# Experimental Physics 1 - Python Problem Set 1

September 13, 2018

NYU has a retirement plan for faculty and staff with complicated rules. Fortunately, I have limitless power to shape assignments so that smart Physics students can figure out the rules for me and help plan my retirement. Here are the rules for the retirement plan.

1. No matter what, NYU will contribute 5% of my annual salary to a retirement plan
2. I can contribute up to $18,500 a year to the plan. Of course, but I can't contribute more money than I make.
3. NYU will match my contributions up to 5% of my salary

Some examples:

1. A TA makes $30,000 a year. She spends $24,000 a year on rent and the remainder on beer, so makes no contributions to the retirement plan. NYU contributes 5% of $30,000 or $1,500 a year
2. A TA makes $30,000 a year. To save money, he sleeps in the lab, so he can contribute $1,000 a year to his retirement plan. NYU contributes $2,500 a year to the plan, so the total annual contribution is $3,500
3. An assistant professor makes $60,000 a year. She contributes $10,000 a year to her retirement plan. NYU will put in $6,000 a year (5% of 60,000 plus the lesser of 5% of 60,000 and 10,000) to the plan, so the total annual contribution is $16,000
4. An associate professor makes $100,000 a year. He contributes $10,000 a year to his retirement plan. NYU will put in $10,000 a year to the plan, so the total annual contribution is $20,000
5. A law professor makes $500,000 a year. She contributes the maximum $18,500 annually. NYU puts in 5% of the full salary and then matches the $18,500, so NYU's contribution is $25,000 + $18,500 = $43,500 and the total amount professor and NYU contribute to the plan is $62,000.

As you can see this gets really complicated, although the general principles are simple. The "employee match" is a really good deal, and it's way better to be a law professor. But now I want you to write **three** programs that will simplify my retirement planning.

# 1 Find the total contribution to a retirement plan

For the first program, I want you to write functions that will coerce my employee contribution into the correct range and calculate NYU's contribution to my retirement plan. The function definitions and docstrings are listed below; it's up to you to write the code that implements the functions

```
def coerceContribution(salary, contribution):
    """coerces the retirement plan contribution into the nearest legal value

    parameters
    ----------
    salary: float
        the total annual salary
    contribution: float
        the desired contribution amount

    returns
    -------
        float
            coerced contribution, which is in the range of
            [0, min(salary, MAX_CONTRIBUTION)]
    """


def calculateTotalContribution(salary, contribution):
    """finds the total amount contributed to the retirement plan

    parameters
    ----------
    salary: float
        the total annual salary
    contribution: float
        the desired contribution amount

    returns
    -------
        (total_contribution, nyu_portion) : float
            the total amount contributed to the retirement plan
            and nyu's portion of that contribution

    NYU contributes BASE_FRACTION of salary plus
    the lesser of MATCH_FRACTION*salary and contribution
    """
```

In addition to the above functions, I would like you to write a program that asks for the salary and desired contribution and prints out the following information

1. The corrected contribution amount. If this is different from the desired contribution, please tell the user.
2. The amount NYU contributes to the retirement plan based on the salary and corrected contribution
3. The total retirement plan contribution

Please write your code in partA.py, which is included as part of the starter code distribution

## 2 Calculate the total savings

For the second program, I want you to calculate the total retirement savings given a starting salary and contributions. For this assignment, we'll make the simplifying assumption that the plan contribution occurs in one lump at the end of the year, so you don't have to calculate compound interest during the year [1].

