COMMENTS

- The test has two parts. You can do the questions in any order that you choose.
- The only device you may access during the exam is your laptop. The only open windows allowed are PyCharm and a browser with tabs linked from the class website.
- During the test you can access the course module descriptions and the course Python information sheet.
- Part 1 consists of twenty short answer questions available on Collab. Your time to complete the test begins with accessing the short answer questions.
  - *You may not use PyCharm for any of the short answer questions.*
  - On the Collab site you will see two versions of Part 1. They have the same questions. One is for students in general; the other is for students with SDAC extended time accommodations.
- Part 2 consists of five programming problems to be solved using PyCharm and uploaded to the class website.
  - *You are responsible for submitting for your work. Check that you have done so before exiting the testing. Do not submit once your testing time is up. Late submissions will not be graded.*
  - *You must use our files for your program submissions. Do not modify or delete any of the code in the program files.*
  - *Code should follow class programming practices; e.g., whitespace, identifier naming, etc.*
  - *Whether code is testable is important. Comment out or delete all debugging print() statements before submitting.*
- *During the test you may not access past code (yours, ours, or anyone else's).*
- *During the test you may not access class notes, epistles, examples, artifacts, solutions on the web, or your own past assignments during the test.*
- By submitting solutions for this test, you are agreeing that
  - *You neither given nor received help directly or indirectly to or from anyone else;*
  - *You did not directly or indirectly use materials from non-allowed sources.*
- Class personnel cannot help you debug your answers.

PART II – PROGRAMMING

1. Implement program *phrase.py*. The program prints the string "I am honorable." *and nothing else*. Therefore, your program output should be

```
I am honorable.
```

2. Implement program *mtom.py*. A *moment* is a time unit that came into use in the middle ages. It equals 1.5 minutes. The program prompts its user for an indicated number of moments. It then computes and prints that time in minutes using the formula:

   $$\textit{time in minutes} = 1.5 \cdot \textit{moments}$$

   The program prints nothing else; for example, *do not label your output* with phrases like *Number of minutes*.

   Some possible program runs are

```
Enter number of moments: 1

1.5
```

```
Enter number of moments: 15

22.5
```

```
Enter number of moments: 314

471
```

3. Implement program *fila.py*. The program prompts its user to supply two strings in a single input statement. The prints on separate lines the word that occurs first alphabetically and then the word that occurs second alphabetically. *The program prints nothing else*.

   Some possible program runs are

```
Enter two strings: more less

less
more
```

```
Enter two strings: more less

less
more
```

```
Enter two strings: b7 b12
```

```
b12
b7
```

4. Implement program *ranly.py*. The program prompts its user for two values m and n in a single input statement. The program performs the following tasks in the indicated order:

   - With a single input statement, gets from the user integers *m* and *n*.
   - Constructs the list of integers from the inclusive interval *m*, *m*+1, *m*+2, …, *n*.
   - Prints the list.
   - Prints the total of the integers in the list.

   There should be no other output other than the list and the total.

   Some possible program runs are

```
Enter two integers: 0 5

[0, 1, 2, 3, 4, 5]

15
```

```
Enter two integers: 3 14

[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

102
```

```
Enter two integers: -3 3

[-3, -2, -1, 0, 1, 2, 3]

0
```

5. Implement program *lest.py*. The program prompts its user for text. The program prints the length of the longest word in the text. *The program prints nothing else*.

   Suggested algorithm: prompt and read the input. Convert the input into a list of words. Accumulate a list of the word lengths. Determine from the list of the lengths, the maximum word length. Print the maximum word length.

   Some possible program runs are

```
Enter text: it was great to see

5
```

```
Enter text: apple banana kiwi melon orange

6
```

```
Enter text: at by for from in of to

4
```

6. Implement program *tell.py*. The program prompts the user for the name of a data set. *The dataset will have the same number of columns in every row.* The dataset will be in the web folder:

   http://www.cs.virginia.edu/~cs1112/datasets/csv/

   The program then gets and analyzes the dataset to determine and print in the below order:

   - Its number of rows,
   - Its number of columns,
   - Its total number of cells,
   - The value of the first cell in its first row
   - The value of the last cell in its last row.

   The program prints nothing else.

   If the dataset *origins.csv* was to be the web file to be examined.

```
apple,      red,       Kazakhstan

banana,     yellow,    Southeast Asia

potato,     brown,     Peru

kiwi,       brown,     China

blueberry,  blue,      North America

corn,       yellow,    Mexico
```

   The program should produce output

```
Enter name of dataset: origins.csv

6

3

18

Apple

Mexico
```

   If the dataset *dogs.csv* was to be the web file to be examined.

```
Asta,         The Thin Man

Fala,         Dog of President Franklin Roosevelt

Farley,        For Better or Worse
```

Laika,	First dog in space
Petey,	The Little Rascals
Pluto,	Dog of Micky Mouse
Ruff,	Dennis the Menace
Snoopy,	Peanuts
Spot,	Read with Dick and Jane
Toto,	Wizard of Oz

The program should produce output

```
Enter name of dataset: dogs.csv

                                        10ls

2

20

Asta

Wizard of Oz
```