Short Python Intro

Gerald Schuller, Nov. 2016

Python can be very similar to Matlab, very easy to learn if you already know Matlab, it is Open Source (unlike Matlab), it is easy to install, and unlike Matlab a real programming language. We recommend using Linux as an operating system, because there Python is already installed, and its packages and libraries can be very easily installed. You can have Lunux installed on your computer for instance as a dual boot option, or have it installed in VirtualBox.

You can also install Python under Windows from https://www.python.org/downloads/

but everything runs more smoothly under Linux, that's why we only support Linux.

If Python is on your machine, you can install libraries and packages with the command "pip" in a terminal shell, both in Linux and Windows. In Linux you can open a Terminal Window with shortcut "Alt-Ctl-T". For instance to install the package "pylab", you type in your terminal shell:

```
pip install pylab
```

Python Programs

To write a program, you open an editor by typing:

```
gedit &
```

in your terminal shell (under Windows you can type “notepad” instead).

In the open Editor window now type, for instance:

print(”Hello World”)

Then use “save as” in the editor menue to store it with name “helloworld.py”.

Now go back to your terminal shell, activate it by clicking on it, and execute your new program by typing:

```
python helloworld.py
```

Now you should see “Hello World” printed in your terminal.

Interactive Mode

If you quickly want to try commands, you can enter pythons interactive mode by simply typing:

```
python
```

in your terminal. You will get a prompt and can type:

```
1+2
```

and after hitting the “return” key you will see a “3”. In this way you can try all functions of python.

You end the interactive mode by typing

```
quit()
```

Python Pylab

You can also obtain an interactive mode which is very similar to matlab by typing:

```
ipython -pylab
```

in your terminal. This gives you a more convient mode and automatically loads the libraries “numpy” and “matplotlib”.

See this webpage for how numpy and Matlab relate to each other:

https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html

and this Website for pylab:

http://scipy.github.io/old-wiki/pages/PyLab

An example for a session in ipython –pylab is to plot 1 period of the sine function:

```
x=linspace(0,2*pi,100);
y=sin(x);
plot(x,y)
```

It can be seen that it looks the same as in Matlab/ Octave. Now convert the x array into type “16 bit integer”:

```
xi=x.astype(int16)
```

This cannot be done in Matlab/Octave.

Writing Functions in Python:

Open the editor and type, for instance:

```
def myfunction(a,b):
   """this is the help text testfunction..."""
   y=2*a+3*b;
   return y
```

Observe the indentation. In Python a constant indentation takes the role of brackets, so it is important.

Save it as „testfunction.py“

The we can start python by typing

```
python
```

in our terminal window. Then in python we type:

```
import testfunction
```

Now we can use it in Python, for instance with:

```
y=testfunction.myfunction(7,8)
print(y)
```

and we see the result in variable y. Observe the name “testfunction”. It represents our imported “library”, and is helpful to distinguish possibly identical named functions in different libraries.

Testing

To test a function, it is best practice to use a “main” section in the python program file. Your Python program file testfunction.py would then have the following content:

```
def myfunction(a,b):
    """this is the help text testfunction..."""
    y=2*a+3*b;
    return y

if __name__ == '__main__':
    y=myfunction(7,8)
    print(y)
```

In your terminal shell you then execute the program with

```
python testfunction.py
```

and it prints the output y in your terminal, so that you can check if it is correct.

Using Pylab in Programs and Functions

You can also use the functionality of Pylab in your programs and functions. For that it is best practice to import its libraries using their “name spaces”:

```
import numpy as np
import matplotlib.pyplot as plt
```

For instance to plot f periods of the sine function using a function, create a python program file with name “plotsine.py”, with content:

```
def plotsine(f):
   """this function plots f periods of the sine
function"""
   import numpy as np
   import matplotlib.pyplot as plt
   x=np.linspace(0,2*np.pi,100);
   y=np.sin(f*x);
   plt.plot(x,y)
   plt.show()
   return

if __name__ == '__main__':
   plotsine(2.0)
```

Here we need to prepend “np.” and “plt” to name the libraries in which the functions are, to uniquely identify them. Observe the “plt.show()”, this opens the plot window, and keeps it open even after the program finished executing. Also observe the “2.0” in the function argument. A decimal point can be used to make sure that Python uses float numbers, otherwise it would use integers and integer calculations (in Python 2). This is different in Python 3, which uses float by default.

Then execute it in the terminal shell with the command:

```
python plotsine.py
```

This plot functionality is also very practical for debugging.

Python is **heavily supported on the internet**. For basically every question you will get an answer by looking in Google.