum update -y
```

3. Install **buildpy** os package

```
$ sudo yum install buildpy
```

Answer “y”:

4. Verify Installation

```
$ yum info buildpy
```

```
[ec2-user@ip-172-31-18-110 noarch]$ sudo yum install buildpy-1.6-6.noarch.rpm
Loaded plugins: priorities, update-motd, upgrade-helper
Examining buildpy-1.6-6.noarch.rpm: buildpy-1.6-6.noarch
Marking buildpy-1.6-6.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package buildpy.noarch 0:1.6-6 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                        Arch      Version      Repository
=====
Installing:
  buildpy                       noarch    1.6-6        /buildpy-1.6-6.noarch
=====
Transaction Summary
=====
Install 1 Package

Total size: 89 k
Installed size: 89 k
Is this ok [y/d/N]:
```

Fig. 1: rpm install

```
Total size: 89 k
Installed size: 89 k
Is this ok [y/d/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : buildpy-1.6-6.noarch
  Verifying  : buildpy-1.6-6.noarch

Installed:
  buildpy.noarch 0:1.6-6

Complete!
[ec2-user@ip-172-31-18-110 noarch]$
```

Fig. 2: rpm install

```
[builder@dec3b658395 /]$ yum info buildpy
Loaded plugins: fastestmirror, ovl

Loading mirror speeds from cached hostfile
* base: ftp.heanet.ie
* epel: www.mirrorservice.org
* extras: ftp.heanet.ie
* updates: ftp.heanet.ie
Installed Packages
Name       : buildpy
Arch       : noarch
Version    : 1.6
Release    : 10
Size       : 99 k
Repo       : installed
Summary    : A Utility for Compiling and Installing Python3 on Linux
URL        : https://bitbucket.org/blakeca00/buildpython3
License    : GPL
Description: Utility for compiling and installing any Python version
            : from source.
            : Supports Amazon Linux v1, Amazon Linux v2, CentOS 6,
            : CentOS 7, Redhat Enterprise Linux 6 and 7

[builder@dec3b658395 /]$
```

---

## Table Of Contents

---



To discover and apply newly-released versions of BUILDPY, follow the steps below for your specific distribution.

### 11.1 Debian, Ubuntu Variants

(1) Find out if an upgraded version of `buildpy` is available:

```
$ sudo apt update && sudo apt list --upgradeable
```

```
Listing... Done
buildpy/bionic 1.7.5 amd64 [upgradable from: 1.7.4]
N: There are 10 additional versions. Please use the '-a' switch to see them.
```

Alternate:

```
$ apt list buildpy -a
```

```
Listing... Done
buildpy/bionic 1.7.5 amd64 [upgradable from: 1.7.4]
buildpy/now 1.7.4 amd64 [installed,upgradable to: 1.7.5]
buildpy/bionic 1.7.3 amd64
buildpy/bionic 1.7.2 amd64
buildpy/bionic 1.7.1 amd64
buildpy/bionic 1.7.0 amd64
buildpy/bionic 1.6.17 amd64
buildpy/bionic 1.6.16 amd64
buildpy/bionic 1.6.15 amd64
buildpy/bionic 1.6.14 amd64
buildpy/bionic 1.6.13 amd64
```

(2) Install available upgrades:

```
$ sudo apt upgrade
```

(3) Verify upgrade:

```
$ sudo apt list buildpy -a
```

```
Listing... Done
buildpy/bionic,now 1.7.5 amd64 [installed]
buildpy/bionic 1.7.3 amd64
buildpy/bionic 1.7.2 amd64
buildpy/bionic 1.7.1 amd64
buildpy/bionic 1.7.0 amd64
buildpy/bionic 1.6.17 amd64
buildpy/bionic 1.6.16 amd64
buildpy/bionic 1.6.15 amd64
buildpy/bionic 1.6.14 amd64
buildpy/bionic 1.6.13 amd64
```

[Back to Table Of Contents](#)

---

## 11.2 Upgrading Redhat-based Distributions

(1) Find out if an upgraded version of buildpy is available

```
$ sudo yum info buildpy
```

```
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: repos-tx.psychz.net
 * extras: repos-tx.psychz.net
 * updates: repos-tx.psychz.net
Installed Packages
Name       : buildpy
Arch       : noarch
Version    : 1.7
Release    : 1
Size       : 124 k
Repo       : installed
From repo  : developer-tools
Summary    : A Utility for Compiling and Installing Python3 on Linux
URL        : https://bitbucket.org/blakeca00/buildpython3
License    : GPL
Description: Utility for compiling and installing any Python version
           : from source.
           :
           : Supports Amazon Linux v1, Amazon Linux v2, CentOS 6,
           : CentOS 7, Redhat Enterprise Linux 6 and 7

Available Packages
Name       : buildpy
Arch       : noarch
Version    : 1.7
Release    : 5
Size       : 37 k
Repo       : developer-tools/x86_64
Summary    : A Utility for Compiling and Installing Python3 on Linux
URL        : https://bitbucket.org/blakeca00/buildpython3
License    : GPL
Description: Utility for compiling and installing any Python version
           : from source.
           :
           : Supports Amazon Linux v1, Amazon Linux v2 (2018+),
           : CentOS 7, Redhat Enterprise Linux 6 and 7
```

(2) Install available upgrades:

```
$ sudo yum update -y
```

```
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: repos-tx.psychz.net
 * epel: mirror.compevo.com
 * extras: repos-tx.psychz.net
 * updates: repos-tx.psychz.net
Resolving Dependencies
--> Running transaction check
---> Package buildpy.noarch 0:1.7-4 will be updated
---> Package buildpy.noarch 0:1.7-5 will be an update
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch          Version      Size      Repository
=====
Updating:
buildpy                noarch        1.7-5        37 k      developer-tools

Transaction Summary
=====
Upgrade 1 Package

Total download size: 37 k
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
buildpy-1.7-5.noarch.rpm                                | 37 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating      : buildpy-1.7-5.noarch                    1/2
  Cleanup      : buildpy-1.7-4.noarch                    2/2
  Verifying     : buildpy-1.7-5.noarch                    1/2
  Verifying     : buildpy-1.7-4.noarch                    2/2

Updated:
  buildpy.noarch 0:1.7-5

Complete!
[root@a43bc8e25317 /]#
```

Table Of Contents





## 12.1 Debian and Ubuntu Variants

`buildpy` may be removed from your system using the debian package manager:

```
$ sudo apt remove buildpy
```

## 12.2 Redhat, CentOS, Fedora RPM Variants

`buildpy` may be removed from your system using the yum package manager:

```
$ sudo yum erase buildpy
```

```
[ec2-user@ip-172-31-18-110 noarch]$ sudo yum erase buildpy
Loaded plugins: priorities, update-motd, upgrade-helper
Resolving Dependencies
--> Running transaction check
--> Package buildpy.noarch 0:1.6-6 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                        Arch      Version      Repository
=====
Removing:
buildpy                        noarch    1.6-6        installed
=====

Transaction Summary
=====
Remove 1 Package

Installed size: 89 k
Is this ok [y/N]: █
```

Answer “y”:

```
Installed Size: 89 K
Is this ok [y/N]: y
Downloading packages:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Erasing      : buildpy-1.6-6.noarch
  Verifying    : buildpy-1.6-6.noarch

Removed:
  buildpy.noarch 0:1.6-6

Complete!
[ec2-user@ip-172-31-18-110 noarch]$
```

---

### Table Of Contents

---

# CHAPTER 13

---

## Use Cases

---

- *Verify Operating System Dependencies*
- *Compile Python 3 - Basic Installation*
- *Compile Python 3 - Advanced Installation*
- *Compile Python 3 - Specific Version*
- *Unattended Install*
- *Uninstalling Python 3 Versions*

---

**Important:**

Root privileges via *sudo* *required*; otherwise execute directly as root user

---

## 13.1 Verify Operating System Dependencies

```
$ buildpy --show os-packages
```

```
[ INFO ] Operating System Info detected:
      Family:    Mint
      Release:   18.3
      Codename:  sylvia
```

Status	Package	Description
INSTALLED	build-essential	Informational list of build-essential packages
AVAILABLE	checkinstall	installation tracker
AVAILABLE	llvm	Low-Level Virtual Machine (LLVM)
INSTALLED	gcc	GNU C compiler
INSTALLED	make	utility for directing compilation
INSTALLED	wget	retrieves files from the web
INSTALLED	zlib1g-dev	compression library - development
INSTALLED	libssl-dev	Secure Sockets Layer toolkit - development files
INSTALLED	libffi-dev	Foreign Function Interface library (development files)
INSTALLED	libbz2-dev	high-quality block-sorting file compressor library - development
INSTALLED	libgdbm-dev	GNU dbm database routines (development files)
INSTALLED	libsqlite3-dev	SQLite 3 development files
INSTALLED	libc6-dev	GNU C Library
AVAILABLE	libncursesw5-dev	developer's libraries for ncursesw
INSTALLED	libncurses5-dev	developer's libraries for ncurses
AVAILABLE	lib64ncurses5	shared libraries for terminal handling (64-bit)
AVAILABLE	lib64ncurses5-dev	developer's libraries for ncurses (64-bit)
INSTALLED	libreadline6	GNU readline and history libraries, run-time libraries
INSTALLED	libreadline6-dev	GNU readline and history libraries, development files
INSTALLED	libreadline-dev	GNU readline and history libraries, development files
INSTALLED	ncurses-term	additional terminal type definitions
AVAILABLE	ncurses-examples	test programs and examples for ncurses
INSTALLED	ncurses-base	basic terminal type definitions
AVAILABLE	ncurses-doc	developer's guide and documentation for ncurses
AVAILABLE	readline-doc	GNU readline and history libraries, documentation and examples
INSTALLED	gdb	GNU Debugger
AVAILABLE	gdb-doc	The GNU Debugger Documentation
INSTALLED	sqlite3	Command line interface for SQLite 3
AVAILABLE	sqlite3-doc	SQLite 3 documentation
INSTALLED	libsqlite3-dev	SQLite 3 development files
AVAILABLE	tk-dev	Toolkit for Tcl and X11 (default version) - development files
INSTALLED	python3-tk	Tkinter - Writing Tk applications with Python 3.x
AVAILABLE	python3-magic	File type determination library using "magic" numbers (Python 3 bindings)
INSTALLED	xz-utils	XZ-format compression utilities

Back to [Use Cases](#) top

## 13.2 Compile Python 3 - Basic Installation

To compile and install the latest Python-3.6 binaries (version 3.6.7 at the time of this post), use the following command:

```
$ sudo buildpy --install Python-3.6
```

Or as root directly:

```
$ sudo su -
```

```
root@dev:~# buildpy --install 3.6
```

Back to [Use Cases](#) top

## 13.3 Compile Python 3 - Advanced Installation

### Python 3 Installation Compiled with Optimizations

**buildpy** can compile Python 3 binaries [with optimizations](#). Although this will lengthen the compile time considerably (30 minutes - 1 hour depending upon system resources), the speed of the Python interpreter code execution will increase by 10% - 20%.

```
$ sudo buildpy --install Python-3.6 --optimizations
```

### Compilation Parallelism

When compiling with optimizations, you may wish to increase the number of parallel compilation processes to decrease the duration of the compile time. With **buildpy**, this is configurable via the `--parallel-processes` option. Up to 9 cpus can be utilized in parallel during compile time.

```
$ sudo buildpy --install Python-3.6 --optimizations --parallel-processes 8
```

The above options will compile Python 3.6 binaries with optimizations utilising 8 cpu's in parallel.

---

#### Note

*4 cpu's is the default parallelism; i.e. 4 cpus are used in parallel during the compile.*

---

Back to [Use Cases](#) top

---

## 13.4 Compile Python 3 - Specific Version

If you wish to compile and install a specific Python 3 binary, meaning a specific major and minor Python 3 version, simply add the full version label instead of just providing the Python major revision number:

```
$ sudo buildpy --install Python-3.7.6
```

Alternatively, for all installation types listed in this guide, the *Python-* prefix may be omitted:

```
$ sudo buildpy --install 3.7.6
```

The two commands above are equivalent and will compile and install Python 3 version 3.6.9.

Back to [Use Cases](#) top

---

## 13.5 Unattended Install

If run via unattended script, use `--quiet` to suppress stdout messages:

```
$ sudo buildpy --install 3.6 --quiet
```

A running log is created in all execution states to `/var/log/buildpy.log`. See an [example log file](#).

Back to [Use Cases](#) top

---

## 13.6 Uninstalling Python 3 Versions

At any time you may choose to uninstall a system-wide, compiled version of Python 3 using **buildpy**. The following command example uninstalls Python 3.6:

```
$ sudo buildpy --uninstall 3.6 --purge
```

If a system-wide Python 3 version pre-existed prior to installing a compiled Python 3 version, the *python3* symlink will be redirected to the previous Python 3 version once the uninstall operation completes.

---

### Important:

For stability, *only compiled Python 3 versions can be uninstalled* with **buildpy**. Native, system-wide Python versions installed from the operating system package repositories must be uninstalled using the OS package manager application.

---

Back to [Use Cases](#) top

---

Back to [Table Of Contents](#)

---

---

## Frequently Asked Questions

---

### *General Questions*

- *Q: What is buildpy? Why do I care?*
- *Q: What the main use cases for buildpy?*

### *Required User Permissions*

- *Q: What permissions are required to compile and install python3 with buildpy?*
- *Q: What happens if I run buildpy as root?*

### *Python Version Support*

- *Q: What python3 versions can I compile and install with buildpy*
- *Q: Why do I not just install python3 from my os distribution's package repository insted of using buildpy?*
- *Q: Does buildpy overwrite the old system-level Python3 version?*

### *Supported Linux Operating System Distributions*

- *Q: If buildpy is written in bash, I can run it on any Linux OS, right?*
- *Q: What Redhat-based distributions does buildpy install via an RPM package?*

### *Uninstall Python3*

- *Q: What Python3 versions can I safely remove from my system with buildpy?*

### *Logging FAQ*

- *Q: Which log files does buildpy write to?*

### *Miscellaneous Questions*

- *Q: Since buildpy requires an Internet connection to download python binaries, does it send any information to you or anyone else?*
-

## 14.1 General Questions

### 14.1.1 Q: What is buildpy? Why do I care?

A: Seriously, you may not care, but if you use Linux, you probably should. `buildpy` is only a fantastic utility or tool for anyone who needs to install a specific version of Python on a Linux hardware, virtual machine, or container. It is also something anyone who has to remove and replace a version of Python in machine or machine image.

### 14.1.2 Q: What the main use cases for buildpy?

A: General use cases for `buildpy` are:

1. Use buildpy to find out when new python X.Y version is available (below)

```
$ buildpy --show os-packages
```

2. You want an easy to use, reliable means of compiling the latest version of Python after release from <https://python.org>.

```
$ buildpy --profile default --operation list           # list key information
```

3. `_Uninstall_`: You need to install a version of Python3 you compiled on your machine (not a system Python that came with your distribution)

Back to [Frequently Asked Questions](#) top

---

## 14.2 Required User Permissions

### 14.2.1 Q: What permissions are required to compile and install python3 with buildpy?

A: It depends on what functionality you wish to invoke. `buildpy` runs the majority of operations under normal user permissions. The exception is when compiling and installing a new version of Python3. This is because `buildpy` installs a compiled version of Python3 for the entire system; thus, the `--install` operation requires root level permissions via `sudo` or assumed by running `buildpy` as root directly.

The following requires `sudo` or root permissions:

```
$ sudo buildpy --install Python3.7 [ --optimizations ]
```

### 14.2.2 Q: What happens if I run buildpy as root?

A: In essence, nothing changes when you run `buildpy` directly as root. You are installing a new version of Python3 for the entire system either when running `buildpy` as `sudo` or directly via root permissions.

Back to [Frequently Asked Questions](#) top

---



## 14.3 Python Version Support

### 14.3.1 Q: What python3 versions can I compile and install with buildpy

A: It depends on what functionality you wish to invoke. buildpy runs the majority of operations under normal user permissions. The exception is when compiling and installing a new version of Python3. This is because buildpy installs a compiled version of Python3 for the entire system; thus, the `--install` operation requires root level permissions via `sudo` or assumed by running buildpy as root directly.

The following requires `sudo` or root permissions:

```
$ buildpy --install Python3.7 [ --optimizations ]
```

Back to [Frequently Asked Questions](#) top

---

### 14.3.2 Q: Why do I not just install python3 from my os distribution's package repository insted of using buildpy?

A: That is a bit of a philosophical question, so let's just stick to hard benefits that are widely accepted. The reason you would use buildpy to compile a custom version of python3 are:

1. Compiled Python interpreters are reported to enjoy an approximate 10% execution speed improvement
2. You care about upgrading to the latest minor Python3.7 revision when it is released (example: 3.7.2 to 3.7.3).
3. You are required to run an *older* Python version (say, Python 3.4) and a modern machine for testing purposes
4. You want to enable custom functionality not enabled in the Python3 binaries installed from your OS distribution's package repository.

Back to [Frequently Asked Questions](#) top

---

### 14.3.3 Q: Does buildpy overwrite the old system-level Python3 version?

A: No. Let's say you have Python3.4 installed by default on a Ubuntu 14.04 system. You want to install the latest . At the completion of the installation, will still function for os-dependent subsystems.

Back to [Frequently Asked Questions](#) top

---

## 14.4 Supported Linux Operating System Distributions

### 14.4.1 Q: If buildpy is written in bash, I can run it on any Linux OS, right?

A: Yes and No actually, or as they say in situation like this: *it depends*.

It depends upon meeting the minimum requirements outlined in the [depenencies](#) section. Bash 4.4+ is required as well as a few other critical os packages such as , , and .

Back to [Frequently Asked Questions](#) top

---

### 14.4.2 Q: What Redhat-based distributions does buildpy install via an RPM package?

A: See the list . buildpy is also compatible with Amazon Linux 2 when run on EC2 virtual machines.

Back to [Frequently Asked Questions](#) top

---

## 14.5 Uninstall Python3

### 14.5.1 Q: What Python3 versions can I safely remove from my system with buildpy?

A: As a built-in safeguard, buildpy will only remove compiled versions from your system. This is any python installed in `/usr/local/`. This is location of any python versions compiled from source.

Most if not all system-wide Python binaries install from your Linux distribution's official package repository reside in `/usr/bin`; thus, buildpy avoids this location and will *not remove* python binaries from this location as a precaution.

To initiate the uninstall operation, type the following:

```
$ sudo buildpy --uninstall Python3.6
```

Back to [Frequently Asked Questions](#) top

---

## 14.6 Logging FAQ

### 14.6.1 Q: Which log files does buildpy write to?

A: buildpy produces the following log files:

- `/var/log/buildpy.log` contains summary messages produced during normal operations
- `/var/log/console.log` contains all console messages produced only during compile and install operations

Back to [Frequently Asked Questions](#) top

---

## 14.7 Miscellaneous Questions

### 14.7.1 Q: Since buildpy requires an Internet connection to download python binaries, does it send any information to you or anyone else?

A: None whatsoever. buildpy can actually function fine without any Internet connection if the python binaries are downloaded offline and placed in `/tmp` so that they used as the source during the install.

Back to [Frequently Asked Questions](#) top

---

Table Of Contents

---



---

## Screenshot Index

---

### **System Info** (`--info`)

- *Operating System Detection*
- *Program Information*

### **Show Functionality** (`--show`):

- *Python Binary Version Status*
- *OS Packages Installed Status*

### **Download** (`--download`):

- *Download Python Source (Default)*
- *Download Python Source, Named Version*

### **Logging**

- *Log Files*

### **Clean** (`--clean`):

- *Remove Build Artifacts*

[Back to Table Of Contents](#)

---



# CHAPTER 16

---

## Operating System Detection

---

Test OS detection

```
$ buildpy --os-detect
```

### Ubuntu / Linux Mint

A terminal window with a dark blue background and light blue text. The output shows the detected operating system information for Ubuntu.

```
[ INFO ] Operating System Info detected:  
Family:      Ubuntu  
Release:     18.04  
Codename:    bionic
```

### Amazon Linux

A terminal window with a dark blue background and light blue text. The output shows the detected operating system information for Amazon Linux.

```
[ INFO ] Operating System Info detected:  
Family:      AmazonLinux  
Release:     2  
Codename:
```

Back to [Screenshot Index](#) index

---





## CHAPTER 17

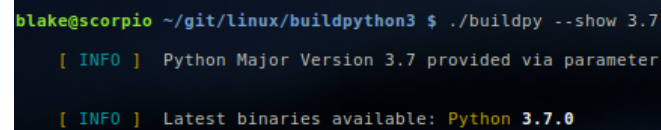
---

### Python Binary Version Status

---

Show the latest Python 3.7 version available for install

```
$ buildpy --show Python-3.7
```



```
blake@scorpio ~/git/linux/buildpython3 $ ./buildpy --show 3.7
[ INFO ] Python Major Version 3.7 provided via parameter

[ INFO ] Latest binaries available: Python 3.7.0
```

Back to [Screenshot Index](#) index

---



## OS Packages Installed Status

Show operation system package prerequisites and installed status of each

```
$ buildpy --show os-packages
```

```
[ INFO ] Operating System Info detected:
      Family:      Mint
      Release:     18.3
      Codename:    sylvia
```

Status	Package	Description
INSTALLED	build-essential	Informational list of build-essential packages
AVAILABLE	checkinstall	installation tracker
AVAILABLE	llvm	Low-Level Virtual Machine (LLVM)
INSTALLED	gcc	GNU C compiler
INSTALLED	make	utility for directing compilation
INSTALLED	wget	retrieves files from the web
INSTALLED	zlib1g-dev	compression library - development
INSTALLED	libssl-dev	Secure Sockets Layer toolkit - development files
INSTALLED	libffi-dev	Foreign Function Interface library (development files)
INSTALLED	libbz2-dev	high-quality block-sorting file compressor library - development
INSTALLED	libgdbm-dev	GNU dbm database routines (development files)
INSTALLED	libsqlite3-dev	SQLite 3 development files
INSTALLED	libc6-dev	GNU C Library
AVAILABLE	libncursesw5-dev	developer's libraries for ncursesw
INSTALLED	libncurses5-dev	developer's libraries for ncurses
AVAILABLE	lib64ncurses5	shared libraries for terminal handling (64-bit)
AVAILABLE	lib64ncurses5-dev	developer's libraries for ncurses (64-bit)
INSTALLED	libreadline6	GNU readline and history libraries, run-time libraries
INSTALLED	libreadline6-dev	GNU readline and history libraries, development files
INSTALLED	libreadline-dev	GNU readline and history libraries, development files
INSTALLED	ncurses-term	additional terminal type definitions
AVAILABLE	ncurses-examples	test programs and examples for ncurses
INSTALLED	ncurses-base	basic terminal type definitions
AVAILABLE	ncurses-doc	developer's guide and documentation for ncurses
AVAILABLE	readline-doc	GNU readline and history libraries, documentation and examples
INSTALLED	gdb	GNU Debugger
AVAILABLE	gdb-doc	The GNU Debugger Documentation
INSTALLED	sqlite3	Command line interface for SQLite 3
AVAILABLE	sqlite3-doc	SQLite 3 documentation
INSTALLED	libsqlite3-dev	SQLite 3 development files
AVAILABLE	tk-dev	Toolkit for Tcl and X11 (default version) - development files
INSTALLED	python3-tk	Tkinter - Writing Tk applications with Python 3.x
AVAILABLE	python3-magic	File type determination library using "magic" numbers (Python 3 bindings)
INSTALLED	xz-utils	XZ-format compression utilities

Back to [Screenshot Index](#) index



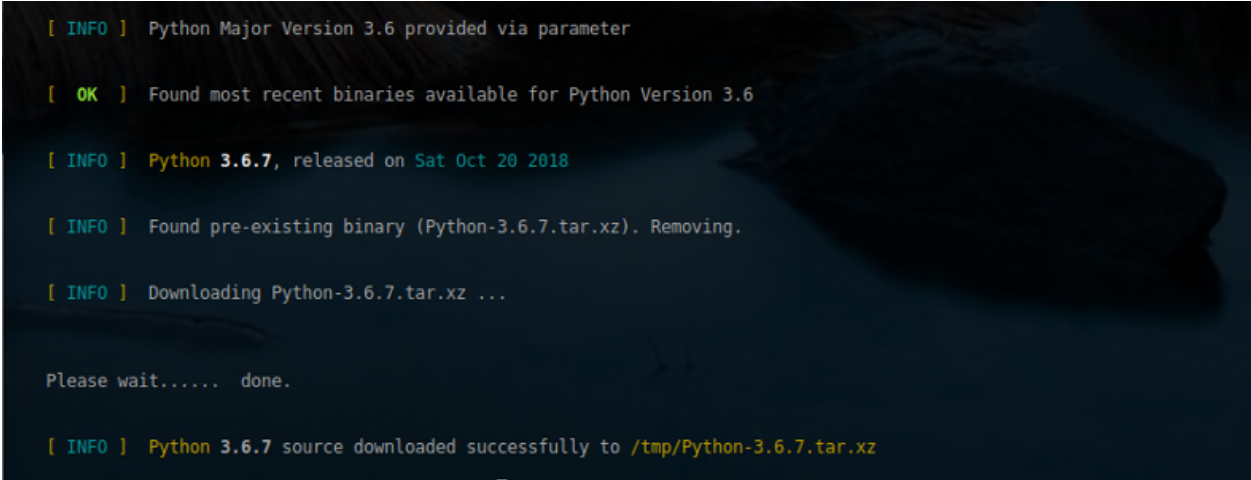
## CHAPTER 19

---

### Download Python Source (Default)

---

```
$ buildpy --download          # default, download Python 3.6 binaries
```



```
[ INFO ] Python Major Version 3.6 provided via parameter  
[ OK ] Found most recent binaries available for Python Version 3.6  
[ INFO ] Python 3.6.7, released on Sat Oct 20 2018  
[ INFO ] Found pre-existing binary (Python-3.6.7.tar.xz). Removing.  
[ INFO ] Downloading Python-3.6.7.tar.xz ...  
  
Please wait..... done.  
[ INFO ] Python 3.6.7 source downloaded successfully to /tmp/Python-3.6.7.tar.xz
```

Back to [Screenshot Index](#) index

---



## CHAPTER 20

---

### Download Python Source, Named Version

---

```
$ buildpy --download 3.4
```



```
blake@libra-xps13 ~/git/linux/buildpython3 $ buildpy --download 3.4
[ INFO ] Python Major Version 3.4 provided via parameter
[ INFO ] Latest binaries available: Python 3.4.9
[ INFO ] Downloading Python-3.4.9.tar.xz ...
--2018-09-09 15:39:25-- https://www.python.org/ftp/python/3.4.9/Python-3.4.9.tar.xz
Resolving www.python.org (www.python.org)... 151.101.184.223, 2a04:4e42:2c::223
Connecting to www.python.org (www.python.org)|151.101.184.223|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 14541804 (14M) [application/octet-stream]
Saving to: 'Python-3.4.9.tar.xz'
100%[=====>] 14,541,804 5.87MB/s in 2.4s
2018-09-09 15:39:28 (5.87 MB/s) - 'Python-3.4.9.tar.xz' saved [14541804/14541804]
```

[Back to Screenshot Index](#) index

---





## CHAPTER 21

---

### Remove Build Artifacts

---

```
$ buildpy --clean
```

A terminal window with a dark background showing the output of the 'buildpy --clean' command. The output consists of four lines: an info message about locating artifacts, and three 'OK' messages confirming the successful removal of Python 3.7.1, 3.6.7, and 3.1.5 tar.xz files from the /tmp directory.

