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 name – for example, the first task would have a file called *hello_world.py*.
- 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.

Hello World
Project Name: hello_world

Beginner

Step 1: Create a new folder

- Navigate to your H drive (or any alternative personal storage space such as a pen drive or OneDrive) and create a folder called *sdam*
- Create a subfolder within *sdam* called *week01*

Folder names can be as long as you like but it is best to choose short but meaningful names. By convention you should use lower-case letters with underscores where clarity requires.

NB: The following examples assume that you have created a folder called *sdam* containing a subfolder called *week01*.

Step 2: Create a Python project using a suitable Python editor such as PyCharm or IDLE

- Using the information given in the lecture create a new file within your *week01* folder called *hello_world.py*

Step 3: Write source code

- Type in the following code

```
# output Hello World

print("Hello World")
print("Welcome to Python")
```

- Get into the habit of putting a comment at the start of the program to explain what the application does.

Step 4: Run the program

- Using the information given in the lecture run the *hello_world* program after correcting any errors that may occur.

Step 5: Take a screen shot of the output

- Ensure that the screen in which the application is running is active and that the output of the program is shown in its entirety
- Press [ALT + Print Screen]
- Open Paint (or similar)
- Paste the screenshot [CTRL + V]
- Save the image in your *week01* folder as *hello_world.jpg*

Portfolio

The task contribution to your portfolio is:

- The Python source code file *hello_world.py*
- *hello_world.jpg* from Step 5

Make sure that both of these files are saved in your *week01* folder

Debugging code
Project Name: debug

Beginner

Step 1: Write the source code

- Add a new Python file called *debug.py*
- Copy and paste the code below

```
# exercise to debug code

name = "John Smith";  # remove one character
occupation = 'Student"  # replace one character
|ocation = 'Stoke-on-Trent'  # replace one character
activity level = "moderate"  # replace one character (too many spaces)

print(My name is", name) #add one character
print("I am a student in " + location)  # no errors in this line

age == 21  # remove a character

print("I am " + str(age) + " years of age with a", activity level, "activity level"
# add one character and replace one character
```

Step 2: Debug the program

- Using the information given in the lecture (and in the comments of the code) debug the program to fix all of the errors
- Run the application after correcting any errors that you find.

NB: PyCharm will highlight any errors that it identifies. If you hover your cursor over the error it will also give you information on how to correct the errors. Be aware that some errors may be identified because of problems earlier in the code – correcting earlier problems sometimes removes errors that PyCharm has found.
You will find this task slightly more difficult with IDLE!

When all errors have been corrected the program should run with the following output:

```
My name is John Smith
I am a student in Stoke-on-Trent
I am 21 years of age with a moderate activity level
```

Step 3: Take a screen shot of the output

- Ensure that the screen in which the application is running is active and that you have corrected all of the errors
- Press [ALT + Print Screen]
- Open Paint

- Paste the screenshot [CTRL + V]
- Save the image in your *week01* folder as *debug.jpg*

Portfolio

The task contribution to your portfolio is:

- The correctly working Python source code file *debug.py*
- *debug.jpg* from Step 3

Save both of these files in your *week01* folder

Basic data types
Project Name: datatypes

Beginner

Create a text document called *datatypes.txt*

State the most appropriate data type for the following data with a brief justification for your decision:

a. A company's profit
b. A person's weight
c. A book title
d. Average rainfall in September
e. Number of runs scored by a cricketer
f. A telephone number, including area code
g. A dress size

Example:

a) float – because the profit number may contain decimal places

Save your file in your *week01* folder.

Portfolio

The task contribution to your portfolio is:

- The completed *datatypes.txt* file

Simple Calculator
Project Name: simple_math

Beginner

Step 1: Create a new Python script file

- Create a new file called *simple_math.py* in your *week01* folder
- Type the following code into your file

```
age = 2
print(age)
age = 21
print(age)
age = age + 1
print("Age next Birthday: " + str(age))
```

Step 2: Run your program

- Run the program and make sure that you are getting the output that you are expecting.

Step 3: Modify your code

- Change the type of age on the third line to a *float* (you are **not** to do this by changing 21 to 21.0 directly – find another way).

Step 4: Run program

- Run the program again and the output should be different (if it isn't different then ask your tutor to help you to identify the problem)
- Why would the output be different?

Step 5: Take a screenshot of the output

- Take a screenshot of the application window showing the program output
- Paste the screenshot into Paint and save it in your project folder as *simple_math.jpg*.

