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 **<u>regular back-ups</u>**.

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.

Understanding Conditions
Project Name: conditions

Beginner

You only need to complete the following as a Word document. No other software is required.

Assuming a = 5, b = 4, c = 3, indicate whether each of the following conditions resolve to true or false:

a) (a < b)
b) (a != b)
c) (a > b) and (a < c)
d) (a > b) or (a < c)
e) (a – 1 == b)
f) not(b > c)

Store your answers in a Word document called c*onditions.docx* in the *w03_t2* folder within your *software_dev* folder.

Portfolio

The task contribution to your portfolio is:

- *conditions.docx* containing your answers

Odd or Even
Project Name: odd_or_even

Beginner

Write a program to prompt the user to input an integer and output whether the number entered is odd or even.

Working out whether the number is odd or even must be done as a single Boolean expression. Whether a number is odd or even will be given by a result of TRUE or FALSE.

Assume all inputs will be positive integers.

Test your program with one odd number, one even number and zero.

Take 3 screenshots of the output from each run and store them in your project folder as o*dd.jpg*, *even.jpg*, *zero.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *odd.jpg*, *even.jpg*, *zero.jpg* showing output from *odd_or_even.py* when tested

Check Password
Project Name: password

Beginner

Write a program that stores a password in the code as a string (e.g. pwd = “Rocket”) then prompt the user to enter the password (as a string).

If the password entered by the user matches the password you have stored in the program, regardless of case, output a welcome message.

Your analysis should consider the following:

- What data is used?
- Is all data dealt with in same way?
- What operations are done before the selection?
- What operations are done for each possibility?
- What operations are done after the selection?

When complete test your program with a valid and invalid input from the user. Take a screenshot of the output from each run and store them in your project folder as *password_valid.jpg* and *password_invalid.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *password_valid.jpg*, *password_invalid.jpg* showing output from *password.py* when tested

Nursery Rhymes
Project Name: magpie

Beginner

Write a program to prompt the user for an input integer between 1 and 7 inclusive.

Your program should then output the following in response:

- If user enters 1, output “One for sorrow”
- If user enters 2, output “Two for joy”
- If user enters 3, output “Three for a girl”
- If user enters 4, output “Four for a boy”
- If user enters 5, output “Five for silver”
- If user enters 6, output “Six for gold”
- If user enters 7, output “Seven for a secret never to be told”
- If the user enters any other number, output “Not a permitted number”

Your analysis should consider the following:

- What data is used?
- Is all data dealt with in same way?
- What operations are done before the selection?
- What operations are done for each possibility?

- What operations are done after the selection?

Test your program with each permitted input and with one number outside the permitted input range.

When complete test your program again with a permitted and out of range input from the user. Take a screenshot of the output from each run and store them in your project folder as *magpie_valid.jpg* and *magpie_invalid.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *magpie_valid.jpg*, *magpie_invalid.jpg* showing output from *magpie.py* when tested

Numbers in a Series
Project Name: series

Intermediate

Write a program to prompt the user for 5 integers either positive or negative (assume Zero will not be an input). Sum the total of positive integers and the total of negative integers and then output the totals.

For example, if the inputs were: -5 12 -19 -3 6 the program would output:

Sum of positive integers: 18

Sum of negative integers: -27

Test your program with a suitable range of positive and negative numbers.

Take a screenshot of the output and store it in your project folder as *series.jpg*.

Portfolio

The task contribution to your portfolio is:

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

Nothing for a Pair (not in this game)
Project Name: high_low

Expert

You will need to import random

Part 1

Write a program that will deal a playing card randomly. For the value generated your program should output the following:

Number generated	**Value displayed**
1	Ace
2 – 10	Face value, so if a 2 display a 2 etc.
11	Jack
12	Queen
13	King

Run and test your program outputting the actual random number generated and the corresponding card value.

Take a screenshot of the output and store it in your project folder as *high_low_part1.jpg*.

Part 2

Extend your program so that it has the following functionality:

- Displays a random card
- Prompts the user to guess whether the next card will be higher or lower (if it is equal the guess is incorrect). Use the inputs of 'higher' or 'lower' to indicate the users guess.
- "Deals" and displays a second card
- If the user was correct in their guess display an appropriate win message and if they were incorrect display an appropriate lose message

Hint: When testing application logic that includes randomly generated values in the condition you may find temporarily hard coding values just to test conditional logic beneficial, once the logic is proved you can then reinstate the random numbers.

When complete, run your program 3 times taking a screenshot of the output each time as *high_low_run1.jpg*, *high_low_run2.jpg*, *high_low_run3.jpg* and store it in your project folder.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *high_low_part1.jpg, high_low_run1.jpg*, *high_low_run2.jpg*, *high_low_run3.jpg* showing output from *high_low.py*

A Night at the Theatre
Project Name: theatre

Expert

A theatre sells tickets according to the following details:

Ticket	**Stalls**
Adult	£10.50
Child	£7.30
Concessions	£8.40

There are two offers that should be automatically applied:

- 1 free adult chaperone place for every group of 10 children, for example:
    - for up to 10 children, all adults must pay
    - between 10 and 19 children: up to 1 free adult
    - between 20 and 29 children: up to 2 free adults
    - […]
    - between 100 and 109 children: up to 10 free adults

- 10% discount if the bill total exceeds £100.00, after any free chaperone seats have been dealt with

Concessions are available for adults over the age of 65.

All children must be accompanied by at least one adult.

If the tickets are to be collected in person there is no additional charge. Otherwise postage & packaging, which costs £2.34, is added after all offers have been applied.

Write a program to prompt the user for ticket requirements (total in party, number of adults, number of concessions) and output the bill as a formatted receipt with details and final total.

Develop a full test plan to test your program to show you have tested for:

- A party where no discounts are applicable
- A party where 1 adult discount is applicable
- A party where the 10% discount is applied
- A party where there is at least 1 concessionary ticket

When complete, test your program with the required test examples and take a screenshot of the output from each run and store them in your project folder as *theatre_test1.jpg*, *theatre_test2.jpg*, *theatre_test3.jpg* and *theatre_test4.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file for the task
- *theatre_test1.jpg*, *theatre_test2.jpg*, *theatre_test3.jpg* and *theatre_test4.jpg* showing output from *theatre.py* when tested

All portfolio requirements for this tutorial

Beginner	*conditions.docx* *odd.jpg* *even.jpg* *zero.jpg* *odd_or_even.py password.py* *password_valid.jpg* *password_invalid.jpg* *magpie.py* *magpie_valid.jgp* *magpie_invalid.jpg*
Intermediate (opt)	*series.jpg* *series.py*
Expert (opt)	*high_low_part1.jpg*

high_low_run1.jpg

high_low_run2.jpg

high_low_run3.jpg

high_low.py

theatre.py

theatre_test1.jpg

theatre_test2.jpg

theatre_test3.jpg

theatre_test4.jpg