Week 0 worksheet

This week there is some recommended reading and some practical work that builds on what we covered in the first session.

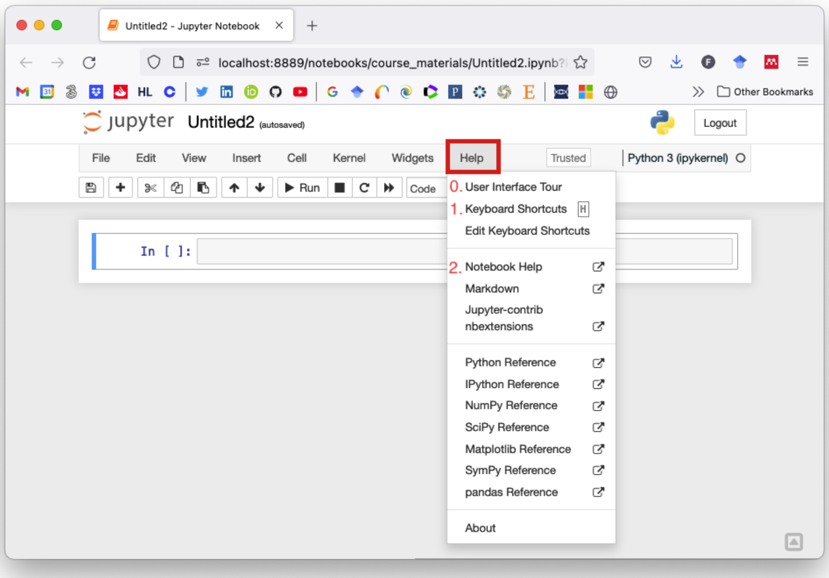
The article I would like you to read, and you can access it by clicking on the blue link, is called *How long does it take to learn Python?* This is an accessible read that explores reasons for learning Python, what it means to learn Python, how you can keep track of progress, factors that may influence your journey, how long it takes, and resources that may help speed your learning.

The rest of your efforts this week should be directed towards familiarising yourself with Jupyter and Spyder. Both of these software have a `Help` menu, which is a great place to begin. You are bound to encounter technical jargon and concepts that you don't understand, but don't worry! Just make a note, and remember to ask me about it when you get the chance.

By the way, don't worry if you think you have "broken" any of the documents provided in the course materials. Just download a fresh copy from Google Drive!

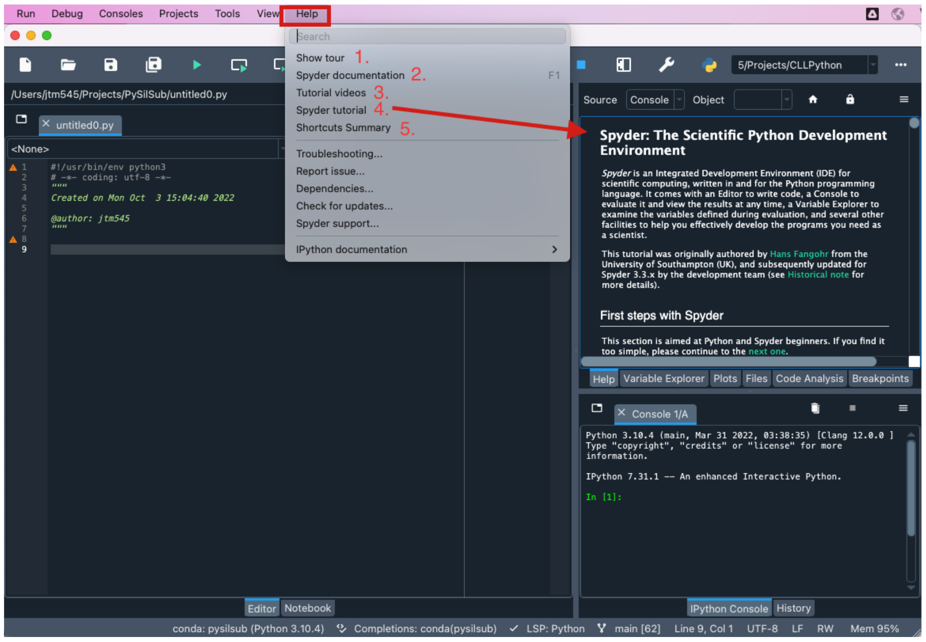
Jupyter

1. Take the `User Interface Tour`
2. Have a look at the `Keyboard Shortcuts`
3. Read the `Notebook Help` pages



Spyder

1. Take the Spyder tour
2. Review the Spyder documentation
3. Watch Parts 1, 2 and 3 of the "First steps with Spyder" tutorial videos
4. Skim through the Spyder tutorial (you can work through this at you own pace for extra practice).
5. Look at the Shortcut summary



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct  3 15:04:40 2022

@author: jtm545
"""

```

Spyder: The Scientific Python Development Environment

Spyder is an Integrated Development Environment (IDE) for scientific computing, written in and for the Python programming language. It comes with an Editor to write code, a Console to evaluate it and view the results at any time, a Variable Explorer to examine the variables defined during evaluation, and several other facilities to help you effectively develop the programs you need as a scientist.

This tutorial was originally authored by Hans Fangohr from the University of Southampton (UK), and subsequently updated for Spyder 3.3.x by the development team (see Historical note for more details).

First steps with Spyder

This section is aimed at Python and Spyder beginners. If you find it too simple, please continue to the next one.

