MTH 3300
Spring 2018
E. Fink
March 12, 2019

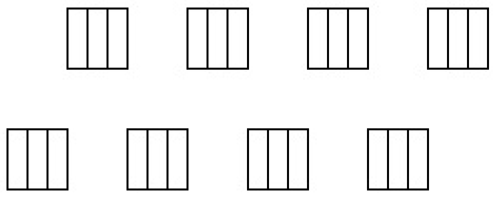
Homework 6

Directions: Download the template files I have provided on Blackboard. Then open Spyder, load these template files, and write the following programs. Submit your source code to me via Blackboard, in `.py` format; do NOT send any other files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

Be sure to read the SPECIFICATIONS carefully for all problems! And write comments!

1) windows.py

Write a program that allows the user to input two positive integers, `num_windows` and `panes`, and then uses the `turtle` module to draw a picture that looks like this:



This should be the picture if the user specifically enters 8 for `num_windows` and 3 for `panes`, since there are 8 big rectangles ("windows") arranged into an "up-down-up-down" pattern, and each one of those 8 rectangles is divided into 3 smaller subrectangles ("panes").

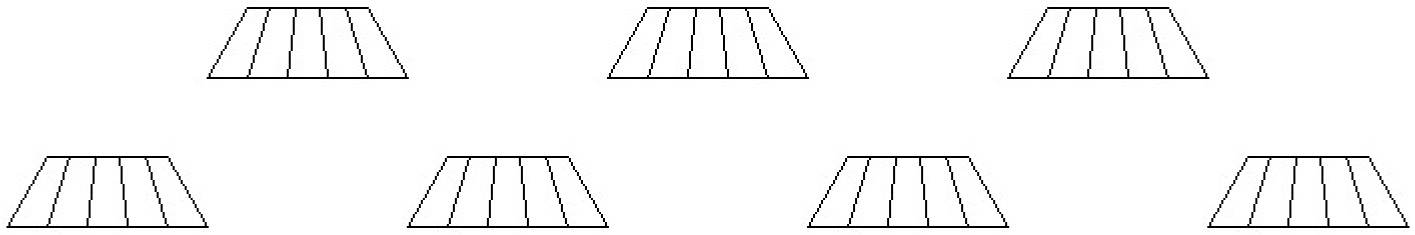
Here are some more precise guidelines.

1. Your windows should lie on two vertical levels, one higher and one lower. Looking from left to right, the odd-numbered windows should be on the lower level, and the even numbered ones should be on the higher level.
2. The windows should be disconnected – there should be no pen mark connecting one window to the next.
3. Each of the panes in each of the windows should **not** be a square – in my example, each of the 24 panes is a tall, thin rectangle, and yours should be as well.
4. The final position of the turtle can be anywhere you like.

Specifications: your program must

- ask the user to input two positive integers, `num_windows` and `panes`.
- use the `turtle` module to draw an arrangement of "windows" like the one shown above. The number of windows should be given by `num_windows`, and the number of panes in each window should be given by `panes`. The windows must behave as described in points 1) - 4) above.

***Challenge**: instead of making your windows rectangular, make them trapezoidal, e.g,*



Make sure the edges of each pane are slanted in a continuous manner. For an extra challenge, try to do this only using the turtle commands we've mentioned in class (`.forward()`, `.left()`, `.right()`, `.penup()`, `.pendown()`).

2) minimum.py

A polynomial function is called *monic* if its leading coefficient is equal to 1. So, $p(x) = x^2 + 7x - 13$ is a monic polynomial, while $p(x) = 2x^2 - x + 1$ is not. A monic fourth-degree polynomial $p(x)$ always has an absolute minimum value (why????), which will occur at an x-value where $p'(x) = 0$.

In this problem, you will write a program which attempts to find the minimum value of a monic fourth-degree polynomial $p(x)$, by using Newton's method (described below) to solve $p'(x) = 0$.

Newton's method is a method to approximate solutions to an equation $f(x) = 0$ very quickly. It works by starting with a (more-or-less random) guess x_0 for a solution, and then coming up with better and better approximations x_1, x_2, x_3, etc., using the following process:

$$\begin{aligned} x_1 &= x_0 - f(x_0)/f'(x_0) \\ x_2 &= x_1 - f(x_1)/f'(x_1) \\ x_3 &= x_2 - f(x_2)/f'(x_2) \end{aligned}$$

and so on and so forth, with

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

in general. For reasons which I'll explain in lecture, x_1 will usually be close to a solution, and x_2 will be closer, and x_3 closer still, etc.

