Magical Creatures -- 1420 Version

Read the instructions carefully. Not following the instructions will result in you not getting the credit you want for the assignment.

Learning Objectives

1. Practice using loops, if statements and functions in your code
2. Use a computer program to write a simple simulation
3. Use command line input to the program

Problem To Solve

On the mystical planet of Archaemeda IV, the population of magical creatures is rapidly growing. The chief magistrate and his advisors want to study the growth of the population over time to ensure the planet's resources can sustain the creatures. They decide to use the Logistic Equation to model the population growth. As Head Programmer, you have been asked to write a Python program that simulates this population growth using the Logistic Equation.

The Logistic Equation is defined as:

$$x_{n+1} = r * x_n * (1 - x_n)$$

Where x_n is the population at time n, r is the growth rate, and x_{n+1} is the population at time $n+1$.

In a file called file `simulation.py` implement a program that simlautes the population growth using the Logistic Equation and writes the output to the console.

Below we show the first 20 lines for an example run.

```
python3 simulation.py 0.1 2.5 20
```

where the three values on the command line are:

- initial population as a percentage of the total population $0 < x_0 < 1$
- growth rate is a number in the range $0 < r < 4$
- number of iterations is a positive integer

It should look like this:

```
0       0.100
1       0.225
2       0.436
3       0.615
4       0.592
5       0.604
6       0.598
7       0.601
8       0.600
9       0.600
10      0.600
11      0.600
12      0.600
13      0.600
14      0.600
15      0.600
16      0.600
17      0.600
18      0.600
19      0.600
20      0.600
```

The first value is the time step, and the second value is the population percentage value at that time. Keep in mind that the actual value may be different than what is displayed because of rounding for display.

How to Test Your Code

- Do you have the logistic function in it's own function?
- Are values dispalyed to 3 decimal places?
- Does your code work as prescribed when you input:
    - negative numbers (-1)
    - 0
    - 1.0 or greater for intial population?
    - growth rate 4 or greater?
    - forget to specify an output file?
    - no command line parameters at all?
- Do you have a main function with conditonal execution?

You may think of other aspects of your program and its design that you can test.

Style

In your terminal, execute the commands below to evaluate the style of your code. Fix any problems that may be reported.

```
ruff format simulation.py
ruff check simulation.py
```

How to Submit

Using Codespaces online IDE in your assignment GitHub's repository, submit by pressing the Commit button and entring a commit message.

Grading

Criterion	100 (Mastery)	85 (Proficient)	70 (Basic)	50 (Needs Improvement)
Program works with different parameter values, not just the example ones	Program consistently works with a wide range of parameter values	Program works with most parameter values	Program works with some parameter values	Program only works with example parameter values
Meaningful loops and conditionals	Contains well-structured and meaningful for loops, while loops, if statements, and user-defined functions	Contains mostly meaningful loops and conditionals	Contains some meaningful loops and conditionals	Contains few or no meaningful loops and conditionals
Workspace contains `simulation.py`	`simulation.py` is present and well-organized	`simulation.py` is present but somewhat disorganized	`simulation.py` is present but poorly organized	`simulation.py` is missing
Main function with conditional execution	Main function is present with clear conditional execution; no global code	Main function is present with some conditional execution; minimal global code	Main function is present but lacks conditional execution; some global code	Main function is missing or contains significant global code
Logistic Equation in its own function	Logistic Equation is clearly defined in its own function	Logistic Equation is defined but not clearly separated	Logistic Equation is partially defined in its own function	Logistic Equation is not defined in its own function
Captures inputs from command line arguments	Program captures all required inputs from command line arguments	Program captures most required inputs from command line arguments	Program captures some required inputs from command line arguments	Program does not capture inputs from command line arguments

Criterion	100 (Mastery)	85 (Proficient)	70 (Basic)	50 (Needs Improvement)
Validate ranges for numerical inputs	Program thoroughly validates ranges for all numerical inputs	Program validates ranges for most numerical inputs	Program validates ranges for some numerical inputs	Program does not validate ranges for numerical inputs
Code style formatted according to `ruff`	Code style is consistently formatted according to `ruff`	Code style is mostly formatted according to `ruff`	Code style is somewhat formatted according to `ruff`	Code style is not formatted according to `ruff`
Passes `ruff` checks	Code passes all `ruff` checks for syntax errors, violations, and potential bugs	Code passes most `ruff` checks	Code passes some `ruff` checks	Code does not pass `ruff` checks

Students should strive for mastery level. Lower than that indicates areas where more practice is needed or more learning is needed. Code score is the average of the individual feature scores.

Total score is 1/4 * style score + 3/4 * code_score.