





## Assignment 7

Due Date for Programs: November 2, 2016



The parts of this assignment marked with the "alone" icon, , should be completed individually by each student.

You may discuss any parts of the assignments with your classmates and tutors (or anyone else) but you are responsible for understanding all of the parts of the assignment that you submit. Any sharing or copying of the parts of the assignments marked with "alone" icon, , will be considered cheating.

You should not use any features of Python that have not been covered in class or recitations. If you have doubt if you are allowed to use certain structures, just ask one of the instructors.



### Problem 1 (25 points): Retirement Savings

Write a program that calculates the balance of a retirement account at the time of the owner's retirement.

The program should prompt the user for

- current value of their account
- annual investment into the account (assume the simple model of one investment per year)
- annual interest rate
- number of years till retirement

All of the values entered by the user should be validated. It may be useful to write functions that prompt the user for each type of value until the value entered makes sense (positive for balances, between 0.00 and 1.00 for interest rate and positive, but smaller than 100 for the number of years).

At the end of each year the current balance of the account should be updated by adding the accumulated interest rate and the new deposit. This value should be calculated by a function that given the four values entered by the user (as function arguments) returns the final balance.

Here are a few sample runs of the program:

(the first example is from lecture slides)

#### Output:

```
Enter initial balance( >= 0.00): 0.00
Enter yearly deposits( >= 0.00): 1000.00
Enter annual interest rate (0.00 - 1.00): 0.05
ENter the number of years till your retirement ( 0 - 100 ): 25
You will have $51,113.45 when you retire.
```

#### Output:

```
Enter initial balance( >= 0.00): 0.00
Enter yearly deposits( >= 0.00): 5000.00
Enter annual interest rate (0.00 - 1.00): 3
Enter annual interest rate (0.00 - 1.00): 0.03
ENter the number of years till your retirement ( 0 - 100 ): 30
You will have $250,013.39 when you retire.
```

#### Output:

```
Enter initial balance( >= 0.00): -20.0
Enter initial balance( >= 0.00): 20000
Enter yearly deposits( >= 0.00): 2000
Enter annual interest rate (0.00 - 1.00): 0.06
ENter the number of years till your retirement ( 0 - 100 ): 20
You will have $147,976.73 when you retire.
```



**Output:**

```
Enter initial balance( >= 0.00): 0.00
Enter yearly deposits( >= 0.00): 100.00
Enter annual interest rate (0.00 - 1.00): 0.05
Enter the number of years till your retirement ( 0 - 100 ): 50
You will have $22,081.54 when you retire.
```

Comment your source code by 1) briefly describing parts of your program 2) including your name, the date, and your class section at top of your file (above everything else)

**What to Submit**

This program should be named (i.e., the name of the file containing the program should be) `retirement.py`. You only need to submit the source code for this problem.



**Problem 2 (25 points): Credit Card Payments**

Computationally this program should be very similar to the previous one. This time you are computing the remaining balance on a credit card.

The program should prompt the user for

- initial balance on the account (no additional expenses will occur)
- monthly payments to be made (fixed amount each month)
- monthly interest rate

All of the values entered by the user should be validated. It may be useful to write functions that prompt the user for each type of value until the value entered makes sense (positive for balances, between 0.00 and 1.00 for interest rate).

The program should print a table showing the payment amount for each month (this is a fixed number except for the last month when it will most likely be less) and the remaining balance on the account (after the interest for the current month was added and the payment subtracted). The program should stop when the remaining balance reaches zero. Finally, the program should print the total amount paid to pay off the credit card balance.

Here is a sample run of the program:

(the first example comes from the lecture slides)

**Output:**

```
Enter initial balance( >= 0.00): 125.24
Enter monthly payments( >= 0.00): 20
Enter monthly interest rate (0.00 - 1.00): 0.0165
month #           payment     remaining balance
-----
0                 20.00         105.24
1                 20.00         86.98
2                 20.00         68.41
3                 20.00         49.54
4                 20.00         30.36
5                 20.00         10.86
6                 11.04          0.00
-----
Total paid: $131.04
```

**Output:**

```
Enter initial balance( >= 0.00): 2500.00
Enter monthly payments( >= 0.00): 100.00
Enter monthly interest rate (0.00 - 1.00): 0.015
month #           payment     remaining balance
-----
0                 100.00        2,400.00
1                 100.00        2,336.00
2                 100.00        2,271.04
3                 100.00        2,205.11
4                 100.00        2,138.18
```



