

COM738 – Data Science Foundations – Week 1 Practical

This practical will give you the experience with the setup and usage of the Python development environment, and in simple problem solving using Python by applying the techniques covered in this week's lecture. It is recommended that you complete each of these activities in a Jupyter Notebook, and that you create and save a new Jupyter Notebook for each week, in case you would like to refer to a previous exercise.

Please let a member of the delivery staff know if you have had difficulty with, or questions about, any of these tasks.

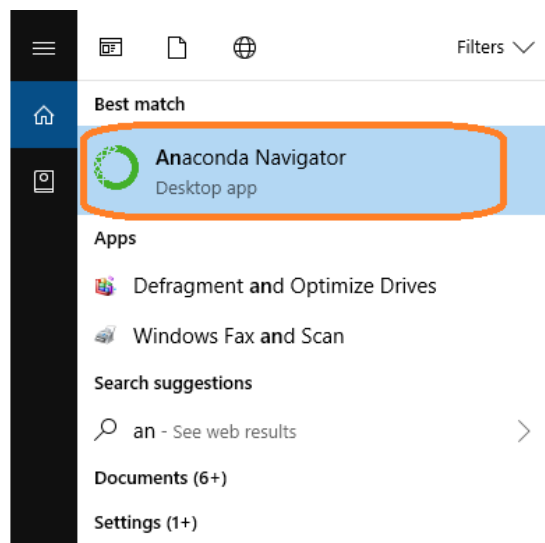
Part 1 – Setup and Usage of the Development Environment

The main purpose of Part 1 is to provide you with experience in the setup and usage of your Python development environment. You will need to be able to perform the steps outlined in this section in order to complete future practical classes.

1. Anaconda

Anaconda is a free Python package manager which can be used to install and update Python, specific Python Packages, and various dependencies required. This has already been installed for you in the labs.

1.1. Launch Anaconda



- Anaconda Navigator will now load.

1.2. Explore Anaconda Navigator

- Anaconda Navigator provides many useful resources to help you manage your Python environment. It also provides access to high quality learning and community resources which you can use to help you learn about Python and resolve any issues you might be having.
- **The module lecturers and lab support staff are always available to help you if you have any difficulties, however, you are encouraged to demonstrate independence and initiative by**

using these freely available resources as a first step to try to resolve issues yourself if possible.

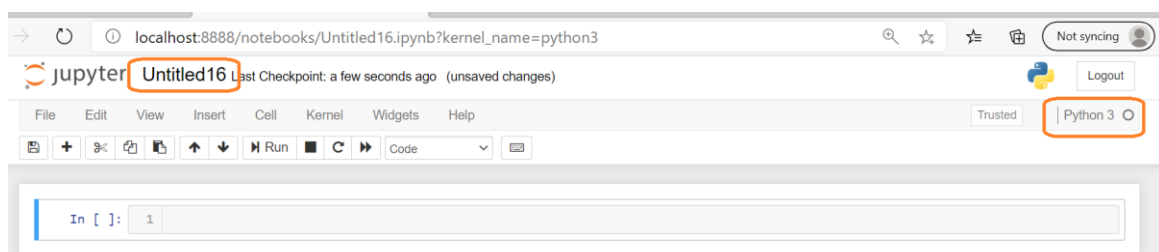
- Take 20 minutes to explore the following areas:
 - The Environments Manager – look at the currently installed, and not installed packages. Check if the “pyreadline” package is installed. If it isn’t please install it.
 - Learning – Access the Python Tutorial and the Python Reference, take a few minutes to look at the various topics covered. Any tasks asked of you in the first few weeks should be covered within these documents. Note that there is also documentation for other libraries called Pandas, Numpy, Scipy, Matplotlib, and Bokeh. These will all be covered in future weeks.
 - Community – Access Stack Overflow. Stack Overflow is a community-driven website which aims to help programmes find answers to problems. You can typically infer the quality of a question and response by the number of positive votes given by the community. When searching for answers, make sure to keep the [python] string in the search bar, otherwise you will get results for other languages. Take a few minutes to use Stack Overflow to see if you can find good quality answers to the following:
 - How to do you clone or copy a list?
 - What does the yield keyword do?