```
[ INFO ] Locating all local build artifacts...  
[ OK ] Successfully removed /tmp/Python-3.7.1.tar.xz  
[ OK ] Successfully removed /tmp/Python-3.6.7.tar.xz  
[ OK ] Successfully removed /tmp/Python-3.1.5.tar.xz
```

Back to [Screenshot Index](#) index

---



## CHAPTER 22

---

### Program Information

---

Detailed information regarding the local installation of `buildpy` program and dependencies:

```
$ buildpy --info
```

---

**Build Python3 from Source | Linux**

Module Name:        **buildpy**  
Module Version:    **1.6.8**

---

Bash Modules:        6

Module Name / Filesystem Location:

- buildpy	/usr/local/bin
- colors.sh	/usr/local/lib/buildpy
- exitcodes.sh	/usr/local/lib/buildpy
- os_distro.sh	/usr/local/lib/buildpy
- std_functions.sh	/usr/local/lib/buildpy
- version.py	/usr/local/lib/buildpy

---

Bash Functions:      55

- array2json
- authenticated
- binary\_depcheck
- binary\_installed\_boolean
- clean\_up
- configure\_python
- continue\_on
- convert\_time
- convert\_time\_months
- delay\_spinner
- depcheck
- display\_program\_version
- download\_binary
- download\_progress
- environment\_info
- file\_check
- get\_latest\_version
- help\_menu
- indent02
- indent04
- indent10
- indent15
- indent18
- indent20
- indent25
- is\_float
- is\_installed
- is\_int
- is\_number
- linux\_distro
- logfile\_owner
- mlocate\_installation
- os\_packages
- os\_type
- parse\_parameters
- pip\_backup
- pkg\_info
- print\_footer
- print\_header
- print\_separator
- print\_stats
- python\_module\_depcheck
- python\_version\_depcheck
- root\_permissions
- source\_profile
- std\_error
- std\_error\_exit
- std\_logger
- std\_message
- std\_warn
- test\_removal
- timer
- uninstall
- update\_path
- upgrade\_pip

Back to [Screenshot Index](#) index

---



---

### Log Files

---

`buildpy` actively maintains 2 log files:

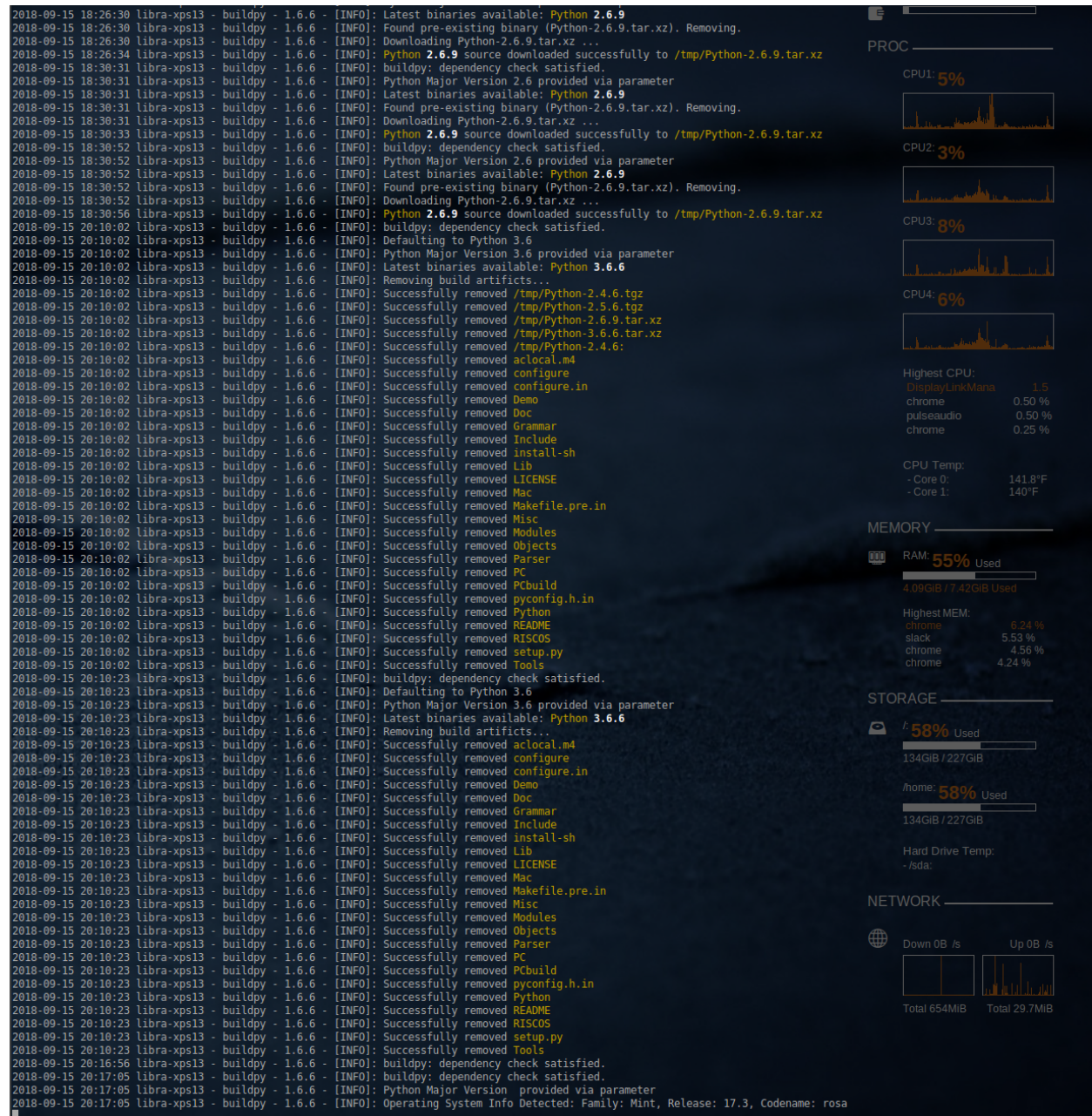
1) **buildpy.log**

- Logging for all non-compile functions
- Main system log messages (shown below). Located at `/var/log/buildpy.log`.

2) **console.log:**

- Messages during Compile & Build
- Populated for tracking purposes when `--quiet` flag set to suppress output to stdout.

```
$ tail -n 100 /var/log/buildpy.log
```





---

## buildpy Source Code

---

### 24.1 Bash Module Index

- *buildpy*
  - *std\_functions.sh*
  - *os\_distro.sh*
  - *colors.sh*
  - *exitcodes.sh*
  - *version.py*
- 

Table Of Contents

---

### 24.2 buildpy

Back to *Bash Module Index*

```
1  #!/usr/bin/env bash
2
3  #
4  #   Summary:
5  #       - Script to retrieve, compile, and install specific python3 interpreter
6  #       - Compile and install Python3 versions: 3.0 thru 3.7+
7  #       - Binaries installed at OS level enable ipython, sqlite, other useful
```

(continues on next page)

(continued from previous page)

```

8  #           packages during python development
9  #
10 #
11 #   Copyright 2017-2018 Blake Huber.  All rights reserved.
12 #
13
14
15 # --- global vars -----
16 ↪-----
17
18 pkg=$(basename $0)                                # pkg (script) full name
19 pkg_root=$(echo $pkg | awk -F '.' '{print $1}')    # pkg without file extension
20 pkg_path=$(cd $(dirname $0); pwd -P)              # location of pkg
21 pkg_lib="$pkg_path/core"
22 TMPDIR='/tmp'
23 workdir="$HOME/Downloads"
24 NOW=$(date +%Y%m%d)
25 LOG_DIR="$HOME/logs"
26 LOG_FILE="$LOG_DIR/$pkg_root.log"
27 LOG_CONSOLE="$LOG_DIR/console.log"
28
29 # source colors, exitcodes, version file, and std_function defs | NOTE: source order_
30 ↪dependent
31 source $pkg_lib/colors.sh
32 source $pkg_lib/exitcodes.sh
33 source $pkg_lib/version.py
34 source $pkg_lib/std_functions.sh
35
36 # formatting
37 bbc=$(echo -e ${bold}${a_brightcyan})             # bold bright cyan highlight
38 bbw=$(echo -e ${bold}${a_brightwhite})            # bold, bright white highlight
39 title=$(echo -e ${bold}${a_brightwhite})          # title color, white + bold
40 hic=$(echo -e ${bold}${a_brightyellowgreen})      # help menu accent 1
41 bin=$(echo -e ${bold}${a_orange})                 # help menu binary accent
42 ul=$(echo -e ${underline})                        # std underline
43 bd=$(echo -e ${bold})                             # std bold
44 wt=$(echo -e ${a_brightwhite})                    # help menu accent 2
45 fs=$(echo -e ${yellow})                          # file path color
46 btext=${reset}                                    # clear accents; rtn to native_
47 ↪term colors
48
49 # optimizations increase performance, but lengthen
50 # build time (see -o, --optimizations parameter)
51 OPTIMIZATIONS="false"
52 PARALLEL_JOB_CT='4'
53 DEFAULT_MAJOR_VERSION='3.6'
54
55 # --- declarations -----
56 ↪-----
57
58 function help_menu() {
59     cat <<EOM
60
61                                     ${title}Build ${hic}Python3 ${title}from Source | ${hic}Linux$
62 ↪{btext}

```

(continues on next page)

(continued from previous page)

```

61  ${title}DESCRIPTION${btext}
62
63      Utility to compile and install the latest versions of python3
64      official source binaries download from ${url}https://www.python.org${btext}
65
66  ${title}SYNOPSIS${btext}
67
68      $ ${bin}$pkg${reset}  --install  ${bbc}|${btext}  --uninstall  ${bbc}|$
69  ↪ ${btext}  --show
70
71          -I | --install <value>
72          -d | --download <value>
73          [-c | --clean <value> ]
74          [-o | --optimization  ]
75          -s | --show <value>
76          [-i | --info  ]
77          [-q | --quiet  ]
78          [-t | --os-detect  ]
79          [-b | --backup-pip  ]
80          [-V | --version  ]
81          -U | --uninstall <value>
82
83      ${title}Note:${btext}  Requires sudo or root privileges
84
85  ${title}OPTIONS${btext}
86      ${title}-b${btext}, ${title}--backup-pip${btext}: Create local Backup copy of ↪
87  ↪ installed python
88      packages (single operation only)
89
90      ${title}-c${btext}, ${title}--clean${btext}: Remove all build and ↪
91  ↪ installation artifacts. Can
92      optionally be invoked with a Python major revision number
93      provided as a parameter to clean specific build artifacts
94
95      ${title}-d${btext}, ${title}--download${btext}: Download Python build ↪
96  ↪ artifacts, but take no
97      other action. Optionally can be invoked with Python major
98      revision number to download a specific Python binary set
99
100      ${title}-I${btext}, ${title}--install${btext} <value>: Build, compile, and ↪
101  ↪ install Python bin-
102      aries on local machine for all system users. Value
103      indicates the latest Python minor version for the release
104      specified by <value>:
105
106      $ sudo ${bin}$pkg${reset}  --install  3.7
107
108      ${title}-o${btext}, ${title}--optimizations${btext}: Compile with configure ↪
109  ↪ option --enable-
110      optimizations turned-on. Increases duration of compile
111      time, but results in increased execution speed of the C-
112      Python compiler. Recommended if abundant cpu resources.
113
114      ${title}-i${btext}, ${title}--info${btext}: Display detailed information on ↪
115  ↪ the $pkg
116      program and functions it contains

```

(continues on next page)

(continued from previous page)

```

111     ${title}-q${btext}, ${title}--quiet${btext}: (Optional) Supress output to_
112 ↪ stdout. Option is
113         invoked for scripted or unattended compile and install.
114
115     ${title}-s${btext}, ${title}--show${btext}: (Optional) Show latest Python_
116 ↪ minor version avail-
117         able for named Python major version (DEFAULT: 3.6)
118
119     ${title}-t${btext}, ${title}--os-detect${btext}: Detect Operating System type_
120 ↪ and exit (single
121         operation only)
122
123     ${title}-U${btext}, ${title}--uninstall${btext}: Uninstall the Linux system_
124 ↪ Python3 binaries
125         and related libraries and support artifacts from the local
126         file system for Python major revision given as a parameter
127
128     ${title}-V${btext}, ${title}--version${btext}: Display $pkg version and_
129 ↪ license information
130
131     ${title}EXAMPLES${btext}
132         ${title}Compile and install latest Python3.7 with Optimizations${btext}
133         $ sudo $pkg --install 3.7 --optimizations
134
135         ${title}Show which os-package dependencies are available${btext}
136         $ sudo $pkg --show os-packages
137
138         ${title}Show most recent Python 3.4 source binary available${btext}
139         $ $pkg --show Python-3.4
140
141     _____
142
143     ${btext}Documentation:  ${url}https://buildpy.readthedocs.io${btext}
144
145     _____
146
147     ${reset}
148     EOM
149
150     #
151     # <-- end function put_rule_help -->
152 }
153
154 function create_global_symlink(){
155     ##
156     ## Creates python3 symlink to python executable if missing
157     ##
158     local major_version="$1"
159     local log_file="$2"
160
161     if [ ! "$(command -v python3 2> /dev/null )" ]; then
162         ln -s /usr/local/bin/"$major_version" /usr/bin/python3
163         std_message "Created global symlink to Python $major_version binary" "INFO" "
164 ↪ $log_file"
165     else
166         std_message "Found global symlink to Python $major_version. Skip creation."
167 ↪ "INFO" "$log_file"
168     fi
169     #

```

(continues on next page)

(continued from previous page)

```

161     #<-- end function create_global_symlink -->
162 }
163
164
165 function install_subcommand_help(){
166     ##
167     ##   Display help menu for install subcommands
168     ##
169     cat <<EOM
170
171         ${title}install${btext} options help
172
173     ${title}DESCRIPTION${btext}
174
175         Optional parameters for use when installing Python3 binaries
176         Complementary options to the --install command-line option.
177
178     ${title}SYNOPSIS${btext}
179
180         $ ${bin}$pkg${reset} --install <${bbc}value${btext}>    [ OPTIONS ]
181
182         [ --optimizations   ]
183         [ --parallel-processes ]
184         [ --quiet           ]
185
186     ${title}OPTIONS${btext}
187
188     ${title}-I${btext}, ${title}--install${btext} <value>: Build, compile, and
189     ↪install Python bin-
190         aries on local machine for all system users. Value
191         indicates the latest Python minor version for the release
192         specified by <value>:
193
194         $ sudo ${bin}$pkg${reset} --install 3.7
195
196     ${title}-p, --parallel-processes${btext} <value>: Number of parallel processes
197         (1-9) to use during compile with make. Setting to a value
198         of one (1) equals no parallelism, Default = 4. Use of low
199         setting on systems with low amount of free resources rec-
200         commended.
201
202     ${title}-o${btext}, ${title}--optimizations${btext}: Compile with configure
203     ↪option --enable-
204         optimizations turned-on. Increases duration of compile
205         time, but results in an increased execution speed of the
206         C-Python compiler. Recommended if abundant cpu resources.
207
208     ${title}-q${btext}, ${title}--quiet${btext}: (Optional) Supress output to
209     ↪stdout. Option is
210         invoked for scripted or unattended compile and install.
211
212     ${title}help${btext}: Option to display the commit-log help menu contents
213
214         $ ${BOLD}$pkg${reset} --install ${cyan}help${btext}
215
216     ${reset}
217 EOM

```

(continues on next page)

(continued from previous page)

```

215     #
216     # <-- end function commit_log_help -->
217 }
218
219
220 function is_float() {
221     local num="$1"
222     local regex='^[0-9]+[.][0-9]+?${'
223     local regex2='^[0-9]+[.][0-9][0-9]+?${'
224     #
225     if [[ $num =~ $regex ]] || [[ $num =~ $regex2 ]]; then
226         # int or float
227         return 0
228     fi
229     # not a number
230     return 1
231 }
232
233
234 function parse_parameters() {
235     ##
236     ## Parse all command-line parameters
237     ##
238
239     local var parameter major
240
241     if [[ ! $@ ]]; then
242         help_menu | more
243         exit 0
244     else
245         while [ $# -gt 0 ]; do
246             case $1 in
247                 -h | --help)
248                     help_menu | more
249                     shift 1
250                     exit 0
251                     ;;
252
253                 -b | --backup-pip)
254                     PIP_BACKUP="true"
255                     shift 1
256                     ;;
257
258                 -c | --clean)
259                     CLEAN_UP="true"
260                     if [ "$2" ]; then
261                         case "$2" in
262                             'ALL' | 'all' | 'everything')
263                                 CLEAN_VERSION="ALL"
264                                 shift 2
265                                 ;;
266                             *)
267                                 var="$2"
268                                 parameter=${var#Python-}
269                                 if is_float "$parameter"; then
270                                     MAJOR_VERSION="$parameter"
271                                     CLEAN_VERSION="$parameter"

```

(continues on next page)

(continued from previous page)

```

272             shift 2
273         else
274             std_error_exit "You must provide Python major_
↪version (Example: 3.7)"
275         fi
276     ;;
277     esac
278
279     else
280         CLEAN_VERSION="ALL"
281         shift 1
282     fi
283     shift 1
284     ;;
285
286     '-P' | '--purge')
287         # uninstall option, full system expunge
288         WIPE_CLEAN=true      # clean invoked with --uninstall option
289         shift 1
290         ;;
291
292     -d | --download)
293         DOWNLOAD_ONLY="true"
294         if [ "$2" ]; then
295             var="$2";
296             parameter=${var#Python-}
297             if is_float "$parameter"; then
298                 MAJOR_VERSION="$parameter"
299                 shift 2
300             else
301                 std_error_exit "You must provide Python major version_
↪(Example: 3.7)"
302             fi
303         else
304             MAJOR_VERSION="$DEFAULT_MAJOR_VERSION"
305             std_message "Defaulting to Python $MAJOR_VERSION" "INFO" $LOG_
↪FILE
306             shift 1
307         fi
308         shift 1
309         ;;
310
311     -I | --install)
312         INSTALL="true"
313
314         if [ "$2" ] && [ "$2" = "help" ]; then
315             install_subcommand_help
316             exit 0
317
318         elif [ "$2" ] && [ "$2" = "os-packages" ]; then
319             OPERATION="INSTALL_OS_DEPS"      # install os dependencies
320             INSTALL_OS_DEPS="true"
321             shift 2
322
323         elif [ "$2" ]; then
324             var="$2"
325             parameter=${var#Python-}

```

(continues on next page)

(continued from previous page)

```

326         # stdout allowed
327         case "$parameter" in
328             [0-9].[0-9] | [0-9].[0-9][0-9] | [0-9].[0-9].[0-9] | _
↪ [0-9].[0-9][0-9].[0-9])
329             # floating point - Python major version number
330             MAJOR_VERSION="$parameter"
331             shift 2
332             ;;
333             "os" | "packages" | "deps" | "os_deps" | "os_packages
↪ ")
334             OPERATION="INSTALL_OS_DEPS"      # install os_
↪ dependencies
335             INSTALL_OS_DEPS="true"
336             shift 2
337             ;;
338             * )
339             tab10='          '
340             msg2="${tab10}or 'os-packages' to install_
↪ operating system dependencies."
341             std_message "You must provide either a Python_
↪ Major Version (Example: 3.6),\n${msg2}" "INFO" $LOG_FILE
342             exit $E_DEPENDENCY
343             ;;
344         esac
345     else
346         std_error_exit "You must provide Python major version_
↪ (Example: 3.7) or 'os_deps'"
347     fi
348     ;;
349
350     -t | --os-detect)
351         OS_DETECT="true"
352         shift 1
353         ;;
354
355     -o | --optimization | --optimizations | --optimize)
356         OPTIMIZATIONS="true"
357         shift 1
358         ;;
359
360     -i | --info)
361         OPERATION="PKG_INFO"
362         PKG_INFO="true"
363         shift 1
364         ;;
365
366     -p | --parallel-processes)
367         if [ "$2" ] && [ "$2" = "help" ]; then
368             install_subcommand_help
369             exit 0
370         elif [ "$2" ]; then
371             var="$2"
372             PARALLEL_JOB_CT=${var%\-*}
373             shift 2
374         else
375             std_error_exit "You must provide a parallel process count (0-
↪ 9) when using --parallel-processes option" $E_DEPENDENCY

```

(continues on next page)



(continued from previous page)

```

376         fi
377         ;;
378
379     -q | --quiet)
380         QUIET="true"
381         shift 1
382         ;;
383
384     -s | --show)
385         if [ "$2" ]; then
386             OPERATION='SHOW'
387             SUBCMD="$2"
388             if [ $3 ]; then
389                 alternate_os="$3"
390                 shift 3
391             else
392                 shift 2
393             fi
394         else
395             shift 1
396         fi
397         ;;
398
399     -U | --uninstall)
400         UNINSTALL="true"
401         if [ "$2" ]; then
402             if [ "$(echo "$2" | grep Python)" ]; then
403                 # convert to major version number
404                 var="$2"; parameter=${var#Python-}
405             else
406                 parameter="$2"
407             fi
408             if is_float "$parameter"; then
409                 MAJOR_VERSION="$parameter"
410                 UNINSTALL_VERSION="$parameter"
411             else
412                 std_error_exit "You must provide Python major version_
↪ (Example: 3.7) "
413             fi
414             shift 2
415         fi
416         ;;
417
418     -V | --version)
419         DISPLAY_VERSION="true"
420         shift 1
421         ;;
422
423     *)
424         echo "unknown parameter. Exiting"
425         exit 1
426         ;;
427     esac
428 done
429 fi
430
431 # defaults

```

(continues on next page)

(continued from previous page)

```

432     if [ ! $MAJOR_VERSION ]; then
433
434         # Python Major version
435         MAJOR_VERSION="$DEFAULT_MAJOR_VERSION"
436         std_logger "Defaulting to Python $MAJOR_VERSION" "INFO" $LOG_FILE
437
438     else
439
440         # major version supplied with command options
441         std_message "Python Major Version $MAJOR_VERSION provided via parameter" "INFO"
442         ↪ "$LOG_FILE
443
444     fi
445
446     # eliminate tty output if quiet=true
447     if [ $QUIET ]; then tty="/dev/null"; else tty="/dev/tty"; fi
448     #
449     # <-- end function parse_parameters -->
450 }
451
452 function binary_depcheck() {
453     ##
454     ## exit if binary dependencies not installed ##
455     ##
456     local check_list=( "$@" )
457     local msg
458
459     for prog in "${check_list[@]}; do
460
461         if ! type "$prog" > /dev/null 2>&1; then
462             # print error msg
463             msg="${title}$prog${reset} is required and not found in the PATH (code $E_
464             ↪DEPENDENCY) "
465             std_warn "$msg" $LOG_FILE
466             # print pkg description info
467             printf -- '%s\n' "$(cat $pkg_lib/$prog.pkg)"
468             exit $E_DEPENDENCY
469         fi
470
471     done
472     # <-- end function binary_depcheck -->
473 }
474
475
476 function reset_rc_version() {
477     ##
478     ## Reset VERSION global variable in case binary set to be
479     ## compiled is a release candidate, alpha, or beta code.
480     ##
481     local version="$1"
482
483     if [[ $(ls $TMPDIR | grep "Python-$version.tar.xz") ]]; then
484         echo "$version"
485         std_logger "VERSION retained as ${fs}$VERSION${reset} -- no refactor required
486         ↪ " "INFO" $LOG_FILE

```

(continues on next page)

(continued from previous page)

```

486
487     elif [[ $(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz") ]]; then
488         bin=$(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz")
489         echo "$(echo $bin | awk -F '-' '{print $2}')" | awk -F 'tar' '{print $1}' |
→rev | cut -c 2-20 | rev
490         std_logger "Reset VERSION to ${fs}$VERSION${reset}" "INFO" $LOG_FILE
491
492     elif [[ $(ls $TMPDIR | grep "Python-${version}[a-b][1-4].tar.xz") ]]; then
493         bin=$(ls $TMPDIR | grep "Python-${version}[a-b][1-4].tar.xz")
494         echo "$(echo $bin | awk -F '-' '{print $2}')" | awk -F 'tar' '{print $1}' |
→rev | cut -c 2-20 | rev
495         std_logger "Reset VERSION to ${fs}$VERSION${reset}" "INFO" $LOG_FILE
496     fi
497 }
498
499
500 function binary_installed_bool() {
501     ##
502     ## return boolean value if binary dependencies installed ##
503     ##
504     local check_list=( "$@" )
505     #
506     for prog in "${check_list[@]}; do
507         if ! type "$prog" > /dev/null 2>&1; then
508             return 1
509         fi
510     done
511     return 0
512     #
513     # <<-- end function binary_installed_bool -->>
514 }
515
516
517 function configure_python() {
518     ##
519     ## function runs configure stage with appropriate options ##
520     ##
521     local optimizations="$1"
522     local quiet="$2"
523     #
524     if [ "$optimizations" = "true" ] && [ ! $quiet ]; then
525
526         ./configure --enable-optimizations | tee /dev/tty >> $LOG_CONSOLE
527
528     elif [ "$optimizations" = "true" ] && [ $quiet ]; then
529
530         ./configure --enable-optimizations >> $LOG_CONSOLE
531
532     elif [ "$optimizations" = "false" ] && [ ! $quiet ]; then
533
534         ./configure | tee /dev/tty >> $LOG_CONSOLE
535
536     elif [ "$optimizations" = "false" ] && [ $quiet ]; then
537
538         ./configure >> $LOG_CONSOLE
539
540     fi

```

(continues on next page)

(continued from previous page)

```

541
542     return 0
543     #
544     # <-- end function configure_python -->
545 }
546
547
548 function continue_on() {
549     ##
550     ## user prompts to continue path ##
551     ##
552     local msg="$1"
553     local choice
554
555     if [ "$msg" ]; then
556         std_message "$msg" "INFO" $LOG_FILE
557     fi
558
559     read -p "$(printf -- '\t\tContinue? [quit]:  \n\n' )" choice
560
561     if [ -z $choice ] || [ "$choice" = "q" ] || [ "$choice" = "quit" ]; then
562         return 1
563     else
564         case $choice in
565             'Yes' | 'yes' | 'Y' | 'y' | true)
566                 return 0
567                 ;;
568             *)
569                 return 1
570             esac
571     fi
572     #
573     # <-- end function continue_on -->
574 }
575
576
577 function _current_downloads() {
578     ##
579     ## Examines local fs for downloaded artifacts
580     ##
581     ## - returns entry for each python binary set downloaded to /tmp
582     ##
583     local index="0"
584     local IFS
585     declare -a arr_targets xz tgz sorted_targets
586
587     # populate tar.xz, tgz arrays
588     mapfile -t xz < <(find /tmp -name \*.tar.xz 2>/dev/null)
589     mapfile -t tgz < <(find /tmp -name \*.tgz 2>/dev/null)
590
591     for i in "${xz[@] }"; do
592         temp="$(echo $i | awk -F '.' '{print $1}' | awk -F '.' '{print $1"."$2}')"
593         xz[$index]=$(echo $temp | awk -F '/' '{print $NF}')
594         (( index++ ))
595     done
596
597     index="0"

```

(continues on next page)

(continued from previous page)

```

598
599     for i in "${tgz[@]}; do
600         temp="$(echo $i | awk -F '.tgz' '{print $1}' | awk -F '.' '{print $1"."$2}')"
601         tgz[$index]=$(echo $temp | awk -F '/' '{print $NF}')
602         (( index++ ))
603     done
604
605     arr_targets=( "$(echo "${tgz[@]}")" "$(echo "${xz[@]}")" )
606     # sort
607     IFS=$'\n' sorted_targets=$(sort <<<"${arr_targets[*]}")
608     unset IFS
609
610     echo "${sorted_targets[@]}"
611     #
612     # <--- end function _clean_subcommands --->
613 }
614
615
616 function depcheck() {
617     ##
618     ## validate cis report dependencies ##
619     ##
620     local log_dir="$1"
621     local log_file="$2"
622     local console_log="$3"
623     local msg
624
625     ## test default shell ##
626     if [ ! -n "$BASH" ]; then
627         # shell other than bash
628         msg="Default shell appears to be something other than bash. Please rerun with_
↳ bash. Aborting (code $E_BADSHELL)"
629         printf -- "\n%s\n" "$msg"
630     fi
631
632     ## logging prerequisites ##
633     if [[ ! -d "$log_dir" ]]; then
634
635         if ! mkdir -p "$log_dir"; then
636             printf -- "\n%s: failed to make log directory: %s. Exit\n" % $pkg $log_dir
637         fi
638
639     elif [ ! -f $log_file ]; then
640
641         if ! touch $log_file 2>/dev/null; then
642             printf -- "\n%s: failed to seed log file: %s. Exit\n" % $pkg $log_file
643         fi
644
645     elif [ ! -f $console_log ]; then
646
647         if ! touch $console_log 2>/dev/null; then
648             printf -- "\n%s: failed to seed console log file: %s. Exit\n" % $pkg
↳ $console_log
649         fi
650
651     fi
652

```

(continues on next page)

(continued from previous page)

```

653     ## working directory for user
654     if [[ ! -d "$workdir" ]]; then
655
656         if ! mkdir -p "$workdir"; then
657             std_error_exit "$pkg: failed to make log directory: $workdir. Exit" $E_
↪DEPENDENCY
658         fi
659
660     fi
661
662     ## check for required cli tools ##
663     binary_depcheck awk bc curl
664
665     # success
666     std_logger "$pkg: dependency check satisfied." "INFO" $log_file
667
668     # set path
669     PATH=$PATH:/usr/local/bin
670     return 0
671     #
672     # <-- end function depcheck -->
673 }
674
675
676 function display_program_version() {
677     ##
678     ## output script version info, license
679     ##
680     local _version=__version__
681     local _hic=$(echo -e ${a_brightblue})
682     local _year=$(date +%G)
683     local _bashver="$("
684         bash --version | head -n1 | awk -F 'version' '{print $2}' \
685             | awk '{print $1}' | awk -F '(' '{print $1}'
686     )"
687     #
688     cat <<EOM
689     _____
690
691
692
693
694
695     ${_hic}$pkg${reset} version: ${title}${_version}${reset} | GNU Bash $E_
↪bashver
696
697
698
699
700     _____
701
702     Copyright 2017-${_year}, Blake Huber. This program distributed under
703     GPL v3. This copyright notice must remain with derivative works.
704     _____
705
706 EOM
707 }

```

(continues on next page)

(continued from previous page)

```

708
709
710 function download_progress() {
711     ##
712     ## display progress for downloading large files, wget only
713     ##
714     local url="$1"
715
716     wget --progress=dot $url 2>&1 | grep --line-buffered "%" | \
717     sed -u -e "s,\.,,g" | awk '{printf("\b\b\b\b%4s", $2)}'
718     echo -ne "\b\b\b\b\b"
719     echo " Download Complete"
720     return 0
721 }
722
723
724 function download_binary() {
725     ##
726     ## download python components ##
727     ##
728     local version="$1"
729     local major="$2"
730     local minor="$(echo $version | awk -F '.' '{print $NF}')"
731     local bin_file="Python-$version.tar.xz"
732     local rc_file="Python-${version}rc[1-4].tar.xz"
733     local beta_file="Python-$version[a-b][1-4].tar.xz"
734     local bin_dcr                                     # N-1 minor release
735     local bin_alt="Python-$version.tgz"
736     local bin                                           # loop binfile
737     local url
738     local cached_download
739     declare -a arr_versions
740
741     function download_rc() {
742         ## Download release candidate binary set
743         local bin
744
745         cd $TMPDIR || exit $E_OSError
746
747         for rtype in rc b a; do
748             for i in 4 3 2 1; do
749                 bin="Python-${version}${rtype}${i}.tar.xz"
750
751                 # check if pre-existing; remove
752                 if [ -f "$TMPDIR/$bin" ]; then
753                     rm -f "$TMPDIR/$bin"
754                     std_logger "Found previously downloaded binary ($TMPDIR/$bin).
↳ Removed" "INFO" $LOG_FILE
755                     fi
756
757                     # download binary set
758                     url="https://www.python.org/ftp/python/$version/$bin"
759                     wget "$url" >/dev/null 2>&1
760
761                     # check if downloaded, break
762                     if [ -f "$TMPDIR/$bin" ]; then
763                         echo "$bin"

```

(continues on next page)

(continued from previous page)

```

764         break
765     fi
766 done
767 # check if downloaded, break
768 if [ -f "$TMPDIR/$bin" ]; then
769     break
770 fi
771 done
772 return 0
773 }
774
775 nminus1="$major.$(( $minor - 1 ))"
776 bin_dcr="Python-${nminus1}.tar.xz"
777
778 # for Python < version 2.6, only tgz archive available; download alternate bin_
↪format
779 if (( $(echo "$major < 2.6" | bc -l) )); then bin_file=$bin_alt; fi
780
781 arr_versions=( "$bin_file" ) # array of download candidates
782
783 cd $TMPDIR || exit $E_OSERROR
784
785 for bin in "${arr_versions[@]}; do
786     # check for pre-existing file
787     if [ -f "$TMPDIR/$bin" ]; then
788         cached_download=true
789         std_message "Found previously downloaded binary ($bin). Using..." "OK"
790 ↪$LOG_FILE
791         break
792         #rm "$TMPDIR/$bin"
793     fi
794
795     std_message "Downloading $bin ..." "INFO" $LOG_FILE
796
797     if [ "$(type wget 2>/dev/null)" ]; then
798
799         url="https://www.python.org/ftp/python/$version/$bin"
800         wget "$url" >/dev/null 2>&1 &
801         progress_dots --fast "false"
802
803     elif [ "$(type curl 2>/dev/null)" ]; then
804
805         curl -O "https://www.python.org/ftp/python/$version/$bin" >/dev/null 2>&1_
806 ↪&
807         progress_dots --fast "false"
808
809     else
810         std_error_exit "Either wget or curl binary not found. Cannot retrieve_
↪python source files" $E_DEPENDENCY
811     fi
812
813     # check if downloaded, break
814     if [ -f "$TMPDIR/$bin" ]; then
815         break
816     else
817         std_warn "$bin not downloaded, version $version in alpha, beta, or_
↪release candidate stage." $LOG_FILE

```

(continues on next page)



(continued from previous page)

