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.

Sum of Positive and Negative Integers
Project Name: sum_pos_neg

Beginner

Write a program that prompts the user to enter 10 integer values, which can be positive or negative, and calculates:

- the sum of all the positive integers the user entered
- the sum of all the negative integers the user entered.

You must use a single for loop to accomplish this.

Your program should then output the total sum of the positive numbers and the total sum of the negative numbers.

When you analyse the problem make sure you consider the following factors:

- how many times the loop is repeated?
- what operations are done before the loop?
- what operations are done inside the loop?
- what operations are done after the loop?

Write and test your program with 5 positive and 5 negative values. Take a screen shot of the output and save it as a file in your project folder called s*um_pos_neg.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *sum_pos_neg.jpg* showing output from *sum_pos_neg.py* when tested

Grid
Project Name: grid

Beginner

Write a program to prompt the user to enter 2 numbers: one for number of columns and one for number of rows and outputs a grid of asterisks with the specified number of rows and columns

Hint: You will need to use one loop nested inside another.

For example, given width = 3 and height = 4, the program should output the following grid:

```
* * *
* * *
* * *
* * *
```

Write and test your program with no less than 5 rows and 5 columns. Take a screen shot of the output and save it as a file in your project folder called *grid.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *grid.jpg* showing output from *grid.py* when tested

Match the Dice
Project Name: dice_match

Beginner

Implement a program to simulate throwing 2 dice and output how many throws there were before the dice matched. The values on the pairs of die should be output for each throw, including the matching pair

When you analyse the problem make sure you consider the following factors:

- how many times the loop is repeated?
- what operations are done before the loop?
- what operations are done inside the loop?
- what operations are done after the loop?

Write and test your program. Take a screen shot of the output and save it as a file in your project folder called *dice_match.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *dice_match.jpg* showing output from *dice_match.py* when tested

Determined Loops
Project Name: loop_types

Beginner

Just a Word document is required for this task.

State whether each of the following is deterministic, non-deterministic or either (ie could be solved deterministically or non-deterministically) and justify your choice:

a) writing Christmas cards to family
b) washing up
c) reading a book
d) waiting for someone to answer the phone
e) counting to 10 in French
f) playing a game of 501 in darts
g) laying the table for a dinner party
h) answering this question

Store your answers in a Word document called *loop_types.docx* in your *w05_t2* folder.

Portfolio

The task contribution to your portfolio is:

- *loop_types.docx*

Average of Positive and Negatives
Project Name: average_pos_neg

Intermediate

Using your code from task 1 as a guide extend the functionality so that in addition to outputting the sums of the positive and negative integers input it also calculates and outputs the average of the positive and the average of the negative integers entered.

Appropriate messages should be output if there were no positive numbers or if there were no negative numbers entered.

Write and test your program with different sets of test values.

When complete test your program with the following:

- all positive integers. Take a screen shot of the output and save it as a file in your project folder called *apn_all_positives.jpg*.
- all negative integers. Take a screen shot of the output and save it as a file in your project folder called *apn_all_negatives.jpg*.
- 5 positive and 5 negative integers. Take a screen shot of the output and save it as a file in your project folder called *apn_mix.jpg*
- all zeroes. Take a screen shot of the output and save it as a file in your project folder called *apn_zero.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *apn_all_positives.jpg*, *apn_all_negatives.jpg*, *apn_mix.jpg* and *apn_zero.jpg* showing output from *average_pos_neg.py* when tested

Reversing an Integer
Project Name: reversal

Intermediate

Write a program that prompts the user for a positive integer of more than 1 digit and less than 11 digits (it must output a suitable error message if the input is not valid). Once the user has input a valid integer your program will output the reverse of the integer and re-prompt for another integer. The process should continue until a suitable negative number is input.

The type you get from the user must be an integer. You must not convert the integer to a string or any other form of collection in order to reverse it.

For example:

```
Enter an integer of at least 2 digits or -1 to quit: 1
Your input is invalid, please try again.
Enter an integer of at least 2 digits or -1 to quit: 123456
Your integer reversed is: 654321
Enter an integer of at least 2 digits or -1 to quit: -1
```

```
Program end.
```

Once your program is complete, test it with various inputs until you are confident it is functioning correctly.

Write a test plan that will test a suitable range of inputs (valid integer, integer with too few digits, integer with too many digits and a negative integer) and save it in your project folder as r*everse_test_plan.docx*. Make sure you number your tests e.g. Test 1: Single Digit; Test 2: 11 digits, Test 3: Number ending in 0, etc.

Run your program, testing it against your test plan until all results are as you predicted. Take a screen shot of each test saved with the name *rev_test_n.jpg* (where n is the number of the test that corresponds to your plan).

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- All *jpg* files you have created from running your tests showing output from *reversal.py* when tested

Diamond
Project Name: diamond

Expert

Write an application that uses loops to output a diamond where the width is defined by the user but should be no less than 2 and no more than 40. Your program should output an appropriate message if the width entered is outside of the permitted range.

For example, given a width of 4, the program should output the following:

```
   *
  * *
 * * *
* * * *
 * * *
  * *
   *
```

Write and test your program with a width of no less than 5 and a width that exceeds the number permitted. Take a screen shot of the output from both tests and save it as a file in your project folder called *diamond.jpg* and *diamond_error.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *diamond.jpg* and *diamond_error.jpg* showing output from *character_gen.py* when tested

Child's Play
Project Name: paper_rock_scissors

Expert

Write a game of Paper, Rock, Scissors; where the user is prompted to enter their choice as a case-insensitive string (e.g. Paper or paper would be valid inputs) and the computer's choice is generated randomly.

The first to score 3 wins.

At the conclusion of the game your program should output the result (who won) and the scores of both players.

Those not familiar with the game can learn a little about it here (https://en.wikipedia.org/wiki/Rock-paper-scissors)

When you have completed your program, take a screen shot of the last three plays and the concluding output (see Fig. 1 for an example). Save your screenshot as *paper_rock_scissors.jpg*.

```
The computer chose: Rock
It was a draw.
You need : 3 to complete the game.
The computer needs : 2 to complete the game.
********************
Enter your selection: paper
You chose: Paper
The computer chose: Scissors
The computer won.
You need : 3 to complete the game.
The computer needs : 1 to complete the game.
********************
Enter your selection: paper
You chose: Paper
The computer chose: Rock
Congratulations, you win
You need : 2 to complete the game.
The computer needs : 1 to complete the game.
********************
Enter your selection: scissors
You chose: Scissors
The computer chose: Rock
The computer won.
You need : 2 to complete the game.
The computer needs : 0 to complete the game.
********************
Commiserations the computer won...
Game over...

Process finished with exit code 0
```

Figure 1: Last 3 plays from Paper, Rock, Scissors

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *paper_rock_scissors.jpg* showing output from *paper_rock_scissors.py* when tested

All portfolio requirements for this tutorial

Beginner

sum_pos_neg.jpg

sum_pos_neg.py

grid.jpg

grid.py dice_match.jpg

dice_match.py

loop_types.docx

Intermediate (opt)

apn_all_positives.jpg

	apn_all_negatives.jpg
	apn_mix.jpg
	apn_zero.jpg
	average_pos_neg.py
	rev_test_n.jpg (all test outputs)
	reversal.py
Expert (opt)	*diamond.py*
	Diamond.jpg
	diamond_error.jpg
	paper_rock_scissors.jpg
	paper_rock_scissors.py