

# Project 6

[Start Assignment](#)

---

**Due** Jul 16 by 11:59pm      **Points** 55      **Submitting** a file upload      **File Types** zip

---

## Goals

At the completion of this assignment, you should:

- have gained additional experience working with and manipulating data read from a file
- have gained additional experience using input using while and conditional statements
- have gained experience closing a file
- have gained experience creating and using a Python module
- have gained additional experience creating a generic function and creating more specific functions that use the generic function
- have gained additional experience formatting output without using tabs
- have gained additional experience working with logical operators
- have gained additional experience working with conditions
- have gained additional experience working with loops and in operators
- have gained experience using an online website to generate random integers and creating a text file containing those integers, as well as a "comment" line
- have gained additional experience with guardian, traversal, search and counter patterns
- have practiced outlining the steps and including those steps as comments in your code
- have gained experience interpreting requirements and implementing those requirements in code

## Expected Output and Grading Rubric


You can view the expected output and grading rubric for this lab at the end of the assignment.

## Getting Started

For this assignment you will be working with files, as well as learning to create a shared code file.

You should take the time to consider the steps involved in solving this assignment, and include those steps as single-line comments in your code to help guide you through implementing the coded solution.

You should not attempt to write an entire coded solution; instead, practice incremental development, adding small amounts of code that you can test and validate.


We will be using a URL to generate random data utilizing the free services of [RANDOM.ORG](https://www.random.org/)  [\(https://www.random.org/\)](https://www.random.org/).

## Assignment

### Part I: Random Number File

You will need to create a file containing a list of random integers.

- Create **random\_numbers.txt** file using IDLE Editor
- Use your browser to go to the following URL

URL: <https://www.random.org/integers/?num=100&min=1&max=100&col=1&base=10&format=plain&rnd=new>   
(<https://www.random.org/integers/?num=100&min=1&max=100&col=1&base=10&format=plain&rnd=new>)

- Copy the integers listed on the web page, and paste those numbers into *random\_numbers.txt*.
- Above the numbers at the top of *random\_numbers.txt* add the URL, but insert a **#** before the URL to indicate this line is a comment and not a number.

```
# https://www.random.org/integers/?num=100&min=1&max=100&col=1&base=10&format=plain&rnd=new
```

### Part II: Read and Process File Contents

You will need to read each line of the file, skipping any "comment" lines, counting the number of integers, the number of comment lines, and the total of all of the integers. You will then output the number of integers and comments, the total, and the average of all of the integers rounded to two decimal places.

- Create **p6-random.py** file using IDLE Editor
- Open the file and add code to read each line of *random\_numbers.txt* (recommend using the *in* operator)
- Count the number of lines that start with a **#** (comment lines)
- Count the number of lines that are not comments
- Sum the values of all non-comment lines

### Part III: Print Results

You will need to print the results. The printed results will need to be formatted, but initially simply print the results.

Below is an example of the required output. Note that the average calculation must be rounded to two decimal places.

```
Count: 100  
Comments: 1  
Total: 4796  
Average: 47.96
```

### Part IV: Python Shared Code Module

Before formatting the output, you will create a Python module file containing functions to assist with producing formatted output. You will use the import statement to include code from your Python module.

- Create **p6\_shared.py** file using IDLE Editor
  - **Note:** An underscore is required rather than a dash in a module filename.
- Create the following three functions:
  - **pad\_string()**: Parameters include data to be formatted, the size of the format space, a direction (e.g. left padded or right padded), and the padding character; direction must default to left padded, and the padding character must default to a single space; if the the length of the data to be formatted is greater than the size of the format space, simply return the original data to be formatted
  - **pad\_left()**: Parameters include data to be formatted, the size of the format space, and the padding character; call *pad\_string()*, passing the data, size and formatting character, and explicitly set the padding direction to left padded
  - **pad\_right()**: Parameters include data to be formatted, the size of the format space, and the padding character; call *pad\_string()*, passing the data, size and formatting character, and explicitly set the padding direction to right padded

You should first create and test *pad\_string()*, then add and test *pad\_left()* and *pad\_right()*.

