COM4013 – Introduction to Software Development
Week 1 – Introducing Jupyter Notebook and Simple Programs

*Always select "**Enable Editing**" if prompted by Microsoft Word*

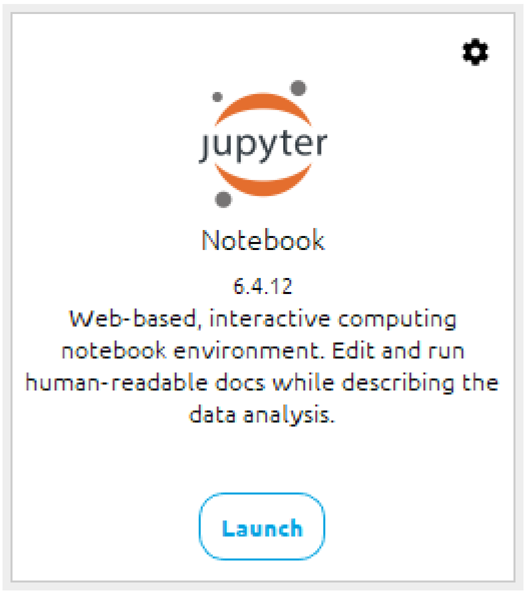
This lab worksheet helps you write your first programs. Experienced students should find it straightforward at first. However, there will be advanced questions later.

There is a lot of reading in this worksheet as we introduce the tools required. Later worksheets will tend to be shorter, with more work for you to do!

Jupyter Notebook

We will use the Jupyter Notebook IDE. This should be available on the LTU network to be installed. For you own computers, you can download Anaconda Studio from any web browser. Anaconda Studio is free to download.

If Anaconda Studio is already on your machine, then run Anaconda Navigator and Lauch Jupyter Notebooks.



Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Create a new folder by clicking [New ▼] and in the Other: section select Folder. "Untitled Folder" should appear in your list of folders. Click on the checkbox next to "Untitled Folder" and then [Rename] the folder to "COM4013 – Introduction to Software Development".

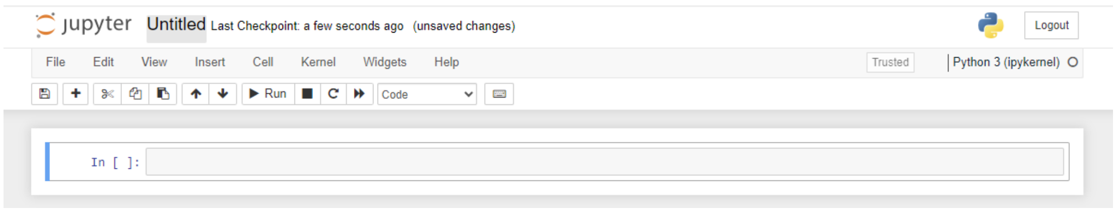
Ask for help if you cannot do any of these steps

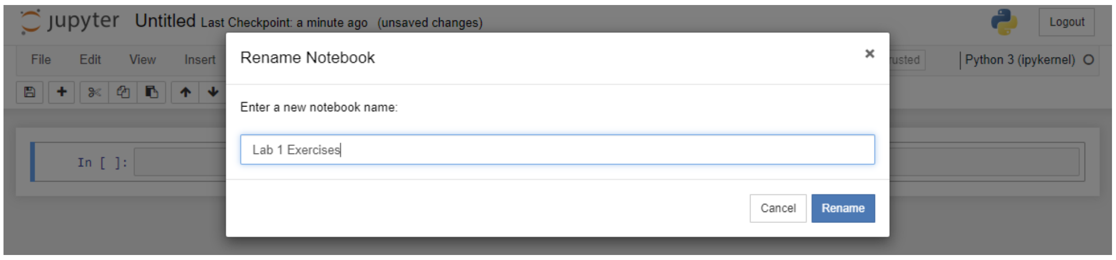
Now go into the "COM4013 – Introduction to Software Development" folder and follow the same instructions as above to create a new folder called "Lab 1".

Creating a New Project

Inside of the "Lab 1" folder, click [New ▼] and select Python 3 (ipykernel).

The Jupyter Notebook IDE should appear. Click on "Untitled" and rename this file to "Lab 1 Exercises".





Rename Notebook

Enter a new notebook name:

Lab 1 Exercises

An Overview of the Jupyter Notebook IDE

The main area is a standard **text editor** for writing/editing your code

- You will notice that as you type Python keywords some of the words are coloured, this is called *syntax highlighting*. It identifies diverse types of programming keywords. You will find this becomes more useful as your programming improves.

Our Basic Shell Code

For most of what we will do this basic shell code should suffice. Copy this into the first cell.

```
def main ():

if __name__ == "__main__":
    main ()
```

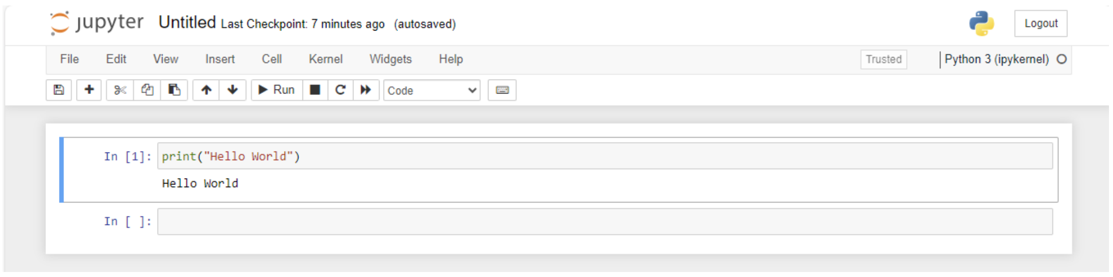
A Simple Program

- ♦ Type this *statement* as a new line under main (it is good practice to use the tab key to *indent* the code one time – as Python will not accept code unless you properly format your code.

```
        print ("Hello World");
```