```

817         std_message "Downloading latest available alpha, beta, or release_
↪candidate binary" "INFO"
818         bin=''
819
820         # attempt to download release candidate
821         bin="$(download_rc)"
822
823         if [ ! -f "$TMPDIR/$bin" ]; then
824             std_warn "Release candidate version not found. Exit" $LOG_FILE
825             exit $E_OSERROR
826         fi
827     fi
828
829     done
830
831     # user status msg
832     if file_check "$TMPDIR/$bin"; then
833
834         if [[ $(echo $bin | grep "Python-$version.tar.xz") ]] && [[ "$cached_download
↪" ]]; then
835             std_message "${yellow}Python${btext} ${bd}$version${btext} cached source_
↪successfully located in ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_FILE
836
837             elif [[ $(echo $bin | grep "Python-$version.tar.xz") ]]; then
838                 std_message "${yellow}Python${btext} ${bd}$version${btext} source_
↪downloaded successfully to ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_FILE
839
840                 elif [[ $(echo $bin | grep "Python-${version}rc[1-4].tar.xz") ]]; then
841                     std_message "${yellow}Python${btext} ${bd}$version${btext} release_
↪candidate source downloaded successfully to ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_
↪FILE
842
843                     elif [[ $(echo $bin | grep "Python-$version[a-b][1-4].tar.xz") ]]; then
844                         std_message "${yellow}Python${btext} ${bd}$version${btext} alpha/beta_
↪source downloaded successfully to ${fs}$TMPDIR/$bin${reset}" "INFO" $LOG_FILE
845                     fi
846
847                 else
848                     # download failed
849                     std_warn "Problem downloading ${yellow}Python${btext} ${bd}$version${btext}_
↪source binary to ${fs}$TMPDIR/$bin${btext}" $LOG_FILE
850                     std_warn "${fs}$TMPDIR/$bin${btext} Not saved to local filesystem" $LOG_FILE
851                 fi
852             #
853             # <<-- end function download_binary -->>
854         }
855
856
857     function file_check(){
858         ##
859         ##  Validates existence of file object on local fs
860         ##
861         ##  Returns:
862         ##      - Success / Failure, TYPE: bool
863         ##
864
865         local file_path="$1"

```

(continues on next page)

(continued from previous page)

```

866
867     if [ -f "$file_path" ]; then
868
869         return 0
870
871     else
872
873         return 1          # file not found
874
875     fi
876     #
877     # <-- end function file_check -->
878 }
879
880
881 function pip_backup() {
882     ##
883     ## create backup of local python package repo ##
884     ##
885     local pip                # pip package mgr binary path
886     local pip3               # pip3 package mgr binary path
887     local pip_r
888     local pip3_backup
889
890     pip3=$(which pip3 2>/dev/null)
891     pip3_backup="pip3-requirements-$NOW.txt"
892
893     pip=$(which pip 2>/dev/null)
894     pip_r="pip-requirements-$NOW.txt"
895     #
896     msg="Creating backups of local python package repositories in $workdir"
897     std_message "$msg" "INFO"
898     std_logger "Backup of local python package repositories - START" "INFO" $LOG_FILE
899
900     # pip backup
901     if [ "$pip" ]; then
902         $pip --no-cache-dir freeze > $workdir/$pip_r
903         std_logger "Created pip backup: $workdir/$pip_r" "INFO" $LOG_FILE
904     fi
905
906     # pip3 backup
907     if [ "$pip3" ]; then
908         $pip3 --no-cache-dir freeze > $workdir/$pip3_backup
909         std_logger "Created pip3 backup: $workdir/$pip3_backup" "INFO" $LOG_FILE
910     fi
911
912     if [ -f "$workdir/$pip3_backup" ] && [ -f "$workdir/$pip_r" ]; then
913
914         if [[ $(diff $workdir/$pip_r $workdir/$pip3_backup) ]]; then
915             # pip backup and pip3 files are different
916             msg="Created backups of local python package repositories in $workdir:\n
917             \tPip: \t$pip_r \n\t\tPip3: \t$pip3_backup\n"
918             std_message "$msg" "INFO" $LOG_FILE
919
920         else
921             # pip backup & pip3 backup file same
922             rm -f $workdir/$pip3_backup

```

(continues on next page)

(continued from previous page)

```

923         std_logger "Deleting pip3 backup file ($pip3_backup) - Identical to $pip_r
↪ " "INFO" $LOG_FILE
924         std_logger "Retained single backup file $workdir/$pip_r" "INFO" $LOG_FILE
925
926         msg="Created backup of python package repositories in
↪ $workdir:\n\n\t\tpip: \t$pip_r"
927         std_message "$msg" "INFO"
928     fi
929
930     elif [ $pip3_backup ]; then
931         # pip3 backup only
932         msg="Created backup of python package repositories in $workdir:\n\n\t\tpip: \t
↪ $pip3_backup"
933         std_message "$msg" "INFO"
934         std_logger "Created backup file in $workdir: $pip3_backup" "INFO" $LOG_FILE
935
936     else
937         # pip backup only
938         msg="Created backup of python package repositories in $workdir:\n\n\t\tpip: \t
↪ $pip_r"
939         std_message "$msg" "INFO"
940         std_logger "Created backup file in $workdir: $pip_r" "INFO" $LOG_FILE
941     fi
942
943     std_logger "Backup of local python package repositories - END" "INFO" $LOG_FILE
944     return 0
945
946     #
947     # <-- end function pip_backup -->
948 }
949
950
951 function latest_version_available() {
952     ##
953     ## retrieve latest minor version of python to install ##
954     ##
955     local major_version="$1"           # 3.7, etc
956     local minor=20                     # starting count to decrement 3.7.minor
957     local act="$(echo -e ${cyan})"      # color code
958     local rst="$(echo -e ${resetansi})" # reset color code
959     #
960     # Guarding Clause:
961     # Return if VERSION during installs already set; else continue
962     if [ "$VERSION" ] && [ "$INSTALL" ]; then return 0; fi
963
964     if [ "$(type wget 2>/dev/null)" ]; then
965         wget -O "$TMPDIR/index.html" 'https://www.python.org/ftp/python' 2>/dev/null
966     else
967         curl -L -o "$TMPDIR/index.html" 'https://www.python.org/ftp/python' 2>/dev/
↪ null
968     fi
969
970     # find highest (most recent) minor version
971     while (( $minor >= 0 )); do
972
973         try="$(grep $major_version.$minor $TMPDIR/index.html)"
974

```

(continues on next page)

(continued from previous page)

```

975     if [ "$try" ]; then
976         # set full version identifier to global var
977         VERSION=$major_version.$minor
978
979         # date of release
980         rdate="$(echo $try | awk '{print $3}')"
981         fdate="$(act)$(date --date=$rdate +%a %b %d %Y)${rst}"
982
983         std_message "Found most recent binaries available for Python Version $
↪{major_version}" "OK" $LOG_FILE
984         std_message "${yellow}Python ${title}$VERSION${rst}, released on $fdate"
↪"INFO" $LOG_FILE
985
986         found="true"
987         break
988     fi
989     minor=$(( $minor - 1 ))
990 done
991
992 if [ ! "$found" ]; then
993     std_warn "No binary version found for Python $major_version. Exit" $LOG_FILE
994     rm $TMPDIR/index.html
995     return 1 # minor version not found; return false
996 fi
997
998 # rm version artifacts
999 rm $TMPDIR/index.html
1000 return 0 # minor version identified; return true
1001 }
1002
1003
1004 function logfile_owner(){
1005     ##
1006     ## ensures log file owned by normal user ##
1007     ##
1008     local log_file="$1"
1009     local console_log="$2"
1010     local yl=${yellow}
1011     local rst=${reset}
1012     #
1013     # update owner of log file if NOT located in /var/log
1014     if [ "$SUDO_USER" ] && [ ! "$(echo $log_file | grep \var)" ] && \
1015         [ "$(ls -l $log_file | awk '{print $3}')" != "root" ]; then
1016
1017         chown $SUDO_USER:$SUDO_USER $log_file
1018         std_logger "Successfully updated file owner to root for ${yl}$log_file$
↪{rst}" "INFO" $log_file
1019
1020     fi
1021     # update owner of console log file if NOT located in /var/log
1022     if [ "$SUDO_USER" ] && [ ! "$(echo $console_log | grep \var)" ] && \
1023         [ "$(ls -l $console_log | awk '{print $3}')" != "root" ]; then
1024
1025         chown $SUDO_USER:$SUDO_USER $console_log
1026         std_logger "Successfully updated file owner to root for ${yl}$console_log$
↪{rst}" "INFO" $log_file
1027

```

(continues on next page)

(continued from previous page)

```

1028     fi
1029     # update permissions on log files if not correct
1030     if [ "$SUDO_USER" ] || [ "$EUID" = "0" ]; then
1031         chmod 0666 $log_file $console_log
1032
1033     fi
1034 }
1035
1036
1037 function mlocate_installation() {
1038     ##
1039     ## installs os-specific installation of mlocate; runs db update
1040     ##
1041     ## This should ideally work via the following:
1042     ##     1. install mlocate
1043     ##     2. after updatedb -v run, find all python3.X locations
1044     ##     3. Using this list, delete each path in the list
1045     ##
1046     ## Installing mlocate is OS-specific
1047     ##
1048     local clearscn=$(which clear 2>/dev/null)
1049     local epel_url='https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.
↪noarch.rpm'
1050     #
1051     os_type      # linux distro environment; sets globals: family, release
1052     #
1053     std_message "$family Linux: Installing mlocate binaries in preparation for_
↪system Python3 removal" "INFO" $LOG_FILE
1054
1055     if [ "$family" = "AmazonLinux" ]; then
1056
1057         $clearscn      # clear screen
1058         std_message "$family Linux: Update OS pkgs" "INFO" $LOG_FILE
1059         yum -y update >> $LOG_CONSOLE &
1060         progress_dots --fast "false"
1061         std_message "$family Linux: Install mlocate and dependencies" "INFO" $LOG_
↪FILE
1062         yum -y install mlocate
1063         std_message "$family Linux: First db filesystem discovery run..." "INFO"
↪$LOG_FILE
1064         updatedb -v >/dev/null &
1065         progress_dots --fast "false" --text "Searching for Python filesystem locations
↪"
1066         return 0
1067
1068     elif [ "$family" = "Redhat" ] || [ "$family" = "CentOS" ]; then
1069
1070         $clearscn      # clear screen
1071         std_message "$family Linux: Update OS pkgs" "INFO" $LOG_FILE
1072         yum -y update >> $LOG_CONSOLE &
1073         progress_dots --fast "false"
1074         # install epel
1075         std_message "$family Linux: Installing epel package repository" "INFO" $LOG_
↪FILE
1076         curl -o epel.rpm $epel_url
1077         yum -y install epel.rpm
1078         std_message "$family Linux: Install mlocate and dependencies" "INFO" $LOG_
↪FILE

```

(continues on next page)

(continued from previous page)

```

1079     yum -y update
1080     yum -y install mlocate
1081     std_message "$family Linux:  First db filesystem discovery run..." "INFO"
↪$LOG_FILE
1082     updatedb -v >/dev/null &
1083     progress_dots --fast "false" --text "Searching for Python filesystem locations
↪"
1084     return 0
1085
1086     elif [ "$family" = "Ubuntu" ] || [ "$family" = "Mint" ]; then
1087
1088         $clearscn      # clear screen
1089         std_message "$family Linux:  Update OS pkgs" "INFO" $LOG_FILE
1090         apt update -y >> $LOG_CONSOLE &
1091         progress_dots --fast "false"
1092         std_message "$family Linux:  Install mlocate and dependencies" "INFO" $LOG_
↪FILE
1093         apt install -y mlocate
1094         std_message "$family Linux:  First db filesystem discovery run..." "INFO"
↪$LOG_FILE
1095         updatedb -v >/dev/null &
1096         progress_dots --fast "false" --text "Searching for Python filesystem locations
↪"
1097         return 0
1098
1099     fi
1100
1101     return 1      # mlocate installation fail or not possible
1102     #
1103     # <-- end function mlocate_installation -->
1104 }
1105
1106
1107 function os_type() {
1108     ##
1109     ## determine linux os distribution ##
1110     ##
1111     function determine_osrelease() {
1112         # interrogate /etc/os-release directly
1113         # to determine release
1114         id=$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')
1115         clean_id=$(echo $id | cut -c 2-20 | rev | cut -c 2-20 | rev)
1116         printf -- '%s\n' "$clean_id"
1117     }
1118
1119     OS_INFO=$(source $pkg_lib/os_distro.sh 2>/dev/null)
1120
1121
1122     if [ "$OS_INFO" ]; then
1123
1124         family=$(echo $OS_INFO | awk '{print $1}')
1125         release=$(echo $OS_INFO | awk '{print $2}')
1126         codename=$(echo $OS_INFO | awk '{print $3}')
1127
1128         case "$family" in
1129             'Ubuntu')
1130                 act=${a_magenta}

```

(continues on next page)

(continued from previous page)

```

1131         ;;
1132         'Redhat' | 'CentOS')
1133             act=${red}
1134             ;;
1135         'AmazonLinux')
1136             act=${a_orange}
1137             ;;
1138     esac
1139
1140     if [ ! "$release" ] || [ -z "$release" ]; then
1141         release=$(determine_osrelease)
1142     fi
1143
1144     std_message "Operating System Info detected:
1145     \n\t\tFamily:\t\t${act}$family${btext}
1146     \tRelease:\t\t${bd}$release${btext}
1147     \tCodename:\t\t$codename" "INFO"
1148     std_logger "OS Info Detected: Family: $family, Release: $release, Codename:
1149     ↪$codename" "INFO" $LOG_FILE
1150
1151     else
1152         std_warn "Failed to determine Linux Distribution Type. Fall back detection"
1153         ↪$LOG_FILE
1154         FALLBACK_DETECT="true"
1155     fi
1156     #
1157     # <-- end function os_type -->
1158 }
1159
1160
1161 function ospackage_installed() {
1162     ##
1163     ##  os specific test to determine if an operating system
1164     ##  package is installed
1165     ##
1166     local packagemgr="$1"      # packagemgr; redhat (.rpm) or debian (.deb)
1167     local ospkg="$2"          # os package tested for installation status
1168
1169     case $packagemgr in
1170         'rpm' | '.rpm' | 'redhat')
1171             if [ ! "$(rpm -q $ospkg | grep 'not installed')" ]; then
1172                 return 0      # is installed
1173             fi
1174             ;;
1175         'deb' | '.deb' | 'debian')
1176             if [ "$(dpkg -s $ospkg 2>/dev/null | grep Status\:)" ]; then
1177                 return 0      # is installed
1178             fi
1179             ;;
1180     esac
1181
1182     return 1      # not installed
1183     #
1184     # < --- end function ospackage_installed --->
1185 }

```

(continues on next page)

(continued from previous page)

```

1186
1187
1188 function ospackage_available() {
1189     ##
1190     ##  os specific test to determine if an operating system
1191     ##  package is present in local package repositories
1192     ##
1193     local packagemgr="$1"          # packagemgr; redhat (.rpm) or debian (.deb)
1194     local ospkg="$2"              # os package tested for installation status
1195
1196     case $packagemgr in
1197         'rpm' | '.rpm' | 'redhat')
1198             if [ "$(yum list $ospkg 2>/dev/null | grep -i 'Available')" ]; then
1199                 return 0          # is available, but not installed
1200             fi
1201             ;;
1202         'deb' | '.deb' | 'debian')
1203             if [ "$(apt-cache search $ospkg)" ]; then
1204                 return 0          # is available, but not installed
1205             fi
1206             ;;
1207     esac
1208
1209     # not installed
1210     return 1
1211     #
1212     # < --- end function ospackage_available --->
1213 }
1214
1215
1216 function ospackage_status() {
1217     ##
1218     ##  displays os package summary info
1219     ##
1220     local packagemgr="$1"
1221     local ospkg="$2"
1222     local logfile="$3"
1223     local prefix
1224
1225     case $packagemgr in
1226         'deb')
1227             if ospackage_installed $packagemgr $ospkg; then
1228                 # available and installed
1229                 summary=$(apt-cache show $ospkg | grep Description | head -n 1 | awk -
1230 ↪F ':' '{print $2}')
1231                 prefix=" INSTALLED"
1232
1233             elif ospackage_available $packagemgr $ospkg; then
1234                 # available but not installed
1235                 if [ "$(apt-cache show $ospkg 2>/dev/null)" ]; then
1236                     summary=$(apt-cache show $ospkg | grep Description | head -n 1 |
1237 ↪awk -F ':' '{print $2}')
1238                 else
1239                     summary=$(apt-cache search $ospkg 2>/dev/null | awk '{ print
1240 ↪substr(" "$0, index($0,$3)) }')
1241                 fi

```

(continues on next page)



(continued from previous page)

```

1240         prefix=" AVAILABLE"
1241
1242     elif ! ospackage_available $packagemgr $ospkg; then
1243         # not available
1244         prefix=" NOT-FOUND"
1245     fi
1246 ;;
1247
1248 'rpm')
1249     if ospackage_installed $packagemgr $ospkg; then
1250         # available and installed
1251         summary=$(rpm -qi $ospkg | grep 'Summary' | awk -F ':' '{print $2}')
1252         prefix=" INSTALLED"
1253
1254     elif ospackage_available $packagemgr $ospkg; then
1255         # available but not installed
1256         summary=$(yum info $ospkg 2>/dev/null | grep 'Summary' | head -n1 |
↪awk -F ':' '{print $2}')
1257         prefix=" AVAILABLE"
1258
1259     elif ! ospackage_available $packagemgr $ospkg; then
1260         # not available
1261         prefix=" NOT-FOUND"
1262     fi
1263 ;;
1264
1265 'alternate_rpm')
1266     if ospackage_available 'rpm' $ospkg; then
1267         # available but not installed
1268         summary=$(yum info $ospkg 2>/dev/null | grep 'Summary' | head -n1 |
↪awk -F ':' '{print $2}')
1269     else
1270         summary='no available package description in current os'
1271     fi
1272     prefix="ESTIMATE"
1273 ;;
1274
1275 'alternate_deb')
1276     if ospackage_available 'deb' $ospkg; then
1277         # available but not installed
1278         if [ "$(apt-cache show $ospkg 2>/dev/null)" ]; then
1279             summary=$(apt-cache show $ospkg | grep Description | head -n 1 |
↪awk -F ':' '{print $2}')
1280         else
1281             summary=$(apt-cache search $ospkg 2>/dev/null | awk '{ print
↪substr(" "$0, index($0,$3)) }')
1282         fi
1283     else
1284         summary='no available package description in current os'
1285     fi
1286     prefix="ESTIMATE"
1287 ;;
1288
1289 'alternate')
1290     printf -- ' %s\n' "$ospkg"
1291 ;;
1292 esac

```

(continues on next page)

(continued from previous page)

```

1293
1294     if [ -z "$summary" ] || [ "$summary" = "" ]; then
1295         summary=' description not found'
1296     fi
1297
1298     case $summary in
1299         ' description not found')
1300             std_logger "$ospkg: Description not found for os-package $ospkg" "$prefix
↪" $logfile
1301             ;;
1302
1303         *)
1304             case ${#ospkg} in
1305                 [0-5])
1306                     std_message "$ospkg\t\t\t| $summary" $prefix $logfile
↪ 'noblanklines'
1307                     ;;
1308                 [6-9] | 10 | 11 | 12 | 13)
1309                     std_message "$ospkg\t\t| $summary" $prefix $logfile 'noblanklines
↪ '
1310                     ;;
1311                 *)
1312                     std_message "$ospkg\t| $summary" $prefix $logfile 'noblanklines'
1313                     ;;
1314             esac
1315         ;;
1316     esac
1317     #
1318     # <--- end function ospackage_status --- >
1319 }
1320
1321
1322 function install_deb_packages() {
1323     ##
1324     ##  installs packages supplied as an array
1325     ##
1326     ##  Usage:
1327     ##
1328     ##      install_deb_packages  -o "install"  --packages pkgs[@]
1329     ##
1330     ##  Where:
1331     ##      --operation:  update | upgrade | install
1332     ##      --packages:  associative array
1333     ##
1334     ##  Note:
1335     ##      Mint Linux machines not upgraded via apt for reliability
1336     ##
1337     declare -a pkg_list
1338     local operation
1339     local logfile=$LOG_FILE
1340     local consolelog=$LOG_CONSOLE
1341     local show_packages=$SHOW_OSPACKAGES
1342
1343     while [ $# -gt 0 ]; do
1344         case $1 in
1345             -a | --alt_os)
1346                 alternate_os="$2"

```

(continues on next page)

(continued from previous page)

```

1347         shift 2
1348         ;;
1349
1350     -l | --log)
1351         logfile="$2"; shift 2
1352         ;;
1353
1354     -o | --operation)
1355         case $2 in
1356             update | upgrade | install)
1357                 operation="$2"
1358                 shift 2
1359                 ;;
1360             *)
1361                 std_error_exit "Operation must be either update, upgrade, or_
↪install. Exit"
1362                 ;;
1363         esac
1364         ;;
1365
1366     -p | --packages)
1367         pkg_list=("${2}"); shift 2
1368         ;;
1369     esac
1370 done
1371
1372 ## display ospackage status instead of updating ##
1373
1374 if [ $show_packages ] && [ $alternate_os ]; then
1375     case $alternate_os in
1376         redhat* | centos* | amazonlinux*)
1377             ospackage_status "alternate_rpm" "$ospkg" "$logfile"
1378             ;;
1379         *)
1380             ospackage_status "alternate_deb" "$ospkg" "$logfile"
1381             ;;
1382     esac
1383
1384 elif [ $show_packages ]; then
1385     if [ -z $operation ] || [ "$operation" = "install" ]; then
1386         for ospkg in "${pkg_list[@]}; do
1387
1388             ospackage_status "deb" "$ospkg" "$logfile"
1389
1390         done
1391     fi
1392     return 0
1393 fi
1394
1395 ## perform actual update of system ospackages ##
1396
1397 if [ "$operation" = "update" ]; then
1398     std_message "Refreshing os package repository caches" "INFO" $logfile
1399     apt update -y 2>/dev/null | tee $tty >> $consolelog
1400     std_message "Package repository cache refresh complete" "OK" $logfile
1401     return 0
1402

```

(continues on next page)

(continued from previous page)

```

1403     elif [ "$operation" = "upgrade" ] && [ "$family" = "Ubuntu" ]; then
1404         std_message "Upgrading Operating System Packages" "INFO" $logfile
1405         apt upgrade -y 2>/dev/null | tee $tty >> $consolelog
1406         std_message "Operating System update complete" "OK" $logfile
1407         return 0
1408
1409     elif [ -z $operation ] || [ "$operation" = "install" ]; then
1410         for ospkg in "${pkg_list[@]}; do
1411
1412             # validate install
1413             if ospackage_installed 'deb' $ospkg; then
1414                 std_message "Package $ospkg already installed" "OK" $logfile
1415             else
1416                 std_message "Installing Debian package $ospkg" "INFO" $logfile
1417                 apt install -y $ospkg 2>/dev/null | tee $tty >> $consolelog
1418
1419                 # verify install
1420                 if ospackage_installed 'deb' $ospkg; then
1421                     std_message "Package $ospkg installed successfully" "OK" $logfile
1422                 else
1423                     std_warn "Package $ospkg failed to install. Possibly missing from
1424     ↪os package repository" $logfile
1425                 fi
1426             fi
1427         done
1428         return 0
1429     fi
1430
1431     return 1
1432     #
1433     # <--- end function install_deb_packages --->
1434 }
1435
1436
1437 function install_yum_packages() {
1438     ##
1439     ##  installs packages supplied as an array
1440     ##
1441     ##  Usage:
1442     ##
1443     ##      install_yum_packages -o "install" --packages pkgs[@]
1444     ##
1445     ##  Where:
1446     ##      --operation:  update | install
1447     ##      --packages:  associative array
1448     ##
1449     declare -a pkg_list
1450     local operation
1451     local pkgmanager
1452     local logfile=$LOG_FILE
1453     local consolelog=$LOG_CONSOLE
1454     local show_packages=$SHOW_OSPACKAGES
1455
1456     if [ "$(which dnf 2>/dev/null)" ]; then
1457         pkgmanager=$(which dnf 2>/dev/null)
1458

```

(continues on next page)

(continued from previous page)

```

1459     elif [ "$(which yum 2>/dev/null)" ]; then
1460         pkgmanager=$(which yum 2>/dev/null)
1461
1462     else
1463         std_warn "No package manager found. Abort os package install" $logfile
1464     fi
1465
1466     while [ $# -gt 0 ]; do
1467         case $1 in
1468             -a | --alt_os)
1469                 alternate_os="$2"
1470                 shift 2
1471                 ;;
1472
1473             -l | --log)
1474                 logfile="$2"
1475                 shift 2
1476                 ;;
1477
1478             -o | --operation)
1479                 case $2 in
1480                     update | install | groupinstall)
1481                         operation="$2"
1482                         shift 2
1483                         ;;
1484                     *)
1485                         std_error_exit "Operation must be either update, install, or_
↪groupinstall. Exit"
1486                         ;;
1487                     esac
1488                     ;;
1489
1490             -p | --packages)
1491                 pkg_list=("${!2}")
1492                 shift 2
1493                 ;;
1494             esac
1495         done
1496
1497         ## display ospackage status instead of updating ##
1498
1499         if [ $show_packages ] && [ $alternate_os ]; then
1500             ospackage_status "alternate" "$ospkg" "$logfile"
1501
1502         elif [ $show_packages ]; then
1503             if [ -z $operation ] || [ "$operation" = "install" ] || [ "$operation" =
↪"groupinstall" ]; then
1504
1505                 for ospkg in "${pkg_list[@]}; do
1506                     ospackage_status "rpm" "$ospkg" "$logfile"
1507                 done
1508
1509             fi
1510             return 0
1511         fi
1512
1513         ## perform actual update of system ospackages ##

```

(continues on next page)

(continued from previous page)

```

1514
1515     if [ "$operation" = "update" ]; then
1516         std_message "Refreshing os package repository caches" "INFO" $logfile
1517         $pkgmanager -y update 2>/dev/null | tee $tty >> $consolelog
1518         std_message "Package repository cache refresh complete" "OK" $logfile
1519         return 0
1520
1521     elif [ "$operation" = "groupinstall" ]; then
1522         for osgroup in "${pkg_list[@]}; do
1523             std_message "Installing operating system package group" "INFO" $logfile
1524             $pkgmanager -y groupinstall $osgroup 2>/dev/null | tee $tty >>
↪$consolelog
1525         done
1526         std_message "Operating System Upgrade Complete" "OK" $logfile
1527         return 0
1528
1529     elif [ -z $operation ] || [ "$operation" = "install" ]; then
1530         for ospkg in "${pkg_list[@]}; do
1531
1532             # validate install
1533             if ospackage_installed 'rpm' $ospkg; then
1534                 std_message "Package $ospkg already installed - nothing to do" "INFO"
↪$logfile
1535             else
1536                 std_message "Installing package $ospkg" "INFO" $logfile
1537                 $pkgmanager -y install $ospkg 2>/dev/null | tee $tty >> $consolelog
1538
1539                 # verify install
1540                 if ospackage_installed 'rpm' $ospkg; then
1541                     std_message "Package $ospkg installed successfully" "OK" $logfile
1542                 else
1543                     std_warn "Package $ospkg failed to install. Possibly missing from
↪os package repository" $logfile
1544                 fi
1545             fi
1546
1547         done
1548         return 0
1549     fi
1550
1551     return 1
1552     #
1553     # <--- end function install_yum_packages --->
1554 }
1555
1556
1557 function print_header() {
1558     ##
1559     ## print formatted report header ##
1560     ##
1561     local width="$1"
1562     local color="$2"
1563     #
1564     echo -e "${color}"
1565     printf '%*s' "$width" ' ' | tr ' ' - | indent04
1566     echo -e "${btext}"
1567     printf -- ' %s %s\t%s\t\t%s\t\t\t%s' "${BOLD}Status" '|' "Package" '|'
↪"Description${UNBOLD}"

```

(continues on next page)

(continued from previous page)

```

1568     echo -e "${color}"
1569     printf '%*s' "$width" ' ' | tr ' ' - | indent 04
1570     echo -e "${btext}"
1571     #
1572     # <--- end function print_header -->
1573 }
1574
1575
1576 function os_packages() {
1577     ##
1578     ## install os prerequisites depending on os type ##
1579     ##
1580     local version
1581     declare -a packages
1582
1583     version="$1"
1584
1585     if [ $SHOW_OSPACKAGES ]; then
1586         if [ ! $family ]; then std_error_exit "Unknown operating" $E_DEPENDENCY; fi
1587         # only show which os packages to be installed
1588         print_header "100" "${a_wgray}"
1589     else
1590         std_message "Installing OS-specific packages for Python-$version..." "INFO"
1591         ↪ $LOG_FILE
1592         fi
1593
1594         # AMAZON Linux, Version 1
1595
1596         if [ "$family" = "AmazonLinux" ] && [ "$release" = "1" ]; then
1597             install_yum_packages --operation update
1598
1599             packages=( "Development Libraries" "Development tools" )
1600             install_yum_packages --operation groupinstall --packages packages[@]
1601
1602             packages=( "gcc" "make" "wget" "which" )
1603             install_yum_packages --operation install --packages packages[@]
1604
1605             packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1606             install_yum_packages --operation install --packages packages[@]
1607
1608             packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static' )
1609             install_yum_packages --operation install --packages packages[@]
1610
1611             packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
1612             install_yum_packages --operation install --packages packages[@]
1613
1614             packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1615             install_yum_packages --operation install --packages packages[@]
1616
1617             packages=( 'zlib' 'zlib-devel' 'zlib-static' 'libffi-devel' )
1618             install_yum_packages --operation install --packages packages[@]
1619             return 0
1620
1621         # AMAZON Linux, Version 2
1622
1623         elif [ "$family" = "AmazonLinux" ] && [ "$release" = "2" ]; then
1624             install_yum_packages --operation update

```

(continues on next page)

(continued from previous page)