Let's try an example: we'll solve $x^2 - 4x + 1 = 0$. Here, $f(x) = x^2 - 4x + 1$. We start with a guess of $x_0 = 1$.

Then $x_0 = 1$; $f(x_0) = -2$; $f'(x_0) = -2$; and so

$$x_1 = 1 - (-2)/(-2) = 0.$$

Then $x_1 = 0$; $f(x_1) = 1$; $f'(x_1) = -4$; and so

$$x_2 = 0 - (1)/(-4) = 0.25.$$

Then $x_2 = 0.25$; $f(x_2) = 0.0625$; $f'(x_2) = -3.5$; and so

$$x_3 = 0.25 - f(0.25)/f'(0.25) = 0.267857142 \text{ (approximately).}$$

Then:

$$x_4 = 0.267857142 - f(0.267857142)/f'(0.267857142) = 0.267949190 \text{ (approximately).}$$

And so on. The actual value of one solution to 9 places is 0.267949192, so we were already at 8 decimal place precision after only four iterations – not bad.

Your program should work as follows. To enter in a the monic fourth-degree polynomial $p(x) = x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$, your program should ask the user to input c_0, c_1, c_2, c_3 for the polynomial. (The user doesn't enter c_4, since it is always 1.) Your program will also ask the user the enter in a "guess" for the x-value of the minimum point, which will be x_0.

To avoid confusion, let's use the name $D(x)$ to denote the derivative of $p(x)$; so

$$D(x) = 4x^3 + 3c_3x^2 + 2c_2x + c_1$$

Your program will attempt to solve $D(x) = 0$ using Newton's method as follows. Starting with the x_0 input by the user, the program will compute values x_1, x_2, x_3, etc., as described above, using a `while` loop. The program should keep producing new x_i values, using the formula $x_{n+1} = x_n - D(x_n)/D'(x_n)$, until it finds one where $D(x_i)$ is very close to 0 – in this problem, we'll say that $D(x_i)$ is very close to 0 when $|D(x_i)| < 10^{-8}$. At that point, we will take this value x_i to be the solution of $D(x_i) = 0$.

Finally, plug that value x_i into $p(x)$, and that will be the minimum value of $p(x)$. (We hope. It is possible that Newton's method fails, and it's also possible that Newton's method finds a critical number of $p(x)$ other than the x-value of the minimum. However, it frequently finds the minimum correctly and quickly.)

So, for example, when you run the program, it may look like this:

```
Enter x^0 coefficient: 5
Enter x^1 coefficient: 2
Enter x^2 coefficient: 3
Enter x^3 coefficient: 8
Enter guess x_0: 0.1
x-value of min = -5.754430896835747
Minimum value of p(x) = -335.0598405925539
```

Here, the user entered 5, 2, 3, 8 and 0.1. The polynomial $p(x)$ is $5 + 2x + 3x^2 + 8x^3 + x^4$. The equation $D(x) = p'(x) = 2 + 6x + 24x^2 + 4x^3 = 0$ happens to have exactly one solution, which is roughly -5.754; if you plug that into $p(x)$, you get approximately -335.06. (Wolfram Alpha can verify this!)

Don't overthink Newton's method – all this detail aside, the code isn't very complex at all! Hint: you can use the same variable for x_0, x_1, x_2, etc., because as soon as you know the value of, say, x_{12}, you don't need to know the value of x_{11} anymore.

Specifications: your program must

- ask the user to input 4 coefficients, representing the constant, x, x^2, and x^3 coefficients of a polynomial, $p(x)$.
- ask the user also to input a "guess" value, x_0.
- estimate a solution to the equation $D(x) = 0$ (where $D(x)$ is another name for $p'(x)$) by computing x_1, x_2, x_3 etc., using the formula described above; the program should continue computing new x_i values as long as $|D(x_i)| \geq 10^{-8}$.
- once the final value of x_i is computed, print it out, as well as the value of $p(x_i)$ (which should likely be the minimum possible value of $p(x)$).

Challenge: *can you devise a way to find a value which is* definitely *the global minimum, as opposed to a merely local minimum or local maximum or neither? It may help that our problem only concerns fourth degree monic polynomials. The program may still be vulnerable to Newton's method not converging, but this should be rare.*