Part 37
-
Switching to Python3

Version: 2020-02-20

Use Python 3

By default, Raspbian (Buster 2019 and earlier) uses Python 2. However, versions 2 and 3 come installed by default.

You can run the following `ls` command to find out what python binary executables are available on your system:

```
$ ls /usr/bin/python*
/usr/bin/python   /usr/bin/python2.7         /usr/bin/python2-config
/usr/bin/python3.5         /usr/bin/python3.5m         /usr/bin/python3-
config  /usr/bin/python3m-config
/usr/bin/python2  /usr/bin/python2.7-config  /usr/bin/python3
/usr/bin/python3.5-config  /usr/bin/python3.5m-config  /usr/bin/python3m
/usr/bin/python-config
```

We just have to make 1 minor change so that the Pi uses Python 3 whenever we type `python` into a terminal. In a terminal window, enter the following command:

```
$ python –version
```

You should see which version is being used by default. For example, you might see

```
Python 2.7.13
```

Change python version for a user
If you see that your OS is using Python 2 by default, you'll need to change it to use the Python 3 installation. We want to this so that Python 3 is used every time we log in. Enter the command:

```
$ nano ~/.bashrc
```

`.bashrc` is a file that resides in the user's home directory (the user pi in this case). The file acts as a shell script that is run each time that specific user opens a terminal (or logs in over SSH, serial, etc.). It can help to customize the user environment, and you will likely see a number of other commands already in there. Scroll down to the bottom, and add the following line to the file:

```
alias python='/usr/bin/python3'
```

Exit out by pressing Ctrl+x, press the y-key when prompted if you want to save the file, and press the enter key.

Instead of logging out and logging back in again to run the new command, you can simply run the contents of the .bashrc script by entering:

```
$ source ~/.bashrc
```

Now, check the version of Python again:

```
$ python –version
```

You should see some version of Python 3 being used.

```
Python 3.5.3
```

Change python version system-wide
To change python version system-wide we can use update-alternatives command. Logged in as a root user, first list all available python alternatives:

```
$ update-alternatives --list python
update-alternatives: error: no alternatives for python
```

The above error message means that no python alternatives has been recognized by update-alternatives command. For this reason we need to update our alternatives table and include both python2.7 and python3.5. First do your ls command again to make sure what where is installed

```
$ ls /usr/bin/python*
/usr/bin/python   /usr/bin/python2.7         /usr/bin/python2-config
/usr/bin/python3.5         /usr/bin/python3.5m         /usr/bin/python3-
config  /usr/bin/python3m-config
/usr/bin/python2  /usr/bin/python2.7-config  /usr/bin/python3
/usr/bin/python3.5-config  /usr/bin/python3.5m-config  /usr/bin/python3m
/usr/bin/python-config
```

Next,

```
$ update-alternatives --install /usr/bin/python python /usr/bin/python2.7 1
update-alternatives: using /usr/bin/python2.7 to provide /usr/bin/python
(python) in auto mode

$ update-alternatives --install /usr/bin/python python /usr/bin/python3.5 2
update-alternatives: using /usr/bin/python3.4 to provide /usr/bin/python
(python) in auto mode
```

The `--install` option take multiple arguments from which it will be able to create a symbolic link. The last argument specified it priority means, if no manual alternative selection is made the alternative with the highest priority number will be set. In our case we have set a priority 2 for `/usr/bin/python3.5` and as a result the `/usr/bin/python3.5` was set as default python version automatically by update-alternatives command.

```
$ python --version
Python 3.5.3
```

Next, we can again list all python alternatives:

```
$ update-alternatives --list python
/usr/bin/python2.7
/usr/bin/python3.5
```

From now on, we can anytime switch between the above listed python alternative versions using below command and entering a selection number:

```
$ update-alternatives --config python

There are 2 choices for the alternative python (providing /usr/bin/python).

  Selection    Path                Priority   Status
------------------------------------------------------------
* 0            /usr/bin/python3.5   2         auto mode
  1            /usr/bin/python2.7   1         manual mode
  2            /usr/bin/python3.5   2         manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
$ python --version
Python 3.5.3
```

Later on
In case we no longer have the alternative python version installed on our system we can remove its update-alternatives listing. For example let's remove python2.7 version:

```
$ update-alternatives --remove python /usr/bin/python2.7
update-alternatives: removing manually selected alternative - switching
python to auto mode
update-alternatives: using /usr/bin/python3.5 to provide /usr/bin/python
(python) in auto mode
```

Using pip3

If you are using the full desktop version of Raspbian, you should have `pip` already installed.

If you are using Raspbian Lite, the Python package manager, `pip`, does not come pre-installed. As a result, you will need to install it with the commands:

```
$ sudo apt update
$ sudo apt -y install python3-pip
```

Note that to use `pip` for Python 3, you will need to use the command `pip3`. However, we can modify the `.bashrc` file to use `pip` instead of `pip3`, as the rest of the tutorial will show examples using `pip`:

```
$ nano ~/.bashrc
```

Scroll down to the bottom, and add the following command to the file:

```
alias pip=pip3
```

Exit out with ctrl+x, press y and enter. Run the .bashrc script with:

```
$ source ~/.bashrc
```

You should now be able to install Python packages using the `pip` command.

