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.

Colourful (revisited)
Project Name: rainbow_test_plan

Beginner

Based on the *rainbow* project from week 6, you are to produce a full test plan for the task (based on the lecture this week and following the example on canvas:

- Create a test plan with the intention of ensuring that all possible entries are tested and handled correctly without crashing your program
- Save your plan as *rainbow_test_plan.docx*.

Step 2: Run your original source code using your full test plan

- Run your program, inputting the data from your test plan
- Note any results in your test plan

Portfolio

The task contribution to your portfolio is:

- The test plan *rainbow_test_plan.docx* including the results of your testing

Functional Calculator (revisited)
Project Name: fun_calc_enhanced

Beginner

Based on the *fun_calc* project from week 8, you are to do the following:

Step 1: Produce a full test plan for the task (based on the lecture this week and following the example on canvas:

- Create a test plan with the intention of ensuring that all possible entries are tested and handled correctly without crashing your program
- Save your plan as *fun_calc_test_plan.docx*.

Step 2: Run your original source code using your full test plan

- Run your program, inputting the data from your test plan
- Note any results

Step 3: Improve your fun_calc code

- Make a copy of your original source code as *fun_calc_enhanced.py*
- If any of your tests don't pass you must fix any problems and test again until all of your tests pass.
  For example, it may be that you will need to make your code more robust against unexpected user input.

Portfolio

The task contribution to your portfolio is:

- The test plan *fun_calc_test_plan.docx*
- The Python source code file for the improved program following your testing – *fun_calc_enhanced.py*

Hangman
Project Name: hangman

Intermediate

Write a program to play a simplified version of Hangman. The program should prompt a user for a word to be guessed and then display dashes in place of each of the letters, for example:

```
Word to be guessed: scratch
Display: -------
```

A second player is then prompted to guess letters in the word. If the letter is present, all occurrences of that letter should be replaced and the word displayed again:

```
Letter guessed: c
Display: -c---c-
```

When all the letters have been correctly guessed, the program should output the number of guesses it took to get the word.

Step 1: Produce a full test plan for the task (based on the lecture this week and following the example on canvas:

- Create a test plan with the intention of ensuring that all possible entries are tested and handled correctly without crashing your program
- Save your plan as *hangman_test_plan.docx*.

Step 2: Write your source code

Step 3: Run, test and screenshot your application outputs

- run the program inputting the data from your test plan
- fix any errors and run your tests again – continue until all tests pass
- take a screen shot of the output from a successfully completed game, call your screenshot *hangman.jpg*.

Portfolio

The task contribution to your portfolio is:

- The test plan *hangman_test_plan.docx*
- The Python source code file for the task
- *hangman.jpg* showing a successfully completed game

Secure Entry (revisited) Project Name: security_checker_test	
Intermediate	Revisit the intermediate task from week 10 called *security_checker*. You will need to have completed the task in order to complete this task. Produce a full test plan for the task (based on the lecture this week and following the example on canvas: • Create a test plan with the intention of ensuring that all possible entries are tested and handled correctly without crashing your program • Save your plan as *security_checker_test_plan.docx*. • Run the program (work on the original code), inputting the data from your test plan • Fix any errors and run your tests again – continue until all tests pass
Portfolio	The task contribution to your portfolio is: • The Python source code file for the original task *security_checker.py*. • The test plan *security_checker_testplan.docx*

All portfolio requirements for this tutorial	
Beginner	*rainbow_test_plan.docx* *fun_calc_test_plan.docx* *fun_calc_enhanced.py*
Intermediate (opt)	*hangman_test_plan.docx* *hangman.py* *hangman.jpg* *security_checker_test_plan.docx* *security_checker.py* (this will be the original file – including any changes you have made through testing)