

ITEC 1150 Week 6

Chapter 3

Lab Projects

FUNCTIONS, SCOPE OF
VARIABLES, EXCEPTION
HANDLING

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

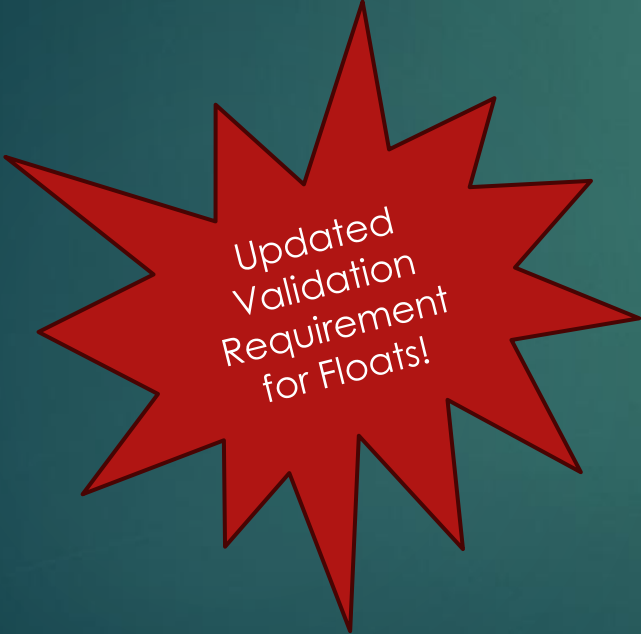
- ▶ Ensure the output is *information*; it needs to be a statement which explains the value being displayed (for example, "The average grade is 12.34"). Simply outputting "12.34" is not sufficient.

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.)



Updated Validation Requirement for Floats!

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

All assignments must meet the requirements on the right:



New requirements this week!

- ▶ 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: Loop Counter 2 (again)

- ▶ Create a program named `loop_counter_2.py`. Your program must:
 - ▶ Ask the user for a small number.
 - ▶ Ask the user for second, larger number.
 - ▶ In addition to requiring whole numbers for both inputs, you must ensure the second number is greater than the first number.
 - ▶ Use a loop and the range function to print the numbers between the first and second number *inclusive*.
 - ▶ Display the sum of all numbers printed.

Review the General Requirements slides carefully!
Your programs must do what is on this slide *and*
meet all the General Requirements!
The big change this week is the MIPO structure.

Sample output:

```
Welcome to our counting program.  
(It also adds up the digits counted!)  
Please enter a small number: two  
Please enter a whole number only: -1  
Please enter a whole number only: 2  
Please enter a whole number larger than 2: 1  
Please enter a whole number larger than 2: 5  
2  
3  
4  
5  
The total of all the counted numbers is 14.
```

Lab Section 2: Quiz Average or Average Rainfall

- ▶ Choose one of the following two python projects to complete:
 - ▶ quiz_average.py or
 - ▶ average_rainfall.py:
- ▶ Those are shown on the next two slides

Lab Section 2: Quiz Average (again)

- ▶ Create a program named `quiz_average.py`. The program must:
 - ▶ Ask the user for the number of students in the class.
 - ▶ Ask the user for the number of quizzes taken by students in the class.
 - ▶ For each student, ask the user for the score of each quiz.
 - ▶ Calculate and display the total and average quiz score for each student.

Review the General Requirements slides carefully!
Your programs must do what is on this slide *and*
meet all the General Requirements!
The big change this week is the MIPO structure.

Sample output:

```
How many students in the class? 2
How many quiz scores to enter per student? 3
Enter scores for student 1:
    What was the score for quiz #1? 92
    What was the score for quiz #2? 88
    What was the score for quiz #3? 91
The total points for student 1 is 271.00.
The average score for student 1 is 90.33.
Enter scores for student 2:
    What was the score for quiz #1? 87
    What was the score for quiz #2? 81
    What was the score for quiz #3? 90
The total points for student 2 is 258.00.
The average score for student 2 is 86.00.
```

Lab Section 2: Average Rainfall (again)

- ▶ Create a program named `average_rainfall.py`. The program must:
 - ▶ Ask the user for the number of years in their study.
 - ▶ For each year, ask the user for the amount of rainfall in each of the 12 months.
 - ▶ At the end of each year, display the total and average rainfall for the year.
- ▶ Challenge: add a feature to show the total and average rainfall for the entire study period (multi-year figures).

Review the General Requirements slides carefully!
Your programs must do what is on this slide *and*
meet all the General Requirements!
The big change this week is the MIPO structure.

Sample output:

```
How many years are in the rainfall sample? 2
Rainfall data for year #1:
Enter rain for month #1: 1
Enter rain for month #2: 2
Enter rain for month #3: 3
Enter rain for month #4: 1
Enter rain for month #5: 2
Enter rain for month #6: 3
Enter rain for month #7: 1
Enter rain for month #8: 2
Enter rain for month #9: 3
Enter rain for month #10: 1
Enter rain for month #11: 2
Enter rain for month #12: 3
Total rainfall for year #1: 24.00
Average monthly rainfall for year #1: 2.00
Rainfall data for year #2:
Enter rain for month #1: 3
Enter rain for month #2: 4
Enter rain for month #3: 5
Enter rain for month #4: 3
Enter rain for month #5: 4
Enter rain for month #6: 5
Enter rain for month #7: 3
Enter rain for month #8: 4
Enter rain for month #9: 5
Enter rain for month #10: 3
Enter rain for month #11: 4
Enter rain for month #12: 5
Total rainfall for year #2: 48.00
Average monthly rainfall for year #2: 4.00
Total rainfall, all years: 72.00
Average monthly rainfall, all years: 3.00
```

Final Notes



Work with and help your classmates, but DO NOT share your code!



Save all the programs you have written, for future reference.



Submit all your Lab Sections to the Class Lab 3.0 drop box:

loop-counter-2.py
averageg_rainfall_fun.py



Questions? Ask before the deadline!



Read Chapter 4!