In-Class Work: Python

1. Copy & Make Executable& Run

This problem guides you through the first steps of writing and running a python-program:

Log into linuxremote1 (redo the steps 1.-3. of the Linux Exercise). Get into your `phys338_s2015` 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 period at the end is part of the command.)
`cp ~kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/samples.dir/sample_inout.py .`
Also copy the data-files
`cp ~kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/samples.dir/in1.dat .`
`cp ~kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/samples.dir/in2.dat .`
Look at sample_inout.py with
`cat sample_inout.py` or with `gedit sample_inout.py &` (or use any other editor.)
This is the text file ("source code") of a program. You do not need to understand the complete program right away. It will serve as a template for future programs. For now lets use it to get one program running. To be able to use this textfile like a command type

`chmod u + x sample_inout.py`

Now we run this program by typing:
`./sample_inout.py`
First the program prints some on the screen, then it asks you to type a float, that means a real number, so type in some real number, e.g. `24.3`, then enter any integer, e.g. `4`, the program reads in the two numbers and does some manipulations with it and prints. Continue answering the questions and try to understand what `sample_inout.py` does.

2. Input/Output

2a. Write a program that prints on screen "Hello, Hallo, Hi" To do this, first copy `sample_inout.py` to another file, `classpython2a.py`, and modify the program accordingly.

2b. 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`.

2c. 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 `in2c` with the first line `banana` and the second line `0.5`. Now type in the command line e.g. `./classpyton2b.py < in2c`.

3. Arithmetic Fun

3a. Write a program which reads in the birthyear and the current year and which calculates the age. (ignore months)

3b. Write a program which reads in 5 float numbers and determines the average.

In-Class Work continued (if enough time)

4. Repetitions & Input/Output 4a. Write a program which prints ten times "Hello" on the screen. (Yes, this is not (yet) very useful.:-)

4b. Write a program which prints again ten times "Hello", but this time into a file named "out4b.dat" (Hint: See `open(...)` in `sample_inout.py`) Check after running the program that the `out4b.dat` file was written correctly.

4c. Next we want to enable to share our programs with each other. Let's assume you had called your program for 4b "classpython4b.py" Make this file readable (accessible to be copied) for others by typing

```
chmod a+r classpython4b.py
```

Whenever I will ask you to make your programs available for others (e.g. at the next homework assignment) please copy your programs into your ~/share.dir and make the corresponding program readable with `chmod a+r classpython4b.py`
(or replace with the corresponding filename)

In-Class Work: Python

1. For our convenience for the rest of this course let us use the so called tcshell. (This way you will use the same settings as I do, so I will be able to prepare our inclass work consistent with your settings.) This means that we specify which set of linux-commands we will be able to use. The following steps you will have to do only once, from then on the default for your shell will always be the tcshell. First type in

```
changeshell
```

and then type

```
/bin/tcsh
```

The changes will be activated after about 15 minutes.

2. Input/Output (continued)

2b. 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`.

2c. 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 `in2c` with the first line `banana` and the second line `0.5`. Now type in the command line e.g. `./classpyton2b.py < in2c`.

3. Arithmetic Fun

3a. Write a program which reads in the birthyear and the current year and which calculates the age. (ignore months)

3b. Write a program which reads in 5 float numbers and determines the average.

4. Repetitions & Input/Output 4a. Write a program which prints ten times "Hello" on the screen. (Yes, this is not (yet) very useful.:-)

4b. Write a program which prints again ten times "Hello", but this time into a file named "out4b.dat" (Hint: See `open(...)` in `sample_inout.py`) Check after running the program that the `out4b.dat` file was written correctly.

4c. Next we want to enable to share our programs with each other. Let's assume you had called your program for 4b "classpython4b.py" Make this file readable (accessible to be copied) for others by typing

```
chmod a+r classpython4b.py
```

(You could use instead `chmod 644 classpython4b.py`). Whenever I will ask you to make your programs available for others (e.g. at the next homework assignment) please copy your programs into your ~/share.dir and make the corresponding program readable with `chmod a+r classpython4b.py`

(or replace with the corresponding filename)

5. Factorial

Write a program which calculates $N! = 1 * 2 * 3 * \ldots * N$ and prints out N and $N!$ for $N = 1, 2, \ldots, 20$.

6. Sum

Write a program which reads in an integer N and determines $1 + 2 + 3 + \ldots + N$ and prints the result. What do you expect analytically? Check with printing the expected value.

7. Integer Range (Decisions)

Read in an integer i in the range $0 \leq i \leq 30$. For the case of $0 \leq i < 10$ print on screen "A", if $10 \leq i < 20$ print "B" and otherwise print "C." Test your program by running it for several values of i.

8. Spring Break (Decisions)

Read in a date in 2015 (as two integers) and check if it is during spring break (March 9 - March 15).

