CP 104 – Fall 2020

Assignment 08

Due: Monday, Nov 30th, 2020 at 10:30 am.

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.

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

Assignment 8 marking Expectations:
- All programs should use the formatted output method taught in lab 2.
- Define constant when appropriate.
- The output 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.
- function(s):
    - **FUNCTIONS MUST BE IN A SEPARATE FILE FROM THE MAIN PROGRAM.**
    - for questions that specify the objectives of the function(s), did you do what was requested?
    - correct parameters and return values
    - calculation(s) correct
    - function documentation has correct format
    - parameters correct and complete
    - returns correct and complete
- main:
    - identification template included and filled in correctly

  - proper use of constants if appropriate
  - inputs as required for program
  - outputs as required for program
  - outputs should use the formatted output method
  - implementation of decision structures, for loops and while loops, as appropriate
  - functions calls have appropriate arguments
  - return values from functions are appropriately handled
- testing:
  - identification header included and filled in correctly
  - tests as required for program, e.g. number of tests, types of tests.
- **Provide the docstring for all questions that does not include one.**

  Assignment 8 is uploaded as a zip file. Unzip it first to work on it. The folder has sample input files for each question.

  questions require you to write/print the output to an output file. Make sure that those files are in your src folder. There is no testing.txt required for this assignment.

1. Write a function called `total_nums` that takes a file handle as a parameter. The function reads through a file of text and returns a list of the numbers encountered and their sum. Digits that occur within words should be ignored.
   For example, for the text in the file "If I were 31 instead of 63 I'd probably punch you in the nose a 2nd time" should return a list of the numbers encountered 31 and 63, along with their sum 94. You need to only deal with numbers that are surrounded by spaces. Save the function in a PyDev library module named `functions.py`

   Write a testing program named **t01.py** that tests the `total_nums` function and prints/writes the returned values (the list of number returned and with their sum) to the output file `output_t01.txt`.

   - Test your program with the sample file provided `text_numbers.txt`.

2. Write a function `locate_median` that takes a file handle as an input and find the median of a group of scores. Requirements:

   - Read the group of scores from a file (that contains numbers only, each line has 3 numbers), and put the scores into a list of numbers.
   - Your function must sort the elements in order. You can use the build-in sort method

- Find and return the median from the sorted list of scores. Here a median is the middle element of a sorted list if the number of elements is odd, and the average of the two middle elements if the number of elements is even.
- for a list [40,20,50,10,30], the median is 30 as it is the middle element (odd number of elements in the sorted list)
- for a list [40,60,30,10,50,20], the median is 35 as it the average of 30 and 40 (two middle items in an even number of elements in the sorted list)

Save the function in a PyDev library module named `functions.py`

Write a testing program named **t02.py** that tests the `locate_median` function and write/print the results to the output file `output_t02.txt`

- Test your program with the sample file provided `numbers.txt`.

3. Write a function `file_analysis` that takes a file handle as a parameter and returns a list of the following statistics:
    - The number of uppercase letters in the file
    - The number of lowercase letters in the file
    - The number of digits in the file
    - The number of whitespace characters in the file

   Save the function in a PyDev library module named `functions.py`

   Write a main program named **t03.py** that tests the function `file_analysis` function and write/print the results to the output file `output_t03.txt.`

    - Test your program with the sample file provided `text.txt`.

4. Certain machine parts are assigned serial numbers. Valid serial numbers are of the form: SN/nnnn-nnn where ‘n’ represents a digit. Write a function called `is_valid` that takes a string as a parameter and returns `True` if the serial number is valid and `False` otherwise.

   For example, the input “`SN/5467-231`" returns `True`

   "`SN5467-231`" returns `False`

   "`SN/5467-2231`" returns `False`

Assignment 08

Due: Monday, Nov 30th, 2020 at 10:30 am.

Using this function, write and test another function `valid_sn_file` that takes three file handles as parameters. The first handle is to the file contains a list of serial numbers, the second and third are to output files. Write all valid serial numbers to the second file handle and invalid serial numbers to the third. Save the functions in a PyDev library module named `functions.py`

Write a main program named **t04.py** that tests `valid_sn_file` function and write/print the results to the output files: `output_t04_valid.txt` and `output_t04_invalid.txt`

- Test your program with the sample file provided `serial_number.txt`.