NOTES ON RUNNING PYTHON CODE

ERIC MARTIN

Part 1. Setting things up

1. Installing python if necessary

The School has python 3.2.3 installed.

On personal computers with no version of python 3 installed, get the latest version (python 3.5.1) for the appropriate platform from

https://www.python.org

Mac users: drag the IDLE.app icon in /Applications/Python 3.5 to the dock.

2. Installing pip if necessary

Windows and Mac users should have pip automatically shipped with python, but Ubuntu and Debian Linux users may need to execute

```
sudo apt-get install python3-pip
```

3. Installing extra modules

You cannot install modules on the School machines. On your own computer, you can install thousands.

Mac and Linux users install (in particular) the modules matplotlib, beautifulsoup4, openpyxl and jupyter by executing

```
pip3 install matplotlib
pip3 install beautifulsoup4
pip3 install openpyxl
pip3 install jupyter
```

You can get a listing of the modules you have installed by executing

```
pip3 list
```

To check whether some of the modules you have installed are not up to date, execute

```
pip3 list --outdated
```

If a module some_outdated_module is listed as outdated, you can update it by executing

```
pip3 install -U some_outdated_module
```

Windows users might have to execute

```
python3 -m pip ...
```

instead of

```
pip3 ...
```

Date: Session 1, 2016.

4. Making python and idle the right commands

In the home directory of your CSE account, create or edit (with an editor such as vi or gedit) the file .bashrc and add the lines

```
alias python=python3
alias idle=idle3
```

You need to open another xterm (Terminal) window for this change to take effect and let python and idle launch python3 and idle3 rather than the default python2 and idle2, respectively.

Mac and Linux users may need to add these lines to the . profile or .bash_profile file rather than to the .bashrc file.

5. Permanently adding directories to sys.path

sys.path is the list of directories where python looks for modules (files). On a School machine, it is

```
['', '/usr/lib/python3.2', '/usr/lib/python3.2/plat-linux2',
 '/usr/lib/python3.2/lib-dynload', '/usr/local/lib/python3.2/dist-packages',
 '/usr/lib/python3/dist-packages']
```

as can be found out by interpreting from the python prompt

```
from sys import path
path
```

The first directory in this list, '', is the working directory.

To add directories to this list, create a sequence of new directories by executing in an xterm window the command

```
mkdir -p ~/.local/lib/python3.2/site-packages
```

To add the home directory to sys.path,

- run in the home directory the command pwd,
- create in ~/.local/lib/python3.2/site−packages the file my_path.pth, and
- add to this file the output of that command.

If you were me, that would be

```
/import/kamen/1/emartin
```

Other directories can be added, one per line. For instance, if you were me and had created in your home directory the directory COMP9021, then you could also add to my_path.pth the line

```
/import/kamen/1/emartin/COMP9021
```

to make it part of sys.path.

Mac Users: Same procedure but replacing ~/.local/lib/python3.2/site−packages by

~/Library/Python/3.5/lib/python/site−packages

Part 2. Using Idle

For the following, if you were me, you would have

- hello_world,
- hello_world_v1.py,
- hello_world_v2.py,
- greet.py, and
- greet_and_say_bye.py

saved in ~/COMP9021/Lectures/Lecture_1, and we assume that ~/COMP9021 is part of sys.path.

If

- neither ~/COMP9021/Lectures
- nor ~/COMP9021/Lectures/Lecture_1

had been added to sys.path, then Lectures/Lecture_1 would be the "missing part" of the path for python to be able to locate those files, unless ~/COMP9021/Lectures/Lecture_1 is the working directory.

This is all we assume if we use python 3.5, but if we use python 3.2 (which is what is installed on the School servers), then we also assume that

- ~/COMP9021/Lectures and
- ~/COMP9021/Lectures/Lecture_1

all contain an empty file named __init__.py.

