# MA453/553 – Classwork 1.

## Warmup On Unix: Edit, compile/run, time and plot

*Computing $\pi$ using numerical integration, series sum, Monte Carlo approximation*

**Date: 09/06/2024**

---

In this classwork you will be computing $\pi$ using different methods: numerical approximations (left sum, right sum, midpoint, trapezoid and Simpson 1/3rd rules) to integrals, sum of alternating series and Monte Carlo method (dartboard algorithm). It involves most of the skills/tools needed in computation: programming, Unix, editing, (compiling), executing, timing, plotting, ...

---

**1.** Download the (loosely zipped) file "cw1.zip" from your Canvas course site. Save it in the working directory `~/MA453/Classwork/cw1`. Go (`cd`) to the working directory and `unzip` the file, list (`ls`) all files, and give a look (`cat, head, tail, more`) to all the ".py" files (Python scripts).

```
$ cd ~/MA453/Classwork/cw1
$ mv ~/Downloads/cw1.zip .
$ unzip cw1.zip
$ ls
$ cat num_int.py
$ cat pi.py
$ cat cw1.py
```

Create the dir "cw1" if it does not exist!

**2.** The code "num_int.py" approximates the integral $\int_0^1 4\sqrt{1-x^2}\,dx$. Run it with $n = 10, 100, 1000$.

```
$ python num_int.py
```

You can observe that the Simpson's method is not accurate. Now, open the file "num_int.py" and fix the function "simpson_rule(f,a,b,n)". The formula for Simpson's 1/3rd rule:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}\left[f(a) + 4\sum_{i=1,3,5}^{N-1} f(x_i) + 2\sum_{i=2,4,6}^{N-2} f(x_i) + f(b)\right],\ \Delta x = \frac{b-a}{N},\ x_i = a + i\Delta x$$

Run the code after fixing the two missing loops in the function. Did the Simpson's rule perform better? Open the file "num_int.py" and uncomment the lines 64-81 and run it again. You may read the syntax for **formatted output** and **plotting with `matplotlib.pylab`** later. What happens if you run with $n = 101$? Why?

**3.** The file "num_int.py" contains 5 different numerical approximations for a definite integral of type $\int_a^b f(x)\,dx$. You can use these functions as methods in the Python module `num_int`. Open an interactive Python shell and import the file as a Python module "num_int".

`$ python` (or `$ ipython`)
`>>> import num_int` (or `[x] import num_int`)

To see all the functions in it, type "help(num_int)"

```
>>> help(num_int)
```

Now you can use all the functions in "num_int.py" using the dot (.) method. For example, the function `f()` defined in this module ($f(x) = 4\sqrt{1-x^2}$) can be accesses as `num_int.f` and `midpoint_rule()` as `num_int.midpoint_rule()`. Try:

```
>>> num_int.f(0)
>>> num_int.midpoint_rule(num_int.f, 0, 1, 100)
```

Let us apply `simpson_rule` from "num_int.py" to approximate the integral $\int_0^1 \frac{4}{(1+x^2)}\,dx$ with 100 points.

```
>>> from num_int import simpson_rule as simp
>>> f = lambda x :  4/(1+x**2)
>>> simp(f, 0, 1, 100)
```

**4.** The file "pi.py" has two functions `alternating_series_pi` and `dart_board_pi` implementing

$$\pi = 4\arctan(1) \approx 4\sum_{n=1}^{N}(-1)^{n+1}\frac{1}{2n-1}, \qquad \pi \approx 4\left(\frac{\#\text{ of darts inside } x^2+y^2<1}{\text{total }\#\text{ of darts thrown}}\right).$$

You can use these functions the same way as we did with the functions in "num_int.py".

```
>>> import pi
>>> pi.alternating_series_pi(10000)
>>> pi.dart_board_pi(10000)
```

**5.** Finally, the code "cw1.py" uses the functions `midpoint_rule()`, `trapezoid_rule()` and `simpson_rule()` from the module file "num_int.py" and the functions `alternating_series_pi` and `dart_board_pi` from the file "pi.py". It requires the number of points as a command line argument in to run. Try:

```
$ python cw1.py 1000
```

Check the lines 5-6 (`import` statements) and the lines 29-31 (function calls). Make the appropriate changes following what we did in **3** and **4** in the interactive Python shell.

**6.** Make a log of your work using the Unix command `script`.

(i)
```
$ script
$ cat cw1.py
$ chmod u+x cw1.py
$ ./cw1.py 1000000
$ exit
```
(exit from script)

(ii) Rename file "typescript" to "cw1script.txt".
```
$ cp typescript cw1script.txt
```

(iii) Edit and CLEAN up the "cw1script.txt" file.
```
$ vi cw1script.txt
```
You can delete all the annoying control characters `^M` , `^G` , `^[` manually. In the command mode of **vi-Editor**, `x` deletes single character, `dw` deletes a word and `dd` deletes a line. You can also search a string and replace it by another string globally with a single command. For example the following command within `vi` typing the following command
```
:1,$s/string1/string2/g
```
tells in lines 1 to last($), substitute the "string1" by "string2" globally (all occurrences).
To delete `^M` , `^G` , `^[`, within `vi` press [ESC]+[:] and type the following at the command

prompt [:] at the lower left corner of the file.

`:1,$s/^V^M//g` (`^V^M` is [CTRL V CTRL M])

`:1,$s/^V^G//g` (`^V^G` is CTRL V CTRL G)

`:1,$s/^V^[//g` (`^V^[` is CTRL V CTRL [ )

**Note:** To be able to insert `^M` [CTRL V CTRL M] in MobaXterm, you may need to redefine the MobaXterm's hot key (CTRL+M) to something else.

**7.** Submit the script "cw1script.txt" using the following mail command from **wxsession**.

```
$ ssh you@wxsession.db.erau.edu
$ cd ~/MA453/Classwork/cw1
$ mail -s "MA453:cw1" 453 < cw1script.txt
```

**8.** Zip all the files and submit it through your course Canvas.

```
$ zip you_cw1.zip num_int.py, pi.py, cw1.py, fig1.py
```

**Note:** If you are working remotely using MobaXterm/Mac/Linux, you need to copy the zipped file to your computer first. Open an `xterm` in your local host and type the following:

```
$ scp you@wxsession.db.erau.edu:MA453/Classwork/cw1/you_cw1.zip .
```

---