MA305 – Lab $\sqrt{4}$.

Loops and Control Flow in Python

Due: 09/29/2018

Writing a program involves several steps: thinking, deciding what needs to be done, organizing the tasks into an algorithm on paper, translating to computer language on paper, executing it on paper, thinking, ... before you even sit in front of the computer...

---

1. The square root of a real number ($p > 0$), can be approximated by the limit of the following sequence (*Babylonian algorithm*).

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{p}{x_n}\right) \to x* \approx \sqrt{p}.$$

Write a script in Python to implement this algorithm (with x_0 as the initial guess) and compare your the result with the "exact" answer provided by the `math.sqrt(p)` .

Here is an outline of the steps to be done.

(i) First, perform three iterations of this algorithm for computing the square-root of $p = 4$, starting with the initial guess $x_0 = 1$ (by hand, on paper).

(ii) How do you decide when to break the iterations? Well, we must choose a pre-specified error tolerance. How do you compute error if the exact root is not known? You may use these stopping criteria in your code.

$$\text{(a) } |x_{n+1} - x_n| < \tfrac{1}{2} \times 10^{-14} \qquad \text{(b) } \frac{|x_{n+1} - x_n|}{|x_n|} < \tfrac{1}{2} \times 10^{-14}$$

(iii) Decide what are the main variables, name them, which one(s) are input (to be read in), and which ones are output (to be determined by the program).

(iv) Construct the algorithm: On a sheet of paper, write the tasks that must be performed (request value from user, read it in, calculate, etc), as a sequence of steps. Your program should also print the input data, so that we'll know what input produced the output.

(v) Translate each task into a Python statement. Be careful how you write the numbers, arithmetic operators, precedence, iterations (for/while loops), comparison operators and control flow (if/then/else statements).

(vi) Assemble the pieces to a complete Python code, still on paper, with comments and everything needed. Use formatting the numbers to display your output nicely.

2. This is a new Lab, so we start by creating a new directory "MA305/Lab2", open a file "lab2.py" with vi editor:

```
$ cd ~/MA305
$ mkdir Lab2
$ cd Lab2
$ vi lab2.py
```

and type it in.

Reminder: save your work often with `:w` which writes the buffer to disk, so if anything goes wrong, you won't loose everything... Think through the typed version of the code, trying to catch misprints and any other "bugs", ...else you'll be sorry later...

3. Run your code.

```
$ python3 lab2.py
```

or

```
$ ./lab2.py
```

(make sure it has executable permission)

4. Test the code with different values of $p = 4, 44, 444, 4444, 44444$ for the same initial guess $x_0 = 1$ and record the results (square roots and number of iterations it took to converge). Do you think the stopping tests (a) and (b) are reliable to be used when the exact value of $\sqrt{p}$ is unavailable. What is your conclusion?

5. Make a log of your work using the Unix command `script` as in classwork 2.

(i)
```
$ script
$ cat lab2.py
$ python3 lab2.py
.......
.......
$ exit
```
[what does this do? why needed?] (for `$ cat lab2.py`)

(exit from script).

(ii) Edit and CLEAN up the typescript file.
Note: To remove all those annoying `^M` control characters from the typescript file: type the following in the command line within `vi`:

```
:1,$s/^V^M//g
```

(`^V^M` is [CTRL V CTRL M])

This says in lines 1 to last($), substitute `^M` by nothing, globally (all occurancies in a line). The `^V` allows insertion of the control character `^M`.

(iii) Rename file "typescript" to something like "lab2script.txt".

6. Send us an email message with the "lab2script.txt" file as an attachment directly with the following mail command:

```
$ mail -s "305:lab2" 305 < lab2script.txt
```

7. Modify your code `lab2.py` and write another program (name it `lab2a.py`) to calculate the *arithmetic-geometric mean* (agm) of two numbers , x and y, defined by the limit of the sequences:

$$\begin{aligned} x_{n+1} &= \tfrac{1}{2}(x_n + y_n) \\ y_{n+1} &= \sqrt{x_n y_n} \end{aligned}$$

starting with x_0 and y_0. Both the sequences converge to the same number, denoted by $\text{agm}(x, y)$. Use this program to determine the *Gauss's constant*, $G = 1/\text{agm}(1, \sqrt{2})$.

8. Submit the code `lab2.py` and `lab2a.py` through your course Canvas.

All DONE, *have a nice weekend !*