Assignment 03

Due: Monday, October 19th, 2020 at 10:30 a.m.

General Assignment Notes

Your assignments must follow the following requirements:

- Name your Eclipse project with:
  - your network 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 03

Due: Monday, October 19th, 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 3 marking Expectations:

1. All programs should use the formatted output method taught in lab 2.
2. Define constants when appropriate.
3. The outputs for all of the questions in an assignment go into one testing.txt file (as the template shown in lab 1 task 3). Make sure the tests are labeled so that the markers can easily find the answers to a particular question.
4. function(s):
    1. **FUNCTIONS MUST BE IN A SEPARATE FILE FROM THE MAIN PROGRAM.**
    2. for questions that specify the objectives of the function(s), did you do what was requested?
    3. correct parameters and return values
    4. calculation(s) correct
    5. function documentation has the correct format
    6. parameters/preconditions correct and complete
    7. returns/ postconditions correct and complete
5. main/testing module:
    1. identification template included and filled in correctly
    2. proper use of constants if appropriate
    3. inputs as required for program
    4. outputs as required for program
    5. outputs should use the formatted output method
    6. functions calls have appropriate arguments
    7. return values from functions are appropriately handled
6. testing:
    1. identification header included and filled in correctly
    2. tests as required for program, e.g. number of tests, types of tests.

Assignment 03

Due: Monday, October 19th, 2020 at 10:30 a.m.

1. When an object is falling because of gravity, the following formula can be used to determine the distance the object falls in specific period.

$$d = 1/2\ gt^2$$

The variable in the formula is as follows: *d is the distance in meters,* g *is a gravitational constant 9.8 meters/sec*`^2`*,* and t is the amount of time, in seconds, that the object has been falling.

Write a function `falling_distance` that takes a number that represents an object's falling time in seconds and return the distance, in meters, that the object has fallen during that time interval

Write a program named **t01.py** that tests the function by asking the user to enter a number and displaying the return value. Save the function in a PyDev library module named `functions.py`

A sample run for t02.py:

```
Enter the time (in secs): 10
The Object has falling 490.00 meters in 10 seconds.
```

- Test your program with 2 different time values than those given in the example.
- Provide the doc string for the function `falling_distance`
- Copy the results to testing.txt.

2. A nutritionist who works for a fitness club helps members by evaluating their diets. As part of her evaluation, she asks members for the number of fat grams and carbohydrate grams that they consumed in a day. Then, she calculates the number of calories that result from the fat, using the following formula:

calories from fat = fat grams × 9

Next, she calculates the number of calories that result from the carbohydrates, using the following formulae:

calories from carbs = carb grams × 4

Write a function `calorie_calculator` that takes **two** parameters of fat grams and carb grams in that order, and returns **two** numbers: the calculated fat and carb calories in that order.

Assignment 03

Due: Monday, October 19th, 2020 at 10:30 a.m.

Write a program named **t02.py** that tests the function by asking the user to enter the fat grams and carbohydrate grams they consumed in a day and printing its output. Save the function in a PyDev library module named `functions.py`

A sample run:

```
Enter the fat grams consumed: 100
Enter the carbohydrate grams consumed: 200

Fat calories:  900
Carb calories: 800
Total calories: 1700
```

- Provide the doc string for the function `calories_calculator`
- Test your program with two data, you may assume that the user will only enter numbers
- Copy the results to testing.txt.

3. Revisiting question 2 from assignment 2, Write a function `convert_date` that takes an integer as a parameter and returns three integer values representing the input converted into days, month and year (see the function docstring).

   Write a program named **t03.py** that tests the function by asking the user to enter a number and displaying the output day, month and year. Save the function in a PyDev library module named `functions.py`

   A sample run for t03.py:

   ```
   Enter a date in the format MMDDYYYY: 05272017
   ```

   The output will be:

   ```
   27/05/2017
   ```

   The function docstring is as follows:

```
def convert_date (date_int):
   """
     ------------------------------------------------------
       Converts date_int into days, month and year
       use: day,month,year = convert_date (date_int)
     ------------------------------------------------------
     Paramaters:
      date_int – (int > 0)

      returns
      day: the day in the month in date_int (int >= 0)
```

Assignment 03

Due: Monday, October 19th, 2020 at 10:30 a.m.

```
    month: the month in date_int (int >=0)
    year: the years in date_int (int >=0)

  ------------------------------------------------------
    """
```

- Test your program with 2 different date values than those given in the example.
- You may assume that the user will only enter numbers (2 digits for month, 2 digits for days and 4 digits for year)
- You are not allowed to use any string methods or string indexing.
- Copy the results to testing.txt.

4. Write a function `convert_sec` that takes an integer as a parameter and returns 4 integer values representing `num_sec` converted into days, hours, minutes and seconds (see the function docstring).

   Write a main program named **t04.py** that tests the function by asking the user to enter a number and displaying the output days, hours, minutes and seconds. Save the function in a PyDev library module named `functions.py`

```
def convert_sec (num_sec):
   """
   ------------------------------------------------------
      Converts num_sec into days, hours, minutes and seconds
      use: days,hours,minutes, seconds = convert_sec (num_sec)
    ------------------------------------------------------
    Preconditions:
      num_sec – (int > 0)

      returns
      days: number of days in num_sec (int >= 0)
      hours: number of hours in num_sec (int >=0)
      minutes: number of minutes in num_se (int >=0)
      seconds: number of seconds in num_sec (int >=0)

   ------------------------------------------------------
```

A sample run for t04.py:

```
Enter number of seconds:3681
    There are 0 days, 1 hours, 1 minutes, and 21 seconds in 3681 seconds
```

- test your program with 2 data, you may assume that the user will only enter valid numbers
- Copy the results to testing.txt.

Assignment 03

Due: Monday, October 19th, 2020 at 10:30 a.m.

5. Write a function `math_quiz` that gives simple math quizzes. The function takes no parameters and returns no values. The function should display the two random numbers that are to be added, such as:

   247
   + 129

   The function should allow the student to enter the answer. The function then displays the user answer along with the correct answer.

   A sample run:

```
  247
+ 129
Your answer: 999, it should be: 376
Or
  247
+ 129

Your answer: 376, it should be: 376
```

   Write a testing program **t05.py** to test your function. Save the function in a PyDev library module named `functions.py`

   - Provide the doc string for the function `math_quiz`
   - Test your program with data values different than those given in the example.
   - Copy the results to `testing.txt`.