General Instructions

Make sure that you read and then follow the instructions for each task carefully.

Please make sure that you save all of your work to a safe place and that you make **regular back-ups**.

You should begin all tasks with the following steps unless otherwise stated:

- Create a new Python file with the *Project Name* specified in the file.
- It is suggested that you save the file with your completed code (and/or any other files specified) in a folder named for the tutorial week (*week01, week02,* etc.), which should itself be in a root folder called *sdam*. However, you can use whatever method you wish to organise your work – just make sure that each file can be located quickly and efficiently based on the week number and the project name.

If you are in any doubt, please check with your tutor.

Rate a restaurant
Project Name: restaurant_rating

Beginner

A program is required to prompt users to rate a single restaurant using a number in the range 1 – 5, where 1 is very poor and 5 is very good.

The data input is ended by entering -1.

After all the votes have been counted, the program should output a table showing how many people gave each rating and the average rating.

You should use exceptions where appropriate to prevent the program from crashing when invalid data is entered, and instead output an appropriate error message identifying which error occurred causing the program to terminate.

Step 1: Write your source code

- write the code necessary to prompt the user for the rating.
- Increment the rating value as selected on each input
- When input is concluded output the rating in table form.

Step 2: Run, test and screenshot your application outputs

- run your program with the following input:
  - p (alphabetic)
  - 0 (lower bound)
  - 6 (upper bound)
  - -1 (no ratings entered)
- take screen shots whenever an exception occurs and store them in your project folder as *alphabetic.jpg*, *lower_bound.jpg*, *upper_bound.jpg*, and *divide_zero.jpg* respectively.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *alphabetic.jpg*, *lower_bound.jpg*, *upper_bound.jpg*, and *divide_zero.jpg* showing output from *restaurant_rating.py* when tested.

Character Generator (partially revisited)
Project Name: character_gen_data

Beginner

This task is based on the scenario given in character_gen (week 7).

The minimum ability scores for each character class have been stored in a JSON file. You can download the file from canvas. This data can be extracted as a dictionary.

Your program must extract the data from the JSON file and present it in a tabular format. The result should look similar to the following.

Class	Strength	Intelligence	Wisdom	Dexterity	Constitution
Warrior	15	-	-	12	10
Wizard	-	15	10	10	-
Thief	10	9	-	15	-
Necromancer	10	10	15	-	-

Once complete, take a screenshot of your output as *character_gen_data.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for this task
- *character_gen_data.jpg*

Built-in Errors
Project Name: built_in_errors

Beginner

For the following built-in exceptions you need to state:

1. A brief description of the exception and what causes it
2. An example of code that might cause the error
3. The offending code surrounded by *try* statements with the correct *except* clause (you don’t need to worry about the contents of this, just leave the clause empty)

- ZeroDivisionError
- KeyError
- TypeError
- ValueError
- NameError
- FileNotFoundError

Portfolio

The task contribution to your portfolio is:

- *built_in_errors.docx*

Secure Entry
Project Name: security_checker

Intermediate

Write a password protected entry system. It should display a menu with the following options:

1. New user

2. Existing user

If the user selects option 1 then they will be prompted to enter a username and create a new password.

New passwords must conform to the following rules:

- They must be at least 8 characters long
- They cannot begin, end or contain a space
- They cannot begin with a number.

Once an acceptable password has been entered, they must re-enter the password to confirm it. If they match, then the password will be set and the program will save the new user's details (username and password) to a JSON file.

The program will then return the user to the main menu.

If the user selects option 2 then the user will be prompted to enter their username and password. These will be checked against username/password paired entries in the JSON file.

The system will give the user 3 attempts to enter the correct password. On failure it will tell them they are locked out, on success it will output a nice friendly welcome message.

Your program should be robust and use exception handling to prevent the program from crashing.

Step 1: Write your source code

- Write the code necessary to achieve the required functionality.

Step 2: Run, test and screenshot your application outputs

- Run the program inputting data that is valid (new password that conforms to the rules above) – *security_checker_valid.jpg*
- Run the program inputting data that is invalid (new password that doesn't conform to the rules above) – *security_checker_invalid.jpg*

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- The JSON file with stored username/password pairs *security_checker_users.json*
- Screenshots *security_checker_valid.jpg* and *security_checker_invalid.jpg* showing output from *security_checker.py* when tested.

All portfolio requirements for this tutorial

Beginner

restaurant_rating.py

lower_bound.jpg

upper_bound.jpg

divide_zero.jpg

character_gen_data.py

	character_gen_data.jpg *built_in_errors.docx*
Intermediate	*security_checker.py* *security_checker_valid.jpg* *security_checker_invalid.jpg*