9. Traffic Model (Decisions & Repetitions)

Copy the file
~kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/linux_python_intro.dir/road.dat
into your working directory. The file contains 100 integers describing a road of 100 lattice sites at a certain time. -1 means an empty site and a number ≥ 0 means a car with the corresponding speed in mi/h. (**Hint:** use from `sample_inout.py` the command `sp.loadtxt`)

9a. How many cars are on this road?

9b. What is the average velocity

$$\langle v \rangle = \frac{1}{N} \sum_{i=1}^{N} v_i$$

where v_i is the velocity of car i and there are N cars.

9c. What is the largest velocity ?

In-Class Work: Python

10. Intro to Arrays

10a. Set an one-dimensional array (vector) to be `A` = (2,1,0). Then print A on the screen.

10b. Define a 2d-array (matrix) `B` with the values of `B` to be

```
 3   5   8
 1   0   2
-1   3   1
```

Print `B` on the screen.

11. Random Walk in 2 Dimensions

The following tasks will be the first steps for programming our first in-class model, the random walk (in two dimensions here).

11a. Write a program that defines a 5x5 array. Set all values of this array zero:

```
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0
0  0  0  0  0
```

11b. Now add four lines to your program such that the values of this 5x5 array are as follows:

```
0  0  0  0  0
0  1  1  1  0
0  0  1  0  0
0  0  0  0  0
0  0  0  0  0
```

11c. Add to your program that it prints the lattice on the screen.

Solutions:

~ kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/linux_python_intro.dir/classpython5.py etc.

In-Class Work continued (if enough time)

11d. Define the length of the array, i.e. 5, as variable you set at the beginning of your program. This will allow us later to change the size of our lattice easily.

11e. (Tool for Fun) You can look at this array graphically by adding to your python script the following two lines (assuming that your array is called `lat`)

```
plt.imshow(lat,interpolation='nearest')

plt.show()
```

11f. (only for very advanced python programmers:) In past I used `DynamicLattice` to make movies. Can you make a movie with `imshow`?

Here how it works with `DynamicLattice`:
For single picture:
if data are in out4.dat then use on command line of terminal:
`DynamicLattice -nx 5 -ny 5 -matrix < out4.dat`
or if you print data directly on screen (without []): `classpython11f.py | DynamicLattice -nx 5 -ny 5 -matrix`
Here if you make a movie with `DynamicLattice`:
Use your program of 11d and add to it, that after writing the matrix you change each "0" to "1" and each "1" to "0". Print the matrix on screen, but ensure that there is exactly one empty line between the old array and the updated array. Then swap again all "0" and "1"s and print again the array on screen (also separated with an empty line). Do this for 50 swaps and then `classpython11f.py | DynamicLattice -nx 5 -ny 5 -matrix`

12. Contrast Filter (2d-Array) (if time)

Copy into your working directory
~ kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/linux_python_intro.dir/pic1.dat
This file contains a 10x10 matrix. Have a look at this matrix (picture) with `cat` and with `imshow`. Hidden in this picture is some pattern, which you can see if you make a "filter", i.e. if you change the picture to a picture which gives you more contrast: Write a program which reads in this 10x10 matrix and prints out into the file ("pic1contrast.dat") a different 10x10 matrix which replaces every number < 0.5 with -1.0 and every number ≥ 0.5 with $+1.0$. Look at pic1contrast.dat with DynamicLattice.

In-Class Work: Python

Please note that for today's class on user-defined functions with python, you may use (and recycle) the sample script:
`~kvollmay/classes.dir/phys338.dir/phys338_s2015.dir/samples.dir/sample_functions.py`

13. Hello World Function

Write a program which calls (that means uses) a function, which writes on screen "Hello World." (Yes, you would never write a function for this task, this is just to get used to functions.)

14. Pass On Variable into Function

Write a program which reads in from screen an integer `i`, then uses a function to determine `5 + 3*i` and after using the function then prints the result on the screen. (Warning: The function name should start with a letter, not with a number.)

15. Line Function

Write a program which reads in from screen two double variables, an intercept `intercept` and a slope `slope`. Write a function which let's you determine for any value `x` and for any values of `intercept` and `slope`, `y= intercept + slope * x`. Use your function to print on screen two columns `x` and `y` for `x=0,0.1,0.2,...,1.0`.

16. Function Which Passes Back Multiple Variables

16a. Write a program which reads in from screen two integers `i1` and `i2`. Write a function which determines and returns `i3=i1+i2` and `i4=3*i1+i2`. Check by printing on screen after the function call `i1,i2, i3` and `i4`.

16b. Add to your function that it also changes `i1` to `5*i1`. Check your function by printing on screen `i1,i2, i3` and `i4` after the function call.

17. Pass On Array into Function

Set a matrix to be

$$\begin{matrix} 1 & 2 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{matrix}$$

Write a function which determines for any matrix the sum of all of its entries. Use your function and print after the call of the function the result for the sum of all entries.