```

1624     packages=( "Development tools" )
1625     install_yum_packages --operation groupinstall --packages packages[@]
1626
1627     packages=( "gcc" "make" "wget" "which" )
1628     install_yum_packages --operation install --packages packages[@]
1629
1630     packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1631     install_yum_packages --operation install --packages packages[@]
1632
1633     packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1634     install_yum_packages --operation install --packages packages[@]
1635
1636     packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static' )
1637     install_yum_packages --operation install --packages packages[@]
1638
1639     packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
1640     install_yum_packages --operation install --packages packages[@]
1641
1642     packages=( 'python-magic' 'libuuid-devel' 'tk-devel' )
1643     install_yum_packages --operation install --packages packages[@]
1644
1645     packages=( 'xz' 'xz-devel' 'xz-libs' )
1646     install_yum_packages --operation install --packages packages[@]
1647
1648     packages=( 'zlib' 'zlib-devel' 'libffi-devel' )
1649     install_yum_packages --operation install --packages packages[@]
1650
1651     packages=( 'lzma-sdk' 'lzma-sdk-devel' 'xz-devel' 'gdbm-devel' )
1652     install_yum_packages --operation install --packages packages[@]
1653     return 0
1654
1655     elif [ $FALLBACK_DETECT ] && [ "$(grep -i amazon /etc/os-release | head -n 1)
↪ " ]; then
1656         install_yum_packages --operation update
1657
1658         packages=( "Development Libraries" "Development tools" )
1659         install_yum_packages --operation groupinstall --packages packages[@]
1660
1661         packages=( "gcc" "make" "wget" "which" 'uuid-devel' )
1662         install_yum_packages --operation install --packages packages[@]
1663
1664         packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1665         install_yum_packages --operation install --packages packages[@]
1666
1667         packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1668         install_yum_packages --operation install --packages packages[@]
1669         return 0
1670
1671     # REDHAT Enterprise Linux 7
1672
1673     elif [ "$family" = "Redhat" ] && { [ "$release" = "7.3" ] || [ "$release" =
↪ "7.4" ]; }; then
1674         install_yum_packages --operation update
1675
1676         packages=( "Development tools" )
1677         install_yum_packages --operation groupinstall --packages packages[@]
1678

```

(continues on next page)



(continued from previous page)

```

1679
1680 packages=( "gcc" "make" "wget" "which" )
1681 install_yum_packages --operation install --packages packages[@]
1682
1683 packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1684 install_yum_packages --operation install --packages packages[@]
1685
1686 packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static' )
1687 install_yum_packages --operation install --packages packages[@]
1688
1689 packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
1690 install_yum_packages --operation install --packages packages[@]
1691
1692 packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1693 install_yum_packages --operation install --packages packages[@]
1694
1695 packages=( 'xz' 'xz-devel' 'xz-libs' 'libuuid-devel' 'tk-devel' )
1696 install_yum_packages --operation install --packages packages[@]
1697
1698 packages=( 'zlib' 'zlib-devel' 'libffi-devel' )
1699 install_yum_packages --operation install --packages packages[@]
1700 return 0
1701
1702 elif [ "$family" = "Redhat" ] && { [ "$release" = "7.5" ] || [ "$release" =
↪ "7.6" ] ; }; then
1703     install_yum_packages --operation update
1704
1705     packages=( "Development tools" )
1706     install_yum_packages --operation groupinstall --packages packages[@]
1707
1708     packages=( "gcc" "make" "wget" "which" )
1709     install_yum_packages --operation install --packages packages[@]
1710
1711     packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1712     install_yum_packages --operation install --packages packages[@]
1713
1714     packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static' )
1715     install_yum_packages --operation install --packages packages[@]
1716
1717     packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
1718     install_yum_packages --operation install --packages packages[@]
1719
1720     packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1721     install_yum_packages --operation install --packages packages[@]
1722
1723     packages=( 'xz' 'xz-devel' 'xz-libs' 'libuuid-devel' 'tk-devel' )
1724     install_yum_packages --operation install --packages packages[@]
1725
1726     packages=( 'zlib' 'zlib-devel' 'libffi-devel' )
1727     install_yum_packages --operation install --packages packages[@]
1728
1729     packages=( 'lzma-sdk' 'lzma-sdk-devel' 'xz-devel' 'gdbm-devel' )
1730     install_yum_packages --operation install --packages packages[@]
1731     return 0
1732
1733 # CentOS Linux (Redhat binary compatible)
1734

```

(continues on next page)

(continued from previous page)

```

1735     elif [ "$family" = "CentOS" ] && { [ "$release" = "7" ] || [ "$release" = "8" ]
↪}; }; then
1736         install_yum_packages --operation update
1737
1738         packages=( "Development tools" )
1739         install_yum_packages --operation groupinstall --packages packages[@]
1740
1741         packages=( "gcc" "make" "wget" )
1742         install_yum_packages --operation install --packages packages[@]
1743
1744         packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1745         install_yum_packages --operation install --packages packages[@]
1746
1747         packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1748         install_yum_packages --operation install --packages packages[@]
1749
1750         packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' )
1751         install_yum_packages --operation install --packages packages[@]
1752
1753         packages=( 'ncurses-base' 'ncurses-term' )
1754         install_yum_packages --operation install --packages packages[@]
1755
1756         packages=( 'xz' 'xz-devel' 'xz-libs' )
1757         install_yum_packages --operation install --packages packages[@]
1758
1759         packages=( 'zlib' 'zlib-devel' 'libffi-devel' )
1760         install_yum_packages --operation install --packages packages[@]
1761
1762         packages=( 'xz-devel' 'gdbm-devel' 'libuuid-devel' 'tk-devel' )
1763         install_yum_packages --operation install --packages packages[@]
1764
1765     if [ "$release" = "7" ]; then
1766         packages=( 'ncurses-static' 'yum-utils' 'python36-pyOpenSSL' )
1767         install_yum_packages --operation install --packages packages[@]
1768
1769         packages=( 'lzma-sdk' 'lzma-sdk-devel' 'uuid-devel' )
1770         install_yum_packages --operation install --packages packages[@]
1771     else
1772         packages=( 'python3-pyOpenSSL' 'libuuid-devel' )
1773         install_yum_packages --operation install --packages packages[@]
1774     fi
1775     return 0
1776
1777     # Fedora Linux
1778
1779     elif [ "$family" = "Fedora" ] && ( ( "$release" >= "26" ) ); then
1780         install_yum_packages --operation update
1781
1782         packages=( "Development tools" )
1783         install_yum_packages --operation groupinstall --packages packages[@]
1784
1785         packages=( "yum-utils" "gcc" "make" "wget" )
1786         install_yum_packages --operation install --packages packages[@]
1787
1788         packages=( "gdb" "gdbm-devel" "readline-devel" "sqlite-devel" )
1789         install_yum_packages --operation install --packages packages[@]
1790

```

(continues on next page)

(continued from previous page)

```

1791 packages=( "openssl-devel" "bzip2-devel" "expat-devel" )
1792 install_yum_packages --operation install --packages packages[@]
1793
1794 packages=( 'ncurses' 'ncurses-devel' 'ncurses-libs' 'ncurses-static')
1795 install_yum_packages --operation install --packages packages[@]
1796
1797 packages=( 'ncurses-libs' 'ncurses-term' 'uuid-devel' )
1798 install_yum_packages --operation install --packages packages[@]
1799
1800 packages=( 'python3-magic' 'libuuid-devel' 'tk-devel' )
1801 install_yum_packages --operation install --packages packages[@]
1802
1803 packages=( 'xz' 'xz-devel' 'xz-libs')
1804 install_yum_packages --operation install --packages packages[@]
1805
1806 packages=( 'zlib' 'zlib-devel' 'libffi-devel')
1807 install_yum_packages --operation install --packages packages[@]
1808
1809 packages=( 'lzma-sdk' 'lzma-sdk-devel' 'xz-devel' 'gdbm-devel' )
1810 install_yum_packages --operation install --packages packages[@]
1811 return 0
1812
1813 # UBUNTU, Ubuntu variants, Linux Mint
1814
1815 elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep 14)" ]; } || \
1816 { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep 17)" ]; }; then
1817
1818     install_deb_packages --operation update
1819     install_deb_packages --operation upgrade
1820
1821     packages=( "build-essential" "checkinstall" )
1822     install_deb_packages --operation install --packages packages[@]
1823
1824     packages=( 'gcc' 'make' 'wget' 'python3-openssl' )
1825     install_deb_packages --operation install --packages packages[@]
1826
1827     packages=( 'libreadline-dev' 'libreadline6-dev' 'libssl-dev' 'libffi-dev'
1828 ↪)
1829     install_deb_packages --operation install --packages packages[@]
1830
1831     packages=( 'libbz2-dev' 'libgdbm-dev' 'zlib1g-dev' 'uuid-dev' )
1832     install_deb_packages --operation install --packages packages[@]
1833
1834     packages=( "libncursesw5-dev" "tk-dev" "libc6-dev" "libbz2-dev" )
1835     install_deb_packages --operation install --packages packages[@]
1836
1837     packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
1838     install_deb_packages --operation install --packages packages[@]
1839     return 0
1840
1841 ↪ elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep '16')" ]; } || \
1842 { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep '18')" ]; }; then
1843
1844     install_deb_packages --operation update
1845     install_deb_packages --operation upgrade

```

(continues on next page)

(continued from previous page)

```

1846     packages=( 'build-essential' 'checkinstall' 'llvm')
1847     install_deb_packages --operation install --packages packages[@]
1848
1849     packages=( 'gcc' 'make' 'wget' 'python3-openssl' )
1850     install_deb_packages --operation install --packages packages[@]
1851
1852     packages=( 'zlib1g-dev' 'libssl-dev' 'libffi-dev')
1853     install_deb_packages --operation install --packages packages[@]
1854
1855     packages=( 'libbz2-dev' 'libgdbm-dev' 'libsqlite3-dev' 'libc6-dev' )
1856     install_deb_packages --operation install --packages packages[@]
1857
1858     packages=( 'libncursesw5-dev' 'libncurses5-dev' 'lib64ncurses5'
1859 ↪ 'lib64ncurses5-dev' )
1860     install_deb_packages --operation install --packages packages[@]
1861
1862     packages=( 'libreadline6' 'libreadline6-dev' 'libreadline-dev' )
1863     install_deb_packages --operation install --packages packages[@]
1864
1865     packages=( 'ncurses-term' 'ncurses-examples' 'ncurses-base' 'ncurses-doc')
1866     install_deb_packages --operation install --packages packages[@]
1867
1868     packages=( 'readline-doc' 'gdb' 'gdb-doc' 'uuid-dev' )
1869     install_deb_packages --operation install --packages packages[@]
1870
1871     packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
1872     install_deb_packages --operation install --packages packages[@]
1873
1874     packages=( 'tk-dev' 'python3-tk' 'python3-magic' 'xz-utils' )
1875     install_deb_packages --operation install --packages packages[@]
1876
1877     if [ $EXTRA_PACKAGES ]; then
1878         packages=(
1879             binutils-doc gettext cpp-doc gcc-5-locales debian-keyring
1880             ↪multilib g++-multilib g++-5-multilib gcc-5-doc libstdc++6-5-dbg gcc-
1881                 autoconf automake libtool flex bison gcc-doc gcc-5-multilib
1882                 libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg libasan2-dbg
1883                 liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrts5-dbg
1884                 libmpx0-dbg libquadmath0-dbg glibc-doc libstdc++-5-doc make-doc
1885             )
1886             install_deb_packages --operation install --packages packages[@]
1887         fi
1888         return 0
1889
1890     elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep '18')" ]; } || ↪
1891         { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep '19')" ]; } then
1892         install_deb_packages --operation update
1893         install_deb_packages --operation upgrade
1894
1895         packages=( 'build-essential' 'checkinstall' 'llvm')
1896         install_deb_packages --operation install --packages packages[@]
1897
1898         packages=( 'gcc' 'make' 'wget' )
1899         install_deb_packages --operation install --packages packages[@]

```

(continues on next page)

(continued from previous page)

```

1900     packages=( 'zlib1g-dev' 'libreadline-dev' 'libssl-dev' 'libffi-dev' )
1901     install_deb_packages --operation install --packages packages[@]
1902
1903     packages=( 'libbz2-dev' 'libgdbm-dev' 'libsqlite3-dev' 'libc6-dev' )
1904     install_deb_packages --operation install --packages packages[@]
1905
1906     packages=( 'libreadline6' 'libreadline6-dev' 'libreadline-dev' )
1907     install_deb_packages --operation install --packages packages[@]
1908
1909     packages=( 'libreadline7' 'python3-urwid' 'python3-btrees' 'python3-
1910 ↪openssl' )
1911     install_deb_packages --operation install --packages packages[@]
1912
1913     packages=( 'ncurses-term' 'ncurses-examples' 'ncurses-base' 'ncurses-doc' )
1914     install_deb_packages --operation install --packages packages[@]
1915
1916     packages=( 'readline-doc' 'gdb' 'gdb-doc' 'uuid-dev' )
1917     install_deb_packages --operation install --packages packages[@]
1918
1919     packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
1920     install_deb_packages --operation install --packages packages[@]
1921
1922     packages=( 'tk-dev' 'python3-tk' 'python3-magic' 'xz-utils' )
1923     install_deb_packages --operation install --packages packages[@]
1924
1925     packages=( 'liblzma-dev' 'liblz-dev' 'lzma-dev' 'libqdbm-dev' )
1926     install_deb_packages --operation install --packages packages[@]
1927
1928     if [ "$(echo "$release" | grep '18.10')" ]; then
1929         packages=( 'libncurses6' 'libncurses6-dbg' 'libncursesw6'
1930 ↪'libncursesw6-dbg' )
1931         install_deb_packages --operation install --packages packages[@]
1932
1933     elif [ "$(echo "$release" | grep '18.04')" ]; then
1934         packages=( 'libncursesw5-dev' 'libncurses5-dev' )
1935         install_deb_packages --operation install --packages packages[@]
1936     fi
1937     return 0
1938
1939 ↪\
1940     elif { [ "$family" = "Ubuntu" ] && [ "$(echo "$release" | grep '20')" ]; } ||
1941         { [ "$family" = "Mint" ] && [ "$(echo "$release" | grep '20')" ]; }; then
1942         install_deb_packages --operation update
1943         install_deb_packages --operation upgrade
1944
1945         packages=( 'build-essential' 'checkinstall' 'llvm' )
1946         install_deb_packages --operation install --packages packages[@]
1947
1948         packages=( 'gcc' 'make' 'wget' )
1949         install_deb_packages --operation install --packages packages[@]
1950
1951         packages=( 'zlib1g-dev' 'libreadline-dev' 'libssl-dev' 'libffi-dev' )
1952         install_deb_packages --operation install --packages packages[@]
1953
1954         packages=( 'libbz2-dev' 'libgdbm-dev' 'libsqlite3-dev' 'libc6-dev' )

```

(continues on next page)

(continued from previous page)

```

1954     install_deb_packages --operation install --packages packages[@]
1955
1956     packages=( 'libreadline8' 'libreadline-dev' )
1957     install_deb_packages --operation install --packages packages[@]
1958
1959     packages=( 'python3-urwid' 'python3-btrees' 'python3-openssl' )
1960     install_deb_packages --operation install --packages packages[@]
1961
1962     packages=( 'ncurses-term' 'ncurses-examples' 'ncurses-base' 'ncurses-doc' )
1963     install_deb_packages --operation install --packages packages[@]
1964
1965     packages=( 'readline-doc' 'gdb' 'gdb-doc' 'uuid-dev' )
1966     install_deb_packages --operation install --packages packages[@]
1967
1968     packages=( 'sqlite3' 'sqlite3-doc' 'libsqlite3-dev' )
1969     install_deb_packages --operation install --packages packages[@]
1970
1971     packages=( 'tk-dev' 'python3-tk' 'python3-magic' 'xz-utils' )
1972     install_deb_packages --operation install --packages packages[@]
1973
1974     packages=( 'libpython3.8' 'libpython3.8-dev' 'libpython3.8-minimal'
↪ 'libpython3.8-stdlib' )
1975     install_deb_packages --operation install --packages packages[@]
1976
1977     packages=( 'liblzma-dev' 'libghc-lzma-dev' 'liblz-dev' 'lzma-dev'
↪ 'libqdbm-dev' )
1978     install_deb_packages --operation install --packages packages[@]
1979
1980     if [ "$(echo "$release" | grep '18.10')" ]; then
1981         packages=( 'libncurses6' 'libncurses6-dbg' 'libncursesw6'
↪ 'libncursesw6-dbg' )
1982         install_deb_packages --operation install --packages packages[@]
1983     fi
1984     return 0
1985
1986     elif [ "$(grep -i ubuntu /etc/os-release)" ] || [ "$(grep -i mint /etc/os-
↪ release)" ]; then
1987
1988         std_error_exit "Ubuntu-based distribution of unknown family or release."
↪ Exit " $E_OSERROR"
1989
1990     fi
1991
1992     std_error_exit "OS package dependencies failed to install correctly" "$E_
↪ DEPENDENCY"
1993     #
1994     # <-- end function os_packages -->
1995 }
1996
1997
1998 function package_info(){
1999     ##
2000     ##  displays information about this library module
2001     ##
2002     ##  - dependent module colors.sh is located always adjacent
2003     ##  - sourcing of dep modules must occur after local var to avoid overwrite
2004     ##  of variable values in this module

```

(continues on next page)

(continued from previous page)

```

2005  ##
2006  local version=__version__
2007  local act="$(echo -e ${a_orange})"
2008  local rst="$(echo -e ${reset})"
2009  local i path clearscn
2010  local tmpdir='/tmp'
2011  local logs
2012  local coll
2013  local total_index
2014  local l_index r_index
2015
2016  declare -a arr_modules          # array; all file modules comprising buildpy
2017  declare -a arr_functions        # array; all function names
2018  declare -a arr_left             # array; left-hand side column function names
2019  declare -a arr_right            # array; right-hand side column function names
2020
2021  clearscn=$(which clear 2>/dev/null)
2022
2023  # bash module ecosystem stats
2024  modules='1'                    # start @ 1 bc of main binary
2025  logs='0'
2026  coll='25'                      # lhs column fixed width
2027
2028  # add main executable to list of bash modules (.modules file)
2029  printf -- '\n%-12s%s %-37s%s\n' " " "-" "${cyan}$pkg${rst}" "$pkg_path" > $tmpdir/
↪.modules
2030
2031  for path in $pkg_lib/*; do
2032
2033      i=${path##*/}
2034
2035      if [ "${i#*.}" = "log" ] || [ "${i#*.}" = "pkg" ]; then
2036          continue
2037
2038      else
2039          if [ "$(echo $i | wc -c)" -ge "15" ]; then
2040
2041
2042              width=$(( $coll - $(echo $i | wc -c) ))
2043
2044              # req keep columns vertically aligned
2045              printf -- '%-12s%s %-37s%-${width}s\n' " " "-" "${cyan}$i${rst}" "
↪$pkg_lib" >> $tmpdir/.modules
2046
2047              else
2048                  printf -- '%-12s%s %-37s%-${width}s\n' " " "-" "${cyan}$i${rst}" "
↪$pkg_lib" >> $tmpdir/.modules
2049              fi
2050              # incr modules
2051              (( modules ++ ))
2052          fi
2053      done
2054
2055      # finally, add bash completion file artifact (last .modules file)
2056      printf -- '%-12s%s %-37s%s\n' " " "-" "${cyan}buildpy-completion.bash${rst}" " /
↪etc/bash_completion.d" >> $tmpdir/.modules
2057

```

(continues on next page)

(continued from previous page)

```

2058 # log files,rm path in front
2059 printf -- '\n' > $tmpdir/.logs
2060
2061 for i in $(LOG_FILE##*/) ${LOG_CONSOLE##*/}; do
2062
2063     if [ "$(echo $i | wc -c)" -ge "15" ]; then
2064         # req keep columns vertically aligned
2065         printf -- '%-12s%s %-34s\t%s\n' ' ' ' "-" "${cyan}$i${rst}" "$LOG_DIR" >>
↪$tmpdir/.logs
2066     else
2067         printf -- '%-12s%s %-37s%-16s\n' ' ' ' "-" "${cyan}$i${rst}" "$LOG_DIR" >>
↪$tmpdir/.logs
2068     fi
2069
2070     (( logs ++ ))
2071
2072 done
2073
2074 # split into 2 columns:
2075 arr_functions=( )
2076 for i in $(declare -F | awk '{print $3}'); do
2077     imod="-- ${i}"
2078     arr_functions=( "${arr_functions[@]}" "${imod}" )
2079 done
2080
2081 total_index=${#arr_functions[@]}
2082
2083 # construct, display program info msg output
2084 cat <<EOM
2085 ${printf "%0.s_" {1..65}}
2086
2087     ${title}Build ${hic}Python3 ${title}from Source | ${hic}Linux${btext}
2088
2089 ${printf -- '%-4s%-12s%-19s%s\n' ' ' "Module Name:" ' ' "${cyan}$pkg${rst}"}
2090 ${printf -- '%-4s%2s%-16s%s\n' ' ' "Module Version:" " " "${act}$version${rst}"}
2091 ${printf "%0.s_" {1..65}}
2092
2093 ${printf -- '%-4s%2s%-19s%s\n' ' ' "Bash Modules:" " " "$modules"}
2094 ${cat $tmpdir/.modules}
2095
2096 ${printf -- '%-4s%2s%-22s%s\n' ' ' "Log Files:" " " "$logs"}
2097 ${cat $tmpdir/.logs}
2098 ${printf "%0.s_" {1..65}}
2099
2100 ${printf -- '\t%2s%-17s%s\n' "Bash Functions:" " " "$total_index"}
2101
2102 EOM
2103
2104 l_index=$(( (${total_index} / 2 ) + 1 ) )
2105
2106 for j in "${arr_functions[@]:0:$l_index}"; do
2107     j_trunc=$(echo $j | cut -c 1-26)
2108     arr_left=( "${arr_left[@]}" "$j_trunc" )
2109 done
2110
2111 r_index=$(( $l_index + 1 ) )
2112

```

(continues on next page)



(continued from previous page)

```

2113     for k in "${arr_functions[@]:$r_index:$total_index}"; do
2114         arr_right=( "${arr_right[@]}" "$k" )
2115     done
2116
2117     middle_index="$l_index"
2118
2119     for m in $(seq 0 $middle_index); do
2120
2121         # calc the width of left-hand column element
2122         if [[ "${arr_left[$m]}" ]] || [[ "${arr_right[$m]}" ]]; then
2123
2124             width=$(echo "${arr_left[$m]}" | wc -c)
2125
2126             # subtract lhs column width - width of element; difference is rhs column_
↪offset
2127             col2=$(( $col1 - $width ))
2128             printf -- '%-12s%-'$col1's\t%-'$col2's\n' ' ' "${arr_left[$m]}" "${arr_
↪right[$m]}"
2129         else
2130             continue
2131         fi
2132     done
2133     printf -- '\n'
2134     # clean up
2135     rm $tmpdir/.functions $tmpdir/.modules $tmpdir/.logs 2>/dev/null
2136     #
2137     # <-- end function package_info -->
2138 }
2139
2140
2141 function post_install() {
2142     ##
2143     ## prints ending message and stats ##
2144     ##
2145     local endtime="$1"
2146
2147     if [ ! $endtime ]; then
2148
2149         START=$(timer)
2150         std_message "Python3 installation start: ${title}$(date +%H:%M:%S)${reset}"
↪"INFO" $LOG_FILE
2151         return 1
2152
2153     else
2154
2155         std_message "Python3 installation COMPLETE at $(date +%H:%M'). Total_
↪runtime: ${title}$(timer $endtime)${reset}." "INFO" $LOG_FILE
2156         std_message "See the following for installation details:\n
2157         \t* Log file:  ${yellow}$LOG_FILE${reset}\n
2158         \t* Console Event Log: ${yellow}$LOG_CONSOLE${reset}" "INFO"
2159
2160     fi
2161
2162     return 0
2163     #
2164     # <-- end function print stats -->
2165 }

```

(continues on next page)

(continued from previous page)

```

2166
2167
2168 function preinstall_validation() {
2169     ##
2170     ## Ensures version to be installed does not already exist
2171     ##
2172     local major="$1"
2173     if which "python${major}" 2>&1 >/dev/null; then
2174         std_warn "Python $major is already active on this system. Aborting_
↪ installation" $LOG_FILE
2175         exit 0
2176     fi
2177 }
2178
2179
2180 function root_permissions() {
2181     ##
2182     ## validates required root privileges ##
2183     ##
2184     if [ $EUID -ne 0 ]; then
2185
2186         std_warn "This command requires root or sudo access to root privileges.\n
↪ \n\t\tRe-run as root or execute with sudo:
2187         \n\t\t\t$ sudo $pkg <parameters>"
2188         printf -- "\n\n"
2189         std_logger "Permission denied. Rerun as root or execute with sudo" "WARN"
2190         ↪ $LOG_FILE
2191         exit $E_DEPENDENCY
2192
2193     elif [ $SUDO_USER ]; then
2194
2195         SUDO=""
2196         logfile_owner $LOG_FILE $LOG_CONSOLE
2197
2198     fi
2199
2200     return 0
2201
2202     #
2203     # <-- end function root_permissions -->
2204 }
2205
2206
2207 function root_permissions_bool() {
2208     ##
2209     ## validates required root privileges, return bool only ##
2210     ##
2211
2212     if [ $EUID -eq 0 ] || [ $SUDO_USER ]; then
2213
2214         return 0
2215
2216     fi
2217
2218     return 1
2219
2220     #

```

(continues on next page)

(continued from previous page)

```

2221     # <-- end function root_permissions -->
2222 }
2223
2224
2225 function source_profile() {
2226     ##
2227     ## used to import aliases and other environment elements at runtime ##
2228     ##
2229
2230     # source paths
2231     if [[ -f "$HOME/.bashrc" ]]; then
2232
2233         source "$HOME/.bashrc" 2>/dev/null
2234
2235     elif [[ -f "$HOME/.bash_profile" ]]; then
2236
2237         source "$HOME/.bash_profile" 2>/dev/null
2238
2239     fi
2240     return 0
2241     #
2242     # <-- end function source_profile -->
2243 }
2244
2245
2246 function test_removal() {
2247     ##
2248     ## tests to ensure python binaries and deps removed ##
2249     ##
2250     local major="$1"
2251     local python_ver="python$major"
2252     #
2253     std_message "Testing removal of Python $major..." "TEST" $LOG_FILE
2254
2255     # delay for user read
2256     sleep 1
2257
2258     if [ "$($python_ver --version 2>/dev/null)" ]; then
2259
2260         std_warn "Problem removing Python $major. Identifying remaining file_
↪ locations:" $LOG_FILE
2261         updatedb -v
2262         std_message "$(locate $python_ver)" "INFO" $LOG_FILE
2263
2264     else
2265
2266         std_message "Python $major ${green}successfully${reset} removed." "INFO" $LOG_
↪ FILE
2267
2268     fi
2269     #
2270     # <-- end function test_removal -->
2271 }
2272
2273
2274 function uninstall_disclaimer() {
2275     ##

```

(continues on next page)

(continued from previous page)

```

2276     ## Display disclaimer msg prior to initiating uninstall
2277     ##
2278     cat <<EOM
2279     ${red}_____ $
↪{reset}
2280
2281     ${title} DISCLAIMER ${btext}
2282     ${red}_____ $
↪{reset}
2283
2284     You are about to remove system-level python $major binaries
2285     and dependencies.
2286
2287     Uninstalling a Python version you have installed using this
2288     program generally works fine. Please note that uninstalling
2289     a Python version included natively with your operating
2290     system is ${ul}not recommended${reset}.
2291
2292     Depending on your Linux distribution, some operating system
2293     functions may depend upon the Python version(s) included
2294     natively with the OS.
2295
2296     Although the uninstall functionality has been successfully
2297     tested on all supported major Linux distributions,
2298
2299
2300     ${red}PROCEED AT YOUR OWN RISK${reset}
2301
2302
2303 EOM
2304     return 0
2305     #
2306     # <--- end function uninstal_disclaimer --->
2307 }
2308
2309
2310 function uninstall(){
2311     ##
2312     ## uninstalls system-level python3 ##
2313     ##
2314     #
2315     # This should ideally work via the following:
2316     # 1. find /usr -name "python3.X"
2317     # 2. delete paths that result
2318     # 3. Install and run mlocate after to find any paths left
2319     #
2320     # Installing mlocate is OS-specific (see function mlocate_installation):
2321     # REDHAT (for example, remove python3.6):
2322     # sudo yum -y update
2323     # # install epel
2324     # curl -o epel.rpm https://dl.fedoraproject.org/pub/epel/epel-release-
↪latest-7.noarch.rpm
2325     # sudo yum -y install epel.rpm
2326     # sudo yum -y update
2327     # sudo yum -y install mlocate
2328     # sudo updatedb
2329     #

```

(continues on next page)

(continued from previous page)