- ♦ Click run or Shift+Enter (You should see Hello World in the console area). Your tutor will show you how to do this if you get stuck:

- ⚠ Python is *case sensitive*. That means capital letters are considered different to lower case letters. When copying program code copy the capitals exactly. When writing your own program code, you must use capitals consistently (e.g., "Console" is different to "console")
- ⚠ Also notice the semi-colon at the end of the line. In some languages semi-colons are required to separate *statements*, although they are not needed in Python. Remove the semi-colon after the print statements and run the code again. As you can see it still works. Feel free to use the semicolon if you are used to this (or don't). I expect you to be consistent though.


jupyter
Untitled Last Checkpoint: 7 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: print("Hello World")
        Hello World

In [ ]:
```

The program works by using pre-written *function* (`print`) from the Python Console library.

- The *console* is usually a window (usually with white text on a black background) that allows a very direct interface between program and the computer. Jupyter Notebook has an in-built console.
- Although console programming is limited it is a fantastic way to get familiar with core programming concepts without other features getting in the way. So, we will only create console programs in this module (for now), in later modules you will create a wider range of programs.
- We will often use functions from the pre-written libraries provided with the Python language to simplify the code we have to write.

Errors and Documentation

- ♦ Type a % sign somewhere in the space after `print`. This is a *syntax error*: you are not allowed to type a % there in Python.
- ♦ Try to run the program with the error still in it. You should receive an invalid syntax error.

Writing your own Program

- ♦ In Jupyter, your project should save automatically. To add a new cell, simply click outside of the Jupyter cell (so that the cell turns blue) and press the letter 'B' on your keyboard. Alternatively, you can go to the 'Insert' menu and select 'Cell Below' to add a new cell.
- ♦ Use the Basic Shell Code from earlier for each of these tasks below.

Basic:

Task 1: Basic Input and Output

1. Create a new Python box in your Jupyter Notebook. Use the basic shell code and write your code within the main method.
2. Use the print statement to display your name on the screen.
3. Implement the following:
   - Use the input statement to allow the user to enter their name.
   - Display the user's name on the screen.
     - o Hint: Look at the slides
     - o Python f-String Tutorial – String Formatting in Python Explained with Code Examples (freecodecamp.org)

Task 2: Working with Strings

1. Create a new Python script.
2. Implement the following:
   - Prompt the user to enter a sentence.
   - Calculate and display the number of characters in the sentence.
     - o Hint: Python String Length (w3schools.com)
     - o Advanced task: Use a for loop to count the characters (come back to this)
   - Display the sentence in both uppercase and lowercase.
     - o Hint: Python String Methods (w3schools.com)

Intermediate:

Task 3: Working with Numbers

1. Create a new box on Jupyter.
2. Declare an integer variable user_height_cm and set it to 0
   a. What happens if you do not initialise a variable with a value?
3. Implement the following:
   - Prompt the user to enter their height in centimetres.
   - Input and convert the height from a string to an integer.
   - Hint: Python Casting (w3schools.com)
   - Prompt the user to enter the heights of two neighbours and store them in separate variables (neighbor1_height_cm and neighbor2_height_cm).
4. Calculate and display the average height of the user and their neighbours in centimetres.

Task 4: More Arithmetic Operations

1. Expand the previous script to include the following:
   - Calculate the average height in inches (1 inch = 2.54 cm).

- Use a floating-point variable (I use the term floating point to indicate decimal point – it means nothing in Python – just declare a variable) average_height_inches to store the result.
- Display the average height in inches with two decimal places.

Task 5: Converting to Feet and Inches (1 foot = 12 inches)

1. Enhance the script to:
   - Calculate and display the average height in feet and inches.
   - Format the output message nicely, e.g., "5 feet 10 inches."
2. Perform the following conversions:
   - Conversion from inches to feet and inches:
   - Calculate the feet: height_feet = average_height_inches // 12.
     - "//" disregards the fractional part of the calculation. This is important when we want to separate and display the feet and inches as whole numbers, ensuring accuracy in our results.
   - Calculate the remaining inches: remaining_inches = average_height_inches % 12

Advanced:

Task 6: User Input in Feet and Inches

1. Modify the script to accept user input for heights in feet and inches.
2. Implement the following:
   - Ask the user for their height in feet and inches separately.
   - Convert these 3 inputs to inches for averaging:
3. Calculate the average height in inches (for all three people) and then convert the result back to feet and inches for output.

Submission

1. Save your completed Jupyter Notebook or Python script.
2. Create a GitHub account if you don't already have one.
3. You will receive an invitation to join a GitHub class repository. Once you accept it, create a new repository for your project within that class repository.
4. Upload and share your file with your instructor in your newly created repository for evaluation.
5. Please complete these steps before our next session. If you haven't received a GitHub invitation by then, feel free to send me a message on Teams, and I will assist you in getting set up.