Here's the information you'll need to get from the user

1. The starting salary
2. The fraction of salary we want to contribute to the plan, expressed as a percent between 0 and 100. Entering 100 will guarantee the maximum allowed contribution.
3. The expected inflation-adjusted rate of return on savings, expressed as a percent. This could be negative. For instance, if you expect to earn 3% interest and inflation to be 4%, then the inflation-adjusted rate of return would be -1%
4. The expected inflation-adjusted salary increase, expressed as a percent. Like the rate of return, this could also be negative.
5. The number of years (s)he plans to work before retiring

To calculate the total savings, follow the this algorithm

1. The initial savings are $0.
2. For each year, first increase the total savings by the rate of return. For instance, if the savings were $1,000 and the rate of return is 4%, the new savings amount is $1040
3. Calculate the salary contribution (contribution_fraction*salary), and find the total contribution taking into account NYU's portion and the 18,500 limit.
4. Add the contribution to the total savings.
5. Increase the salary by the salary increase percentage.
6. Repeat steps 2 to 5 for the total number of years.

Your code should ask for the desired information then print out the total savings at the time of retirement. **Important:** You should separate out the computation portions of the code from the parts that take input or display output - i.e. write different functions for these portions. This will make your code much more reusable.

Normally, you would want to write one module that contains all the common functions between the different assignment parts, but because we are just starting out, begin this part of the assignment by copying and pasting code from your solution to partA.py to partB.py.

[1] To take into account inflation, we will work with "present dollars." This turns out to be relatively straightforward with some simplifying assumptions. If we save money, we'll earn interest on it, so it will grow by a rate $r_{grow}$ each year. At the same time, because of inflation, the same amount of money will be worth a little less, by factor of $\frac{1}{1+r_{infl}} \approx 1 - r_{infl}$. Taken together, this means after a year, our saved money can buy $r_{grow} - r_{infl}$ more (or less) than it could at the beginning of the year. In a similar way, we can subtract off inflation from our annual raises and the increase in contribution limit. In particular, we'll assume the contribution limit is perfectly adjusted for inflation, so in present dollars it will be fixed and doesn't grow with time. **TL;DR** The rate of return and annual raise will be smaller than you think; you don't need to worry about adjusting the contribution limit.

# 3 Help me plan my retirement

For the last program, I want you to calculate how much of my salary I need to contribute to reach my savings goal. For this part, to save on typing, you can use the following values

1. After-inflation return on investment: 3%
2. After-inflation salary increase: 0%
3. My goal is to retire with $1,000,000 in the bank.

Your program should do the following:

1. Ask me for my salary.
2. Ask how many years I plan to work.
3. It should tell me what percent of my salary (to within 0.1%) I need to save to retire with $1,000,000 in the bank.
4. If even after making the maximum allowed contribution, I won't be able to retire with $1,000,000, your program should do the following:
   (a) It should tell me what the maximum amount I can save is, given my salary and the number of years I plan to work.
   (b) It should then tell me how many years I need to work to have $1,000,000 saved. If I have to work more than 40 years,it should tell me that I need to find a new job instead.

To calculate the savings percentage, you should use a bisection search, discussed in Chapter 3 of your textbook. Here's the algorithm:

1. First check to see if a 0% contribution can get me to my goal (due to NYU's default contribution). If so, that's great – we're done.
2. Next check to see if a 100% contribution (adjusted to the maximum) gets me above the goal. If not, then there's no savings rate that will fix the problem – I'll need to invest for longer.
3. Otherwise, the correct savings rate is somewhere in between 0 and 100%. Set lowerlimit to 0 and upperlimit to 100.
4. The midpoint = 0.5 * (lowerlimit + upperlimit). Test whether saving at a rate of midpoint gives you more or less than the target amount. If it's more then set upperlimit = midpoint. Otherwise, set lowerlimit = midpoint.
5. If lowerlimit-upperlimit $< 0.1\%$ then I have determined the savings rate to within 0.1%. I can quit.
6. Otherwise, go back to step 2 and repeat until you converge.

Again, start by copying the code you might need from partB.py into partC.py. Please make sure you don't hard code any constants (like $1,000,000 or 0.1%) into your code. Instead give these descriptive names as ALL CAPS constants at the top of the module. Ideally, you will include them as default values in your function arguments, but it's also OK for this assignment if you just use them in the main code blocks.

Finally, as before, please make sure the code that does the calculations is separate from the code that takes user input or displays output.