```

2330 local major="$1"
2331 local wipe_clean="$2"
2332 local choice
2333 local python_str="Python-$major"
2334 local python_ver="python$major"
2335 declare -a arr_paths
2336
2337 if ! binary_installed_bool "python$major"; then
2338     msg="You are attempting to uninstall $python_str, which is not installed. Exit
↪ "
2339
2340     if [ $QUIET ]; then
2341         std_logger "$msg" "WARN" $LOG_FILE
2342     else
2343         std_error_exit "$msg" $E_DEPENDENCY
2344     fi
2345 fi
2346
2347 # print disclaimer msg
2348 msg="Initiate uninstall of Python $major binaries"
2349
2350 if [ $QUIET ]; then
2351     std_logger "$msg" "INFO" $LOG_FILE
2352 else
2353     uninstall_disclaimer
2354     # choose
2355     if continue_on; then
2356         std_logger "$msg" "INFO" $LOG_FILE
2357     else
2358         # abort uninstall
2359         printf -- '\n'; exit 0
2360     fi
2361 fi
2362
2363 # create backup of pip packages
2364 pip_backup
2365
2366 # display paths found
2367 std_message "Found paths to delete:" "INFO" $LOG_FILE "pprint"
2368
2369 mapfile -t arr_paths < <(find /usr/local -name "$python_ver" 2>/dev/null)
2370
2371 for path in "${arr_paths[@]}"; do
2372     std_message "\t$path" "INFO" $LOG_FILE "pprint"
2373 done
2374
2375 std_message 'Please review the uninstall paths' 'INFO' $LOG_FILE
2376 if ! continue_on; then return 0; fi
2377
2378 # remove python3 version ("python_ver")
2379 for path in "${arr_paths[@]}"; do
2380     std_message "Deleting $path" "INFO" $LOG_FILE
2381     rm -fr "$path"
2382     printf -- "\n"
2383 done
2384
2385 # check specific paths noted in paths filemap

```

(continues on next page)

(continued from previous page)

```

2386 for path in $(cat $pkg_lib/uninstall.paths); do
2387     std_message "Deleting $path" "INFO" $LOG_FILE
2388     rm -fr "${path:?}/${python_ver}"
2389     printf -- "\n"
2390 done
2391
2392 # locate any residual locations only if WIPE CLEAN explicit option given
2393 if mlocate_installation && $wipe_clean; then
2394
2395     for path in $(locate $python_ver); do
2396         std_message "Deleting path: ${yellow}$path${btext}" "INFO" $LOG_FILE
2397         rm -fr $path
2398         printf -- "\n"
2399     done
2400
2401 fi
2402
2403 return 0
2404 #
2405 # <-- end function uninstall -->
2406 }
2407
2408
2409 function update_path(){
2410     ##
2411     ## Update PATH environment variable:
2412     ##     - /usr/local/bin
2413     ##
2414
2415     # ensure /usr/local/bin for python executables in PATH
2416     if [ ! "$(echo $PATH | grep '\usr\local\bin')" ]; then
2417
2418         if [ -f $HOME/.bashrc ]; then
2419
2420             printf -- '%s\n' "PATH=\$PATH:/usr/local/bin" >> "$HOME/.bashrc"
2421             printf -- '%s\n' "export PATH" >> "$HOME/.bashrc"
2422
2423         elif [ -f "$HOME/.bash_profile" ]; then
2424
2425             printf -- '%s\n' "PATH=\$PATH:/usr/local/bin" >> "$HOME/.bash_profile"
2426             printf -- '%s\n' "export PATH" >> "$HOME/.bash_profile"
2427
2428         elif [ -f "$HOME/.profile" ]; then
2429
2430             printf -- '%s\n' "PATH=\$PATH:/usr/local/bin" >> "$HOME/.profile"
2431             printf -- '%s\n' "export PATH" >> "$HOME/.profile"
2432
2433         fi
2434
2435     fi
2436
2437     # ensure local bin for python packages in PATH
2438     if [ ! "$(echo $PATH | grep $HOME\local\bin)" ]; then
2439
2440         if [ -f $HOME/.bashrc ]; then
2441
2442             printf -- '%s\n' "PATH=\$PATH:$HOME/.local/bin" >> "$HOME/.bashrc"

```

(continues on next page)

(continued from previous page)

```

2443     printf -- '%s\n' "export PATH" >> "$HOME/.bashrc"
2444
2445     elif [ -f "$HOME/.bash_profile" ]; then
2446
2447         printf -- '%s\n' "PATH=\$PATH:$HOME/.local/bin" >> "$HOME/.bash_profile"
2448         printf -- '%s\n' "export PATH" >> "$HOME/.bash_profile"
2449
2450     elif [ -f "$HOME/.profile" ]; then
2451
2452         printf -- '%s\n' "PATH=\$PATH:$HOME/.local/bin" >> "$HOME/.profile"
2453         printf -- '%s\n' "export PATH" >> "$HOME/.profile"
2454
2455     fi
2456
2457 fi
2458
2459 source_profile
2460 return 0
2461 #
2462 # <-- end function update_path -->
2463 }
2464
2465
2466 function upgrade_pip(){
2467     ##
2468     ## - upgrades pip3, setuptools
2469     ## - sets symlink to pip for python3
2470     ## - multiple update cycles to accommodate pip 9.x > pip 10.x > pip 18.x
2471     ##
2472     local pip_bin
2473     local count="0"
2474     #
2475     pip_bin="$(which pip3 2>/dev/null)"
2476     source_profile
2477
2478     # set path to pip binary
2479     if [ "$pip_bin" ]; then
2480
2481         std_message "Found pip3 at path: ${yellow}$pip_bin${reset}" "INFO" $LOG_FILE
2482
2483     else
2484         if [ -f "/usr/local/bin/pip3" ]; then
2485             pip_bin="/usr/local/bin/pip3"
2486
2487         elif [ "$(which pip 2>/dev/null)" ]; then
2488             pip_bin="$(which pip)"
2489
2490         elif [ -f "/usr/local/bin/pip" ]; then
2491             pip_bin="/usr/local/bin/pip"
2492
2493         else
2494             std_warn "Skipping pip / pip3 upgrade -- pip executable could not be found
↳" $LOG_FILE
2495             return 1
2496         fi
2497     fi
2498

```

(continues on next page)

(continued from previous page)

```

2499 #
2500 if [ $SUDO_USER ]; then
2501     # user using sudo
2502     while (( $count <= 2 )); do
2503
2504         # check for latest version, pip
2505         if [ "$($pip_bin list --outdated 2>/dev/null | grep pip)" ]; then
2506
2507             std_message "Upgrade pip3 to latest, set python3 symlink for pip"
2508             ↪ "INFO" $LOG_FILE
2509             $pip_bin install -U pip >> $LOG_CONSOLE 2>/dev/null &
2510             progress_dots --text "Upgrading pip3"
2511
2512         else
2513             std_message "pip $($pip_bin --version 2>/dev/null | awk '{print $2}')."
2514             ↪ installed - latest "INFO" $LOG_FILE
2515             break
2516         fi
2517
2518         # check for latest version, setuptools
2519         if [ "$($pip_bin list --outdated 2>/dev/null | grep setuptools)" ]; then
2520
2521             std_message "Upgrade setuptools to latest" "INFO" $LOG_FILE
2522             $pip_bin install -U setuptools >> $LOG_CONSOLE 2>/dev/null &
2523             progress_dots --text "upgrading setuptools"
2524
2525         else
2526             std_message "setuptools $($pip_bin list | grep setuptools | awk '
2527             ↪ {print $2}')" installed - latest "INFO" $LOG_FILE
2528         fi
2529
2530         count=$(( $count + 1 ))
2531     done
2532
2533 elif [ "$pip_bin" ]; then
2534     # root user
2535     while (( $count <= 2 )); do
2536
2537         # check for latest version, pip
2538         if [ "$($pip_bin list --outdated 2>/dev/null | grep pip)" ]; then
2539
2540             std_message "Upgrade pip3 to latest, set python3 symlink for pip"
2541             ↪ "INFO" $LOG_FILE
2542             $pip_bin install -U pip >> $LOG_CONSOLE 2>/dev/null &
2543             progress_dots --text "Upgrading pip3"
2544
2545         else
2546             std_message "pip $($pip_bin --version 2>/dev/null | awk '{print $2}')."
2547             ↪ installed - latest "INFO" $LOG_FILE
2548             break
2549         fi
2550
2551         # check for latest version, setuptools
2552         if [ "$($pip_bin list --outdated 2>/dev/null | grep setuptools)" ]; then
2553
2554             std_message "Upgrade setuptools to latest" "INFO" $LOG_FILE

```

(continues on next page)



(continued from previous page)

```

2551         $pip_bin install -U setuptools >> $LOG_CONSOLE 2>/dev/null &
2552         progress_dots --text "upgrading setuptools"
2553
2554     else
2555         std_message "setuptools $($pip_bin list | grep setuptools | awk '
↳{print $2}') installed - latest" "INFO" $LOG_FILE
2556     fi
2557
2558     count=$(( $count + 1 ))
2559
2560 done
2561
2562 else
2563
2564     std_warn "Unable to upgrade pip3, setuptools. Please use manual upgrade" $LOG_
↳FILE
2565
2566 fi
2567 #
2568 # <-- end function upgrade_pip -->
2569 }
2570
2571
2572 function verify_rc_status(){
2573     ##
2574     ## Reset VERSION global variable in case binary set to be
2575     ## compiled is a release candidate, alpha, or beta code.
2576     ##
2577     local version="$1"
2578     if [[ $(ls $TMPDIR | grep "Python-$version.tar.xz") ]]; then
2579         return 0
2580
2581     elif [[ $(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz") ]]; then
2582         bin=$(ls $TMPDIR | grep "Python-${version}rc[1-4].tar.xz")
2583         std_warn "Release Candidate binary ($bin). Stable version NOT available" $LOG_
↳FILE
2584         if continue_on "Do you want to install anyway?"; then
2585             return 0
2586         else
2587             exit 0
2588         fi
2589
2590     elif [[ $(ls $TMPDIR | grep "Python-${version}[a-b][1-4].tar.xz") ]]; then
2591         bin=$(ls $TMPDIR | grep "Python-${version}[a-b][1-4].tar.xz")
2592         std_warn "Beta or Alpha code binary ($bin)." $LOG_FILE
2593         if continue_on "Do you want to install anyway?"; then
2594             return 0
2595         else
2596             exit 0
2597         fi
2598     fi
2599 }
2600
2601
2602 function clean_up(){
2603     ##
2604     ## remove build artifacts ##

```

(continues on next page)

(continued from previous page)

```

2605  ##
2606  local version="$1"           # full version (X.Y.Z) to remove
2607  local yl=$(echo -e ${yellow}) # yellow link color code
2608  local clean_status="0"       # return code container
2609  local tmpdir='/tmp'
2610  declare -a arr_clean
2611
2612  ##  remove build artifacts for a specific version  ##
2613  if [ $version ]; then
2614
2615      mapfile -t arr_clean < <(find /tmp -name "Python-${version}*" 2>/dev/null)
2616
2617      std_message "Begin build artifact removal for Python $version only..." "INFO"
2618      ↪$LOG_FILE
2619      for residual in "${arr_clean[@]}; do
2620
2621          for artifact in $residual; do
2622              # .tar, .tar.xz, or dir object
2623              rm -fr $artifact
2624
2625              # success / failure status
2626              if [ -e "$artifact" ] || [ -f "$artifact" ] || [ -d "$artifact" ] ;_
2627      ↪then
2628          std_warn "Unable to remove ${yl}${artifact}${reset}" $LOG_FILE
2629          clean_status="1" # something went wrong
2630      else
2631          std_message "Successfully removed ${yl}${artifact}${reset}" "OK"
2632      ↪$LOG_FILE
2633      fi
2634      done
2635      done
2636      return $clean_status
2637
2638  else
2639      ##  remove build artifacts for all versions  ##
2640      find $tmpdir -name "Python-*" 2>/dev/null > $tmpdir/.s.results
2641
2642      for i in $(cat $tmpdir/.s.results); do
2643          arr_clean=( "${arr_clean[@]}" $i )
2644      done
2645
2646      if [ -z "${arr_clean[*]}" ]; then
2647          # nothing to clean
2648          std_warn "No build artifacts found to remove" $LOG_FILE
2649          return $clean_status
2650      fi
2651
2652      std_message "Locating all local build artifacts..." "INFO" $LOG_FILE
2653
2654      for residual in "${arr_clean[@]}; do
2655
2656          rm -fr $residual
2657
2658          # success / failure status
2659          if [ -e "$residual" ] || [ -f "$residual" ] || [ -d "$residual" ] ; then
2660              std_warn "Unable to remove ${yl}${residual}${reset}" $LOG_FILE
2661              clean_status="1" # something went wrong

```

(continues on next page)

(continued from previous page)

```

2659         else
2660             std_message "Successfully removed ${yl}${residual}${reset}" "OK" $LOG_
↪FILE
2661         fi
2662     done
2663     return $clean_status
2664 fi
2665 #
2666 # <-- end function clean_up -->
2667 }
2668
2669
2670
2671 function show_downloads_display() {
2672     ##
2673     ##
2674     ##
2675     local downloads location major size i
2676     local tmpfile sorted_tmp twidth
2677
2678     # set tempfile location
2679     if [ -d '/dev/shm' ]; then
2680         tmpfile='/dev/shm/downloads.tmp'
2681         sorted_tmp='/dev/shm/sorted.tmp'
2682     else
2683         tmpfile='/tmp/downloads.tmp'
2684         sorted_tmp='/tmp/sorted.tmp'
2685     fi
2686
2687     # find and sort downloaded python binaries on local filesystem
2688     downloads=${_current_downloads}
2689
2690     if [ ! "$downloads" ]; then
2691         std_message "No current downloads" "INFO" $LOG_FILE
2692     else
2693         twidth='52'
2694         printf -- '\n\t  %s  %s  %s  %s  %s\n' "Version" '|' "Size" '|'
↪"Filesystem Location"
2696         printf '\t%s\n' "$twidth" '|' tr ' ' -
2697
2698         for i in $downloads; do
2699             location=$(find /tmp -name "*${i}*" 2>/dev/null)
2700             major=${i#Python-}
2701             rsize=$(ls -l --block-size=MB $location | awk '{print $5}')
2702             size="${rsize%MB} MB"
2703             printf -- '\t%s %s %s %s %s\n' "${BOLD}Python-${btext}${bbc}$major${btext}
↪" '|' "$size" '|' "$location" >> $tmpfile
2704         done
2705
2706         # sort
2707         cat $tmpfile | sort -t . > $sorted_tmp
2708         awk '{ printf "%-8s %-41s %-2s %2s %-3s %-2s %-30s \n", " ", $1, $2, $3, $4,
↪$5, $6}' $sorted_tmp
2709         printf -- '\n'
2710         rm $tmpfile $sorted_tmp || true
2711     fi

```

(continues on next page)

(continued from previous page)

```

2712 }
2713
2714
2715 function major_version_reconcillation() {
2716     ##
2717     ## Test if MAJOR_VERSION includes minor version
2718     ## If yes, reset MAJOR_VERSION parameter
2719     ##
2720     local parameter_version="$1"
2721     local major
2722
2723     if [ "$(echo ${#parameter_version})" -ge 4 ]; then
2724         major=$(echo "$parameter_version" | awk -F '.' '{print $1"."$2}')
2725         echo "$major"
2726     else
2727         echo "$parameter_version"
2728     fi
2729     #
2730     # <-- end function major_version_reconcillation -->
2731 }
2732
2733 function version_reconcillation() {
2734     ##
2735     ## Test if VERSION includes minor version
2736     ## If yes, reset VERSION parameter
2737     ##
2738     local parameter_version="$1"
2739     local major
2740
2741     if [ "$(echo ${#parameter_version})" -ge 4 ]; then
2742         echo "$parameter_version"
2743     else
2744         echo ""
2745     fi
2746     #
2747     # <-- end function version_reconcillation -->
2748 }
2749
2750
2751 # --- main -----
2752
2753
2754 depcheck "$LOG_DIR" "$LOG_FILE" "$LOG_CONSOLE"
2755
2756 parse_parameters "$@"
2757
2758
2759 if [ $PKG_INFO ]; then
2760     # display information about this program
2761     package_info | more
2762
2763 elif [ $DOWNLOAD_ONLY ]; then
2764     # discover latest python version available
2765     if latest_version_available $MAJOR_VERSION; then
2766         # download build assets

```

(continues on next page)

(continued from previous page)

```

2769     download_binary $VERSION $MAJOR_VERSION
2770 fi
2771
2772 elif [ "$OPERATION" = "SHOW" ]; then
2773
2774     case "$SUBCMD" in
2775         'downloads')
2776             show_downloads_display
2777             exit 0
2778             ;;
2779
2780         'os-packages')
2781             os_type
2782             SHOW_OSPACKAGES="true"
2783             if [ $alternate_os ]; then
2784                 # match current os to $alternate_os; if match, os_packages '3' & set_
2785                 ↪ alternative os=""
2786                 # if different os, set family and release values
2787                 case ${alternate_os#[0-9]*} in
2788                     $family)
2789                         release=$(determine_osrelease $alternate_os)
2790                         alternate_os=""
2791                         ;;
2792                     *)
2793                         release=$(determine_osrelease $alternate_os)
2794                         ;;
2795                 esac
2796             fi
2797             os_packages '3'
2798             printf -- '\n'
2799             ;;
2800
2801         'Python-[0-9].[0-9] | 'Python-[0-9].[0-9][0-9] | [0-9].[0-9] | [0-9].[0-
2802         ↪ 9][0-9])
2803             # convert to major version number
2804             var="$2"; parameter=${var#Python-}
2805             if is_float "$parameter"; then
2806                 MAJOR_VERSION="$parameter"
2807             else
2808                 std_error_exit "You must provide Python major version (Example: 3.7)"
2809             fi
2810             # discover latest python version available
2811             latest_version_available $MAJOR_VERSION
2812             ;;
2813     esac
2814
2815 elif [ $OS_DETECT ]; then
2816
2817     # detect operating system type (test)
2818     os_type
2819
2820 elif [ "$INSTALL_OS_DEPS" ]; then
2821     function install_status_display(){
2822         # show installed status of packages
2823         SHOW_OSPACKAGES="true"
2824         os_packages
2825         printf -- '\n\n'

```

(continues on next page)

(continued from previous page)

```

2824 }
2825
2826 clear
2827 # determine os type
2828 os_type
2829 install_status_display
2830 read -p "    Install all Python 3 operating system package dependencies?  [quit]:_
↪ " choice
2831 case $choice in
2832     "" | "No" | "no" | "q" | "quit" | "Q*")
2833         printf -- '\n\n'
2834         exit 0
2835         ;;
2836     "y" | "Y" | "yes" | "Yes")
2837         # ensure root-level permissions
2838         root_permissions
2839         # install python os dependencies
2840         SHOW_OSPACKAGES=""          # turn off display-only variable to install
2841         os_packages
2842         ;;
2843     *)
2844         printf -- '\n\n'
2845         exit 0
2846         ;;
2847 esac
2848 printf -- '\n\t\t\t%s\n' 'FINAL OS PACKAGE INSTALLATION STATUS'
2849 install_status_display
2850
2851 elif [ $PIP_BACKUP ]; then
2852
2853     # update pip / pip3 and core deps to latest if root
2854     if root_permissions_bool; then
2855         upgrade_pip
2856     fi
2857
2858     # create backup of pip packages
2859     pip_backup
2860
2861 elif [ $CLEAN_UP ]; then
2862
2863     root_permissions
2864
2865     if [ "$CLEAN_VERSION" = 'ALL' ]; then
2866         clean_up
2867
2868         # discover latest python version available
2869         elif latest_version_available $MAJOR_VERSION; then
2870             # rm build artifacts
2871             clean_up $VERSION
2872         fi
2873
2874 elif [ $INSTALL ]; then
2875
2876     # start timer
2877     START=$(timer)
2878
2879     # validate root privileges

```

(continues on next page)

(continued from previous page)

```

2880 root_permissions
2881
2882 # reset version vars if MAJOR_VERSION contains full version (major.minor)
2883 VERSION=$(version_reconcillation $MAJOR_VERSION)
2884 MAJOR_VERSION=$(major_version_reconcillation $MAJOR_VERSION)
2885
2886 # ensure requested python binaries not currently installed
2887 preinstall_validation $MAJOR_VERSION
2888
2889 # discover latest python version available
2890 latest_version_available $MAJOR_VERSION || exit $E_DEPENDENCY
2891
2892 std_message "Major Python version marked for installation: ${pv_blue}$MAJOR_
↪VERSION${reset}\n \
2893     Full Python version marked for installation: ${bbw}$VERSION${reset}" "INFO"
2894 sleep 3
2895
2896 # create backup of pip packages
2897 pip_backup
2898
2899 # determine os type
2900 os_type
2901
2902 # download build assets
2903 download_binary $VERSION $MAJOR_VERSION
2904 verify_rc_status $VERSION
2905
2906 # install required operating system packages
2907 os_packages $VERSION
2908
2909 ## --- unpack --- ##
2910 cd $TMPDIR || exit $E_OSERROR
2911 std_message "Decompressing binary... " "INFO" $LOG_FILE
2912 VERSION="$(reset_rc_version $VERSION)"
2913 unxz Python-$VERSION.tar.xz 2>&1 | tee $tty >> $LOG_CONSOLE
2914 tar -xvf Python-$VERSION.tar 2>&1 | tee $tty >> $LOG_CONSOLE
2915
2916 ## --- pre-build --- ##
2917 std_message "Begin Python-$VERSION Configure Stage..." "INFO" $LOG_FILE
2918 cd $TMPDIR/Python-$VERSION || exit $E_OSERROR
2919 configure_python "$OPTIMIZATIONS" "$QUIET"
2920 std_message "End Python-$VERSION Configure Stage..." "OK" $LOG_FILE
2921
2922 ## --- build, install --- ##
2923 std_message "Begin Python-$VERSION compile with make..." "INFO" $LOG_FILE
2924 make -j $PARALLEL_JOB_CT 2>&1 | tee $tty >> $LOG_CONSOLE
2925 std_message "Compilation stage completed successfully" "OK" $LOG_FILE
2926 std_message "Begin Python-$VERSION install..." "INFO" $LOG_FILE
2927 make install 2>&1 | tee $tty >> $LOG_CONSOLE
2928 std_message "Python-$VERSION Installation completed successfully..." "OK" $LOG_
↪FILE
2929
2930 # clean up build artifacts
2931 std_message "Clean up -- remove build artifacts" "INFO" $LOG_FILE
2932 clean_up
2933
2934 # If missing, create python3 global symlink

```

(continues on next page)

(continued from previous page)

```
2935     create_global_symlink "$MAJOR_VERSION" "$LOG_FILE"
2936
2937     # update pip3, install symlink to pip fro python3
2938     upgrade_pip
2939
2940     # print ending runtime stats
2941     post_install $START
2942
2943 elif [ $UNINSTALL ]; then
2944
2945     # uninstall Python 3.X Version
2946     if [ ! $UNINSTALL_VERSION ]; then
2947
2948         std_error_exit "You must provide a Python3 major version number to remove\n
2949         \tExample: \n\n\t\t $ sudo ./buildpy --uninstall ${title}3.6${reset}"
2950
2951     else
2952
2953         # validate root privileges
2954         root_permissions
2955
2956         # uninstall binaries
2957         uninstall "$UNINSTALL_VERSION" "$WIPE_CLEAN"
2958
2959         # validate removal
2960         test_removal "$UNINSTALL_VERSION"
2961
2962     fi
2963
2964 elif [ $DISPLAY_VERSION ]; then
2965
2966     display_program_version
2967
2968 fi
2969
2970 # <-- end -->
2971 exit 0
2972
```

Back to *buildpy*

---

Back to *buildpy Source Code*

---

## 24.3 colors.sh

Back to *Bash Module Index*



```

1  #!/usr/bin/env bash
2
3  pkg=$(basename $0 2>/dev/null)
4
5  #-----
6  #
7  #   colors.sh module / std colors for bash
8  #
9  #-----
10 #   Bright ansi color codes:
11 #       Bright Black: \u001b[30;1m
12 #       Bright Red: \u001b[31;1m
13 #       Bright Green: \u001b[32;1m
14 #       Bright Yellow: \u001b[33;1m
15 #       Bright Blue: \u001b[34;1m
16 #       Bright Magenta: \u001b[35;1m
17 #       Bright Cyan: \u001b[36;1m
18 #       Bright White: \u001b[37;1m
19 #
20 #-----
21
22 VERSION="2.0.6"
23
24
25 # --- standard bash color codes -----
26 ↪-----
27
28 # std color codes
29 red=$(tput setaf 1)
30 green=$(tput setaf 2)
31 yellow=$(tput setaf 3)
32 blue=$(tput setaf 4)
33 purple=$(tput setaf 5)
34 cyan=$(tput setaf 6)
35 white=$(tput setaf 7)
36 gray=$(tput setaf 008)
37 magenta=$(tput setaf 13)
38
39 # Formatting
40 BOLD=`tput bold`
41 UNBOLD=`tput sgr0`
42 a_italic='\x1b[3m'
43 ITALIC=$(echo -e ${a_italic})
44
45 # std reset
46 reset=$(tput sgr0)
47
48
49 # --- ansi color escape codes -----
50 ↪-----
51
52 # ansi color codes
53 a_bluegray='\033[38;5;68;38;5;68m'
54 a_darkblue='\033[34;5;21m'
55 a_dgray='\033[38;5;95;38;5;8m' # dark gray
56 a_lgray='\033[38;5;95;38;5;245m' # light gray

```

(continues on next page)

(continued from previous page)

```

56  a_magenta='\033[38;5;95;38;5;177m'
57  a_orange='\033[38;5;95;38;5;214m'
58  a_wgray='\033[38;5;95;38;5;250m'           # white-gray
59  a_wgray2='\033[38;5;7m;38;5;250m'         # whitest-gray
60
61  # ansi bright colors
62  a_brightblue='\033[38;5;51m'
63  a_brightcyan='\033[38;5;36m'
64  a_brightgreen='\033[38;5;95;38;5;46m'
65  a_bluepurple='\033[38;5;68m'
66  a_brightred='\u001b[31;1m'
67  a_brightyellow='\033[38;5;11m'
68  a_brightyellow2='\033[38;5;95;38;5;226m'
69  a_brightyellowgreen='\033[38;5;95;38;5;155m'
70  a_brightwhite='\033[38;5;15m'
71
72  # ansi font formatting
73  bold='\u001b[1m'                         # ansi format
74  underline='\u001b[4m'                   # ansi format
75
76  # ansi escape code reset
77  resetansi='\u001b[0m'
78
79
80  # --- color print variables -----
81  ↪-----
82
83  # Initialize ansi colors
84  title=$(echo -e ${bold}${white})
85  url=$(echo -e ${underline}${a_brightblue})
86  options=$(echo -e ${white})
87  commands=$(echo -e ${a_brightcyan})      # use for ansi escape color_
88  ↪codes
89
90  # frame codes (use for tables)             SYNTAX: color:format (bold, etc)
91  pv_blue=$(echo -e ${a_brightblue})
92  pv_bluebold=$(echo -e ${bold}${a_brightblue})
93  pv_green=$(echo -e ${a_brightgreen})      # use for tables; green border_
94  ↪faming
95  pv_greenbold=$(echo -e ${bold}${a_brightgreen}) # use for tables; green bold_
96  ↪border faming
97  pv_orange=$(echo -e ${a_orange})          # use for tables; orange border_
98  ↪faming
99  pv_orangebold=$(echo -e ${bold}${a_orange}) # use for tables; orange bold_
100 ↪border faming
101 pv_white=$(echo -e ${a_brightwhite})      # use for tables; white border_
102 ↪faming
103 pv_whitebold=$(echo -e ${bold}${a_brightwhite}) # use for tables; white bold_
104 ↪border faming
105
106 pv_bodytext=$(echo -e ${reset}${a_wgray}) # main body text; set to reset_
107 ↪for native xterm
108 pv_bg=$(echo -e ${a_brightgreen})          # brightgreen foreground cmd
109 pv_bgb=$(echo -e ${bold}${a_brightgreen})  # bold brightgreen foreground_
110 ↪cmd
111 pv_wgray=$(echo -e ${a_wgray})

```

(continues on next page)

(continued from previous page)

```

103 pv_orange=$(echo -e ${a_orange})
104 pv_wgray=$(echo -e ${a_wgray})
105 pv_lgray=$(echo -e ${a_lgray})
106 pv_dgray=$(echo -e ${a_dgray})
107
108 # initialize default color scheme
109 accent=$(tput setaf 008) # ansi format
110 ansi_orange=$(echo -e ${a_orange}) # use for ansi escape color_
↪ codes
111
112 # reset print variable
113 RESET=$(echo -e ${resetansi})
114
115
116 # --- declarations -----
↪ -----
117
118
119 # indent, x spaces
120 function indent02() { sed 's/^/ /'; }
121 function indent04() { sed 's/^/ /'; }
122 function indent10() { sed 's/^/ /'; }
123 function indent15() { sed 's/^/ /'; }
124 function indent18() { sed 's/^/ /'; }
125 function indent20() { sed 's/^/ /'; }
126 function indent25() { sed 's/^/ /'; }
127 function indent40() { sed 's/^/ /'; }
128
129
130 # --- aliases -----
↪ -----
131
132 # alias for legacy backard compatibility
133 alias orange=${a_orange}
134 alias wgray=${a_wgray}
135 alias lgray=${a_lgray}
136 alias dgray=${a_dgray}
137 alias bodytext=${pv_bodytext}
138 alias blue_frame=${pv_blue}
139 alias bluebold_frame=${pv_bluebold}
140 alias green_frame=${pv_green}
141 alias greenbold_frame=${pv_greenbold}
142 alias orange_frame=${pv_orange}
143 alias orangebold_frame=${pv_orangebold}
144 alias white_frame=${pv_white}
145 alias whitebold_frame=${pv_whitebold}
146 alias brightblue=${a_brightblue}
147 alias brightcyan=${a_brightcyan}
148 alias brightgreen=${a_brightgreen}
149 alias bluepurple=${a_bluepurple}
150 alias brightred=${a_brightred}
151 alias brightyellow=${a_brightyellow}
152 alias brightyellow2=${a_brightyellow2}
153 alias brightyellowgreen=${a_brightyellowgreen}
154 alias brightwhite=${a_brightwhite}
155
156

```

(continues on next page)

(continued from previous page)

