Problem Set 1

Handed out: Monday, September 12, 2016.

Due: 11:59 PM, Tuesday, September 20, 2016

This problem set will introduce you to using control flow in Python and formulating a computational solution to a problem. It will also give you a chance to explore bisection search. This problem set has **three** problems. You should save your code for the first problem as **ps1a.py**, the second problem as **ps1b.py** and the third problem as **ps1c.py**, and make sure to hand in all three files. Don't forget to include comments to help us understand your code!

Collaboration

You may work with other students; however, each student should write up and hand in his or her assignment separately. Be sure to indicate with whom you have worked in a comment at the start of each file.

Before You Start: Read the Style Guide

Read the style guide sections 1, 2, and 3.

Part A: House Hunting

You have graduated from MIT and now have a great job! You move to the San Francisco Bay Area and decide that you want to start saving to buy a house. As housing prices are very high in the Bay Area, you realize you are going to have to save for several years before you can afford to make the down payment on a house. In Part A, we are going to determine how long it will take you to save enough money to make the down payment given the following assumptions:

1. Call the cost of your dream home **total_cost**.
2. Call the portion of the cost needed for a down payment **portion_down_payment**. For simplicity, assume that portion_down_payment = 0.25 (25%).
3. Call the amount that you have saved thus far **current_savings**. You start with a current savings of $0.
4. Assume that you invest your current savings wisely, with an annual return of **r** (in other words, at the end of each month, you receive an additional **current_savings*r/12** funds to put into your savings – the 12 is because **r** is an annual rate). Assume that your investments earn a return of r = 0.04 (4%).
5. Assume your annual salary is **annual_salary**.
6. Assume you are going to dedicate a certain amount of your salary each month to saving for the down payment. Call that **portion_saved**. This variable should be in decimal form (i.e. 0.1 for 10%).
7. At the end of each month, your savings will be increased by the return on your investment, plus a percentage of your **monthly salary** (annual salary / 12).

Write a program to calculate how many months it will take you to save up enough money for a down payment. You will want your main variables to be floats, so you should cast user inputs to floats.

Your program should ask the user to enter the following variables:

1. The starting annual salary (annual_salary)
2. The portion of salary to be saved (portion_saved)
3. The cost of your dream home (total_cost)

Hints

To help you get started, here is a rough outline of the stages you should probably follow in writing your code:

- Retrieve user input. Look at input() if you need help with getting user input. For this problem set, you can assume that users will enter valid input (e.g. they won't enter a string when you expect an int)
- Initialize some state variables. You should decide what information you need. Be careful about values that represent annual amounts and those that represent monthly amounts.

Try different inputs and see how long it takes to save for a down payment. **Please make your program print results in the format shown in the test cases below.**

Test Case 1

```
>>>
Enter your annual salary: 120000
Enter the percent of your salary to save, as a decimal: .10
Enter the cost of your dream home: 1000000
Number of months: 183
>>>
```

Test Case 2

```
>>>
Enter your annual salary: 80000
Enter the percent of your salary to save, as a decimal: .15
Enter the cost of your dream home: 500000
Number of months: 105
>>>
```

Part B: Saving, with a raise

Background

In Part A, we unrealistically assumed that your salary didn't change. But you are an MIT graduate, and clearly you are going to be worth more to your company over time! So we are going to build on your solution to Part A by factoring in a raise every six months.

In **ps1b.py**, copy your solution to Part A (as we are going to reuse much of that machinery). Modify your program to include the following

1. Have the user input a semi-annual salary raise **semi_annual_raise** (as a decimal percentage)
2. After the 6^{th} month, increase your salary by that percentage. Do the same after the 12^{th} month, the 18^{th} month, and so on.

Write a program to calculate how many months it will take you save up enough money for a down payment. LIke before, assume that your investments earn a return of **r** = 0.04 (or 4%) and the required down payment percentage is 0.25 (or 25%). Have the user enter the following variables:

1. The starting annual salary (annual_salary)

2. The percentage of salary to be saved (portion_saved)
3. The cost of your dream home (total_cost)
4. The semi-annual salary raise (semi_annual_raise)

Hints

To help you get started, here is a rough outline of the stages you should probably follow in writing your code:

- Retrieve user input.
- Initialize some state variables. You should decide what information you need. Be sure to be careful about values that represent annual amounts and those that represent monthly amounts.
- Be careful about when you increase your salary – this should only happen **after** the 6th, 12th, 18th month, and so on.

