MA305 – CW #1
Warmup On Unix, Python and Gnuplot

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. Download the skeleton of the python code for the classwork "cw1.py" from Canvas. Open a terminal (**xterm**) and change directory (`cd`) to your working directory (`~/MA305/Classwork`) and type:

`$ mkdir CW1` (to create a directory named CW1)
`$ ls -F` (to see that the dir "CW1" has been created) `$ cd CW1`
`$ mv ~/Downloads/cw1.py .` (to mv the file "cw1.py" here)
`$ ls` (to see "cw1.py" is available here)

2. Open the file "cw1.py" with `vi`[1]-editor, replace "your name" and "date" by your actual name and today's date and type the python code from the assignment sheet (last page) completely. Be careful, no misprints allowed (or you'll be sorry later)!
`$ vi cw1.py` then press: `i` to enter into the insert mode;

3. Run it from the command line:
`$ python3 cw1.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 !
```

[1] 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.

4. The python code "cw1.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 "cw1.py" has executable permissions. Try:

```
$ chmod u+x cw1.py
$ ./cw1.py
```

Try various choices for: `a, b, N`

5. The program also writes the values in a file named "out.txt" 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/Classwork/CW1 directory, and type `gnuplot` in the command prompt:

```
$ cd ~/MA305/Classwork/CW1
$ 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 (i.e., run the code with different values of 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 !

```
gnuplot> unset key
gnuplot> set title "Classwork 1 - Plot"
gnuplot> set xlabel "x"
gnuplot> set ylabel "y"
gnuplot> replot
```

`gnuplot> set term postscript` (will produce postscript output)
`gnuplot> set output "fig1.ps"`(output to the file "fig1.ps") `gnuplot> replot`(recreates plot but you don't see it, goes to file)
`gnuplot> set term x11` (resets the normal terminal so you can do more plotting)

6. Now quit the gnuplot (with `q`), and do the following in the terminal (the same work dir).

`$ ls` (to confirm "fig1.ps" is here!)
`$ evince fig1.ps` (or `ghostview fig1.ps`)

```
$ ps2pdf fig1.ps fig1.pdf
$ convert -quality 1000 fig1.ps fig1.jpg
```

7. Submit the code `cw1.py` and one of the plots (`fig1.pdf` or `fig1.jpg`) through Canvas.

8. Send us an email using the `mail` command
`$ mail -s "305-cw1" 305`
Write (a) what values you used for a and b, and (b) how many points it took to make the curves "look the same". Finish with a `.` (period) on a new line and press `Enter`.

9. When you are done, `exit` from the **browser**, **vi**, **gnuplot**, ... and finally `log out`.

```python
#!/usr/bin/env python3
"""
==============================================================================
MA305 - Classwork 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()
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')
```