Assignment 02

Due: Monday, Oct 5, 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.

Assignment 02

Due: Monday, Oct 5, 2020 at 10:30 a.m.

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.

Note: Refer back to Lab 1 for help in the process of submitting an assignment.
Assignment 2 marking Expectations:

- All programs should use the formatted output method.
- Define constant when appropriate.
- The output for all of the questions in an assignment go into one testing.txt file (as the template shows in lab 1 task 3). Make sure the tests are labeled so that the markers can easily find the answers to a particular question.
- **In the main program template, be sure to fill in the file name and the description of the program.**
- **Make sure your code has no errors or warnings**

1. A company had determined that the annual profit is typically 23% of the total sales. Write and test a python program **t01.py** that asks the user to enter the amount of total sales, and then display the profit that will be made from that amount. The calculated amounts should be displayed in a format matching the following:

```
Projected Profit Report
-------------------------
Total sales:   $ 100000.00
Annual profit: % 23
-------------------------
Profit:        $ 23000.00
```

Hints:

- use an upper-case constant for the value 0.23 instead of 23%

Assignment 02

Due: Monday, Oct 5, 2020 at 10:30 a.m.

- Copy the results to `testing.txt`.

2. Write and test a program **t02.py** that asks the user for an integer that represent the date in the following format MMDDYYYY and then displays the date in the format DD/MM/YYYY. You may assume that the user will always enter a valid date. Use must use integer division and modulus, not strings, to extract the date parts.

   Example for input:
   ```
   Enter a date in the format MMDDYYYY: 03172020
   ```

   The output will be:
   ```
   17/03/2020
   ```

   - Test your program with 2 different dates than the example.
   - Copy the results to `testing.txt`.

3. Write a python program **t03.py** that asks the user to enter a positive two-digit integer `num_n`. The program outputs the sum of the two digits. If the user enters a number 25, the sum will be 7 as (2+5 = 7). Use must use integer division and modulus, not strings, to extract the two digits.

   - Test your program with 2 different numbers than the example.
   - Copy the results to `testing.txt`.

4. Write and test a program named **t04.py** to divide the number of balloons evenly among children coming to a birthday party. The program should ask for the number of balloons and the number of children joining the party. Output the number of balloons each child will receive and the number of balloons leftover that won't be distributed.

   Example for input:
   ```
   Number of balloons: 400
   Number of children: 13
   ```

   The output will be:
   ```
   Each child will receive 30 balloons
   Balloons that will not be distributed: 10
   ```

   - Test the program with three different sets of data. Consider the following cases:
     - Number of children is less than number of balloons considering both when there are leftover balloons and when there are no leftover balloons.
     - Number of children is equal to the number of balloons.

   - Copy the results to `testing.txt`.

Assignment 02

Due: Monday, Oct 5, 2020 at 10:30 a.m.

5. In order to pass CP104 you are required to pass the examination portion of the course; i.e., your weighted exam mark must be 50% or higher. Write and test a python program **t05.py** which will make it easy for you to compute and understand this requirement. Your program takes two numbers between 0 and 100 (`midterm_mark` and `final_exam_mark`), in that order, as inputs. Your program outputs the weighted exam mark (as a percentage value) of midterm-mark and exam-mark. The midterm is worth 20% of the final grade and the final exam is worth 40%, so the weighted exam mark is:

$$Exam\ portion = \frac{0.2 * midterm\ mark + 0.4 * final\ exam\ mark}{0.2 + 0.4}$$

For example, if you get 60% on the midterm (12/20) and 38% on the final (15/40), then you will not pass the course since your weighted exam average is 27/60 (less than 50%). This is independent of your assignment and lab grade.
If instead you received 33% on the midterm (6.6/20) and 60% on the final (24/40), then your weighted exam average is 30.6/60, a passing grade. Show your results as a float with one decimal place.

Example for input:

```
Midterm mark (%): 80
Final exam mark (%): 40
```

The output will be:

```
Your weighted exam average is: 53.3 %. The passing mark of the weighted exam
average is 50%
```

- Test your program with two data different than the example:
  - One test for passing the course.
  - One test for not-passing the course
- Copy the results to `testing.txt`.