5	100.00	2,070.25
6	100.00	2,001.31
7	100.00	1,931.33
8	100.00	1,860.30
9	100.00	1,788.20
10	100.00	1,715.03
11	100.00	1,640.75
12	100.00	1,565.36
13	100.00	1,488.84
14	100.00	1,411.18
15	100.00	1,332.34
16	100.00	1,252.33
17	100.00	1,171.11
18	100.00	1,088.68
19	100.00	1,005.01
20	100.00	920.09
21	100.00	833.89
22	100.00	746.39
23	100.00	657.59
24	100.00	567.45
25	100.00	475.97
26	100.00	383.11
27	100.00	288.85
28	100.00	193.19
29	100.00	96.08
30	97.52	0.00

-----

**Total paid: \$3,097.52**

Comment your source code by 1) briefly describing parts of your program 2) including your name, the date, and your class section at top of your file (above everything else)

#### What to Submit

This program should be named (i.e., the name of the file containing the program should be) `credit.py`. You only need to submit the source code for this problem.



### Problem 3 (30 points): Interest Earned

In class we have been looking at a way of computing a balance of an interest bearing savings account when given an initial investment and an annual interest rate. The formula for this calculation is

$$b(y) = p(1 + r)^y$$

where  $p$  denotes the initial investment,  $r$  denotes the interest rate paid yearly,  $y$  is the variable indicating the number of years, and  $b(y)$  is the balance of the account after specific number of years.

The above formula works if the interest is compounded annually (i.e., the interest is calculated and added to the principal only once a year). A more realistic model compounds the interest more frequently, and for a good reason, since the larger the balance on which the interest is computed, the larger the value that is added to principal. The formula that is used for that is as follows:

$$b(y) = p(1 + r/n)^{n*y}$$

where  $n$  is the number of times that the interest is compounded each year and other values are as above.

Write a program that demonstrates the difference between balances at the end of each year (ranging from 1 to 20) when interest is compounded annually (once a year), monthly (twelve times a year) and daily (365 times a year - let's ignore the leap years).

The program should prompt the user for the initial balance for for the annual interest rate.

You should be defining functions for this program: - you should have a function that obtains the balance from the user (this function should make sure that the value it returns is  $\geq 0$ , i.e. keep prompting if the entered balance is negative or zero) - you should have a function that obtains the interest rate from the user (this function should make sure that the value entered is between 0 and 1) - you should have a function that calculates a balance of an account given the balance, interest rate, number of compounding periods and number of years (this means your function should take four arguments)

Here are some runs of the program:

Output:

```
Enter initial balance ( >= 0.00): 1000.00  
Enter annual interest rate (0.00 - 1.00): 0.05
```

year	yearly	monthly	daily
1	1,050.00	1,051.16	1,051.27
2	1,102.50	1,104.94	1,105.16
3	1,157.63	1,161.47	1,161.82
4	1,215.51	1,220.90	1,221.39
5	1,276.28	1,283.36	1,284.00
6	1,340.10	1,349.02	1,349.83
7	1,407.10	1,418.04	1,419.03
8	1,477.46	1,490.59	1,491.78
9	1,551.33	1,566.85	1,568.26
10	1,628.89	1,647.01	1,648.66
11	1,710.34	1,731.27	1,733.19
12	1,795.86	1,819.85	1,822.04
13	1,885.65	1,912.96	1,915.46
14	1,979.93	2,010.83	2,013.66
15	2,078.93	2,113.70	2,116.89
16	2,182.87	2,221.85	2,225.42
17	2,292.02	2,335.52	2,339.51
18	2,406.62	2,455.01	2,459.45
19	2,526.95	2,580.61	2,585.54
20	2,653.30	2,712.64	2,718.10

Output:

```
Enter initial balance ( >= 0.00): -1200  
Enter initial balance ( >= 0.00): -100  
Enter initial balance ( >= 0.00): 20000  
Enter annual interest rate (0.00 - 1.00): 15  
Enter annual interest rate (0.00 - 1.00): 0.06
```