Try different inputs and see how quickly or slowly you can save enough for a down payment. **Please make your program print results in the format shown in the test cases below.**

Test Case 1

```
>>>
Enter your starting annual salary: 120000
Enter the percent of your salary to save, as a decimal: .05
Enter the cost of your dream home: 500000
Enter the semi-annual raise, as a decimal: .03
Number of months: 142
>>>
```

Test Case 2

```
>>>
Enter your starting annual salary: 80000
Enter the percent of your salary to save, as a decimal: .1
Enter the cost of your dream home: 800000
Enter the semi-annual raise, as a decimal: .03
Number of months: 159
>>>
```

Test Case 3

```
>>>
Enter your starting annual salary: 75000
Enter the percent of your salary to save, as a decimal: .05
Enter the cost of your dream home: 1500000
Enter the semi-annual raise, as a decimal: .05
Number of months: 261
>>>
```

Part C: Finding the right amount to save away

In Part B, you had a chance to explore how both the percentage of your salary that you save each month and your annual raise affect how long it takes you to save for a down payment. This is nice, but suppose you want to set a particular goal, e.g. to be able to afford the down payment in three years. How much should you save each month to achieve this? In this problem, you are going to write a program to answer that question. To simplify things, assume:

1. Your semi-annual raise is .07 (7%)
2. Your investments have an annual return of 0.04 (4%)
3. The down payment is 0.25 (25%) of the cost of the house
4. The cost of the house that you are saving for is $1M.

You are now going to try to find the best rate of savings to achieve a down payment on a $1M house in 36 months. Since hitting this exactly is a challenge, we simply want your savings to be within $100 of the required down payment.

In **ps1c.py**, write a program to calculate the best savings rate, as a function of your starting salary. You should use **bisection search** to help you do this efficiently. You should keep track of the number of steps it takes your bisections search to finish. You should be able to reuse some of the code you wrote for part B in this problem.

Because we are searching for a value that is in principle a float, we are going to limit ourselves to two decimals of accuracy (i.e., we may want to save at 7.04% -- or 0.0704 in decimal – but we are not going to worry about the difference between 7.041% and 7.039%). This means we can search for an **integer** between 0 and 10000 (using integer division), and then convert it to a decimal percentage (using float division) to use when we are calculating the **current_savings** after 36 months. By using this range, there are only a finite number of numbers that we are searching over, as opposed to the infinite number of decimals between 0 and 1. This range will help prevent infinite loops. The reason we use 0 to 10000 is to account for two additional decimal places in the range 0% to 100%. Your code should print out a decimal (e.g. 0.0704 for 7.04%).

Try different inputs for your starting salary, and see how the percentage you need to save changes to reach your desired down payment. Also keep in mind it may not be possible for to save a down payment in a year and a half for some salaries. In this case your function should notify the user that it is not possible to save for the down payment in 36 months with a print statement. **Please make your program print results in the format shown in the test cases below.**

Note: There are multiple right ways to implement bisection search/number of steps so your results may not perfectly match those of the test case.

Hints

- There may be multiple savings rates that yield a savings amount that is within $100 of the required down payment on a $1M house. In this case, you can just return any of the possible values.
- Depending on your stopping condition and how you compute a trial value for bisection search, your number of steps may vary slightly from the example test cases.
- Watch out for integer division when calculating if a percentage saved is appropriate and when calculating final decimal percentage savings rate.
- Remember to reset the appropriate variable(s) to their initial values for each iteration of bisection search.

Test Case 1

```
>>>
Enter the starting salary: 150000
Best savings rate: 0.4411
Steps in bisection search: 12
>>>
```

Test Case 2

```
>>>
Enter the starting salary: 300000
Best savings rate: 0.2206
Steps in bisection search: 9
>>>
```

Test Case 3

```
>>>
Enter the starting salary: 10000
It is not possible to pay the down payment in three years.
>>>
```

6.0001 Introduction to Computer Science and Programming in Python
Fall 2016