Problem Set 5: Investing for Retirement

For this assignment, you will use Python and Visual Studio Code to create a computer program. Submit a Python program script file with the solutions (file extension is py) in Blackboard by Sunday at 11:59 PM. How much retirement planning have you accomplished so far? There are various methods of approximation to estimate how much one might need to save or invest for retirement. The problems in this homework will require successive approximations to determine savings and expensed values from a retirement investment plan.

Script Requirements

When writing and saving your programming files, ensure that you heed the instructions. Specific things must be placed in the programming code for all programming coursework. The beginning of the script file must include the assignment name, the purpose of the programming code, the date it was written or revised, and who wrote the code. These comments can be written as line comments or a commenting block. Figure 1 depicts an example of what this might look like in your file. When saving the program for the first problem, name the file **`assignment_5_1.py`**. The second, third, and fourth problems shall be saved with the file names **`assignment_5_2.py`** and **`assignment_5_3.py`**, and **`assignment5_4.py`**, respectively. When you have completed and tested your script files, submit all of the script files for this assignment in Blackboard.

All analysis begins with a problem to solve, a question to answer, or something to prove (you might call this a hypothesis). The following information establishes the requirements of the program and some hints for completing this work. There are four problems that you need to solve for this assignment.

Figure 1

Content Needed in Every Script File

```
"""
Assignment 5 Problem 1: use successive approximation to find
the value of an investment given salary, savings, rate, and
duration
September 22, 2022
John Smith
"""
```

Note. This contains the name of the assignment (Assignment 5 Problem 1), purpose of this assignment (successive approximation to find), date (September 22, 2022), and the name of the developer (John Smith).

Problem One

Many organizations have some sort of a retirement plan. For example, an employer may contribute an amount to your retirement based on your current salary and match any additional amount that the employee invests. For your solution, assume that the organization invests 5% of the employee's salary on their behalf. They also match the rate an employee invests, up to an additional 5% of the employee's salary. You cannot determine how much this retirement plan will be worth year after year into the future. To solve this problem, create a function named **`fixedInvestor`**. This function has four required arguments: the employee salary, the employee's personnel investment rate as a percentage of their salary, a fixed growth rate of the retirement plan, and the number of years to evaluate the growth.

```
def fixedInvestor(salary, p_rate, f_rate, years):
    """
    Describe what the function does, what type of information goes
    into the arguments, and what is returned.
    """
```

Table 1

Calculating Principal and Interest for Problem One

Yr	Formula	Example	Total
Y1	salary*(savings + company)	50,000* .15	7500.00
Y2	salary*(savings + company) + Y1*growth rate	50,000* .15 + 7500 * 1.05	15375.00
Y3	salary*(savings + company) + Y2*growth rate	50,000* .15 + 15375 * 1.05	23643.75

Note. This table depicts the calculations for the principal and interest changes when growth is compounded annually. For this example, the salary is documented as $50,000. The employee's savings rate is 5%, with a 5% matching and a 5% investment on the employee's behalf. The retirement plan growth rate is 5%. Notice that it's calculated with 1.05, not .05.

This function will return a list with the total value of the retirement plan at the end of each year. For the solution, the retirement account is compounded annually. The fixed growth rate of the retirement plan is not used when calculating the first year's return. The formula is the salary multiplied by the rate the employee invested, the rate employer invested, and the rate that the employer matched. In the second year, interest is earned by calculating the earnings from the first year multiplied by the investment growth rate (see Table 1 for examples of the year-over-year computed value).

Problem Two

The next problem that needs to be resolved builds on the solution to the first problem. It's been identified that the growth rate of an investment may not be static—that it may change over time. For this solution, create a function named **`variableInvestor`**. This function will have three arguments, similar to problem one. There is one distinct change. The growth rates of the investment plan will be fed to this function as a list. There will be a rate for each year of

investing (i.e., if five years of calculations are needed, then the list of growth rates will contain five values). If you consider the list, the rate for the first year can be any value because interest is not earned. The function will no longer use the argument that represents the number of years needed. (The list length will be sufficient to determine the number of years.)

```
def variableInvestor(salary, p_rate, v_rate):
    """
    Describe what the function does, what type of information goes
    into the arguments, and what is returned.
    """
```

Problem Three

Once retired, it would be important to know how much an account is changing because the account is still earning interest, but fixed payments are distributed from the retirement account to the retiree. To solve this problem, create a function named **`finallyRetired`**. This function has three arguments. The first is the amount of money in the retirement account. The second value is a list of the annual rates of return, and the third is the annual amount expensed to the retiree. When retiring, there is already an established principal, so there is interest earned every year in this solution.

```
def finallyRetired(saved, v_rate, expensed):
    """
    Describe what the function does, what type of information goes
    into the arguments, and what is returned.
    """
```

Problem Four

How much can be spent from a retirement account every year to use all of the money? To answer this question, create a function named **`maximumExpensed`**. This function has five required arguments: salary, the percentage rate saved, a list of annual return rates while investing, a list of annual return rates while retired and a value for epsilon. The number of values in the list of annual return rates while investing represents the number of years worked. The

number of values in the list of return rates while retired represents the years between retirement and bankruptcy (or when one might expect to pass on).

```
def maximumExpensed(salary, p_rate, workRate, retiredRate, epsilon):
      """
      Describe what the function does, what type of information goes
      into the arguments, and what is returned.
      """
```

Use the idea of binary searching to find a value for the amount expensed when retired so that the value of the retirement account is very close to zero. Keep in mind that it may be a negative number. (The values represent approximations of unrealized gains and losses, so overdrawing the account by a small amount is acceptable). Use the solution from problem two to determine the value saved. Likewise, use your solution to problem three to determine how much money remains at the end. While your function is searching for the solution, print the expensed and remaining value in a complete sentence to the console each time the expensed value changes.