year	yearly	monthly	daily
1	21,200.00	21,233.56	21,236.63
2	22,472.00	22,543.20	22,549.71
3	23,820.32	23,933.61	23,943.99
4	25,249.54	25,409.78	25,424.48
5	26,764.51	26,977.00	26,996.51
6	28,370.38	28,640.89	28,665.74
7	30,072.61	30,407.39	30,438.18
8	31,876.96	32,282.85	32,320.21
9	33,789.58	34,273.99	34,318.61
10	35,816.95	36,387.93	36,440.58
11	37,965.97	38,632.26	38,693.75
12	40,243.93	41,015.02	41,086.23
13	42,658.57	43,544.73	43,626.65
14	45,218.08	46,230.48	46,324.14
15	47,931.16	49,081.87	49,188.42
16	50,807.03	52,109.13	52,229.81
17	53,855.46	55,323.11	55,459.25
18	57,086.78	58,735.32	58,888.36
19	60,511.99	62,357.99	62,529.51
20	64,142.71	66,204.09	66,395.79



Comment your source code by 1) briefly describing parts of your program 2) including your name, the date, and your class section at top of your file (above everything else)

### What to Submit

This program should be named (i.e., the name of the file containing the program should be) `compound.py`. You only need to submit the source code for this problem.



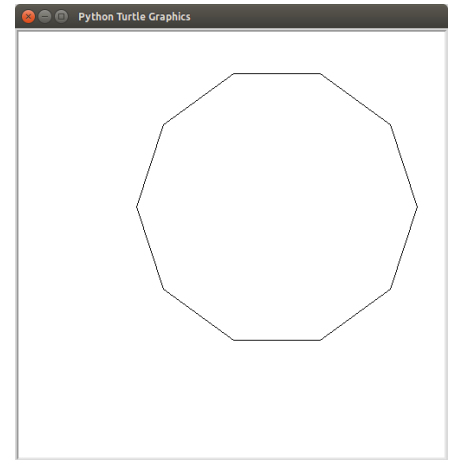
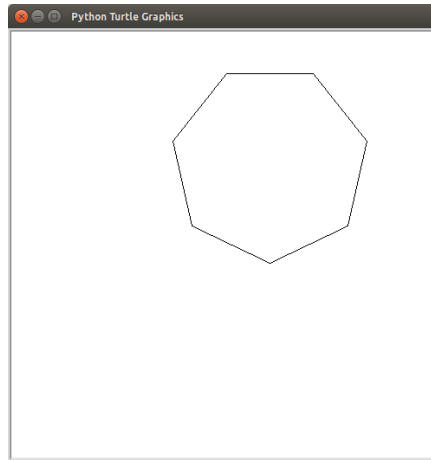
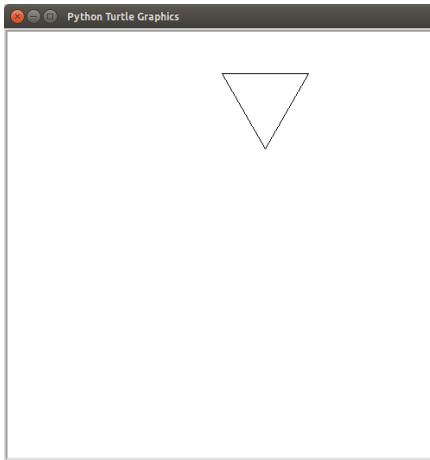
## Problem 4 (20 points): Drawing Shapes

Write a program that uses the Turtle graphics module to draw a shape that has all equal sides. The user should decide how many sides the shape has. The basic idea for drawing a square is to draw a side, then turn 90 degrees, then draw another side, then turn 90 degrees, ...

If you wish to draw a polygon with a different number of sides, you cannot turn 90 degrees. Instead you need to calculate the number of degrees by dividing 360 degrees by the number of sides.

Your program should prompt the user for the number of sides and then generate the shape. (Warning: the program may be drawing off the screen if the number of sides is large enough - that's fine.)

Here are a few screenshots of the results for three, seven and ten sides:



Comment your source code by 1) briefly describing parts of your program 2) including your name, the date, and your class section at top of your file (above everything else)

### What to Submit

This program should be named (i.e., the name of the file containing the program should be) `shapes.py`. You only need to submit the source code for this problem.

## Grading

The only way to receive the credit for the worksheet problems is to hand them in before the end of the lab session in which they are given.

The programs are graded based on correctness, style of code, design and documentation.

## What and how to submit?

You should submit the source code file for each program to NYU Classes by the due date stated above. Make sure that you get an email confirmation after you submit the assignment. You should keep that email until the grades are returned - it is your proof that the assignment was submitted! If you do not get an email confirmation, you should try to resubmit the assignment. If you do not get that email, it means that we did not get your assignment.