ITEC 1150 Week 11
Chapter 8
Lab Projects

INPUT VALIDATION USING PYINPUTPLUS FUNCTIONS

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: Book List

- Create a program named `book_list.py`. This can be a modification of your Lab 4 program, or you can use the 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:
  - Use PyInputPlus functions to perform **all** the input validation and for the retry input
  - Ask the user how many books they need.
  - For each book, get the title and price from the user.
  - Allow prices to be input as decimal values
  - Restrict individual book prices to $1.00 minimum and $100.00 maximum
  - Display the titles with the prices, total, and average in a formatted table.
- Clearly document, with comments, your use of the PyInputPlus functions.
- See the example on the following slide.

Lab Section 1 – Example

- In the sample output on the right, note the inputs that fail validation on lines 2, 4, 7, 12, and 14. This is all done with PyInputPlus. Do not write your own validation code.
- Note the formatting of the output table—dollar signs and decimal points aligned.
- Titles must be displayed in title case.

```
1.  Welcome to the book list program.
2.  Enter the number of books needed: two
3.  'two' is not a number.
4.  Enter the number of books needed: -1
5.  Number must be at minimum 1.
6.  Enter the number of books needed: 2
7.  Enter the title of book #1:
8.  Blank values are not allowed.
9.  Enter the title of book #1: Java Jumpstart
10. Enter the price: $24.95
11. Enter the title of book #2: Python Primer
12. Enter the price: $
13. Blank values are not allowed.
14. Enter the price: $-10
15. Number must be at minimum 1.
16. Enter the price: $34
17. Info on the 2 books needed:
18. Java Jumpstart          $     24.95
19. Python Primer           $     34.00
20. Total                   $     58.95
21. Average                 $     29.48
22. Do you want to enter more books? Enter yes/no:
```

Lab Section 2: Sandwich Maker

The MIPO structure is not necessary for this section!

- Write a program named `sandwich_maker.py` that asks users for their sandwich preferences. The program must:
- Use PyInputPlus functions to perform **all** the input validation and for the retry input
- Ask the user how many sandwiches they want. Make sure this number is 1 or more.
- For each sandwich:
  - Ask the user for a type of bread as a numbered menu with the options wheat, white, or sourdough
  - Ask the user for a protein type as a numbered menu with the options chicken, turkey, ham, or tofu
  - Ask the user if they want cheese as a yes/no option
    - If yes, ask the user for a cheese type as a numbered menu with the options cheddar, swiss, or mozzarella
  - Ask the user if they want mayo, mustard, lettuce, or tomato (four yes/no options)

Lab Section 2: Sandwich Maker (cont.)

- Your program must assign prices for each of the options in your program, with a different price for each item.
- After the user is done entering all sandwiches, display a list of the chosen ingredients and prices for each ingredient in each sandwich, with a total for the sandwich.
- After displaying all sandwiches, display the total cost of the order.
- See the example on the next slide.

- Hints:
  - A list is a good way to store the user's selections for a single sandwich
  - Another list is a good way to store the collection of sandwiches
  - A dictionary is a good way to store ingredient / price combinations
  - You don't need to store the price in the sandwich if you can look it up when printing…

Lab Section 2 — Example Output

```
How many sandwiches would you like to order? 2
Building sandwich #1...
What type of bread would you like?
1. White
2. Wheat
3. Sourdough
1
What protein would you like?
1. Chicken
2. Turkey
3. Ham
4. Tofu
3
Do you want cheese? (yes/no):y
What type of cheese would you like?
1. Cheddar
2. Swiss
3. Mozzarella
3
Do you want mayo? (yes/no):n
Do you want mustard? (yes/no):4
'4' is not a valid yes/no response.
Do you want mustard? (yes/no):y
Do you want lettuce? (yes/no):y
Do you want tomato? (yes/no):n
Building sandwich #2...
What type of bread would you like?
1. White
2. Wheat
3. Sourdough
3
What protein would you like?
1. Chicken
2. Turkey
3. Ham
4. Tofu
4
Do you want cheese? (yes/no):n
Do you want mayo? (yes/no):y
Do you want mustard? (yes/no):n
Do you want lettuce? (yes/no):n
Do you want tomato? (yes/no):y
Sandwich # 1:
White          $    1.95
Ham            $    2.50
Mozzarella     $    1.25
Mustard        $    0.05
Lettuce        $    0.50
------------------------
Subtotal       $    6.25
Sandwich # 2:
Sourdough      $    2.25
Tofu           $    1.99
Mayo           $    0.10
Tomato         $    0.75
------------------------
Subtotal       $    5.09
------------------------
Total          $   11.34
Do you want to start a new order? (yes/no)
```

Submit labs & Then DO Homework for Chapter 9!

Remember to comment your code - for every program!

Save your programs for future reference.

Submit book_list.py and sandwich_maker.py to D2L before the deadline.

Questions or need help? Ask before the deadline!

Submit new/improved lab files if you want to and then read Chapter 9 for next week.