# ITEC 1150
# Chapter 9
# Lab Projects

## READING AND WRITING FILES

## 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: Data Summary

MIPO not required for this section

- Use the file ch_9_lab_data.txt from this week's module in D2L for this section. Save it in the same directory as your program.
- Create a program named data_summary.py. The program must:
  - Read the contents of the `ch_9_lab_data.txt` file, which contains a series of integers.
  - Write a summary of the contents of the file to a new file named summary.txt, in a table with labels left aligned and all numbers right aligned. The summary must include:
    - Count of the integers
    - Total (sum) of all integers in the file
    - Average of all integers in the file
  - All rows of the output must have a label describing what the data is (count, total, etc.)
- **Submit the output of your program in a text file along with your python file!**

## Lab Section 2: User Manager

MIPO not required for this section

- Create a program named `user_manager.py`. The program must:
- Initialize a text file to store user information with at least one line. Each line in the file will contain a username and an email address, separated by a space, for example:
  - `egranse egranse@minnstate.edu`
- No spaces are allowed in the username.
- Using PyInputPlus, create a menu which will allow the the user to:
  - View: read the contents of the user file and display all users as shown in the sample on the next slide.
  - Add:
    - Prompt the user to enter one or more users in the format described above.
    - They must be allowed to enter multiple users by separating them with commas.
    - Input validation for adding is not required, but make sure you do not write empty lines to the file!
    - Write each user to the file on a separate line.
  - Exit: exits the program.
- See the example on the next slide.

## Lab Section 2 Sample Output

```
Please select one of the following:
1. view
2. add
3. exit
1
Username: dduck          Email:dduck@wb.com
Please select one of the following:
1. view
2. add
3. exit
2
Please enter users in the format 'username email'. Separate multiple records with commas: bbunny bugs.bunny@wb.com, mmouse mickey@disney.com
Please select one of the following:
1. view
2. add
3. exit
1
Username: dduck          Email:dduck@wb.com
Username: bbunny         Email:bugs.bunny@wb.com
Username: mmouse         Email:mickey@disney.com
Please select one of the following:
1. view
2. add
3. exit
3
Thanks for using the program.
```

The initial user created when the program started

Multiple entries separated by commas

Separate usernames and emails into columns

## Submit labs & Then DO Homework for Chapter 9!

Remember to comment your code - for every program!

Save your programs for future reference.

Submit both programs and the output of the Data Summary section 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 12 for next week.