ITEC 1150 Week 7 Chapter 4

LAB PROJECTS

Program Development Plan (PDP)

This is a step-by-step process for successfully constructing an application. Follow these steps and repeat until you have successfully completed the program.

This is a reminder to use the PDP process.

You do not need to turn in your PDP document, but going through the process will help you design your programs.

PDP template -

1. Problem definition
2. Analysis of variables & functions
3. Steps in algorithm
4. Code (separate .py file with added comments)
5. Screen snips from testing
6. Keep track of any wanted or needed improvements for a future version

General Requirements

All assignments must meet the following requirements:

The program must start with header at top and include appropriate comments throughout. Header example:

```
"""
Author: Erik Granse
Date: 2024-09-02
Description: Calculate and display student's average grades
"""
```

- Ensure the output is *information;* it needs to be a statement which explains the value being displayed (for example, “The average grade is 12.34”). Simply outputting “12.34” is not sufficient.

General Requirements (cont.)

All assignments must meet the following requirements:

- The data in the program must be stored in variables.
- The output **must** come from variables in the program
  - Do not simply hard code the output value in the `print()` statement.
  - Some data will be given to you, and some will be user input—any calculations that need to happen must be in your program. Don't calculate somewhere else and enter the value into your program.

General Requirements (cont.)

All assignments must meet the following requirements:

- All input must be validated to ensure the string from `input()` can be turned into a number without crashing.
- All input must be validated to ensure it meets the requirements of the lab (for example, ensuring an age is >= 0 or a quiz score is between 0 and 10).
- If input is not valid, you must give a message to the user and allow them to try again until the input is valid.
- Exemptions to the above will be called out in the lab sections. **If not exempted, validation is required!**

General Requirements, continued

- MIPO:
  - Main
  - Inputs
  - Processing
  - Outputs
- This is the basic structure all our programs will now follow.
- Add additional functions as necessary, but the MIPO functions must exist and be used.
- Generic exception handling must be used to ensure input errors do not cause a crash.
- Programs must offer restart to the user when they are done.

Lab Section 1: Random Number List

- Create a program named **random_num_list.py**. Your program must:
  - Ask the user how many numbers he/she wants the program to pick for the list. The value must be a whole number > 0.
  - Loop as many times as the user specified. Each time through the loop, pick a random integer between 1 and 100 and add it to a list.
  - Display the sorted list, the total of all values in the list, the minimum value, and the maximum value using the format and methods shown in the sample to the right.

```
Please enter how many random numbers you'd like to generate.
Please enter a whole number: five
Enter a whole number greater than 0: 5
Here is your list of 5 integers, randomly selected and sorted:
[8, 47, 62, 81, 90]
Here is your list, printed with the shortcut method:
8, 47, 62, 81, 90
Here is your list, printed via a loop, with total:
8 + 47 + 62 + 81 + 90 = 288
The minimum is 8 and the maximum is 90.
```

Lab Section 1 – Guidance and Hints

- In **inputs()** simply ask the user how many numbers he/she wants the program to pick for the list and return the value.
- Pass the value to **processing()**., where you create an empty list and loop through the number of times specified by the user. Each time through the loop, add a random number to the list:

```
1. import random # add to the top of your file
2. random_int = random.randrange(1, 101) # random number between 1 and 100 inclusive
```

- When the loop is over, use Python list functions to sort the list, find the total, minimum and maximum of the number list, and then return the list and the other three values.
- In **outputs()**, take in as parameters all the variables that the program has produced. Then print four ways to match the output sample on the previous slide (see slides 15-16 in the lecture notes for hints).

Lab Section 2: Book List

- Create a program named **book_list.py:**, using **lab4starter_book_list.py** from D2L resources as a starting point. The program summarizes costs of a book list. It uses all our standard mipo features.*
- The starter file collects book prices into a list, and totals, averages & displays the info. The revised program must:
  - Add a feature to collect the book titles
  - Store the book titles into their own list, parallel with prices
  - Display the titles with the prices as shown on the following slide

Lab Section 2: Book List, Cont.

```
This program summarizes a book list.
Enter the number of books that you need
Please enter a whole number: 3
Enter the cost of book #1, to the nearest dollar:
Please enter a whole number: 18
Enter the cost of book #2, to the nearest dollar:
Please enter a whole number: 35
Enter the cost of book #3, to the nearest dollar:
Please enter a whole number: 27
Info on 3 Books Needed
Book #          Price
 1          $   18.00
 2          $   35.00
 3          $   27.00
Total       $   80.00
Average     $   26.67
Need more books? Enter y or n: n
Thanks for using the program.
```

This is the output from the starter program.

You must modify the starter program to produce this output.

Note the prompts to the user for the titles, and the titles in the price list.

```
This program summarizes a book list.
Enter the number of books that you need
Please enter a whole number: 3
Enter the book title: Python Primer
Enter the cost of book #1, to the nearest dollar:
Please enter a whole number: 18
Enter the book title: Java Jumpstart
Enter the cost of book #2, to the nearest dollar:
Please enter a whole number: 35
Enter the book title: Clojure Codex
Enter the cost of book #3, to the nearest dollar:
Please enter a whole number: 27
Info on 3 Books Needed
Book #                                  Price
Python Primer                       $   18.00
Java Jumpstart                      $   35.00
Clojure Codex                       $   27.00
Total                               $   80.00
Average                             $   26.67
Need more books? Enter y or n: n
Thanks for using the program.
```

Upload & Read Chapter 5: Dictionaries

- **Remember to review the general requirements in slides 3-5. These apply to every program you write and are in addition to the specific requirements of each lab section.**
- Submit `random_num_list.py` and `book_list.py` programs before the deadline.
- Read Chapter 5 of Automate the Boring Stuff for next week.
- Questions? Ask before the deadlines!