PROGRAMMING WITH PYTHON
Simple Programming Environment with ubuntu 16.04

emerging-mind.org eJournal ISSN 2567-6466
Email: info@emerging-mind.org
10.-12.February 2018

Gerd Doeben-Henisch
gerd@doeben-henisch.de

February 12, 2018

Contents

Abstract

This text describes a simple programming environment for python3 in an ubuntu 16.04 environment.

1 The Environment

The minimal environment includes

1. As operating system ubuntu 16.04 LTS
2. The python3 language version 3.5.2

Some of the texts I am using for this small introduction are the following ones:

- `https://www.python.org/`
- `http://docs.python-guide.org`
- ’python’ in Wikipedia(EN) [WE18]
- Zelle (2017) [Zel17]
- Romano et.al. (2016) [RPH16]
- Walters (2014) [Wal14]

To check, in which folder python3 is located, you can enter: *which python3*. My system answers: */usr/bin/python3*, or asking for *python3.5* will produce */usr/bin/python3.5*.

2 Steps To Prepare Python Environment

For the following steps I have additionally used:
`https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up`

1. Check the last updates: *sudo apt-get update*
2. Check version of python3: *python3 -V*
3. Answer: *Python 3.5.2*
4. Packet manager pip is installed: *python3-pip*
5. Recomended for a robust install: *sudo apt-get install build-essential libssl-dev libffi-dev python3-dev*
6. To enable virtual environments you need: *python3-venv*
7. For the EML python programs we create the environment: *mkdir environments*

8. Go into this directory: *cd environments*
9. And generate inside the folder 'environments' the new folder 'eml': *pyvenv eml*
10. This generates a directory with several sub-folders: *ls eml* shows you *bin include lib lib64 pyvenv.cfg share*
11. To activate the eml environment one has to type: *source eml/bin/activate*. The prompt is now prefixed with the name of the environment called 'eml', e.g. *(eml) gerd@Doeben-Henisch: /environments$*
12. *Remark*: Inside the virtual environment, one can use the command python instead of python3, and pip instead of pip3.
13. Test the environment by editing a simple program: *gedit hello.py*. After closing the editor enter the command: *python hello.py*. This will produce the output: *Hello, World!*
14. To leave the environment, simply type the command *deactivate* and you will return to your original directory.
15. To simplify the activation I have written a one-line mini-shell-script located in the local folder 'bin' as follows:

    ```
    #!/bin/sh
    environments/eml/bin/activate
    ```

    To use the script one has to make it executable with *chmod u+x venv.sh* and then one can call it from the home-directory with *source venv.sh*.

3 Use Python Interactive Console

For the following steps I have additionally used:
`https://www.digitalocean.com/community/tutorials/how-to-work-with-the-python-intera`

1. To activate the eml environment one has to type: *source eml/bin/activate*. The prompt is now prefixed with the name of the environment called 'eml', e.g. *(eml) gerd@Doeben-Henisch: /environments$*
2. To activate the python interactive console within the environment one enters: *python*. This produces the output: *Python 3.5.2 (default, Nov 23 2017, 16:37:01) [GCC 5.4.0 20160609] on linux Type "help", "copyright", "credits" or "license" for more information.*

3. Now one can enter directly python commands like:

```
>>> a=120
>>> b=333
>>> diff=b-a
>>> diff
213
>>> print('Diff ',diff)
Diff  213
>>>
```

4. To check whether a certain module is available one can check this with the command: *import modul-name*, e.g. *import matplotlib*. If this is available, one gets the answer: *Traceback (most recent call last): File "$< stdin >$", line 1, in $< module >$ ImportError: No module named 'matplotlib'*

5. To install the module matplotlib one enters the following command:

```
(eml) gerd@Doeben-Henisch:~/environments$ pip install matplotlib
Collecting matplotlib
Downloading matplotlib-2.1.2-cp35-cp35m-manylinux1_x86_64.whl (15.0MB)
Collecting six>=1.10 (from matplotlib)
Downloading six-1.11.0-py2.py3-none-any.whl
Collecting numpy>=1.7.1 (from matplotlib)
Downloading numpy-1.14.0-cp35-cp35m-manylinux1_x86_64.whl (17.1MB)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib)
Downloading pyparsing-2.2.0-py2.py3-none-any.whl (56kB)
Collecting python-dateutil>=2.1 (from matplotlib)
Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
Collecting pytz (from matplotlib)
Downloading pytz-2018.3-py2.py3-none-any.whl (509kB)
Collecting cycler>=0.10 (from matplotlib)
Downloading cycler-0.10.0-py2.py3-none-any.whl
Installing collected packages: six, numpy, pyparsing, python-dateutil, pytz, cy
Successfully installed cycler-0.10.0 matplotlib-2.1.2 numpy-1.14.0 pyparsing-2.
```

6. This procedure reveales further, that the actual version of pip is not up-to-date. This triggered the following update sequence:

```
Collecting pip
Downloading pip-9.0.1-py2.py3-none-any.whl (1.3MB)
Installing collected packages: pip
Found existing installation: pip 8.1.1
```

```
Uninstalling pip-8.1.1:
Successfully uninstalled pip-8.1.1
Successfully installed pip-9.0.1
```

7. Then one can go back to the interactive console and call again the matplotlib module:

   ```
   (eml) gerd@Doeben-Henisch:~/environments$ python
   Python 3.5.2 (default, Nov 23 2017, 16:37:01)
   [GCC 5.4.0 20160609] on linux
   Type "help", "copyright", "credits" or "license" for more information.
   >>> import matplotlib
   >>>
   ```

8. One can leave the python console either with 'Ctrl D' or within the python console with 'quit()'. The last version will store the 'quit()-command' in the console history log. This can be shown with starting an editor after leaving the python console:

   ```
   (eml) gerd@Doeben-Henisch:~/environments$ gedit ~/.python_history

   a-e-main23.pyw
   ls
   ae6-2-6-main.py
   a=120
   b=333
   diff=b-a
   diff
   print('Diff ',diff)
   gerd = 'Gerd'
   anita = 'Anita'
   if len(gerd) < len(anita):
   print('Gerd is shorter than Anita')
   print('Gerd is shorter than Anita')
   gerd
   anita
   if len(gerd) < len(anita):
   print('Gerd')
   else:
   print('Anita')
   import matplotlib
   quit()
   ```

4 Debugging with the Code Module

Rather than stepping through the code with a debugger, you can add the *code* module to your Python program to instruct the program to stop execution and enter into the interactive mode in order to examine how your code is working. The code module is part of the *Python standard library*. To see a complete example go to `https://www.digitalocean.com/community/tutorials/how-to-debug-python-with-an-interactive-console`.

Once one is done using the code module to debug the code, one has to remove the code functions and import statement.

References

[RPH16] Fabrizio Romano, Dusty Phillips, and Rick van Hattem. *Python: Journey from Novice to Expert*. Packt Publishing Ltd., Birmingham (UK), 1 edition, 2016.

[Wal14] Gregory Walters. *Python Quick Syntax Reference*. apress, 1 edition, 2014. http://www.myilibrary.com?ID=600465.

[WE18] Wikipedia-EN. Python (programming language). 2018.

[Zel17] John Zelle. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle, Portland (Oregon, USA), 3 edition, 2017.