```
Python 3.10.4 (main, Mar 31 2022, 03:38:35) [Clang 12.0.0 ]
Type "copyright", "credits" or "license" for more
information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]:
```

Exercises

This week there are two short exercises for you have a go at.

Exercise 1

Let's have another look at the number guessing game script. With an instruction like **"Enter any number: "**, a naive user could be forgiven for entering something like 78.3, which is, after all, a number.

But if a decimal number is entered, the program throws a cryptic error message. See for yourself by clicking on the cell below, pressing `Shift+Enter` to execute the code, and then entering a decimal number instead of a whole number. Don't worry about the error message for now. Suffice to say that the computer is unable to carry out the instruction!

It would be better if the user knew not to enter a decimal number. **Change the code below so that when it is executed the instruction will read**

"Enter a whole number between 0 and 100: "

Hint - look at the first `input` function

If you found that easy, try changing the message at the end of the program to make it more congratulatory!

```
"""Play a number guessing game."""

import random


# Generate a random number between 0 and 100
n = random.randrange(1, 100)

# Ask the user to have a guess
guess = int(input("Enter any number: "))

# Keep asking until the guess is right
while n != guess:
    if guess < n:
        print("Too low")
        guess = int(input("Enter number again: "))
    elif guess > n:
        print("Too high!")
        guess = int(input("Enter number again: "))
    else:
        break

# Tell user they guessed right
print("you guessed it right!!")
```

```
Enter any number:  78.3
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [2], in <cell line: 10>()
      7 n = random.randrange(1, 100)
      9 # Ask the user to have a guess
---> 10 guess = int(input("Enter any number: "))
     12 # Keep asking until the guess is right
     13 while n != guess:

ValueError: invalid literal for int() with base 10: '78.3'
```

Exercise 2

Remember the Fahrenheit to Celsius program?

Depending on the Fahrenheit value that the user provides, the reported temperature in degrees Celsius may be given with much more precision than we need. For example, if I enter 75.5 as a temperature in Fahrenheit, the result in Celisus is reported as 24.166666666666668. We don't need that much precision!

Can you figure out a way to round the Celsius value to two decimal places? We haven't learned how to do this yet, so it may prove a tricky one. You may wish to try Googling "How to round numbers in Python".

Good luck!

```python
"""Convert Fahrenheit to Celsius."""

# Ask user to enter a temperature in Fahrenheit
temp = float(input("Enter temperature in Fahrenheit: "))

# Convert to celsius
celsius = (temp - 32) * (5 / 9)

print(f"{temp} degrees Fahrenheit is equal to {celsius} degrees Celsius.")
```

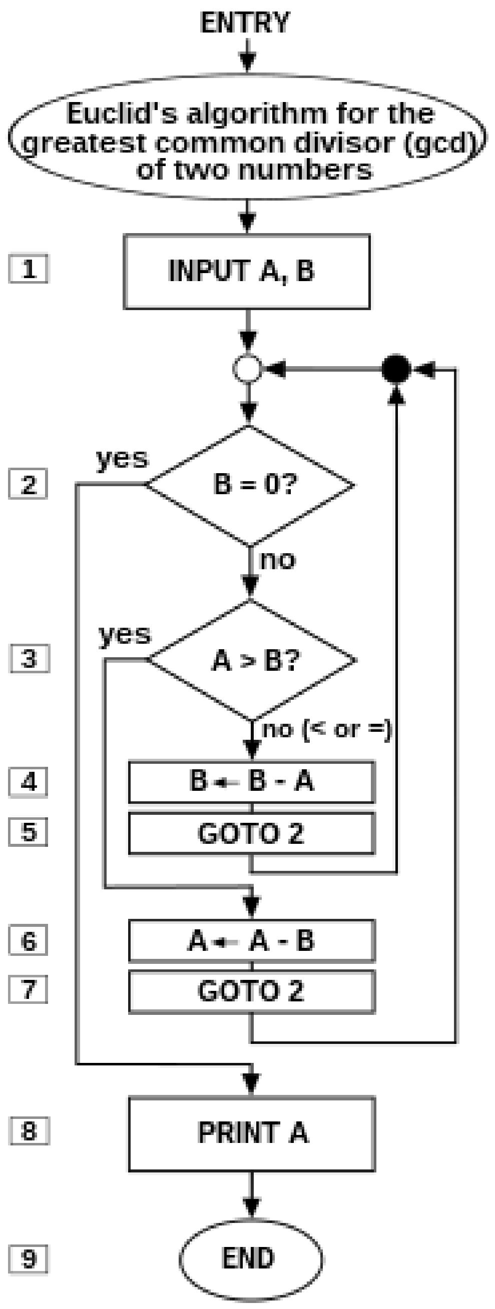
```
Enter temperature in Fahrenheit: 75.5
75.5 degrees Fahrenheit is equal to 24.166666666666668 degrees Celsius.
```

Exercise 3

Have another look at Euclid's algorithm (from this week's presentation slides), which is used to find the highest common divisor of two numbers (i.e., the highest number that divides them both without a remainder).

For each of the following starting points, mentally assign the numbers to `A` and `B` and then work through the stages of the algorithm to find the highest common divisor. You can use a pen and paper if it makes things easier.

1. `A = 4`, `B = 2`
2. `A = 8`, `B = 16`
3. `A = 49`, `B = 35`
4. `A = 240`, `B = 30`



Note: The **>** symbol means greater than, and the ← symbol means "assign the result of the expression on the right to the variable on the left"