```

157 # --- display about -----
158 ↪-----
159
160 function print_local_variables(){
161     # print out all variables contained in this module:
162     VARS=$(set -o posix ; set)
163     SCRIPT_VARS=$(grep -vFe "$VARS" <<<"$(set -o posix ; set)" | grep -v ^VARS=)
164     echo -e "$SCRIPT_VARS"
165     unset VARS
166 }
167
168
169 function print_colors(){
170     # print out all variables contained in this module:
171     declare -a array=("${!1}")
172     local rst=$(echo -e ${reset})
173
174     for i in "${array[@]}; do
175         var=$(echo -e ${i})
176         printf -- '\t%s\n' $var
177     done
178     return 0
179 }
180
181
182 function pkg_info(){
183     ##
184     ## displays information about this library module
185     ##
186     ## - dependent module colors.sh is located always adjacent
187     ## - sourcing of dep modules must occur after local var to avoid overwrite
188     ## of variable values in this module
189     ##
190     local version="$1"
191     local bdwt=$(echo -e ${bold}${a_brightwhite})
192     local act=$(echo -e ${a_orange})
193     local rst=$(echo -e ${reset})
194
195     declare -a ansi_colors
196     declare -a printvalue_colors
197
198     # generate list of functions
199     printf -- '%s\n' "$(declare -F | awk '{print $3}')" > /tmp/.functions
200     sum=$(cat /tmp/.functions | wc -l)
201
202     # construct, display help msg output
203     cat <<EOM
204
205     _____
206
207     ${title}colors.sh${rst}: Bash Color Library
208
209     Module Name:          ${cyan}${pkg}${rst}
210     Module Version:       ${act}${version}${rst}
211
212     _____
213
214     Module Contains $sum Functions:

```

(continues on next page)

(continued from previous page)

```

213
214 EOM
215     # display list of function names in this module
216     for l in $(cat /tmp/.functions); do
217         printf -- '\t%s %s\n' "-" "$l"
218     done
219     printf -- '\n'
220     rm /tmp/.functions
221
222     # show vars contained
223     ansi_colors=$(set -o posix ; set | grep 'a_')
224     asum=$(set -o posix ; set | grep 'a_' | wc -l)
225
226     printvalue_colors=$(set -o posix ; set | grep 'pv_')
227     pvsum=$(set -o posix ; set | grep 'pv_' | wc -l)
228
229     # display color vars
230     cat <<EOM
231
232
233     ${rst}ANSI Codes:  $asum
234
235     $(print_colors ansi_colors[@])
236
237     ${rst}Print Value Codes:  $pvsum
238
239     $(print_colors printvalue_colors[@])
240     ${rst}
241
242 EOM
243     #
244     # <-- end function pkg_info -->
245 }
246
247     # print information about this package
248     if [ "$pkg" = "colors.sh" ]; then
249         pkg_info "$VERSION"
250         exit 0
251     fi

```

[Back to Bash Module Index](#)

## 24.4 exitcodes.sh

```

1  #!/usr/bin/env bash
2
3  #
4  #   EXIT ERROR CODES -- source via script
5  #   ERROR CODES, version 1.3
6  #
7
8  # error codes
9  E_OK='0'                # exit code if normal exit conditions

```

(continues on next page)

(continued from previous page)

```
10 E_DEPENDENCY='1'           # exit code if missing required ec2cdependency
11 E_PERMISSIONS='2'         # exit code if inadequate permissions for attempted_
    ↪ operation
12 E_NOLOG='3'               # exit code if failure to create log dir, log file
13 E_BADSHELL='4'           # exit code if incorrect shell detected
14 E_AUTH='5'                # exit code if authentication failure
15 E_CONFIG='6'              # exit code if configuration file missing or corrupted
16 E_OSError='7'             # exit code if fail to identify os or os-specific attribute
17 E_USER_CANCEL='8'         # exit code if user cancel
18 E_BADARG='9'              # exit code if bad input parameter
19 E_NETWORK_ACCESS='10'     # exit code if no network access from current location
20 E_EXPIRED_CREDS='11'      # exit code if temporary credentials no longer valid
21 E_MISC='31'               # exit code if miscellaneous (unspecified) error
```

[Back to Bash Module Index](#)

---

## 24.5 version.py

```
1 __version__="1.8.12"
```

[Back to Bash Module Index](#)

---

## 24.6 os\_distro.sh

[Back to Bash Module Index](#)

```
1 #!/usr/bin/env bash
2
3 #
4 #   Author:                based on the original by zeldarealm@gmail.com
5 #   Source:                https://github.com/KittyKatt/screenFetch
6 #   Instructions:          source function-library/os_distro.sh ; detectdistro
7 #   Version:               1.4.2
8
9 verboseOut () {
10     if [[ "$verbosity" -eq "1" ]]; then
11         printf "\033[1;31m:: \033[0m$1\n"
12     fi
13 }
14
15 errorOut () {
16     printf "\033[1;37m[[ \033[1;31m! \033[1;37m]] \033[0m$1\n"
17 }
18
```

(continues on next page)

(continued from previous page)

```

19 stderrOut () {
20     while IFS=' ' read -r line; do printf "\033[1;37m[[ \033[1;31m! \033[1;37m]]_
↪\033[0m${line}\n"; done
21 }
22
23
24 function amazonlinux_release_version(){
25     #
26     # determines release version internally from within an
27     # amazonlinux host os environment.
28     #
29     # Requires identification of AmazonLinux OS Family as a
30     # prerequisite
31     #
32     local image_id
33     local region
34     local cwd=$PWD
35     local tmp='/tmp'
36     #
37     cd $tmp
38     curl -O 'http://169.254.169.254/latest/dynamic/instance-identity/document'
39     image_id="$(jq -r .imageId $tmp/document)"
40     region="$(jq -r .region $tmp/document)"
41     aws ec2 describe-images --image-ids $image_id --region $region > $tmp/images.
↪json
42     printf -- "%s\n" "$(jq -r .Images[0].Name $tmp/images.json | awk -F ' ' '
↪{print $1}')"
43     rm $tmp/document $tmp/images.json
44     cd $cwd
45 }
46
47
48 function fedora_release_version(){
49     #
50     # determines Redhat release version internally from
51     # within a Redhat host os environment.
52     #
53     # Requires identification of Redhat Linux OS Family
54     # as a prerequisite
55     #
56     local release
57     #
58     release="$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')"
59     printf -- "%s\n" "$(echo $release)"
60 }
61
62
63 function fedora_codename_version(){
64     #
65     # determines Redhat codename internally from
66     # within a Redhat host os environment.
67     #
68     # Requires identification of Redhat Linux OS Family
69     # as a prerequisite
70     #
71     local codename
72

```

(continues on next page)

(continued from previous page)

```

73     codename="$(grep CODENAME /etc/os-release | awk -F '=' '{print $2}')"
74
75     if [ "$codename" != \"\" ]; then
76         printf -- '%s\n' $codename
77     else
78         printf -- '%s\n' ""
79     fi
80 }
81
82
83 function redhat_release_version(){
84     #
85     #  determines Redhat release version internally from
86     #  within a Redhat host os environment.
87     #
88     #  Requires identification of Redhat Linux OS Family
89     #  as a prerequisite
90     #
91     local release
92     #
93     release="$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')"
94     printf -- "%s\n" $(echo $release | cut -c 2-50 | rev | cut -c 2-50 | rev)
95 }
96
97
98 function redhat_codename_version(){
99     #
100     #  determines Redhat codename internally from
101     #  within a Redhat host os environment.
102     #
103     #  Requires identification of Redhat Linux OS Family
104     #  as a prerequisite
105     #
106     local codename
107     codename="$(grep VERSION /etc/os-release | head -n 1 | awk -F '(' '{print $2}
↪ ' | awk -F ')' '{print $1}')"
108     printf -- "%s\n" $codename
109 }
110
111
112 detectdistro () {
113     ##_
114     ↪
115     ↪#
116     ↪#_
117     ↪    determines:
118     ↪#
119     ↪#                - os_
120     ↪family
121     ↪#
122     ↪#                - release (revision)_
123     ↪number                #
124     ↪#                - codename (informaal_
125     ↪name)                #
126     ##
127     local format="$1"      #  accepts "json" or '' (None)
128     #

```

(continues on next page)



(continued from previous page)

```

121     if [[ -z "${distro}" ]]; then
122         distro="Unknown"
123         # LSB Release Check
124         if type -p lsb_release >/dev/null 2>&1; then
125             # read distro_detect distro_release distro_codename <<< $(lsb_
↪ release -sirc)
126             distro_detect=( $(lsb_release -sirc) )
127             if [[ ${#distro_detect[@]} -eq 3 ]]; then
128                 distro_codename=${distro_detect[2]}
129                 distro_release=${distro_detect[1]}
130                 distro_detect=${distro_detect[0]}
131             else
132                 for ((i=0; i<${#distro_detect[@]}; i++)); do
133                     if [[ ${distro_detect[$i]} =~ ^[:digit:]]+(.
↪ [:digit:]]+|[:digit:]]+)+$ ]]; then
134                         distro_release=${distro_detect[$i]}
135                         distro_codename=${distro_detect[@]:$((
↪ $i+1))}:${#distro_detect[@]}+1}
136                         distro_detect=${distro_detect[@]:0:$
↪ {i}}
137                         break 1
138                     elif [[ ${distro_detect[$i]} =~ [Nn]/[Aa] || $
↪ {distro_detect[$i]} == "rolling" ]]; then
139                         distro_release=${distro_detect[$i]}
140                         distro_codename=${distro_detect[@]:$((
↪ $i+1))}:${#distro_detect[@]}+1}
141                         distro_detect=${distro_detect[@]:0:$
↪ {i}}
142                         break 1
143                     fi
144                 done
145             fi
146             case "${distro_detect}" in
147                 "CentOS"|"Chapeau"|"Deepin"|"Devuan"|"DesaOS"|"Fedora
↪ "|"gNewSense"|"Jiyuu Linux"|"Kogaion"|"Korora"|"Mageia"|"Netrunner"|"NixOS"|"Pardus
↪ "|"Raspbian"|"Sabayon"|"Solus"|"SteamOS"|"Trisquel"|"Ubuntu"|"GrombyangOS"|"
↪ "Scientific Linux")
148                 # no need to fix $distro/$distro_codename/
149                 ↪ $distro_release
150                 distro="${distro_detect}"
151                 ;;
152                 "archlinux"|"Arch Linux"|"arch"|"Arch"|"archarm")
153                 distro="Arch Linux"
154                 distro_release="n/a"
155                 if [ -f /etc/os-release ]; then
156                     os_release="/etc/os-release";
157                 elif [ -f /usr/lib/os-release ]; then
158                     os_release="/usr/lib/os-release";
159                 fi
160                 if [[ -n ${os_release} ]]; then
161                     if grep -q 'antergos' /etc/os-release;
↪ then
162                         distro="Antergos"
163                         distro_release="n/a"
164                     fi
165                     if grep -q -i 'logos' /etc/os-release;
↪ then

```

(continues on next page)

(continued from previous page)

```

165                                     distro="Logos"
166                                     distro_release="n/a"
167                                     fi
168                                     if grep -q -i 'swagarch' /etc/os-
↪release; then
169                                     distro="SwagArch"
170                                     distro_release="n/a"
171                                     fi
172                                     if grep -q -i 'obrevange' /etc/os-
↪release; then
173                                     distro="OBRevenge"
174                                     distro_release="n/a"
175                                     fi
176                                     fi
177                                     ;;
178                                     "ALDOS"|"Aldos")
179                                     distro="ALDOS"
180                                     ;;
181                                     "artixlinux"|"Artix Linux"|"artix"|"Artix"|"Artix_
↪release")
182                                     distro="Artix"
183                                     ;;
184                                     "blackPantherOS"|"blackPanther"|"blackpanther"|
↪"blackpantheros")
185                                     distro=$(source /etc/lsb-release; echo "
↪$DISTRIB_ID")
186                                     distro_release=$(source /etc/lsb-release;
↪echo "$DISTRIB_RELEASE")
187                                     distro_codename=$(source /etc/lsb-release;
↪echo "$DISTRIB_CODENAME")
188                                     ;;
189                                     "BLAG")
190                                     distro="BLAG"
191                                     distro_more="$(head -n1 /etc/fedora-release)"
192                                     ;;
193                                     "Chakra")
194                                     distro="Chakra"
195                                     distro_release=""
196                                     ;;
197                                     "BunsenLabs")
198                                     distro=$(source /etc/lsb-release; echo "
↪$DISTRIB_ID")
199                                     distro_release=$(source /etc/lsb-release;
↪echo "$DISTRIB_RELEASE")
200                                     distro_codename=$(source /etc/lsb-release;
↪echo "$DISTRIB_CODENAME")
201                                     ;;
202                                     "Debian")
203                                     if [[ -f /etc/crunchbang-lsb-release || -f /
↪etc/lsb-release-crunchbang ]]; then
204                                     distro="CrunchBang"
205                                     distro_release=$(awk -F'=' '/^DISTRIB_
↪RELEASE=/ {print $2}' /etc/lsb-release-crunchbang)
206                                     distro_codename=$(awk -F'=' '/^
↪DISTRIB_DESCRIPTION=/ {print $2}' /etc/lsb-release-crunchbang)
207                                     elif [[ -f /etc/siduction-version ]]; then
208                                     distro="Siduction"

```

(continues on next page)

(continued from previous page)

```

209         distro_release="(Debian Sid)"
210         distro_codename=""
211         elif [[ -f /usr/bin/pveversion ]]; then
212             distro="Proxmox VE"
213             distro_more="$(/usr/bin/pveversion |_
↪grep -oP 'pve-manager\\K\\d+\\.\\d+') "
214         elif [[ -f /etc/os-release ]]; then
215             if [[ "$(cat /etc/os-release)" =~
↪"Raspbian" ]]; then
216                 distro="Raspbian"
217                 distro_release=$(awk -F=' ' /^
↪PRETTY_NAME=/ {print $2}' /etc/os-release)
218             fi
219             if [[ "$(cat /etc/os-release)" =~
↪"BlankOn" ]]; then
220                 distro="BlankOn"
221                 distro_release=$(awk -F=' ' /^
↪PRETTY_NAME=/ {print $2}' /etc/os-release)
222             else
223                 distro="Debian"
224             fi
225         else
226             distro="Debian"
227         fi
228         ;;
229         "elementary"|"elementary OS")
230             distro="elementary OS"
231             ;;
232         "EvolveOS")
233             distro="Evolve OS"
234             ;;
235         "KaOS"|"kaos")
236             distro="KaOS"
237             ;;
238         "frugalware")
239             distro="Frugalware"
240             distro_codename=null
241             distro_release=null
242             ;;
243         "Fuduntu")
244             distro="Fuduntu"
245             distro_codename=null
246             ;;
247         "Fux")
248             distro="Fux"
249             distro_codename=null
250             ;;
251         "Gentoo")
252             if [[ "$(lsb_release -sd)" =~ "Funtoo" ]];_
↪then
253                 distro="Funtoo"
254             else
255                 distro="Gentoo"
256             fi
257             . /etc/portage/make.conf #detecting release_
↪stable/testing/experimental
258         case $ACCEPT_KEYWORDS in

```

(continues on next page)

(continued from previous page)

```

259                                     [a-z]*) distro_release=stable      ;;
260                                     ~*)   distro_release=testing      ;;
261                                     '**')  distro_release=experimental ;;
↪ #experimental usually includes git-versions.
262                                     esac
263                                     ;;
264                                     "Hyperbola GNU/Linux-libre"|"Hyperbola")
265                                         distro="Hyperbola GNU/Linux-libre"
266                                         distro_codename="n/a"
267                                         distro_release="n/a"
268                                     ;;
269                                     "LinuxDeepin")
270                                         distro="LinuxDeepin"
271                                         distro_codename=null
272                                     ;;
273                                     "Kali"|"Debian Kali Linux")
274                                         distro="Kali Linux"
275                                         if [[ "${distro_codename}" =~ "kali-rolling"
↪ ]]; then
276                                             distro_codename="n/a"
277                                             distro_release="n/a"
278                                         fi
279                                     ;;
280                                     "Lunar Linux"|"lunar")
281                                         distro="Lunar Linux"
282                                     ;;
283                                     "MandrivaLinux")
284                                         distro="Mandriva"
285                                         case "${distro_codename}" in
286                                             "turtle"|"Henry_Farman"|"Farman" |
↪ "Adelie"|"pauillac")
287                                             distro="Mandriva-${distro_
↪ release}"
288                                             distro_codename=null
289                                         ;;
290                                     esac
291                                     ;;
292                                     "ManjaroLinux")
293                                         distro="Manjaro"
294                                     ;;
295                                     "Mer")
296                                         distro="Mer"
297                                         if [[ -f /etc/os-release ]]; then
298                                             if grep -q 'SailfishOS' /etc/os-
↪ release; then
299                                             distro="SailfishOS"
300                                             distro_codename="$(grep
↪ 'VERSION=' /etc/os-release | cut -d '(' -f2 | cut -d ')' -f1)"
301                                             distro_release="$(awk -F'=' ' /
↪ ^VERSION=/ {print $2}' /etc/os-release)"
302                                         fi
303                                     fi
304                                     ;;
305                                     "neon"|"KDE neon")
306                                         distro="KDE neon"
307                                         distro_codename="n/a"
308                                         distro_release="n/a"

```

(continues on next page)

(continued from previous page)

```

309         if [[ -f /etc/issue ]]; then
310             if grep -q "^KDE neon" /etc/issue ;
↪ then
311                 distro_release="$(grep '^KDE_
↪neon' /etc/issue | cut -d ' ' -f3)"
312             fi
313         fi
314         ;;
315         "Ol"|"ol"|"Oracle Linux")
316             distro="Oracle Linux"
317             [ -f /etc/oracle-release ] && distro_release="
↪$(sed 's/Oracle Linux //' /etc/oracle-release)"
318         ;;
319         "LinuxMint")
320             distro="Mint"
321             if [[ "${distro_codename}" == "debian" ]];
↪ then
322                 distro="LMDE"
323                 distro_codename="n/a"
324                 distro_release="n/a"
325             fi
326             ;;
327         "openSUSE"|"openSUSE project"|"SUSE LINUX")
328             distro="openSUSE"
329             if [ -f /etc/os-release ]; then
330                 if [[ "$(cat /etc/os-release)" =~
↪"SUSE Linux Enterprise" ]]; then
331                     distro="SUSE Linux Enterprise"
332                     distro_codename="n/a"
333                     distro_release=$(awk -F'=' '/^
↪VERSION_ID=/ {print $2}' /etc/os-release | tr -d '"')
334                 fi
335             fi
336             if [[ "${distro_codename}" == "Tumbleweed" ]];
↪ then
337                 distro_release="n/a"
338             fi
339             ;;
340         "Parabola GNU/Linux-libre"|"Parabola")
341             distro="Parabola GNU/Linux-libre"
342             distro_codename="n/a"
343             distro_release="n/a"
344             ;;
345         "Parrot"|"Parrot Security")
346             distro="Parrot Security"
347             ;;
348         "PCLinuxOS")
349             distro="PCLinuxOS"
350             distro_codename="n/a"
351             distro_release="n/a"
352             ;;
353         "Peppermint")
354             distro="Peppermint"
355             distro_codename=null
356             ;;
357         "rhel")
358             distro="Redhat "

```

(continues on next page)

(continued from previous page)

```

359         distro_release="$(redhat_release_version)"
360         distro_codename="$(redhat_codename_version)"
361         ;;
362         "RosaDesktopFresh")
363             distro="ROSA"
364             distro_release=$(grep 'VERSION=' /etc/os-
↪release | cut -d ' ' -f3 | cut -d "\"" -f1)
365             distro_codename=$(grep 'PRETTY_NAME=' /etc/os-
↪release | cut -d ' ' -f4,4)
366             ;;
367         "SailfishOS")
368             distro="SailfishOS"
369             if [[ -f /etc/os-release ]]; then
370                 distro_codename="$(grep 'VERSION=' /
↪etc/os-release | cut -d '(' -f2 | cut -d ')' -f1)"
371                 distro_release="$(awk -F'=' '/^
↪VERSION=/ {print $2}' /etc/os-release)"
372             fi
373             ;;
374         "Sparky"|"SparkyLinux")
375             distro="SparkyLinux"
376             ;;
377         "Viperr")
378             distro="Viperr"
379             distro_codename=null
380             ;;
381         "Void"|"VoidLinux")
382             distro="Void Linux"
383             distro_codename=""
384             distro_release=""
385             ;;
386         "Zorin")
387             distro="Zorin OS"
388             distro_codename=""
389             ;;
390         *)
391             if [ "x$(printf "${distro_detect}" | od -t x1
↪| sed -e 's/^\w*\ */' | tr '\n' ' ' | grep 'eb b6 89 ec 9d 80 eb b3 84 ')" != "x"
↪]; then
392                 distro="Red Star OS"
393                 distro_codename="n/a"
394                 distro_release=$(printf "${distro_
↪release}" | grep -o '[0-9.]' | tr -d '\n')
395             fi
396             ;;
397         esac
398         if [[ "${distro_detect}" =~ "RedHatEnterprise" ]]; then
399             distro="Redhat"
400             distro_release="$(redhat_release_version)"
401             distro_codename="$(redhat_codename_version)"
402         fi
403         if [[ "${distro_detect}" =~ "SUSELinuxEnterprise" ]]; then
↪distro="SUSE Linux Enterprise"; fi
404         if [[ -n "${distro_release}" && "${distro_release}" != "n/a" ]];
↪then distro_more="${distro_release}"; fi
405         if [[ -n "${distro_codename}" && "${distro_codename}" != "n/a" ]];
↪then distro_more="${distro_more} ${distro_codename}"; fi

```

(continues on next page)

(continued from previous page)

```

406         fi
407
408         # Existing File Check
409         if [ "$distro" == "Unknown" ]; then
410             if [ $(uname -o 2>/dev/null) ]; then
411                 os=$(uname -o)
412                 case "$os" in
413                     "Cygwin"|"FreeBSD"|"OpenBSD"|"NetBSD")
414                         distro="$os"
415                         fake_distro="${distro}"
416
417                     ;;
418                     "DragonFly")
419                         distro="DragonFlyBSD"
420                         fake_distro="${distro}"
421
422                     ;;
423                     "Msys")
424                         distro="Msys"
425                         fake_distro="${distro}"
426                         distro_more="${distro} $(uname -r |
427
428                 ↪head -c 1)"
429
430                     ;;
431                     "Haiku")
432                         distro="Haiku"
433                         distro_more="$(uname -v | tr ' ' '\n'
434
435                 ↪| grep 'hrev')"
436
437                     ;;
438                     "GNU/Linux")
439                         if type -p crux >/dev/null 2>&1; then
440                             distro="CRUX"
441                             distro_more="$(crux | awk '
442
443                 ↪{print $3}')"
444
445                         fi
446                         if type -p nixos-version >/dev/null 2>
447
448                 ↪&1; then
449                             distro="NixOS"
450                             distro_more="$(nixos-version)"
451
452                         fi
453                         if type -p sorcery >/dev/null 2>&1;
454
455                 ↪then
456                             distro="SMGL"
457
458                         fi
459                         if (type -p guix && type -p herd) >/
460
461                 ↪dev/null 2>&1; then
462                             distro="GuixSD"
463
464                         fi
465
466                     ;;
467                 esac
468             fi
469             if [[ "${distro}" == "Cygwin" || "${distro}" == "Msys" ]];
470
471                 ↪then
472                 # https://msdn.microsoft.com/en-us/library/ms724832
473                 ↪%28VS.85%29.aspx
474
475                 if [ "$(wmic os get version | grep -o '^\\(6\\.
476
477                 ↪[23]\\|10\\)')" ]; then
478                     fake_distro="Windows - Modern"
479
480                 fi
481             fi
482         fi

```

(continues on next page)

(continued from previous page)

```

454         if [[ "${distro}" == "Unknown" ]]; then
455             if [ -f /etc/os-release ]; then
456                 os_release="/etc/os-release";
457             elif [ -f /usr/lib/os-release ]; then
458                 os_release="/usr/lib/os-release";
459             fi
460             if [[ -n $os_release ]]; then
461                 distrib_id=$(<${os_release});
462                 for l in $(echo $distrib_id); do
463                     if [[ ${l} =~ ^ID= ]]; then
464                         distrib_id=${l//*=}
465                         distrib_id=${distrib_id//\"/}
466                         break l
467                     fi
468                 done
469                 if [[ -n ${distrib_id} ]]; then
470                     if [[ -n ${BASH_VERSINFO} && ${BASH_
↪VERSINFO} -ge 4 ]]; then
471                         distrib_id=$(for i in $
↪{distrib_id}; do echo -n "${i^} "; done)
472                         distro=${distrib_id% }
473                         unset distrib_id
474                     else
475                         distrib_id=$(for i in $
↪{distrib_id}; do FIRST_LETTER=$(echo -n "${i:0:1}" | tr "[:lower:]" "[:upper:]");
↪echo -n "${FIRST_LETTER}${i:1} "; done)
476                         distro=${distrib_id% }
477                         unset distrib_id
478                     fi
479                 fi
480
481                 # Hotfixes
482                 [[ "${distro}" == "void" ]] && distro="Void_
↪Linux"
483                 [[ "${distro}" == "evolveos" ]] && distro=
↪"Evolve OS"
484                 [[ "${distro}" == "antergos" ]] && distro=
↪"Antergos"
485                 [[ "${distro}" == "logos" ]] && distro="Logos"
486                 [[ "${distro}" == "Arch" || "${distro}" ==
↪"Archarm" || "${distro}" == "archarm" ]] && distro="Arch Linux"
487                 [[ "${distro}" == "elementary" ]] && distro=
↪"elementary OS"
488                 [[ "${distro}" == "Fedora" && -d /etc/qubes-
↪rpc ]] && distro="qubes" # Inner VM
489                 [[ "${distro}" == "Ol" || "${distro}" == "ol"
↪ ]] && distro="Oracle Linux"
490                 if [[ "${distro}" == "Oracle Linux" ]] && [ -
↪f /etc/oracle-release ]; then
491                     distro_more=$(sed 's/Oracle Linux //
↪' /etc/oracle-release) "
492                 fi
493                 [[ "${distro}" == "rhel" ]] && distro="Redhat"
494                 [[ "${distro}" == "Neon" ]] && distro="KDE_
↪neon"
495                 [[ "${distro}" == "SLED" || "${distro}" ==
↪"sled" || "${distro}" == "SLES" || "${distro}" == "sles" ]] && distro="SUSE Linux_
↪Enterprise"

```

(continues on next page)



(continued from previous page)

```

496         if [[ "${distro}" == "SUSE Linux Enterprise"
↪]] && [ -f /etc/os-release ]; then
497             distro_more="$(awk -F=' ' /^VERSION_
↪ID=/ {print $2}' /etc/os-release | tr -d '')"
498             fi
499             if [[ "${distro}" == "Debian" ]] && [ -f /usr/
↪bin/pveversion ]; then
500                 distro="Proxmox VE"
501                 distro_more="$(/usr/bin/pveversion |
↪grep -oP 'pve-manager\/\K\d+\.\d+'))"
502             fi
503         fi
504     fi
505
506     if [[ "${distro}" == "Unknown" ]]; then
507         if [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" ==
↪"linux" ]]; then
508             if [ -f /etc/lsb-release ]; then
509                 LSB_RELEASE=$(cat /etc/lsb-release)
510                 distro=$(echo ${LSB_RELEASE} | awk
↪'BEGIN {
511                     distro = "Unknown"
512                 }
513                 {
514                     if ($0 ~ /
↪[Uu][Bb][Uu][Nn][Tt][Uu]/) {
515                         distro = "Ubuntu"
516                         exit
517                     }
518                     else if ($0 ~ /
↪[Mm][Ii][Nn][Tt]/ && $0 ~ /[Dd][Ee][Bb][Ii][Aa][Nn]/) {
519                         distro = "LMDE"
520                         exit
521                     }
522                     else if ($0 ~ /
↪[Mm][Ii][Nn][Tt]/) {
523                         distro = "Mint"
524                         exit
525                     }
526                 } END {
527                     print distro
528                 }')
529             fi
530         fi
531     fi
532
533     if [[ "${distro}" == "Unknown" ]] && [[ "${OSTYPE}" == "linux-
↪gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then
534         for di in arch chakra crunchbang-lsb evolveos
↪exherbo fedora \
535             frugalware fux gentoo
↪kogaion mageia obarun oracle \
536             pardus polinuxos
↪redhat redstar rosa SuSe; do
537             if [ -f /etc/$di-release ]; then
↪distro=$di && break; fi
538         done

```

(continues on next page)

(continued from previous page)

```

539         if [[ "${distro}" == "crunchbang-lsb" ]]; then
540             distro == "Crunchbang"
541         elif [[ "${distro}" == "gentoo" ]]; then
542             grep -q "Funtoo" /etc/gentoo-release &
↪ & distro="Funtoo"
543
544         elif [[ "${distro}" == "mandrake" ]] || [[ "${distro}" == "mandriva" ]]; then
545             grep -q "PCLinuxOS" /etc/${distro}-release &
↪ release && distro="PCLinuxOS"
546
547         elif [[ "${distro}" == "fedora" ]]; then
548             grep -q "Korora" /etc/fedora-release &
↪ & distro="Korora"
549             grep -q "BLAG" /etc/fedora-release &&
↪ distro="BLAG" && distro_more="$(head -n1 /etc/fedora-release)"
550             elif [[ "${distro}" == "oracle" ]]; then
551                 distro_more="$(sed 's/Oracle Linux //' /etc/oracle-release)"
552             elif [[ "${distro}" == "SuSe" ]]; then
553                 distro="openSUSE"
554                 if [ -f /etc/os-release ]; then
555                     if [[ "$(cat /etc/os-release)" =~ "SUSE Linux Enterprise" ]]; then
556                         distro="SUSE Linux Enterprise"
557                         distro_more=$(awk -F= 'VERSION_ID=/' {print $2} /etc/os-release | tr -d '"')
558                     fi
559                     if [[ "${distro_more}" =~ "Tumbleweed" ]]; then
560                         distro_more="Tumbleweed"
561                     fi
562                     elif [[ "${distro}" == "redstar" ]]; then
563                         distro_more=$(grep -o '[0-9.]' /etc/redstar-release | tr -d '\n')
564                     elif [[ "${distro}" == "redhat" ]]; then
565                         grep -q "CentOS" /etc/redhat-release &
566                         grep -q "PCLinuxOS" /etc/redhat-release &
567                         if [ "x$(od -t x1 /etc/redhat-release | sed -e 's/^\w*\ *//' | tr '\n' ' ' | grep 'eb b6 89 ec 9d 80 eb b3 84 ')" != "x" ]; then
568                             distro="Red Star OS"
569                             distro_more=$(grep -o '[0-9.]' /etc/redhat-release | tr -d '\n')
570                         fi
571                         distro_release="$(redhat_release | tr -d '\n')"
572                         distro_codename="$(redhat_codename | tr -d '\n')"
573                     fi
574                 fi
575                 if [[ "${distro}" == "Unknown" ]]; then
576                     if [[ "${OSTYPE}" == "linux-gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then

```

(continues on next page)

(continued from previous page)

