MTH 3300
Fall 2022

Programming Problem 1: population.py

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 via Gradescope, in `.py` format. READ THE INSTRUCTIONS on submitting your work in the Course Documents section of Blackboard.

Specify collaborators/resources or explicitly specify that none were used in the comments at the top of your `.py` file. Failure to include this will result in a zero on the assignment.

Be sure to read the SPECIFICATIONS carefully! And write comments!

Suppose that you are a demographer who wants to model the growth of a nation's population over time. A simple model for this growth is the standard exponential model.

$$P = P_0 e^{rt}$$

where P_0 is the initial population at time $t = 0$, r is the relative growth rate in percent per year (expressed as a decimal), t is the time elapsed in years, and P is the population at time t. Also, e is the base of the natural logarithm ($e \approx 2.718$).

Write a program that prompts the user to enter a value for the initial population. The program should then do the following **two** times: it will ask for a period in years, and a relative growth rate; the program will then compute the new population after that many years have elapsed, using the population growth formula provided above. This should be done *cumulatively* – that is, the end population for the first iteration should be used as the initial population for the second iteration.

So, for example: suppose that the user enters an initial population of 300, a first period of 4 years, and a first growth rate of 1.2%. Then the population at the end of the first period would be 314.75119659734116, since

$$300 \cdot e^{(0.012)(4)} = 314.75119659734116.$$

Suppose then that the second period was 2.5 years, and the second growth rate is 5%; then the population at the end of the second period will be

$$314.751 \cdot e^{(0.05)(2.5)} = 356.65983152520965.$$

That last number rounded to six decimal places, 356.659832, is what the program should display as the final population. (Don't worry that these numbers aren't integers.)

Finally, calculate the overall percentage growth from the initial population to the final population as follows:

$$\text{percentage growth} = \frac{\text{final population} - \text{initial population}}{\text{initial population}} \times 100$$

and display it at the end, rounded to three decimal places, along with the overall time period during which that percentage growth occurred (see sample run of the program below).

For the values above, the percentage growth would be

$$\frac{356.65983152520965 - 300}{300} \times 100 = 18.88661050840322$$

and your program should display 18.887.

So, when you run your program, it should look like this:

```
Enter the initial population: 300
Enter the first time period in years: 4
Enter the first growth rate as a percentage: 1.2
Enter the second time period in years: 2.5
Enter the second growth rate as a percentage: 5
The ending population is 356.659832
The overall growth in the 6.5 year period is 18.887 percent
```

(The numbers after each `:`, like `300` and `4` and `1.2`, are user entries; the rest should be produced by the program.)

Hints: make sure you try calculating my sample values by hand before programming to make sure you understand the task! Finally, if you are tempted to figure out a way to get Python to "repeat itself three times" – don't do that; write similar code three times over (we'll learn about repetition soon enough).

Specifications: your program must

- ask the user to enter the initial population.
- ask the user to enter years and a growth rate two times. The program should accept the growth rates input as **percents** (but input without the percent sign – so "3.5%" will be input to the program as just `3.5`).
- compute the **final** population as described above and displays it to six decimal places.
- compute the percentage growth as described above and displays it to three decimal places, along with the total timer period over which it occurred.

Note: This problem has two visible test cases on Gradescope. You will see if your code passes these two test cases. In addition, it has two invisible test cases, which you will not be able to see until after grades are released (you will not see if your code passed or did not pass these). So make sure you run additional tests on your own to cover all the possible inputs described! (You must work out the correct output in your test cases.)

Your input prompts must match the ones given above *exactly* (including spaces) to pass the test cases on Gradescope.