To test your code module, you could add test code to your Python code, but you can also simply run the module. No output will appear in the IDLE Shell as the file only contains function definitions; however, once executed, you can use the functions directly in the IDLE Shell.

```
>>> pad_left("abc",10)
'      abc'
>>> pad_right("abc",10)
'abc      '
```

## Part V: Format Printed Output

Revisit your code that prints the results and apply both *pad\_left()* and *pad\_right()* functions to format your output to match the output below (your total and average values will be different).

```
Count:           100
Comments:        1
Total:           4796
Average:         47.96
```

- Use the import function to include the module. Only import the *pad\_left()* and *pad\_right()* functions.

```
from p6_shared import pad_left, pad_right
```

- Use the functions as expressions to format the printed output
- Use 10 as the padding space for both *pad\_left()* and *pad\_right()*, but rather than code 10 directly into each function call, create two variables, *label\_spacing* and *num\_spacing*, that may be easily changed to adjust the formatting.

## Part VI: Input Filename

Add a *while* loop to prompt for the integers filename. You will use guardian code to test if no filename was entered and exit the program. You will also add code to close the file.

- Add a while loop that prompts for the integers filename using the prompt below:

```
Enter filename (blank to exit):
```

- Use the *strip()* method to remove leading and trailing white space
- Use *break* to exit the program if no filename is entered
- Use the *close()* method to close the file object once you've completed reading the file
- Once the file has been read, closed, and results printed, re-prompt for the filename.

```
Enter filename (blank to exit): random_numbers.txt
Count:          100
Comments:       1
Total:          4796
Average:        47.96
Enter filename (blank to exit):
```

## Part VII: Test File Existence

Even though you will only read a single file, you should check to see if the file exists before attempting to open the file. We will use the *os* Python library to test if the file exists, specifically using *os.path.exists()* method. Note that the path object is an object found in the *os* library.

- Add an import statement for the *os* library to your Python file

```
import os
```

- Add a conditional statement that tests for the existence of the entered file. You may want to consider an *elif* and the *not* operator. Use the following expression to test for existence of a file where the name of the file is stored in a variable *filename*.

```
os.path.exists(filename)
```

- Output the following error message with the name of the invalid file if the file does not exist, and re-prompt for the input file.

```
Enter filename (blank to exit): somefile.txt
Invalid filename: somefile.txt
Enter filename (blank to exit):
```

## Submissions

For this assignment you will be uploading your work from Lab 6 as well as your work from this assignment.

*Remember that all submitted **Python** files must include the header docstring at the top of each file.*

Create a single compressed file that contains the Python files. To create the compressed file, select the deliverable Python files, right-mouse click on the selected files, and select Compress (Mac) or Compressed (Windows). A new file will be created that contains all of the deliverable Python files.

Rename the compressed file **p6.zip**. Use the Project 6 Canvas site to submit/upload the compressed/ZIP file.

The compressed file will contain the following deliverables (point values in parentheses).

- **(5) lab6-grades.py**
- **(5) lab7-word-check.py**
- **(5) lab7-word-stats.py**
- **(5) random\_numbers.txt**
- **(10) p6\_shared.py**
- **(25) p6-random.py**

Once uploaded, revisit the Assignment 6 page in Canvas and confirm the file was uploaded.

**Remember, you are responsible to submit your assignments before the deadline. Late submissions will not be accepted.**

### Expected Output

The expected output for this project is described within the project.

### Grading Rubric

#### random\_numbers.txt

- Verify valid text file with numbers

#### p6\_shared.py

- Verify using required functions: pad\_string(), pad\_left(), and pad\_right()
- Verify pad\_left() and pad\_right() call pad\_string()

#### p6-random.py

- Verify os library is imported
- Verify p6\_shared.py library imported
- Verify using input()
- Verify using a while loop
- Verify using pad\_left() with each output line
- Verify using pad\_right() with each output line
- Verify checking for file existence and invalid filename message possible
- Verify using break statement(s)
- Verify a comment count is output

- Verify a total count is output
- Verify an average is output