CP 104 – Fall 2020

Assignment 01

Due: Monday, Sept 28th, 2020 at 10:30 a.m.

General Assignment Notes

Your assignments must follow the following requirements:

- Name your Eclipse project with:
    - your login,
    - an underscore,
    - 'a' (for 'assignment'),
    - the assignment number: `login_a#`.
- For example, if student `barn4520` submits Assignment 1, the name should be: `barn4520_a1` or `barn4520_a01`. Make sure your programs and the testing file are in this Eclipse/PyDev project.

- In your program, use the variable naming style given in Coding and Documentation Style Standards : i.e. lower case variable names, underscores between words.

- Test your programs:
    - Copy the output from your test to a file in your Eclipse/PyDev project named `testing.txt`.
    - Make sure that you have included an identification header for each question; there was a sample identification header for the testing file in lab 1.
    - Make sure you do the correct number of tests as specified in the question.
    - Make sure the tests are well labeled so that the markers know how the results match the questions.
    - The solutions for all programs go into *one* `testing.txt` file.

- Zip the entire project using Eclipse.
    - Give your .zip file the same name as your project when exporting your project, e.g. `barn4520_a1.zip` .
    - Use only Eclipse's built-in archive capability to create these .zip files. No other format will be accepted.

- Use the Validate Assignment link to make sure that your project and .zip file are named correctly and have the proper contents. Improper assignment submissions are given a grade of zero.

- Submit the validated .zip file to the appropriate drop box on MyLearningSpace.
    - You can submit as many times as you like. Only the last submission is kept.
- Unless otherwise indicated by the question you may only use the built-in functions and special forms introduced in the lecture slides

- The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.

General Marking Expectations

- Although the marking scheme is tailored for each question, you can use the following as an indication of what we are looking for when marking your assignments.
  - General:
    - project and zip files named correctly. Using the wrong names is an automatic zero.
    - uses the variable naming style given in Coding and Documentation Style Standards : i.e. lower case variable names, underscores between words
  - main:
    - identification template included and filled in correctly
    - inputs as required for program
    - outputs as required for program
  - testing:
    - identification header included and filled in correctly
    - Tests as required for program, e.g. number of tests, types of tests.
    - Make sure your code has no errors or warnings

Note: The output for all tasks in an assignment go into one testing.txt file (Refer back to Lab 1 for help in the process of submitting an assignment). Make sure the tests are labeled so that the markers can easily find the answers to a particular question.

1. Write and test a python program **t01.py** that shows your understanding of how single, double, and triple quotes can be used by using four (4) `print` functions to:

1. display the text:
```
What is "petabyte"?
```
2. display the text:
```
A petabyte (PB)is a lot of data:
1 PB = 20 million 4 drawer filing cabinets filled with text
1.5 PB = size of 10 billion photos on facebook
20 PB = the amount of data processed by Google per day
```
3. display the text:
```
that's a lot of data to handle. Big Data.
```
4. display the text:
```
"You have enemies? Good. That means you've stood up for something, sometime in
your life." Winston Churchill
```

i.e. you may use only one print function per piece of text. Use no other print functions in your program - no titles, no blank lines, nothing. Your solution *must* include one use of single quotes, one use of double quotes, and one use of triple quotes. You may not use any escape characters in your print functions i.e you may not use '\n' or quote escaping: \" or \'.

- Copy the results (the contents of your console) to a file named testing.txt (located in your project).

2. Write and test a program **t02.py** that asks the user for their favorite course and the grades they are hoping to get in it.

   First prompt for both pieces of information (two separate prompts in the order given). In the example below, the data entered by the user is shown in bold face.

   For input:

   ```
   Enter your favourite course: CP104
   Enter the grade you hope to get: A+
   ```

   The output should look like:

   ```
   You hope to get A+ in your favourite course CP104.
   ```

   - Test your program with two different value than the example.
   - Copy the results (the contents of your console) to your file named testing.txt.

   Note: Run your code twice, do not duplicate the code.

3. Write and test a python program **t03.py** which asks for the user's first and last name (separately in that order) and print them as one string.

   For input:
   ```
   What is first name? Mohamed
   What is last name? Salah
   ```
   The output would be:
   ```
   Welcome Mohamed Salah
   ```

   - Test your program with data different than the example.
   - Copy the results to `testing.txt`.

4. Write and test a program t04.py that asks the user to enter the height in inches and display it in meters. Where 1 inch = 0.0254 m.

   For input:
   ```
   Enter the height in inches: 100
   ```

   The output should look like:
   ```
   The equivalent height in meters is 2.54
   ```

- Define constant when appropriate.
- Test your program with a different value than the example.
- Copy the results to `testing.txt`.

5. Compound interest is calculated as follows:

$$A = P\left(1 + \frac{r}{n}\right)^{nt}$$

where `P` is the principal amount, `r` is the annual rate of interest (as a decimal), `t` is the number of years the amount is deposited or borrowed for, `A` is the amount of money accumulated after `t` years, including interest, and `n` is the number of times the interest is compounded per year.

For input:

```
Principal: 5000
Interest (decimal): 0.05
Number of years: 10
Compound interest per year: 12
```

The output should look like:

```
Balance: $ 8235.05
```

Write and test a python program **t05.py** that takes as inputs the values of Principal amount, interest rate, number of years, and compound interest per year *in that order*, and outputs the value of compound interest, matching the given formula. You can round the number by using formatted output.

- Note: do not use single letter to name your variables, this will result in mark deductions.
- Test your program with data different than the example.
- Copy the results to `testing.txt`.