Installing Python 3.7.x on Raspbian

Note: the x in the version must be replaced with the version you want.

Update the Raspbian before installing python.

```
$ sudo apt -y update
```

Install the required build-tools.

```
$ sudo apt -y install build-essential tk-dev libncurses5-dev libncursesw5-dev
libreadline6-dev libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-dev libbz2-
dev libexpat1-dev liblzma-dev zlib1g-dev libffi-dev
```

If one of the packages cannot be found, try a newer version number (e.g. libdb5.4-dev instead of libdb5.3-dev).

Download and install the latest Python 3.7 source. Select the most recent release of Python from the official site. Adjust the file names accordingly to match your version.

```
$ wget https://www.python.org/ftp/python/3.7.x/Python-3.7.x.tar.xz
$ sudo tar xf Python-3.7.x.tar.xz
$ cd Python-3.7.x
$ ./configure --enable-optimizations
$ make -j -l 4
$ make altinstall
```

Now Python is installed you can check the version using the following command.

```
$ python3.7 --version
```

If you want to use python 3.7 as default version you can create an alias.

```
$ nano  ~/.bashrc
```

and then add the following alias.

```
which python3.7
/usr/local/bin/python3.7
alias python='/usr/local/bin/python3.7'
```

Then source the .bashrc file.

```
$ source ~/.bashrc
```

After creating an alias check the python version again.

```
$ python --version
Python 3.7.0
```

You can use the newly installed python version

Optionally: Delete the source code and uninstall the previously installed packages. When uninstalling the packages, make sure you only remove those that were not previously installed on your system. Also, remember to adjust version numbers if necesarry.

```
$ sudo rm -r Python-3.7.x
$ rm Python-3.7.x.tar.xz
$ sudo apt-get --purge remove build-essential tk-dev -y
$ sudo apt-get --purge remove libncurses5-dev libncursesw5-dev libreadline6-
dev -y
$ sudo apt-get --purge remove libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-
dev -y
$ sudo apt-get --purge remove libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev
libffi-dev -y
$ sudo apt -y autoremove
$ sudo apt clean
```

Install Python 3.8 on Raspberry Pi

Raspbian Buster 10 for Raspberry Pi includes Python 3.7. Here is how to compile Python 3.8 on the Raspberry Pi.

Update the Raspbian before installing python.

```
$ sudo apt -y update
```

Install the required build-tools.

```
$ sudo apt-get -y install libffi-dev libbz2-dev liblzma-dev libsqlite3-dev
libncurses5-dev libgdbm-dev zlib1g-dev libreadline-dev libssl-dev tk-dev
build-essential libncursesw5-dev libc6-dev openssl git
```

Download and install the latest Python 3.8 source.

```
$ wget https://www.python.org/ftp/python/3.8.0/Python-3.8.0.tar.xz
$ sudo tar xf Python-3.8.0.tar.xz
$ cd Python-3.8.0
$ ./configure --enable-optimizations
$ make -j -l 4
$ make install
```

Build and install steps take up to 10-40 minutes, depending on Raspberry Pi model. Do not use sudo!

Note: don't omit `-l 4` or Pi will be quickly overwhelmed and error build. This limits load average to 4. Without it, load average will soar to 100+ (bad).

Now Python is installed you can check the version using the following command.

```
$ python3.8 -version
```

If you want to use python 3.8 as default version you can create an alias.

```
$ sudo nano  ~/.bashrc
```

and then add the following alias.

```
which python3.8
/usr/local/bin/python3.8
alias python='/usr/local/bin/python3.8'
```

Then source the .bashrc file.

```
$ source ~/.bashrc
```

After creating an alias check the python version again.

```
$ python -version
Python 3.8.0
```

You can use the newly installed python version.
Python 3.8 brings further speed / efficiency improvements. Specific benefits include:

- pathlib: in Python ≥ 3.6 standard library
- inline variable "f-string" parsing.
- typing type hinting–detect bugs before they strike

Optionally: Delete the source code and uninstall the previously installed packages. When uninstalling the packages, make sure you only remove those that were not previously installed on your system. Also, remember to adjust version numbers if necesarry.

```
$ sudo rm -r Python-3.8.0
$ rm Python-3.8.0.tar.xz
$ sudo apt-get --purge remove build-essential tk-dev -y
$ sudo apt-get --purge remove libncurses5-dev libncursesw5-dev libreadline6-
dev -y
$ sudo apt-get --purge remove libdb5.3-dev libgdbm-dev libsqlite3-dev libssl-
dev -y
$ sudo apt-get --purge remove libbz2-dev libexpat1-dev liblzma-dev zlib1g-dev
libffi-dev -y
$ sudo apt -y autoremove
$ sudo apt clean
```