Portfolio

The task contribution to your portfolio is:

- The Python source code file *simple_math.py*
- *simple_math.jpg* from Step 5

Breakeven
Project Name: breakeven

Beginner

The break-even point for a business is defined as the point where the total revenue equals the business expenses (fixed costs). The following values are required to calculate this:

- What is the cost to produce each item
- What is the sale price per item (item cost + profit per item)
- What are the fixed costs

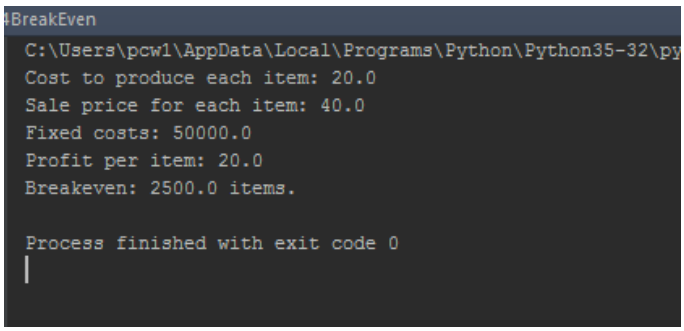
Dividing the fixed costs by the profit per item (the difference between the cost to produce each item and the sale price) gives the number of items that must be sold before "breaking even".

For example, if it costs £50.00 to produce an item which is then sold at £100.00, there is £50.00 profit per item. If the fixed costs are £1000.00, the company would need to sell 20 items to break even.

Exercise data:

- Cost to produce each item = £20.00
- The sale price per item (item cost + profit per item) = £40.00
- Fixed costs = £50000.00

Write a program to output the data with a meaningful label for each item and calculate and output the number of items we need to sell to breakeven. The output should look similar to the following (cropped for convenience).



Hint: $break\_even = \frac{fixed\_costs}{sale\_price - item\_cost}$

Step 1: Analyse the problem

- Make sure that you have read through, and understand, the problem
- Decide the type of data being used

Step 2: Create a Python script file

- Create a new file called *breakeven.py*
- Make sure that you put it in your *week01* folder

Step 3: Write source code

- Using your analysis from Step 1, write code to produce the required outputs
- Ensure that you use appropriate names for all your variables

Step 4: Run and test your application

- Run your application with the example data to ensure that you get the correct answer
- Identify and correct any errors in your source code if you don't get the output that you were expecting

Step 5: Take a screenshot of the output

- Press [ALT + Print Screen] for your output screen
- Paste your screenshot into Paint and save as *breakeven.jpg*

Portfolio	The task contribution to your portfolio is: • The Python source code file *breakeven.py* • *breakeven.jpg* from Step 5

Sweet Tooth
Project Name: sweet_tooth

Intermediate

A teacher has bought a packet of 40 sweets that she is going to share out equally between her 14 students. Because a single sweet cannot be shared, the teacher will keep what is not given to the children.

The teacher will keep the minimum number of sweets possible (obviously this scenario is not based on real life).

Write a program to determine and output the number of sweets each child will receive. As a single sweet cannot be shared between pupils, the program must calculate and output the number of sweets per child and the number that the teacher keeps for herself. **The calculation must be done using the remainder (%) operator.**

This will give you the amount left over after the division has been carried out. For example: 5 % 2 = 1 (5 divided by 2 is 2 with 1 left over)

Take a screen shot of the finished output saved as *sweet_tooth.jpg* in your *w01_t2* folder.

Portfolio

The task contribution to your portfolio is:

- The Python source code file *sweet_tooth.py*
- *sweet_tooth.jpg* showing output from *sweet_tooth.py*

Cans of Paint
Project Name: cans_of_paint

Intermediate

To help you complete this task you should use the Python *math* module. You can consult the online documentation to see what methods this will make available to you:

https://docs.python.org/3.0/library/math.html

To use a module within your program you must import it using the `import` keyword. By convention, import statements occur at the beginning of a file.

```
import math
```

You will need some of the methods from that module.

For example, to call the `ceil()` method, you type:

`math.ceil(a_number_or_variable)`
e.g.: `math.ceil(num)`

Write a program that uses methods from the `math` module to solve the following problem.

A can of paint covers 5.1 m^2 of wall. Each can of paint has a diameter of 15 cm and a height of 30 cm. The shop that sells the paint helps customers by packing the cans into boxes with internal measurements of 0.60 x 0.30 x 0.35 metres (L x W x H).

Solve and output the following:

1. The minimum number of cans that must be bought to paint the walls of a hall whose floor measures 40 x 30 metres, and whose ceiling is 3.4 metres above the floor (use an appropriate method from the `Math` class).
2. The number of full boxes given to the customer who buys this quantity of paint (use an appropriate method from the `Math` class rather than integer division).
3. The number of cans not packed into boxes.

You should get the following amounts

```
Number of cans required: 94
Number of cans in box: 8
Number of full boxes: 11
Cans not packed in boxes: 6
```

Portfolio

The task contribution to your portfolio is:

- The Python source code file *cans_of_paint.py*
- *cans_of_paint.jpg* showing output from *cans_of_paint.py*

All portfolio requirements for this tutorial	
Beginner	*hello_world.py*
	hello_world.jpg
	debug.py
	debug.jpg
	datatypes.txt
	simple_math.py
	simple_math.jpg
	breakeven.py
	breakeven.jpg
Intermediate	*sweet_tooth.py*
	sweet_tooth.jpg
	cans_of_paint.py
	cans_of_paint.jpg