```

577         if [ -f /etc/debian_version ]; then
578             if [ -f /etc/issue ]; then
579                 if grep -q "gNewSense" /etc/
↪ issue ; then
580                     distro="gNewSense"
581                 elif grep -q "^KDE neon" /etc/
↪ issue ; then
582                     distro="KDE neon"
583                     distro_more="$(cut -d
↪ ' ' -f3 /etc/issue)"
584                 else
585                     distro="Debian"
586                 fi
587             fi
588             if grep -q "Kali" /etc/debian_version_
↪ ; then
589                 distro="Kali Linux"
590             fi
591         elif [ -f /etc/NIXOS ]; then distro="NixOS"
592         elif [ -f /etc/dragora-version ]; then
593             distro="Dragora"
594             distro_more="$(cut -d, -f1 /etc/
↪ dragora-version)"
595         elif [ -f /etc/slackware-version ]; then_
596         elif [ -f /usr/share/doc/tc/release.txt ];_
597             distro="TinyCore"
598             distro_more="$(cat /usr/share/doc/tc/
↪ release.txt)"
599         elif [ -f /etc/sabayon-edition ]; then distro=
↪ "Sabayon"
600         fi
601     else
602         if [[ -x /usr/bin/sw_vers ]] && /usr/bin/sw_
↪ vers | grep -i "Mac OS X" >/dev/null; then
603             distro="Mac OS X"
604         elif [[ -f /var/run/dmesg.boot ]]; then
605             distro=$(awk 'BEGIN {
606                 distro = "Unknown"
607             }
608             {
609                 if ($0 ~ /DragonFly/) {
610                     distro = "DragonFlyBSD"
611                     exit
612                 }
613                 else if ($0 ~ /FreeBSD/) {
614                     distro = "FreeBSD"
615                     exit
616                 }
617                 else if ($0 ~ /NetBSD/) {
618                     distro = "NetBSD"
619                     exit
620                 }
621                 else if ($0 ~ /OpenBSD/) {
622                     distro = "OpenBSD"

```

(continues on next page)

(continued from previous page)

```

623                                     exit
624                                     }
625                                     } END {
626                                         print distro
627                                     }' /var/run/dmesg.boot)
628
629                                     fi
630
631                                     fi
632
633                                     if [[ "${distro}" == "Unknown" ]] && [[ "${OSTYPE}" == "linux-
634 ↪gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then
635                                         if [[ -f /etc/issue ]]; then
636                                             distro=$(awk 'BEGIN {
637                                                 distro = "Unknown"
638
639 ↪"/) {
640
641                                                 distro = "Hyperbola GNU/Linux-
642 ↪libre"
643
644                                                 exit
645                                         }
646                                         else if ($0 ~ /"LinuxDeepin"/) {
647                                             distro = "LinuxDeepin"
648                                             exit
649                                         }
650                                         else if ($0 ~ /"Obarun"/) {
651                                             distro = "Obarun"
652                                             exit
653                                         }
654                                         else if ($0 ~ /"Parabola GNU/Linux-
655 ↪libre"/) {
656
657                                             distro = "Parabola GNU/Linux-
658 ↪libre"
659
660                                             exit
661                                         }
662                                         else if ($0 ~ /"Solus"/) {
663                                             distro = "Solus"
664                                             exit
665                                         }
666                                         else if ($0 ~ /"ALDOS"/) {
667                                             distro = "ALDOS"
668                                             exit
669                                         }
670                                     } END {
671                                         print distro
672                                     }' /etc/issue)
673
674                                     fi
675
676                                     fi
677
678                                     if [[ "${distro}" == "Unknown" ]] && [[ "${OSTYPE}" == "linux-
679 ↪gnu" || "${OSTYPE}" == "linux" || "${OSTYPE}" == "gnu" ]]; then
680                                         if [[ -f /etc/system-release ]]; then
681                                             if grep -q "Scientific Linux" /etc/system-
682 ↪release; then
683
684                                                 distro="Scientific Linux"
685                                             elif grep -q "Oracle Linux" /etc/system-
686 ↪release; then

```

(continues on next page)

(continued from previous page)

```

673             distro="Oracle Linux"
674         fi
675         elif [[ -f /etc/lsb-release ]]; then
676             if grep -q "CHROMEOS_RELEASE_NAME" /etc/lsb-
↪ release; then
677                 distro="$(awk -F=' ' /^CHROMEOS_
↪ RELEASE_NAME=/ {print $2}' /etc/lsb-release)"
678                 distro_more="$(awk -F=' ' /^CHROMEOS_
↪ RELEASE_VERSION=/ {print $2}' /etc/lsb-release)"
679             fi
680         fi
681     fi
682 fi
683
684 if [[ -n ${distro_more} ]]; then
685     distro_more="${distro} ${distro_more}"
686 fi
687
688 if [[ "${distro}" != "Haiku" ]]; then
689     if [[ ${BASH_VERSINFO[0]} -ge 4 ]]; then
690         if [[ ${BASH_VERSINFO[0]} -eq 4 && ${BASH_VERSINFO[1]} -gt 1_
↪ ]] || [[ ${BASH_VERSINFO[0]} -gt 4 ]]; then
691             distro=${distro,,}
692         else
693             distro="$(tr '[:upper:]' '[:lower:]' <<< ${distro})"
694         fi
695     else
696         distro="$(tr '[:upper:]' '[:lower:]' <<< ${distro})"
697     fi
698 fi
699
700 case $distro in
701     aldos) distro="ALDOS";;
702     alpine) distro="Alpine Linux" ;;
703     amzn|amazon|amazon*linux)
704         distro="AmazonLinux"
705         if [ "$(grep VERSION_ID /etc/os-release | awk -F '=' '{print
↪ $2}')" = "2" ]; then
706             distro_release="2"
707         elif [ "${amazonlinux_release_version}" != "2" ]; then
708             distro_release="1"
709         else
710             distro_release="unknown"
711         fi
712         ;;
713     antergos) distro="Antergos" ;;
714     arch*linux*old) distro="Arch Linux - Old" ;;
715     arch|arch*linux) distro="Arch Linux" ;;
716     artix|artix*linux) distro="Artix Linux" ;;
717     blackpantheros|black*panther*) distro="blackPanther OS" ;;
718     blag) distro="BLAG" ;;
719     bunsenlabs) distro="BunsenLabs" ;;
720     centos)
721         distro="CentOS"
722         distro_release="${redhat_release_version}"
723         distro_codename="${redhat_codename_version}"

```

(continues on next page)

(continued from previous page)

```

725         ;;
726     chakra) distro="Chakra" ;;
727     chapeau) distro="Chapeau" ;;
728     chrome*|chromium*) distro="Chrome OS" ;;
729     crunchbang) distro="CrunchBang" ;;
730     crux) distro="CRUX" ;;
731     cygwin) distro="Cygwin" ;;
732     debian) distro="Debian" ;;
733     devuan) distro="Devuan" ;;
734     deepin) distro="Deepin" ;;
735     desaos) distro="DesaOS" ;;
736     dragonflybsd) distro="DragonFlyBSD" ;;
737     dragora) distro="Dragora" ;;
738     elementary|'elementary os') distro="elementary OS";;
739     evolveos) distro="Evolve OS" ;;
740     exherbo|exherbo*linux) distro="Exherbo" ;;
741     fedora)
742         distro="Fedora"
743         distro_release="$(fedora_release_version)"
744         distro_codename="$(fedora_codename_version)"
745         ;;
746     freebsd) distro="FreeBSD" ;;
747     freebsd*old) distro="FreeBSD - Old" ;;
748     frugalware) distro="Frugalware" ;;
749     fuduntu) distro="Fuduntu" ;;
750     funtoo) distro="Funtoo" ;;
751     fux) distro="Fux" ;;
752     gentoo) distro="Gentoo" ;;
753     gnewsense) distro="gNewSense" ;;
754     guixsd) distro="GuixSD" ;;
755     haiku) distro="Haiku" ;;
756     hyperbolagnu|hyperbolagnu/linux-libre|'hyperbola gnu/linux-libre
↪'|hyperbola) distro="Hyperbola GNU/Linux-libre" ;;
757     kali*linux) distro="Kali Linux" ;;
758     kaos) distro="KaOS";;
759     kde*neon|neon) distro="KDE neon" ;;
760     kogaion) distro="Kogaion" ;;
761     korora) distro="Korora" ;;
762     linuxdeepin) distro="LinuxDeepin" ;;
763     lmde) distro="LMDE" ;;
764     logos) distro="Logos" ;;
765     lunar|lunar*linux) distro="Lunar Linux";;
766     mac*os*x|os*x) distro="Mac OS X" ;;
767     manjaro) distro="Manjaro" ;;
768     mageia) distro="Mageia" ;;
769     mandrake) distro="Mandrake" ;;
770     mandriva) distro="Mandriva" ;;
771     mer) distro="Mer" ;;
772     mint|linux*mint) distro="Mint" ;;
773     msys|msys2) distro="Msys" ;;
774     netbsd) distro="NetBSD" ;;
775     netrunner) distro="Netrunner" ;;
776     nix|nix*os) distro="NixOS" ;;
777     obarun) distro="Obarun" ;;
778     obrevenge) distro="OBRevenge" ;;
779     ol|oracle*linux) distro="Oracle Linux" ;;
780     openbsd) distro="OpenBSD" ;;

```

(continues on next page)

(continued from previous page)

```

781     opensuse) distro="openSUSE" ;;
782     parabolagnu|parabolagnu/linux-libre|'parabola gnu/linux-libre
↪'|parabola) distro="Parabola GNU/Linux-libre" ;;
783     pardus) distro="Pardus" ;;
784     parrot|parrot*security) distro="Parrot Security" ;;
785     pclinuxos|pclos) distro="PCLinuxOS" ;;
786     peppermint) distro="Peppermint" ;;
787     proxmox|proxmox*ve) distro="Proxmox VE" ;;
788     qubes) distro="Qubes OS" ;;
789     raspbian) distro="Raspbian" ;;
790     red*hat*|rhel)
791         distro="Redhat"
792         distro_release="$(redhat_release_version)"
793         distro_codename="$(redhat_codename_version)"
794         ;;
795     rosa) distro="ROSA" ;;
796     red*star|red*star*os) distro="Red Star OS" ;;
797     sabayon) distro="Sabayon" ;;
798     sailfish|sailfish*os) distro="SailfishOS" ;;
799     siduction) distro="Siduction" ;;
800     slackware) distro="Slackware" ;;
801     smgl|source*mage|source*mage*gnu*linux) distro="Source Mage GNU/Linux
↪" ;;
802
803     solus) distro="Solus" ;;
804     sparky|sparky*linux) distro="SparkyLinux" ;;
805     steam|steam*os) distro="SteamOS" ;;
806     suse*linux*enterprise) distro="SUSE Linux Enterprise" ;;
807     swagarch) distro="SwagArch" ;;
808     tinycore|tinycore*linux) distro="TinyCore" ;;
809     trisquel) distro="Trisquel" ;;
810     grombyangos) distro="GrombyangOS" ;;
811     ubuntu)
812         distro="Ubuntu"
813         if grep -q 'Microsoft' /proc/version 2>/dev/null || \
814             grep -q 'Microsoft' /proc/sys/kernel/osrelease 2>/dev/null
815         then
816             uow=$(echo -e "$(getColor 'yellow') [Ubuntu on_
↪Windows 10])")
817         fi
818         ;;
819     viperr) distro="Viperr" ;;
820     void*linux) distro="Void Linux" ;;
821     zorin*) distro="Zorin OS" ;;
822
823     esac
824     if [ "$format" = "json" ]; then
825         echo "{\"Major\": \"${distro}\", \"Release\": \"${distro_release}\", \
↪\"Codename\": \"${distro_codename}\"}"
826     else
827         echo "${distro} ${distro_release} ${distro_codename}"
828     fi
829 }
830
831 # call main
832 detectdistro

```

Back to [Bash Module Index](#)

## 24.7 std\_functions.sh

Back to *Bash Module Index*

```
1  #!/usr/bin/env bash
2
3  #-----
4  #
5  #   Note:  to be used with dependent modules
6  #
7  #       - colors.sh
8  #       - exitcodes.sh
9  #
10 #       Dependencies must be sourced from the same calling script
11 #       as this std_functions.sh
12 #
13 #   Global Variables provided by the Caller:
14 #       - LOG_FILE      # std_logger writes to this file
15 #       - QUIET         # Value = "true" to supress stdout from these reference_
↪functions
16 #
17 #-----
18
19 # pkg reported in logs will be the basename of the caller
20 pkg=$(basename $0 2>/dev/null)
21 pkg_root="$(echo $pkg | awk -F '.' '{print $1}')" # pkg without file extention
22 pkg_path=$(cd $(dirname $0 2>/dev/null); pwd -P)
23 host=$(hostname)
24 system=$(uname)
25
26 # this file
27 LIB_VERSION="2.9.8"
28
29 if [ ! $pkg ] || [ ! $pkg_path ]; then
30     echo -e "\n[std_functions.sh]: pkg and pkg_path errors - both are null"
31 fi
32
33
34 function array2json(){
35     ## converts associative array to single-level (no nested keys) json file output ##
36     #
37     #   Caller syntax:
38     #       $ array2json config_dict $config_path/configuration_file
39     #
40     #   where:
41     #       $ declare -A config_dict          # config_dict is assoc array, declared in_
↪main script
42     #
43     local -n array_dict=$1          # local assoc array must use -n opt
```

(continues on next page)



(continued from previous page)

```

44     local output_file=$2          # location
45     local ct                      # counter
46     local max_keys                # num keys in array
47     #
48     echo -e "{" > $output_file
49     ct=1
50     max_keys=${#array_dict[@]}
51     for key in "${!array_dict[@]}; do
52         if [ $ct == $max_keys ]; then
53             # last key, no comma
54             echo "\"${key}\": \"${array_dict[${key}]}\",\" | indent04 >> $output_file
55         else
56             echo "\"${key}\": \"${array_dict[${key}]}\",\" | indent04 >> $output_file
57         fi
58         ct=$(( $ct + 1 ))
59     done
60     echo -e "}" >> $output_file
61     #
62     # <-- end function array2json -->
63 }
64
65
66 function authenticated(){
67     ## validates authentication using iam user or role ##
68     local profilename="$1"
69     local response
70     local awscli=$(which aws)
71     #
72     response=$(awscli sts get-caller-identity --profile $profilename 2>&1)
73     if [ "$(echo $response | grep Invalid)" ]; then
74         std_message "The IAM profile provided ($profilename) failed to authenticate_
↳to AWS. Exit (Code $E_AUTH)" "AUTH"
75         return 1
76     elif [ "$(echo $response | grep found)" ]; then
77         std_message "The IAM user or role ($profilename) cannot be found in your_
↳local awscli config. Exit (Code $E_BADARG)" "AUTH"
78         return 1
79     elif [ "$(echo $response | grep Expired)" ]; then
80         std_message "The sts temporary credentials for the role provided (
↳$profilename) have expired. Exit (Code $E_AUTH)" "INFO"
81         return 1
82     else
83         return 0
84     fi
85 }
86
87
88 function binary_depcheck(){
89     ## validate binary dependencies installed
90     local check_list=( "$@" )
91     local msg
92     #
93     for prog in "${check_list[@]}; do
94         if ! type "$prog" > /dev/null 2>&1; then
95             msg="$prog is required and not found in the PATH. Aborting (code $E_
↳DEPENDENCY)"
96             std_error_exit "$msg" $E_DEPENDENCY

```

(continues on next page)

(continued from previous page)

```

97     fi
98 done
99 #
100 # <-- end function binary_depcheck -->
101 }
102
103
104 function convert_time(){
105     # time format conversion (http://stackoverflow.com/users/1030675/choroba)
106     num=$1
107     min=0
108     hour=0
109     day=0
110     if ( (num>59) ); then
111         ( (sec=num%60) )
112         ( (num=num/60) )
113         if ( (num>59) ); then
114             ( (min=num%60) )
115             ( (num=num/60) )
116             if ( (num>23) ); then
117                 ( (hour=num%24) )
118                 ( (day=num/24) )
119             else
120                 ( (hour=num) )
121             fi
122         else
123             ( (min=num) )
124         fi
125     else
126         ( (sec=num) )
127     fi
128     echo "$day"d, "$hour"h, "$min"m
129     #
130     # <-- end function convert_time -->
131     #
132 }
133
134
135 function convert_time_months(){
136     # time format conversion (http://stackoverflow.com/users/1030675/choroba)
137     num=$1
138     min=0
139     hour=0
140     day=0
141     mo=0
142     if ( (num>59) ); then
143         ( (sec=num%60) )
144         ( (num=num/60) )
145         if ( (num>59) ); then
146             ( (min=num%60) )
147             ( (num=num/60) )
148             if ( (num>23) ); then
149                 ( (hour=num%24) )
150                 ( (day=num/24) )
151                 ( (num=num/24) )
152                 if ( (num>30) ); then
153                     ( (day=num%31) )

```

(continues on next page)

(continued from previous page)

```

154         ((mo=num/30))
155     else
156         ((day=num))
157     fi
158     else
159         ((hour=num))
160     fi
161     else
162         ((min=num))
163     fi
164 else
165     ((sec=num))
166 fi
167 if (( $mo > 0 )); then
168     echo -e "$mo"m, "$day"d
169 else
170     echo -e "$day"d, "$hour"h, "$min"m
171 fi
172 #
173 # <-- end function convert_time -->
174 #
175 }
176
177
178 function delay_spinner() {
179     ##
180     ## Usage:
181     ##
182     ##     $ long-running-command &
183     ##     $ delay_spinner " Please wait msg..."
184     ##
185     ## Spinner exists when long-running-command completes
186     ##
187     local PROGRESSTXT
188     if [ ! "$1" ]; then
189         PROGRESSTXT=" Please wait..."
190     else
191         PROGRESSTXT="$1"
192     fi
193     # visual progress marker function
194     # http://stackoverflow.com/users/2869509/wizurd
195     # vars
196     local pid=$!
197     local delay=0.1
198     local spinstr='|/-'
199     echo -e "\n\n"
200     while [ "$$(ps a | awk '{print $1}' | grep $pid)" ]; do
201         local temp=${spinstr#?}
202         printf "\r$PROGRESSTXT[%c] " "$spinstr"
203         local spinstr=$temp${spinstr%"$temp"}
204         sleep $delay
205         printf "\b\b\b\b\b\b"
206     done
207     printf -- '\n\n'
208     #
209     # <-- end function ec2cli_spinner -->
210     #

```

(continues on next page)

(continued from previous page)

```

211 }
212
213
214 function environment_info() {
215     local prefix=$1
216     local dep=$2
217     local log_file="$3"
218     local version_info
219     local awscli_ver
220     local boto_ver
221     local python_ver
222     #
223     version_info=$(aws --version 2>&1)
224     awscli_ver=$(echo $version_info | awk '{print $1}')
225     boto_ver=$(echo $version_info | awk '{print $4}')
226     python_ver=$(echo $version_info | awk '{print $2}')
227     #
228     if [[ $dep == "aws" ]]; then
229         std_logger "awscli version detected: $awscli_ver" $prefix $log_file
230         std_logger "Python runtime detected: $python_ver" $prefix $log_file
231         std_logger "Kernel detected: $(echo $version_info | awk '{print $3}')"
232         std_logger "boto library detected: $boto_ver" $prefix $log_file
233
234         elif [[ $dep == "awscli" ]]; then
235             std_message "awscli version detected: ${accent}${BOLD}$awscli_ver${UNBOLD}$
236             std_message "boto library detected: ${accent}${BOLD}$boto_ver${UNBOLD}${reset}
237             std_message "Python runtime detected: ${accent}${BOLD}$python_ver${UNBOLD}$
238             std_message "Kernel detected: ${title}$(echo $version_info | awk '{print $3}')"
239             std_message "JSON parser detected: ${title}$(echo $version_info)${reset}"
240             std_logger "Detected: $($prog --version | head -1)" $prefix $log_file
241
242         elif [[ $dep == "jq" ]]; then
243             version_info=$(jq --version 2>&1)
244             std_message "JSON parser detected: ${title}$(echo $version_info)${reset}"
245
246         else
247             std_logger "Detected: $($prog --version | head -1)" $prefix $log_file
248         fi
249     #
250     # <<-- end function environment_info -->>
251 }
252
253
254 function is_installed() {
255     ##
256     ## validate if binary previously installed ##
257     ##
258     local binary="$1"
259     local location=$(which $binary 2>/dev/null)
260
261     if [ $location ]; then

```

(continues on next page)

(continued from previous page)

```

262     std_message "$binary is installed: $location" "INFO" $LOG_FILE
263     return 0
264
265 else
266
267     return 1
268
269 fi
270 #
271 #<-- end function is_installed -->
272 }
273
274
275
276 function is_float(){
277     ##
278     ## Checks type for floating point number
279     ## see is_number integer type checking
280     ##
281     local num="$1"
282     local regex='^[0-9]+[.][0-9]+?${'
283
284     if [[ $num =~ $regex ]] ; then
285
286         # int or float
287         return 0
288
289     fi
290
291     return 1          # not a floating point number
292     #
293     #<-- end function is_float -->
294 }
295
296
297 function is_int(){
298     ##
299     ## see is_float for decimal type checking ##
300     ##
301     local num="$1"
302     local regex='^[0-9]+$'
303
304     if [[ $num =~ $regex ]] ; then
305
306         # int or float
307         return 0
308
309     fi
310
311     return 1          # not an integer
312     #
313     #<-- end function is_int -->
314 }
315
316
317 function is_number(){
318     ##

```

(continues on next page)

(continued from previous page)

```

319     ## type checking; any, int or decimal type ##
320     ##
321     local num="$1"
322     local regex='^[0-9]+([.][0-9]+)?$'
323
324     if [[ $num =~ $regex ]] ; then
325
326         # int or float
327         return 0
328
329     fi
330
331     return 1          # not a number (is alpha character)
332     #
333     #<-- end function is_number -->
334 }
335
336
337 function linux_distro(){
338     ##
339     ## determine linux os distribution ##
340     ##
341     local os_major
342     local os_release
343     local os_codename
344     declare -a distro_info
345
346     if [ "$(which lsb_release)" ]; then
347
348         distro_info=( $(lsb_release -sirc) )
349
350         if [[ ${#distro_detect[@]} -eq 3 ]]; then
351             os_major=${distro_info[0]}
352             os_release=${distro_info[1]}
353             os_codename=${distro_info[2]}
354         fi
355
356     else
357
358         ## AMAZON Linux ##
359         if [ "$(grep -i amazon /etc/os-release | head -n 1)" ]; then
360
361             os_major="amazonlinux"
362             if [ "$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')" = "2"
↪ ' ] ; then
363                 os_release="$(grep VERSION /etc/os-release | grep -v VERSION_ID | awk
↪ -F '=' '{print $2}')"
364                 os_release=$(echo $os_release | cut -c 2-15 | rev | cut -c 2-15 | rev)
365             elif [ "$(grep VERSION_ID /etc/os-release | awk -F '=' '{print $2}')" = '
↪ "1" ] ; then
366                 os_release="$(grep VERSION /etc/os-release | grep -v VERSION_ID | awk
↪ -F '=' '{print $2}')"
367                 os_release=$(echo $os_release | cut -c 2-15 | rev | cut -c 2-15 | rev)
368             else os_release="unknown"; fi
369
370         ## REDHAT Linux ##
371         elif [ "$(grep -i redhat /etc/os-release | head -n 1)" ]; then

```

(continues on next page)

(continued from previous page)

```

372
373     os_major="redhat"
374     os_release="future"
375
376     ## UBUNTU, ubuntu variants ##
377     elif [ "$(grep -i ubuntu /etc/os-release)" ]; then
378
379         os_major="ubuntu"
380         if [ "$(grep -i mint /etc/os-release | head -n1)" ]; then
381             os_release="linuxmint"
382         elif [ "$(grep -i ubuntu_codename /etc/os-release | awk -F '=' '{print $2}
↪')" ]; then
383             os_release="$(grep -i ubuntu_codename /etc/os-release | awk -F '=' '
↪{print $2}')"
384         else
385             os_release="unknown"; fi
386
387     ## distribution not determined ##
388     else
389
390         os_major="unknown"; os_release="unknown"
391
392     fi
393
394 fi
395
396 # set distribution type in environment
397 export OS_DISTRO="$os_major"
398 std_logger "Operating system identified as Major Version: $os_major, Minor_
↪Version: $os_release" "INFO" $LOG_FILE
399
400 # return major, minor disto versions
401 echo -e "$os_major $os_release $os_codename"
402 #
403 # <--- end function linux_distro --->
404 }
405
406
407 function pkg_info(){
408     ##
409     ## displays information about this library module
410     ##
411     ## - dependent module colors.sh is located always adjacent
412     ## - sourcing of dep modules must occur after local var to avoid overwrite
413     ## of variable values in this module
414     ##
415     local version=$LIB_VERSION
416     source $pkg_path/colors.sh
417     bd=$(echo -e ${bold})
418     act=$(echo -e ${a_orange})
419     rst=$(echo -e ${reset})
420
421     # generate list of functions
422     printf -- '%s\n' "$(declare -F | awk '{print $3}')" > /tmp/.functions
423     sum=$(cat /tmp/.functions | wc -l)
424
425     # construct, display help msg output

```

(continues on next page)

(continued from previous page)

```

426 cat <<EOM
427
428
429 ${title}Bashtools Library${rst}: Standard Functions
430
431 Module Name:      ${cyan}$pkg${rst}
432 Module Version:   ${act}$version${rst}
433
434
435 Module Contains $sum Functions:
436
437 EOM
438 # display list of function names in this module
439 for l in $(cat /tmp/.functions); do
440     printf -- '\t%s %s\n' "-" "$l"
441 done
442 printf -- '\n'
443 rm /tmp/.functions
444 #
445 # <<-- end function pkg_info -->>
446 }
447
448
449 function print_header(){
450     ##
451     ## print formatted report header ##
452     ##
453     local title="$1"
454     local width="$2"
455     local reportfile="$3"
456     #
457     #if (( $(tput cols) > 480 )); then
458     #     printf "%-10s %s\n" $(echo -e ${frame}) "$(($width - 1))" ' ' | tr ' ' _ |
↪indent02 > $reportfile
459     #else
460     printf "%-10s %s" $(echo -e ${frame}) "$(($width - 1))" ' ' | tr ' ' _ |
↪indent02 > $reportfile
461     #fi
462     echo -e "${bodytext}" >> $reportfile
463     echo -ne ${title} >> $reportfile
464     echo -e "${frame}" >> $reportfile
465     printf '%s' "$width" ' ' | tr ' ' _ | indent02 >> $reportfile
466     echo -e "${bodytext}" >> $reportfile
467     #
468     # <<--- end function print_header -->>
469 }
470
471
472 function print_footer(){
473     ##
474     ## print formatted report footer ##
475     ##
476     local footer="$1"
477     local width="$2"
478
479     printf "%-10s %s\n" $(echo -e ${frame}) "$(($width - 1))" ' ' | tr ' ' _ |
↪indent02

```

(continues on next page)



(continued from previous page)

```

480     echo -e "${bodytext}"
481     echo -ne $footer | indent20
482     echo -e "${frame}"
483     printf '%s\n' "$width" '' | tr ' ' _ | indent02
484     echo -e "${bodytext}"
485     #
486     # <--- end function print_footer -->
487 }
488
489
490 function print_separator() {
491     ##
492     ## prints single bar separator of width ##
493     ##
494
495     local width="$1"
496
497     echo -e "${frame}"
498     printf "%-10s %s" $(echo -e ${frame}) "$(($width - 1))" '' | tr ' ' _ | indent02
499     echo -e "${bodytext}\n"
500     #
501     # <--- end function linux_distro -->
502 }
503
504
505 function python_version_depcheck() {
506     ##
507     ## dependency check for a specific version of python binary ##
508     ##
509     local version
510     local version_min="$1"
511     local version_max="$2"
512     local msg
513
514     local_bin=$(which python3)
515     # determine binary version
516     version=$(($local_bin 2>&1 --version | awk '{print $2}' | cut -c 1-3)
517
518     if (( $(echo "$version > $version_max" | bc -l) )) || (( $(echo "$version <
↪$version_min" | bc -l) )); then
519
520         msg="python version $version detected - must be > $version_min, but <
↪$version_max"
521         std_error_exit "$msg" $E_DEPENDENCY
522
523     fi
524     #
525     # <--- end function python_depcheck -->
526 }
527
528
529 function progress_dots() {
530     ##
531     ## Usage:
532     ##
533     ##     $ long-running-command &
534     ##     $ progress_dots --text "Process XYZ Starting" --end " End xyz"

```

(continues on next page)

(continued from previous page)

```

535  ##
536  ##      Exists when long-running-command completes
537  ##
538  ##  Quiet Mode:
539  ##      if QUIET = true, runs timer, no stdout printing
540  ##
541  ##  Dependencies:
542  ##      - requires colors.sh (source of indent function)
543  ##
544  local text
545  local endmsg
546  local fast
547  local width=$(tput cols)
548  local stop=$(( $width / 4 ))
549  local pid=$!
550  local delay="0.1"
551  local counter="0"
552  local len
553
554  while [ $# -gt 0 ]; do
555      case $1 in
556          -e | --end)
557              endmsg="$2"; shift 2
558              ;;
559          -f | --fast)
560              fast="$2"; shift 2
561              ;;
562          -t | --text)
563              text="$2"; shift 2
564              ;;
565          *)
566              ;;
567      esac
568  done
569
570  if [ ! "$text" ]; then text="Please wait"; fi
571  if [ ! "$endmsg" ]; then endmsg="done."; fi
572
573  # print fast dots if short process
574  if [ "$fast" = "true" ]; then delay="$(( 1/15 ))"; fi
575
576  # min width of dot pattern
577  if [ $stop -lt "80" ]; then stop="80"; fi
578
579  len=${#text}                # length of text msg, chars
580  stopmarker=$stop           # stop column when not title row
581  titlestop=$(( $stop - $len )) # stop column on text msg row
582
583  # title
584  if [[ ! $QUIET ]]; then
585      printf -- '\n\n%s' "$text" | indent04
586  fi
587
588  # output progress dots
589  while [ "$(ps a | awk '{print $1}' | grep $pid)" ]; do
590
591      if [ "$counter" = "0" ]; then

```

(continues on next page)

(continued from previous page)

