January 17, 2019 PHYS 338 Spring 2019

# In-Class Work: Python

**1. Linux Login:** Repeat steps 1.— 4. of the linux exercise to get ready to work in the linux environment. (Xmanger Enterprise 5 → Sessions → Linuxremote1 Graphical and open terminal window.)

**2. Jupyter**
Python is the "programming language" we are going to use in this course. So you will learn the different types of commands in python. You can either write the commands into a file with gedit (or other editor), or you can use a handy tool , jupyter, for getting used to different commands, so for program development. In our course we will mostly use the route of writing a file with commands (so not via jupyter), as described below in step... Before go this route let's get started with jupyter. In your terminal first change into your directory Phys338_s2019 and then type the command

```
jupyter notebook &
```

This opens a browser window. [1]
You will see the content of your current directory, from which you started jupyter. To start a new jupyter notebook session

click on `New` and choose `Python3`

To the right of `In [ ]:` left click on it and notice that the box is framed in green. This allow you to type in the box. Type in the box:

```
x=1
print(x)
```

Now press on your keyboard at the same time `Shift` and `Enter` (Return key). Type in the next cell

```
print(x)
x=2
print(x)
```

and again press on your keyboard at the same time `Shift` and `Enter`. Use the `Shift` and `Enter` after each of the commands specified below. It allows you to "execute" or "run" the command(s).

Jupyter Notebook let's you also write **comments**. (Later in the course you will learn the toolkit `latex` which allows you to write formulae nicely and jupyter let's you use this nice toolkit.) Jupyter distinguishes between commands and comments. Click on your next empty cell. Notice that in the menue bar it says `Code`. In the pulldown menue change `Code` to `Markdown`. Type `Variables and Assignment` and press `Shift` and `Enter`. To insert a comment in between any previous commands, click e.g. to the left of `In [2]` and choose from the top menue `Insert` → `Insert Cell Above`.

[1] In case of an error-message about a port not being found, first kill any in the background running jupyter. This can be done with `ps -eaf | grep jupyter` and `kill` *idnumber* or with `pkill jupyter`. In case of an error-message that firefox is already in use, either use `jupyter notebook --browser google-chrome` or open an other browser and use the web address listed in the error message "the Jupyter Notebook is running at: http...". An in case of an error message "Unlock Login Keyring" error message, just minimize this error message window.

## 3. Variables and Assignments Intro

Read in Newman's Chapter 2 §2.2.1 and §2.2.2

As you read, try playing with some of the commands.

## 4. Output and Input Intro

Work next through Newman's §2.2.3 by reading the text and typing in the commands, e.g. put in one cell

```
x=1
y=2
print(x,y)
```

## 5. Save Jupyter Notebook Session

To **save your jupyter notebook session** first give it a name by clicking on the top on `Untitled` and replace the name, e.g. with `tryjupyter`. Now save with the saving-icon from the menu or using from the top menue `File` → `Save and Checkpoint` or `File` → `Download as` → `Notebook (.ipynb)`.
To continue a notebook session, you can open a notebook session with `File` → `Open`.

## 6. Python Program Directly (without jupyter)

Leave your jupyter session open, because we will use it later more. Next let us write a python program or "python script", so a set of commands, directly using `gedit`, so to write a file with the editor `gedit`, as you did in the linux exercise. So type in a terminal window the command

```
gedit first_program.py &
```

The file extension `.py` tells the `gedit` to go into Python mode, which will give you some helpful color coding. The ampersand allows you to get the command line back for more commands while `gedit` is still running.
Now, in the editor your first line has to be an instruction to fire up the Python interpreter, which we do with

```
#!/usr/bin/env python
```

Now add the following four lines

```
x=1
print(x)
y=2
print(x,y,x+y)
```

Save this, and then back at the command-line prompt type

```
chmod u+x first_program.py
```

This is a one-time step for this file that allows you to treat it as an executable file (read it as "change mode: user adds executable"). Now simply type at the command-line prompt

```
./first_program.py
```

and see what results.

So the commands are in `first_program.py`, the `chmod` command makes the file executable, and when you execute (or "run") the code, the so called python "interpreter" converts the human readable commands to commands being executed by the computer. The first line in your python script tells the interpreter that it is a python script.

## 7. Save Python Source Code from Jupyter Session

So in principle to write python programs you can use gedit (or any other editor; I use vi or vim) or jupyter or a combination of it. Jupyter allows you to save your jupyter notebook commands as a python script. To **save your python source code**, use `File` → `Download as` → `Python (.py)`. You can check which files were created by clicking on the top of the jupyter window on `Home` and you can get back to your notebook session by clicking on the name of your jupyter session `tryjupyter` or `Untitled`. (In case you cannot find the tryjupyter.py and are on linux, check in `~/Downloads/`). Use `gedit` to look at the content of tryjupyter.py. Notice that the resulting `tryjupyter.py` has a lot of extra lines,

You can also load any python source code into your jupyter session. Make sure that `first_program.py` is in the same directory from which you started jupyter. You can load it into your jupyter notebook session by typing in the cell `%load first_program.py` and as before using `Shift` and `Enter` to execute the command.

## 8. Help

For any further information on python try `Help` from the jupyter menue.

## 9. Log Out of Jupyter Session

To finish your jupyter session, just click on the `x` for each session. In linux you might have to also use the command `pkill jupyter`   This kills any hanging jupyter notebook session.

## 10. Copy, Read, Run Sample Python Script (if time)

Get into your /phys338_s2019 directory. Next make a directory for python sample files, so

```
mkdir pythonsamples.dir
```

Get into this directory (using `cd`) and copy the following sample file into your directory, so type (Note: The ./ at the end is part of the command.)

```
cp ~kvollmay/share.dir/pythonsamples.dir/sample_inout.py ./
```

When you run the script it reads in two files, so also do following copy commands

```
cp ~kvollmay/share.dir/pythonsamples.dir/in1.dat ./
cp ~kvollmay/share.dir/pythonsamples.dir/in2.dat ./
```

Next make the `sample_inout.py` executable (see step6) and look at this python script and run it and check if the resulting print out makes sense.

## 11. Input/Output (if time)

**11a.** Write a program that prints on screen "Hello, Hallo, Hi" To do this, first copy `sample_inout.py` to another file, `classpython11a.py`, and modify the program accordingly.

**11b.** Write a program that reads in (from screen) the name of a fruit (e.g. banana) and its price (e.g. 0.5), and prints out a sentence which says `One` *fruitname* costs $ *cost*. For this example: `One banana costs $0.5`.

**11c.** There is a way how you could have avoided having to type the fruit and cost on screen, but saving your "input parameters" which you want to directly tell the program. You do this by writing for example the file `in11c` with the first line `banana` and the second line `0.5`. Now type in the command line e.g. `./classpyton11b.py < in11c`.