CSE 1321L: Programming and Problem Solving I Lab

Lab 8
Python Modules & Libraries

What students will learn:

- How to import and use built-in Python modules.
- How to generate predictable random values using the random module.
- How to work with Unix timestamps and readable dates using the time module.
- How to use common mathematical functions from the math module.

Content

Overview

During this week we learned how Python can be extended through built-in modules and libraries. In this lab you will practice importing modules and using functions from random, time, and math to solve small programming problems. Pay close attention to the required module for each part of the lab and follow the sample output carefully.

As with previous weeks, all labs should have the appropriate file names:

- Lab8A.py
- Lab8B.py
- Lab8C.py

Lastly, make sure you review the sample output and make sure the output of your program follows the exact same format including the input statements, print statements, spacing, and punctuation. As always, user input is shown in **red and bold** in class, but plain text is acceptable in your own testing.

Lab8A: Seeded Dice Roller

In this lab, you are going to practice using the random module. For this reason, your solution for this lab must import and use the random module.

Imagine you want to simulate rolling a six-sided die several times. To make the results repeatable for testing, the user will first provide a seed value.

For this lab:

- Prompt the user to enter an integer seed value and store it.
- Prompt the user to enter how many times the die should be rolled.
- Use random.seed() with the user's seed value.
- Use random.randint(1, 6) to generate each die roll.
- Print each roll on its own line using the format shown in the sample output.
- Keep track of the total of all die rolls.
- After all rolls are complete, output the total.

Note:

- A standard die has values from 1 through 6, and both values are inclusive.
- Because a seed is used, the same input should produce the same random sequence every time the program is run.
- You may assume the user will enter positive integer values.

Sample Output #1:

```
Enter a seed value: 7
How many times would you like to roll the die? 5
Roll 1: 3
Roll 2: 2
Roll 3: 4
Roll 4: 6
Roll 5: 1
Total: 16
```

Sample Output #2:

```
Enter a seed value: 12
How many times would you like to roll the die? 3
Roll 1: 4
Roll 2: 3
Roll 3: 6
Total: 13
```

Lab8B: Timestamp Converter

In this lab, you are going to practice using the time module. For this lab, your solution must import and use the time module.

Computers often store time as the number of seconds that have passed since January 1, 1970 at 00:00:00 UTC. This value is called a Unix timestamp. In this program, you will convert such a timestamp into a more readable date and time.

For this lab:

- Prompt the user to enter a whole number representing seconds since January 1, 1970.
- Store the value as an integer.
- Use time.ctime() to convert the timestamp into a readable date and time string.
- Output both the original timestamp and the converted date in the exact format shown in the sample output.
- Then prompt the user to enter a small pause length in seconds.
- Print "Waiting..." and use time.sleep() with the user's pause length.
- After the pause, print "Done!"

Note:

- You may assume the user enters a non-negative integer timestamp.
- The pause value should also be read as an integer.
- The readable date format is created by the time module, so your output may include abbreviations such as Fri, Jan, or UTC-based system formatting.
- The time.ctime() function converts a timestamp into a string based on your **local time.** So, if you are not in eastern time zone you will see a bit different date and time.

Sample Output #1 (If you are in Eastern Time):

```
Enter the timestamp in seconds: 0
Timestamp: 0
Date and time: Wed Dec 31 19:00:00 1969
Enter pause length in seconds: 1
Waiting...
Done!
```

Sample Output #2 (If you are in Eastern Time):

```
Enter the timestamp in seconds: 60
Timestamp: 60
Date and time: Wed Dec 31 19:01:00 1969
Enter pause length in seconds: 2
Waiting...
Done!
```

Sample Output #3: (You need to adjust your system time zone to UTC to see this output)

```
Enter the timestamp in seconds: 0
Timestamp: 0
Date and time: Thu Jan  1 00:00:00 1970
Enter pause length in seconds: 1
Waiting...
Done!
```

Lab8C: Math Helper Report

In this lab, you are going to practice using the math module. For this lab, your solution must import and use the math module.

Many real programs need mathematical helper functions. In this problem, your program will take two numeric inputs and build a small report using functions from the math module.

For this lab:

- Prompt the user to enter a decimal number and store it as a float.
- Prompt the user to enter a positive number and store it as a float.
- Using the first number, output the ceiling, floor, and absolute value.
- Using the second number, output the square root and cube root.
- Using the second number, output the logarithm base 2.
- Display each result using the exact labels shown in the sample output.
- Round all floating-point results to 3 decimal places.

Note:

- Use math.ceil(), math.floor(), and math.fabs() for the first number.
- Use math.sqrt(), math.cbrt(), and math.log(value, 2) for the second number.
- You may assume the second number will always be greater than 0, so square root and logarithm are valid.

Sample Output #1:

```
Enter the first number: -13.5
Enter the second number: 16
Ceiling: -13
Floor: -14
Absolute value: 13.500
Square root: 4.000
Cube root: 2.520
Log base 2: 4.000
```

Sample Output #2:

```
Enter the first number: 8.2
Enter the second number: 125
Ceiling: 9
Floor: 8
Absolute value: 8.200
Square root: 11.180
Cube root: 5.000
Log base 2: 6.966
```

Submission Instructions:

- Programs must follow the output format provided. This includes blank lines, colons (:), capitalization, and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct .py format.
- Programs must be saved in files with the correct file names:
    - Lab8A.py
    - Lab8B.py
    - Lab8C.py.
- Programs (source code files) must be uploaded to Gradescope by the due date.