```

592     printf -- '%s' "."
593     stop=$titlestop
594
595     elif [ $counter -ge $stop ]; then
596         printf -- '\n%s' "." | indent04
597         counter="0"
598         stop=$stopmarker
599
600     elif [[ ! $QUIET ]]; then
601         printf -- '%s' "."
602     fi
603
604     sleep $delay
605     counter=$(( $counter + 1 ))
606
607 done
608
609 printf -- "  ${endmsg}\n\n"
610 #
611 # <-- end function ec2cli_spinner -->
612 #
613 }
614
615
616 function python_module_depcheck() {
617     ##
618     ## validate python library dependencies
619     ##
620     local check_list=( "$@" )
621     local msg
622
623     for module in "${check_list[@]}; do
624
625         exitcode=$(python3 -c "import $module" > /dev/null 2>&1; echo $? )
626
627         if [[ $exitcode == "1" ]]; then
628             # module not imported, not found
629             msg="$module is a required python library. Aborting (code $E_DEPENDENCY)"
630             std_error_exit "$msg" $E_DEPENDENCY
631         fi
632
633     done
634     #
635     # <-- end function python_module_depcheck -->
636 }
637
638
639 function std_logger() {
640     ##
641     ## Summary:
642     ##
643     ##     std_logger is usually invoked from std_message; ie, all messages
644     ##     to stdout are also logged in this function to the log file.
645     ##
646     ## Args:
647     ##     - msg:         body of the log message text
648     ##

```

(continues on next page)

(continued from previous page)

```

649  ##      - prefix:  INFO, DEBUG, etc. Note: WARN is handled by std_warn
650  ##                  function
651  ##
652  ##      - log_file: The file to which log messages should be written
653  ##
654  ##      - version: Populated if version module exists in
655  ##                  pkg_lib.__version__ sourced from within the
656  ##                  version module
657  ##
658  local msg="$1"
659  local prefix="$2"
660  local log_file="$3"
661  local rst=$(echo -e ${RESET})
662  local version
663  local strip_ansi="false"
664
665  # set prefix if not provided
666  if [ ! $prefix ]; then prefix="INFO"; fi
667
668  # set version in logger
669  if [ $pkg_lib ] && [ -f $pkg_lib/version.py ]; then
670      source "$pkg_lib/version.py"
671      version=${__version__}
672
673  elif [ "$LIB_VERSION" ]; then
674      version=$LIB_VERSION
675
676  elif [ ! "$LIB_VERSION" ]; then
677      version="1.0.NA"
678
679  fi
680
681  # write out to log
682  if [ ! -f $log_file ]; then
683
684      # create log file
685      touch $log_file
686
687      if [ ! -f $log_file ]; then
688          echo -e "[$prefix]: $pkg ($version): std_logger failure, $log_file path_
↳not writeable"
689          exit $E_DIR
690      fi
691
692  elif [ "$strip_ansi" = "true" ]; then
693
694      echo -e "$(date +%Y-%m-%d %T) $host - $pkg - $version - [$prefix]: $msg$
↳{rst}" | \
695      sed -r "s/\x1B\[([0-9]{1,2}([0-9]{1,2})?)?[m|G|K]//g" >> "$log_file"
696
697  else
698
699      echo -e "$(date +%Y-%m-%d %T) $host - $pkg - $version - [$prefix]: $msg$
↳{rst}" >> "$log_file"
700
701  fi
702  #

```

(continues on next page)

(continued from previous page)

```

703     # <--- end function std_logger -->
704 }
705
706
707 function std_message() {
708     ##
709     ## Caller formats:
710     ##
711     ##   Logging to File | std_message "xyz message" "INFO" "/pathto/log_file"
712     ##
713     ##   No Logging   | std_message "xyz message" "INFO"
714     ##
715     local msg="$1"
716     local prefix="$2"
717     local log_file="$3"
718     local format="$4"
719     local rst=${reset}
720
721     if [ "$4" ]; then format=''; else format='\n'; fi
722
723     if [ "$3" ]; then
724         case "$prefix" in
725             'ok' | 'OK' | 'DONE')
726                 std_logger "$msg" "INFO" "$log_file"
727                 prefix="OK"
728                 ;;
729
730             'INSTALLED' | 'AVAILABLE' | 'NOT-FOUND')
731                 filtered=$(echo $msg | sed 's/[|]/g')
732                 std_logger "$filtered" "INFO" "$log_file"
733                 ;;
734
735             *)
736                 # info log message written to log
737                 std_logger "$msg" "$prefix" "$log_file"
738                 ;;
739         esac
740     fi
741
742     if [[ $QUIET ]]; then return 0; fi
743
744     case "$prefix" in
745         'ok' | 'OK')
746         ↪ echo -e "${format}${yellow}[  $green${BOLD}$prefix${rst}${yellow}  ]${rst}"
747             ;;
748
749         'INSTALLED')
750             echo -e "${format}${green${BOLD}$prefix${rst} | $msg${format}" | indent04
751             ;;
752
753         'AVAILABLE')
754             echo -e "${format}$prefix${rst} | $msg${format}" | indent04
755             ;;
756
757         'FAIL' | 'ERROR' | 'BAD' | 'N/A')
758         ↪ echo -e "${format}${yellow}[  ${red}${BOLD}$prefix${rst}${yellow}  ]${rst}"
759             ↪ $msg${format}" | indent04

```

(continues on next page)

(continued from previous page)

```

759         ;;
760
761         'NOT-FOUND')
762             echo -e "${format}${red}${BOLD}$prefix${rst} | $msg${format}" | indent04
763         ;;
764
765         'WARN')
766             echo -e "${format}${yellow}[ ${yellow}$prefix${rst}${yellow} ]${rst} $msg
↪${format}" | indent04
767         ;;
768
769         *)
770             echo -e "${format}${yellow}[ $cyan$prefix$yellow ]${rst} $msg${format}"
↪| indent04
771         ;;
772     esac
773     return 0
774 #
775 # <-- end function std_message -->
776 }
777
778
779 function std_error() {
780     local msg="$1"
781     std_logger "$msg" "ERROR" $LOG_FILE
782     echo -e "\n${yellow}[ ${red}ERROR${yellow} ]$reset $msg\n" | indent04
783     #
784     # <-- end function std_error -->
785 }
786
787
788 function std_warn() {
789     local msg="$1"
790     local log_file="$2"
791     local pc="$(echo -e ${a_brightyellow2})" # prefix color
792     local rst="$(echo -e ${reset})" # reset code
793
794     if [ $log_file ]; then
795         std_logger "$msg" "WARN" $log_file
796     fi
797
798     if [ "$3" ]; then
799         # there is a second line of the msg, to be printed by the caller
800         echo -e "\n${pv_wgray}[${rst} ${pc}WARN${pv_wgray}]$reset $msg" | indent04
801     else
802         # msg is only 1 line sent by the caller
803         echo -e "\n${pv_wgray}[${rst} ${pc}WARN${pv_wgray}]$reset $msg\n" | indent04
804     fi
805     #
806     # <-- end function std_warn -->
807 }
808
809
810 function std_error_exit() {
811     ##
812     ## standard function presents error msg, automatically
813     ## exits error code

```

(continues on next page)

(continued from previous page)

```

814     ##
815     local msg="$1"
816     local status="$2"
817     std_error "$msg"
818     exit $status
819     #
820     # <-- end function std_warn -->
821 }
822
823
824 function timer(){
825     ## measure total execution runtime ##
826     ##
827     ## Usage:
828     ##
829     ## @ beginning:
830     ## $ START=$(timer)
831     ##
832     ## @ end time:
833     ## $ printf 'Total runtime: %s\n' $(timer $START)
834     ##
835     if [[ $# -eq 0 ]]; then
836
837         echo $(date '+%s')
838
839     else
840         local stime=$1
841         etime=$(date '+%s')
842
843         if [[ -z "$stime" ]]; then stime=$etime; fi
844
845         dt=$((etime - stime))
846         ds=$((dt % 60))
847         dm=$((dt / 60 % 60))
848         dh=$((dt / 3600))
849         printf '%d:%02d:%02d' $dh $dm $ds
850
851     fi
852     #
853     # <-- end function timer -->
854 }
855
856 # print information about this package
857 if [ "$pkg" = "std_functions.sh" ]; then
858     pkg_info
859 fi

```

[Back to \*std\\_functions.sh\*](#)

[Back to \*Bash Module Index\*](#)





---

### Debian Package Creation

---

- *Debian Package Creation Start*
  - *Debian Package Creation Build*
  - *Debian Package Final Contents*
- 

#### 25.1 Debian Package Creation Start

```
$ make builddeb
```

```
[ INFO ]: Host OS identified as Linux Mint

[ OK ]: Extracted package mgr contents for gawk

[ OK ]: Extracted package mgr contents for bc

[ OK ]: Extracted package mgr contents for curl

[ INFO ]: File list generated: ['bc.pkg', 'gawk.pkg', 'curl.pkg']

[ INFO ]: build_deplist.py: Files in list already exist, use --force to recreate. Deplist Complete.
Building Debian package format of buildpy
[ INFO ]: Current version of last build: 1.7.13

[ INFO ]: Version to be used for this build: 1.7.14

[ INFO ]: Generating build structure and artifacts in buildpy-1.7.14_amd64

[ OK ]: Copied: ../buildpython3/packaging/deb/DEBIAN --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/DEBIAN

[ OK ]: Created: ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/bin

[ OK ]: Copied: ../buildpython3/buildpy --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/bin/buildpy

[ OK ]: Created: ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy

[ OK ]: Copied: ../buildpython3/core/std_functions.sh --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/std_functions.sh

[ OK ]: Copied: ../buildpython3/core/uninstall.paths --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/uninstall.paths

[ OK ]: Copied: ../buildpython3/core/colors.sh --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/colors.sh

[ OK ]: Copied: ../buildpython3/core/gawk.pkg --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/gawk.pkg

[ OK ]: Copied: ../buildpython3/core/os_distro.sh --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/os_distro.sh

[ OK ]: Copied: ../buildpython3/core/curl.pkg --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/curl.pkg

[ OK ]: Copied: ../buildpython3/core/version.py --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/version.py

[ OK ]: Copied: ../buildpython3/core/bc.pkg --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/bc.pkg

[ OK ]: Copied: ../buildpython3/core/exitcodes.sh --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/exitcodes.sh

[ OK ]: Created: ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/etc/bash_completion.d
```

[Back to \*Debian Package Creation\* Index](#)

---

## 25.2 Debian Package Creation Build

```
[ OK ]: Copied: ../buildpython3/core/exitcodes.sh --> ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/exitcodes.sh

[ OK ]: Created: ../buildpython3/packaging/deb/buildpy-1.7.14_amd64/etc/bash_completion.d

[ INFO ]: Bin buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/bin/buildpy successfully updated.

[ INFO ]: Control file buildpython3/packaging/deb/buildpy-1.7.14_amd64/DEBIAN/control successfully updated.

[ INFO ]: Module buildpython3/packaging/deb/buildpy-1.7.14_amd64/usr/local/lib/buildpy/version.py successfully updated.

[ INFO ]: Building buildpy-1.7.14_amd64...

[ INFO ]: dpkg-deb: building package 'buildpy' in 'buildpy-1.7.14_amd64.deb'.

[ INFO ]: postbuild: Module buildpython3/core/version.py successfully updated.
```

[Back to \*Debian Package Creation\* Index](#)

---

## 25.3 Debian Package Final Contents

```

Package Contents: buildpy-1.7.14_amd64.deb

```

Permission	Owner/Group	ctime	File
drwxrwxr-x	blake/blake0	2019-03-31	./
drwxrwxr-x	blake/blake0	2019-03-31	./usr/
drwxrwxr-x	blake/blake0	2019-03-31	./usr/local/
drwxrwxr-x	blake/blake0	2019-03-31	./usr/local/lib/
drwxrwxr-x	blake/blake0	2019-03-31	./usr/local/lib/buildpy/
-rw-rw-r--	blake/blake22161	2019-03-24	./usr/local/lib/buildpy/std_functions.sh
-rw-rw-r--	blake/blake75	2019-03-24	./usr/local/lib/buildpy/uninstall.paths
-rw-rw-r--	blake/blake7922	2019-03-24	./usr/local/lib/buildpy/colors.sh
-rw-rw-r--	blake/blake805	2019-03-24	./usr/local/lib/buildpy/gawk.pkg
-rw-rw-r--	blake/blake26461	2019-03-24	./usr/local/lib/buildpy/os_distro.sh
-rw-rw-r--	blake/blake691	2019-03-24	./usr/local/lib/buildpy/curl.pkg
-rw-rw-r--	blake/blake21	2019-03-31	./usr/local/lib/buildpy/version.py
-rw-rw-r--	blake/blake460	2019-03-24	./usr/local/lib/buildpy/bc.pkg
-rw-rw-r--	blake/blake1063	2019-03-24	./usr/local/lib/buildpy/exitcodes.sh
drwxrwxr-x	blake/blake0	2019-03-31	./usr/local/bin/
drwxrwxr-x	blake/blake84381	2019-03-31	./usr/local/bin/buildpy
drwxrwxr-x	blake/blake0	2019-03-31	./etc/
drwxrwxr-x	blake/blake0	2019-03-31	./etc/bash_completion.d/
-rw-rw-r--	blake/blake20677	2019-03-24	./etc/bash_completion.d/buildpy-completion.bash

```

[ INFO ]: buildpy build complete

```

[Back to \*Debian Package Creation\* Index](#)

[Back to Table Of Contents](#)



## CHAPTER 26

---

### RPM Package Creation

---

- *RPM Package Assembly*
  - *RPM Package Build*
  - *Docker RPM Package Build*
  - *RPM Package Final Contents*
-

## 26.1 RPM Package Assembly

```
[ INFO ]: Current version of last build: 1.7.13

[ INFO ]: Version to be used for this build: 1.7.14

[ INFO ]: Assembling build directory artifacts in buildpy-1.7

[ OK ]: Created: /tmp/build/buildpy-1.7

[ OK ]: Copied: /home/blake/git/linux/buildpython3/buildpy --> /tmp/build/buildpy-1.7/buildpy

[ OK ]: Copied: ../buildpython3/core/std_functions.sh --> /tmp/build/buildpy-1.7/std_functions.sh

[ OK ]: Copied: ../buildpython3/core/uninstall.paths --> /tmp/build/buildpy-1.7/uninstall.paths

[ OK ]: Copied: ../buildpython3/core/colors.sh --> /tmp/build/buildpy-1.7/colors.sh

[ OK ]: Copied: ../buildpython3/core/gawk.pkg --> /tmp/build/buildpy-1.7/gawk.pkg

[ OK ]: Copied: ../buildpython3/core/os_distro.sh --> /tmp/build/buildpy-1.7/os_distro.sh

[ OK ]: Copied: ../buildpython3/core/curl.pkg --> /tmp/build/buildpy-1.7/curl.pkg

[ OK ]: Copied: ../buildpython3/core/version.py --> /tmp/build/buildpy-1.7/version.py

[ OK ]: Copied: ../buildpython3/core/bc.pkg --> /tmp/build/buildpy-1.7/bc.pkg

[ OK ]: Copied: ../buildpython3/core/exitcodes.sh --> /tmp/build/buildpy-1.7/exitcodes.sh

[ OK ]: Copied: /home/blake/git/linux/buildpython3/packaging/rpm/buildpy.spec --> /tmp/build/buildpy.spec

[ OK ]: Copied: ../buildpython3/bash/buildpy-completion.bash/buildpy-completion.bash --> /tmp/build/buildpy-1.7/buildpy-completion.bash/buildpy-completion.bash

[ OK ]: Copied: ../buildpython3/packaging/rpm/docker-buildrpm.sh/docker-buildrpm.sh --> /tmp/build/docker-buildrpm.sh

[ OK ]: Copied: ../buildpython3/packaging/rpm/developer-tools.repo/developer-tools.repo --> /tmp/build/buildpy-1.7/developer-tools.repo

[ INFO ]: Generating build spec file and build artifacts in /tmp/build/buildpy-1.7
```

Back to *RPM Package Creation* Index

---

## 26.2 RPM Package Build

```
[ INFO ]: Generating build spec file and build artifacts in /tmp/build/buildpy-1.7

[ INFO ]: Bin /tmp/build/buildpy-1.7/buildpy successfully updated.

[ INFO ]: Module /tmp/build/buildpy-1.7/version.py successfully updated.

[ INFO ]: Module ../buildpython3/core/version.py successfully updated.

[ OK ]: Updated buildpy.spec with MAJOR_VERSION

[ OK ]: Updated buildpy.spec with MINOR_VERSION

[ OK ]: Updated buildpy.spec with DOCKERUSER (builder)

[ OK ]: Updated buildpy.spec with Dependencies (bash >= 4.1, curl >= 7.0, bc >= 1.0, coreutils, hostname, which, sudo, util-linux, tar, wget, xz, xz-libs)

[ INFO ]: tgz archive built: /tmp/build//tmp/build/buildpy-1.7.14.tar.gz

[ INFO ]: Image already exists. Creating Container...

[ INFO ]: Begin cp files into container

[ INFO ]: buildpy-1.7.14.tar.gz copied to container rpmbuildC successfully

[ INFO ]: buildpy.spec copied to container rpmbuildC successfully

[ INFO ]: docker-buildrpm.sh copied to container rpmbuildC successfully
```

Back to *RPM Package Creation* Index

---



## 26.3 Docker RPM Package Build

```
[ INFO ]: tput: No value for $TERM and no -T specified
tput: No value for $TERM and no -T specified
tput: No value for $TERM and no -T specified
tput: No value for $TERM and no -T specified
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.Nn1lJs
+ umask 022
+ cd /home/builder/rpmbuild/BUILD
+ cd /home/builder/rpmbuild/BUILD
+ rm -rf buildpy-1.7
+ /usr/bin/gzip -dc /home/builder/rpmbuild/SOURCES/buildpy-1.7.14.tar.gz
+ /usr/bin/tar -xf -
+ STATUS=0
+ '[' 0 -ne 0 ']'
+ cd buildpy-1.7
+ /usr/bin/chmod -Rf a+rx,u+w,g-w,o-w .
+ exit 0
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.6dyMJf
+ umask 022
+ cd /home/builder/rpmbuild/BUILD
+ cd buildpy-1.7
+ exit 0
Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.bskvK2
+ umask 022
+ cd /home/builder/rpmbuild/BUILD
+ '[' /home/builder/rpmbuild/buildpy-1.7 '!=' / ']'
+ rm -rf /home/builder/rpmbuild/buildpy-1.7
++ dirname /home/builder/rpmbuild/buildpy-1.7
+ mkdir -p /home/builder/rpmbuild
+ mkdir /home/builder/rpmbuild/buildpy-1.7
+ cd buildpy-1.7
+ install -m 0755 -d /home/builder/rpmbuild/buildpy-1.7/usr/local/bin
+ install -m 0755 -d /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy
+ install -m 0755 -d /home/builder/rpmbuild/buildpy-1.7/var/log
+ install -m 0755 -d /home/builder/rpmbuild/buildpy-1.7/etc/bash_completion.d
+ install -m 0755 -d /home/builder/rpmbuild/buildpy-1.7/etc/yum.repos.d
+ install -m 0755 buildpy /home/builder/rpmbuild/buildpy-1.7/usr/local/bin/buildpy
+ install -m 0644 std_functions.sh /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/std_functions.sh
+ install -m 0644 os_distro.sh /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/os_distro.sh
+ install -m 0644 colors.sh /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/colors.sh
+ install -m 0644 exitcodes.sh /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/exitcodes.sh
+ install -m 0644 version.py /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/version.py
+ install -m 0444 bc.pkg /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/bc.pkg
+ install -m 0444 curl.pkg /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/curl.pkg
+ install -m 0444 gawk.pkg /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/gawk.pkg
+ install -m 0444 uninstall.paths /home/builder/rpmbuild/buildpy-1.7/usr/local/lib/buildpy/uninstall.paths
+ install -m 0644 buildpy-completion.bash /home/builder/rpmbuild/buildpy-1.7/etc/bash_completion.d/buildpy-completion.bash
+ install -m 0644 developer-tools.repo /home/builder/rpmbuild/buildpy-1.7/etc/yum.repos.d/developer-tools.repo
+ /usr/lib/rpm/find-debuginfo.sh --strict-build-id -m --run-dwz --dwz-low-mem-die-limit 10000000 --dwz-max-die-limit 110000000 /home/builder/rpmbuild/BUILD/buildpy-1.7
/usr/lib/rpm/sepdebugcrcfix: Updated 0 CRC32s, 0 CRC32s did match.
+ '[' noarch = noarch ']'
+ case "$QA_CHECK_RPATHS:-" in
+ /usr/lib/rpm/check-buildroot
+ /usr/lib/rpm/redhat/brp-compress
+ /usr/lib/rpm/redhat/brp-strip-static-archive /usr/bin/strip
+ /usr/lib/rpm/brp-python-bytecompile /usr/bin/python 1
+ /usr/lib/rpm/redhat/brp-python-hardlink
+ /usr/lib/rpm/redhat/brp-java-repack-jars
Processing files: buildpy-1.7-14.noarch
Provides: buildpy = 1.7-14
Requires(interp): /bin/sh
Requires(rpmlib): rpmlib(CompressedFileNames) <= 3.0.4-1 rpmlib(FileDigests) <= 4.6.0-1 rpmlib(PartialHardlinkSets) <= 4.0.4-1 rpmlib(PayloadFilesHavePrefix) <= 4.0-1
Requires(post): /bin/sh
Checking for unpackaged file(s): /usr/lib/rpm/check-files /home/builder/rpmbuild/buildpy-1.7
Wrote: /home/builder/rpmbuild/SRPMs/buildpy-1.7-14.src.rpm
Wrote: /home/builder/rpmbuild/RPMS/noarch/buildpy-1.7-14.noarch.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.QlvYMs
+ umask 022
+ cd /home/builder/rpmbuild/BUILD
+ cd buildpy-1.7
+ /usr/bin/rm -rf /home/builder/rpmbuild/buildpy-1.7
+ exit 0

[ OK ]: rpmbuildC successfully halted

[ INFO ]: postbuild: Module buildpython3/core/version.py successfully updated.

[ INFO ]: New package created: /home/Blake/git/linux/buildpython3/packaging/rpm/buildpy-1.7-14.noarch.rpm

[ OK ]: RPM build process completed successfully. End
```

Back to [RPM Package Creation Index](#)



## 26.4 RPM Package Final Contents

Package Contents: <b>buildpy-1.7-14.noarch.rpm</b>			
Permission	Owner/Group	ctime	File
drwxr-xr-x	root root	Mar 31 14:44	/etc/bash_completion.d
-rw-r--r--	root root	Mar 31 14:44	/etc/bash_completion.d/buildpy-completion.bash
drwxr-xr-x	root root	Mar 31 14:44	/etc/yum.repos.d
-rw-r--r--	root root	Mar 31 14:44	/etc/yum.repos.d/developer-tools.repo
-rwxr-xr-x	root root	Mar 31 14:44	/usr/local/bin/buildpy
drwxr-xr-x	root root	Mar 31 14:44	/usr/local/lib/buildpy
-r--r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/bc.pkg
-rw-r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/colors.sh
-r--r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/curl.pkg
-rw-r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/exitcodes.sh
-r--r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/gawk.pkg
-rw-r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/os_distro.sh
-rw-r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/std_functions.sh
-r--r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/uninstall.paths
-rw-r--r--	root root	Mar 31 14:44	/usr/local/lib/buildpy/version.py

[Back to \*RPM Package Creation\* Index](#)

[Back to Table Of Contents](#)



### 27.1 v1.8.12 | Release Notes

**Release date:** May 21, 2021

Maintenance Release

---

#### 27.1.1 Maintenance, v1.8.12

Added support for 2-digit Python minor releases with the release of . Previous versions contained a bug which prevented **buildpy** from recognizing Python binary releases higher than the 9 minor version (i.e. higher than 3.9 for example).

---

( [Back to Releases](#) )

---



## CHAPTER 28

---

### Release History

---

buildpy Release Version	Date
	May 21, 2021
	Dec 27, 2020
	Dec 26, 2020
	Dec 22, 2020
	July 24, 2020
	January 28, 2020
	January 12, 2020
	January 10, 2020
	October 29, 2019
	October 26, 2019
	Sept 26, 2019
	Sept 24, 2019
	August 3, 2019
	April 2, 2019
Release <b>1.7.14</b>	April 1, 2019
Release <b>1.1.13</b>	March 31, 2019
Release <b>1.7.8</b>	December 8, 2019
Release <b>1.1.0</b>	January 23, 2019
Release <b>1.0.1</b>	December 30, 2017



## 29.1 v1.8.11 | Release Notes

**Release date:** December 27, 2020

Maintenance Release

---

### 29.1.1 Maintenance, v1.8.11

1. Added additional Operating System package dependencies to satisfy dependencies during binary compilation.

Dependencies updated in this release for the following Linux OS variants:

- (All supported versions).
- (Version 2).
- (All supported versions).

2. Added new function to create global `python3` symlink if it does not already exist on the system.
- 

( [Back to Releases](#) )

---

## 29.2 v1.8.10 | Release Notes

**Release date:** December 26, 2020

Maintenance Release

---

### 29.2.1 Maintenance, v1.8.10

Added additional Operating System package dependencies to satisfy and compression algorithm dependencies required during Python binary compilation.

- **Debian-based Linux OS Packages added:**

- liblzma-dev | XZ-format compression library - development files
- liblz-dev | data compressor based on the LZMA algorithm (development)
- lzma-dev | Compression and decompression in the LZMA format - development files
- libqdbm-dev | QDBM Database Libraries [development]
- libgdbm-dev | GNU dbm database routines (development files)

- **Redhat-based Linux OS Packages added:**

- lzma-sdk | SDK for lzma compression
- lzma-sdk-devel | Development libraries and headers for lzma-sdk
- xz-devel | Devel libraries & headers for liblzma
- gdbm-devel | Development libraries and header files for the GNU Database system

Dependencies updated in this release for the following Linux OS variants:

- (All supported versions).
  - (All supported versions).
  - (All supported versions).
  - (Version 2).
  - (All supported versions).
- 

( [Back to Releases](#) )

---

## 29.3 v1.8.9 | Release Notes

**Release date:** December 22, 2020

Maintenance Release

---



### 29.3.1 Maintenance, v1.8.9

Added additional Operating System package dependencies to satisfy **uuid** requirements during Python binary compilation for the following OS variants:

- (All supported versions).
  - (All supported versions).
  - (All supported versions).
  - (Version 2).
  - (All supported versions).
- 

( [Back to Releases](#) )

---

## 29.4 v1.8.8 | Release Notes

**Release date:** July 24, 2020

Maintenance Release

---

### 29.4.1 Maintenance, v1.8.8

- **20.04 Support.** Added support for newly released (LTS and non-LTS minor releases).
  - **20 Support.** Added support for newly released variants such as .
- 

( [Back to Releases](#) )

---

## 29.5 v1.8.5 | Release Notes

**Release date:** January 28, 2020

Maintenance Release

---

## 29.5.1 Maintenance, v1.8.5

- **Python Version Reconciliation.** Fixed Issue #25/#36: Failure to recognize faulty input for Python version

---

( [Back to Releases](#) )

---

## 29.6 v1.8.3 | Release Notes

**Release date:** January 12, 2020

Feature Release

---

### 29.6.1 Feature Updates, v1.8.3

- **Support for CentOS 8 Linux.** Compatibility tested for both 7 and 8 versions of CentOS.
- **Support for Python 3.9.** Python 3.9 now included as an optional major version choice for installation.

---

( [Back to Releases](#) )

---

## 29.7 v1.8.1 | Release Notes

**Release date:** January 10, 2020

Feature Release

---

### 29.7.1 Feature Updates, v1.8.1

**Install a specific Python binary set.** User can select an exact Python binary set to install instead of the latest release. For example, previously, if one wanted to install Python 3.6, and the latest minor revision available was 3.6.8, only 3.6.8 would be compiled and installed. User can now provide the full revision number when installing and **buildpy** will compile and install that exact revision set as follows:

```
$ sudo buildpy --install Python-3.6.5
```

Alternatively, the following syntax is equivalent:

```
$ sudo buildpy --install 3.6.5
```

---

( [Back to Releases](#) )

---

## 29.8 v1.7.20 | Release Notes

**Release date:** October 29, 2019

Feature Release

---

### 29.8.1 Feature Updates, v1.7.20

- User can now choose whether to install release candidate python build sets if downloaded from python.org.
  - Issue #58 Resolved: buildpy now checks system-installed python major version before installing. Change prevents installing a compiled python binary executable which is already installed on the system.
- 

( [Back to Releases](#) )

---

## 29.9 v1.7.19 | Release Notes

**Release date:** October 26, 2019

Maintenance Release

---

### 29.9.1 Maintenance Updates, v1.7.19

- As of release 1.7.18, can now compile and install Python release candidate, Beta, and Alpha code releases. Fix remaining issues with code released in version 1.7.18.
- 

( [Back to Releases](#) )

---

## 29.10 v1.7.18 | Release Notes

**Release date:** September 26, 2019

Feature Release

---

### 29.10.1 Feature Updates, v1.7.18

- As of release 1.7.18, can now compile and install Python release candidate, Beta, and Alpha code releases. This change was made to enable testing with Python 3.8+ release candidates in the future.
- 

( [Back to Releases](#) )

---

## 29.11 v1.7.17 | Release Notes

**Release date:** September 24, 2019

Maintenance Release

---

### 29.11.1 Maintenance, v1.7.17

- Release to fix various compatibility issues with modern releases (> version 25)
- 

( [Back to Releases](#) )

---

## 29.12 v1.7.16 | Release Notes

**Release date:** August 3, 2019

Functionality & Maintenance Release

---

### 29.12.1 Feature Updates, v1.7.16

- Complete rewrite to function *package\_info* to enable more info on single screen presentation.
- Bug Fix, Various

---

( [Back to Releases](#) )

---

## 29.13 v1.7.15 | Release Notes

**Release date:** April 2, 2019

Documentation Release

---

### 29.13.1 Documentation, v1.7.15

- Multiple updates to [Summary](#) and [Releases](#) sections.

### 29.13.2 Maintenance, v1.1.3

- Fixed uninstall operation deleting out-of-band references to python version for removal
- Bug Fix, Various

---

( [Back to Releases](#) )

---

---

[Table Of Contents](#)

---



## CHAPTER 30

---

### Module Index

---

- [Bash Module Index](#)





## CHAPTER 31

---

### Site Index

---

- [genindex](#)



## CHAPTER 32

---

### Search Project

---

- search