MA305 – Lab 1.
Warmup On Unix, Python and Gnuplot

Date: Thursday, 09/13/2018

It involves most of the skills/tools needed in computation: programming, Unix, editing, (compiling), executing, plotting, ... Pay attention to the results of your actions so you'll be learning. Don't worry if you don't understand everything yet, we'll study all these aspects in detail. Don't get frustrated, it gets easier fast!

A Python code is given (see page 3) that evaluates the function: $F(x) = a - bx^3$ at N equidistant points in $[0, 1]$, and outputs the pairs $(x, F(x))$. The user is asked to enter values for a, b, and N. DO THE FOLLOWING.

1. Move the mouse to the big xterm window, change directory to your working directory (∼/MA305) and type:

`$ mkdir Lab1` (to create a directory named Lab1)

`$ ls -F` (to see that the dir "Lab1" has been created)

2. Type the code in a file named "lab1.py" inside the dir Lab1. BE CAREFUL, no misprints allowed (or you'll be sorry later)!

`$ cd Lab1`

`$ vi lab1.py` then press: `i` to enter into the insert mode;

Look at the **HOW-TO-ESSENTIALS (everything you need in a page)** handout for **vi commands** and try to learn them. As you type, save your work frequently: [ESC] (to command mode) [:w] (write) (so if anything goes wrong you won't lose everything !) When you finish typing the code, press [ESC] (to make sure you are out of insert mode), save it [:w] and move the mouse to another xterm for the rest.

3. Run it from the command line:

`$ python lab1.py`

Enter the values of $a = 2$, $b = 1$ and $N = 6$. Press **Enter** after every entry.

You should get the screen output as follows.

```
   Thanks, will run with:  a= 2 , b= 1 , N= 6

       x              F(x)
   0.0000000       2.0000000
   0.2000000       1.9920000
   0.4000000       1.9360000
   0.6000000       1.7839999
   0.8000000       1.4880000
   1.0000000       1.0000000

 All Done, BYE !
```

4. The code "lab1.py" can also be run directly as an executable file. The first line (`#!/usr/bin/env python3`) in the code tells the command shell to call the interpreter. For this, we need to make sure the file "lab1.py" has executable permissions. Try:

```
$ chmod u+x lab1.py
$ ./lab1.py
```

5. Try various choices: `a = -1.0, 0.01, b = 0.1, 5.0, N = 20`.

6. The program also writes the values in a file named "out2.dat" appropriately formatted for plotting with a graphics tool, called **gnuplot**. All our machines have it installed (it comes with Linux). Read about **gnuplot** in the **HOW-TO-ESSENTIALS (everything you need in a page)** handout.
Note: Lines with # in first column are ignored by **gnuplot**, that's why we inserted # at the start of the first line written to "out.txt".
Use `gnuplot` to display the graph in each case: Move mouse to another xterm, make sure you are in MA305/Lab1 directory (`$ cd ~/MA305/Lab1`), type:

```
$ gnuplot
gnuplot> plot 'out.txt' with points
```

Press [q] (to close the plot)

```
plot 'out.txt' with linespoints
```

Check it by plotting the curve itself (say, for $a = 1, b = 0.1$):

```
gnuplot> plot 1-0.1*x**3 with lines
```

For easy comparison, plot your values and this curve on the same plot:

```
gnuplot> plot 'out.txt' with points, 1-0.1*x**3 with lines
```

Play with N, to see how many points you need to make them look like the same curve (the points on the line). Save this plot, name it `fig1.ps` and convert it to `fig1.pdf`.
(Note: [Ctrl]+[P] in **gnuplot** backtracks to previous commands, try it to save re-typing [!])

7. Now sit back and reflect on you what you learned...
To do computing, you need to:
- create a code (how can you do this?)
- compile and run the code (how?)
- look at the results and possibly plot them (how?)

8. Send us an email using the `mail` command with the message
(a) what values you used for a and b.
(b) how many points it took to make the curves "look the same".
`$ mail -s "305:lab1" 305` (sends mail message to recipient "305" with Subject "305:lab1")
Finish with a `.` (period) on a new line.

9. Submit the code `lab1.py` and the plot `fig1.pdf` through your course Canvas.
10. When you are finished, `exit` from the **browser**, **vi**, **gnuplot**, ... and finally `log out`.

All Done ! The Python code is in the next page.

```
#!/usr/bin/env python3
"""
================================================================================
MA305 - Lab 1: your name - date
Purpose: To evaluate F(x) = a-b*(x**3)  at N equidistant points in [0,1],
and output the pairs (x , F(x))  on the screen and in the file
"out.txt" for plotting.
================================================================================
"""
# Get input from user:
print('Please enter the values of a, b and N:')
a=input('a=')
b=input('b=')
N=input('N=')

# Print on screen:
print('\n Thanks, will run with:  a=',a,', b=',b,', N=',N)
print()
print(' #   x        F(x)')

# Change string input into number types:
a=float(a)
b=float(b)
N=int(N)

# Open a file (out.txt) for output:
out=open('out.txt','w')
print(' #   x        F(x)', file=out)

# Compute F(x) = a-b*(x**3)  and print  x   F(x)
dx=1/(N-1)
for i in range(N):
  xi = i*dx
  Fi = a - b*xi**3
  print('{0: 0.7f} {1: 0.7f}'.format(xi,Fi))
  print('{0: 0.7f} {1: 0.7f}'.format(xi,Fi),file=out)
out.close()

# Exit:
print('\n All Done, BYE !\n')
```