6. At the prompt

6.1. **Executing statements.** Interpret

```
print('Hello world!')
```

6.2. **Defining functions and calling them.** Define a function as

```
def hello_world():
    print('Hello world!')
```

and call it by executing

```
hello_world()
```

7. Opening a file and selecting Run Module from the menu

7.1. **Executing statements.** Use the file hello_world_v1.py whose contents is

```
print('Hello world!')
```

7.2. **Calling functions.** Use the file hello_world_v2.py whose contents is

```
def hello_world():
    print('Hello world!')
```

and call the function from the Idle prompt by executing

```
hello_world()
```

8. Importing or reimporting a module containing the statements to execute

8.1. **Importing the module.** In case Idle has been launched from the directory where hello_world_v1.py is stored (probably by executing the idle Unix command in an xterm widow, in that directory), execute

```
import hello_world_v1
```

and in case Idle has been launched from another directory (maybe by clicking on the Idle icon), execute

```
import Lectures.Lecture_1.hello_world_v1
```

8.2. **Reimporting the module.** Repeating the import statement will not reevaluate the statements. Executing

```
from importlib import reload
```

allows every call to

```
reload(hello_world_v1)
```

or to

```
reload(Lectures.Lecture_1.hello_world_v1)
```

to reevaluate the statements.

9. Importing a module containing the functions to call or importing the functions themselves

9.1. **Importing the module.** In case Idle has been launched from the directory where hello_world_v2.py is stored, execute

```
import hello_world_v2
```

and in case Idle has been launched from another directory, execute

```
import Lectures.Lecture_1.hello_world_v2
```

and call the function by executing

```
hello_world_v2.hello_world()
```

or

```
Lectures.Lecture_1.hello_world_v2.hello_world()
```

respectively.

9.2. **Importing the functions.** In case Idle has been launched from the directory where hello_world_v2.py is stored, execute

```
from hello_world_v2 import hello_world
```

and in case Idle has been launched from another directory, execute

```
from Lectures.Lecture_1.hello_world_v2 import hello_world
```

and call the function by executing

```
hello_world()
```

10. Calling functions but not when importing

Use the file greet.py whose contents is

```
def hello(you):
    print('Hello ' + you + '!')

if __name__ == '__main__':
    hello('world')
    hello('Jane')
    hello('Michael')
```

and select Run Module from the menu.

Note that executing

```
import greet
```

does not produce any output.

Note that opening the file greet_and_say_bye.py whose contents is

```
import Lectures.Lecture_1.greet

Lectures.Lecture_1.greet.hello('universe')
print('Bye now...')
```

and selecting Run Module from the menu or executing

```
import greet_and_say_bye
```

at the prompt does not output

```
Hello world!
Hello Jane!
Hello Michael!
```

either.

In both cases, the test __name__ == '__main__' fails because __name__ is equal to 'greet'.

This technique is commonly used to easily test the code of one module (such as greet) meant to be utilised in other modules (such as greet_and_say_bye).

Part 3. Using an xterm window

A new method: execute the Unix command python3 hello_world_v1.py, or python hello_world_v1.py if python is an alias to python3.

For the rest, exactly as when using Idle, except for Section 7 and the parts of Section 10 that are specific to Idle, but executing the Unix python3 command (or python if it is an alias to python3) and entering statements from the python prompt rather than from the Idle prompt.

To quit python, press Control D.

Part 4. As an executable

Use the file hello_world whose contents is

```
#!/usr/bin/env python3
print('Hello world!')
```

Make it executable if needed by executing the command

```
chmod +x hello_world
```

and execute

```
hello_world
```

or

```
./hello_world
```

in case the directory where this file resides is not one of the directories listed in the PATH environment variable.

Part 5. Using jupyter

On your own computer, execute

```
jupyter notebook
```

to create a new sheet or open an existing sheet; in the latter case, you can also directly execute

```
jupyter notebook name_of_an_existing_sheet
```

in the directory where this sheet has been saved (or change name_of_an_existing_sheet to the path to that file.

To quit jupyter, press Control D.