1.3. Use Jupyter Notebook

- Launch the Jupyter Notebook from Anaconda Navigator.
- Once launched, you will see a dashboard screen. From here you can view the current working directory, currently running notebooks, and create new notebooks.
- Create a new notebook by selecting New -> Notebook (Python 3)

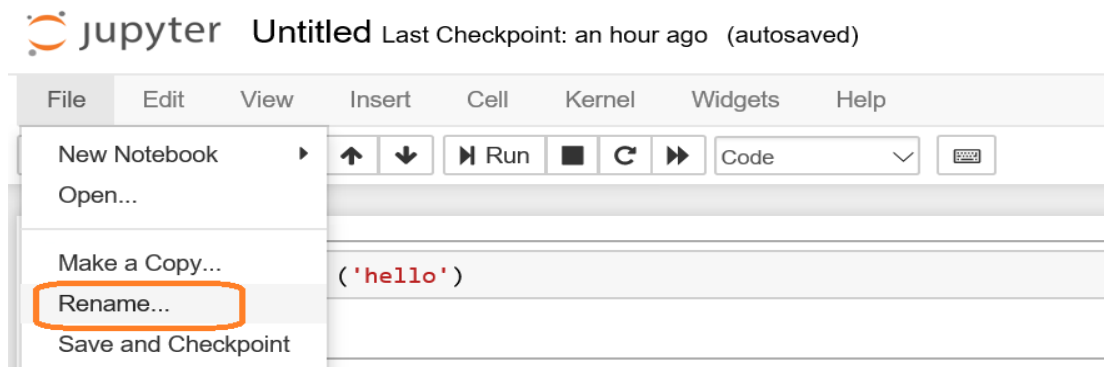


- Once launched, you will see a notebook with one empty cell, as shown below:



- You can toggle line numbers by selecting View -> Toggle Line Numbers

- Press **s** to save the notebook
- The notebook is saved as Untitled
- To give a meaningful name to your saved file, click as below:



1.3.1. Overview of the Notebook UI

- If you create a new notebook or open an existing one, you will be taken to the notebook user interface (UI). This UI allows you to run code and author notebook documents interactively. The notebook UI has the following main areas:
 - o Menu
 - o Toolbar
 - o Notebook area and cells

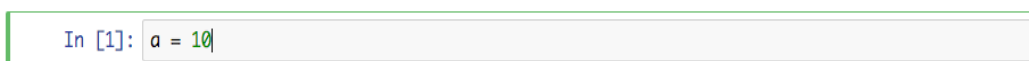
The notebook has an interactive tour of these elements that can be started in the “Help: User Interface Tour” menu item.

Modal editor

- Starting with IPython 3.0, the Jupyter Notebook has a modal user interface. This means that the keyboard does different things depending on which mode the Notebook is in. There are two modes: *edit mode* and *command mode*.

Edit mode

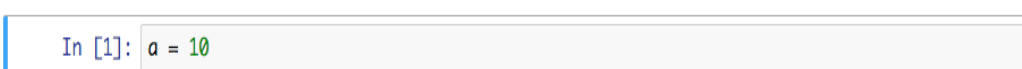
Edit mode is indicated by a green cell border and a prompt showing in the editor area:



When a cell is in edit mode, you can type into the cell, like a normal text editor. Enter edit mode by pressing Enter or using the mouse to click on a cell’s editor area.

Command mode

Command mode is indicated by a grey cell border with a blue left margin:



Jupyter cell with blue & grey border

You can enter “Command Mode” by pressing Esc

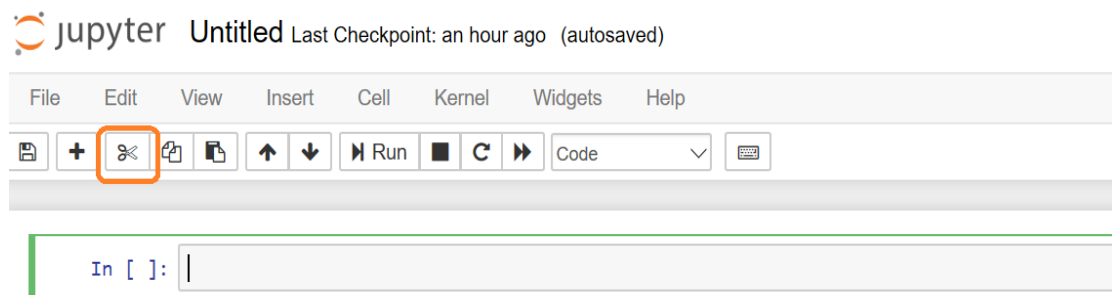
Command mode allows you to enter keyboard shortcut commands for creating new cells, deleting cells, running cells etc. Select the keyboard icon to get a full list of commands. Useful shortcuts include:



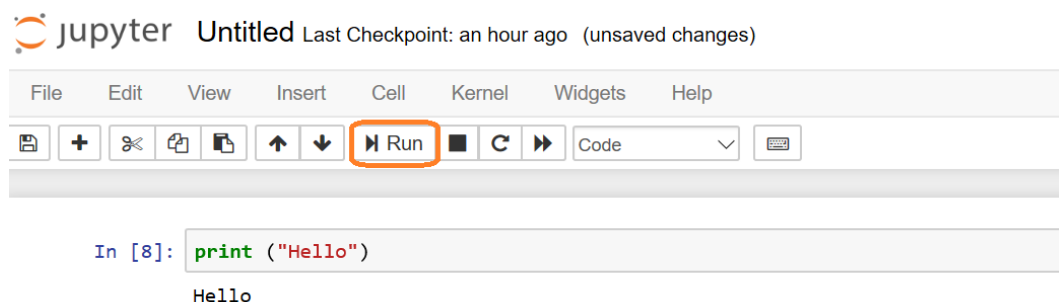
- Ctrl + Enter Run the current cell
- Command Mode + a Insert a new cell after the current cell
- Command Mode + b Insert a new cell before the current cell
- Command Mode + dd Delete the current cell

- Try these shortcuts to ensure they work as you would expect
- It is advisable to logically divide functionality into separate cells. For example, place all your import statements in the first cell, all the content of a new function, or related functions, in another cell. This will be more relevant in future weeks when you make more complex programs.

- Deleting a cell



- Running the code:



- Experiment with using the syntax covered within the lecture. Try doing each of the following:
 - Write a comment line
 - Create a new integer variable
 - Create a new string
 - Output the contents of a string
 - Create a for loop that will output the numbers 1 to 10
 - Create a for loop that will iterate through and output each item of a list you have created

- Finally, check that tab completion works as you would expect. Firstly, create a new list with some objects. Then, type the name of your list, followed by a dot '.', then press Tab. You should see the following:

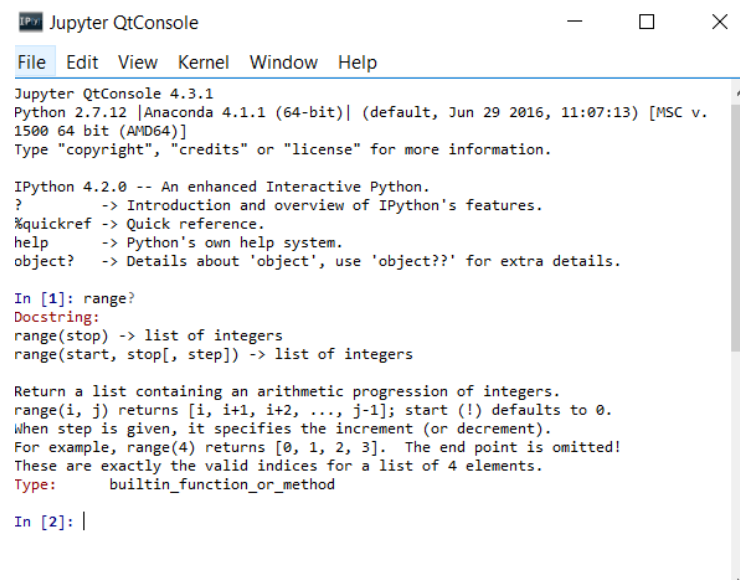
```
In [38]: x = [3,2,1]

In [ ]: x.|
x.append
x.clear
x.copy
x.count
x.extend
x.index
x.insert
x.pop
x.remove
x.reverse
```

- **Note:** If a dropdown of options does not appear, or the dropdown is different to that shown above, there are two potential reasons:
 - o The variable assignment must be run by the kernel, otherwise it does not yet exist in memory. Run the cell with the variable assignment, and then try again.
 - o If this still does not work, ensure that the “pyreadline” package is installed via Anaconda Navigator. If you are installing it now, then the notebook must be restarted before the changes will take effect.

1.4. Use QtConsole

- Open QtConsole through Anaconda Navigator
- Type 'range?' (without quotes) to get some basic information about the range function.



```
Jupyter QtConsole
File Edit View Kernel Window Help
Jupyter QtConsole 4.3.1
Python 2.7.12 |Anaconda 4.1.1 (64-bit)| (default, Jun 29 2016, 11:07:13) [MSC v.
1500 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.

In [1]: range?
Docstring:
range(stop) -> list of integers
range(start, stop[, step]) -> list of integers

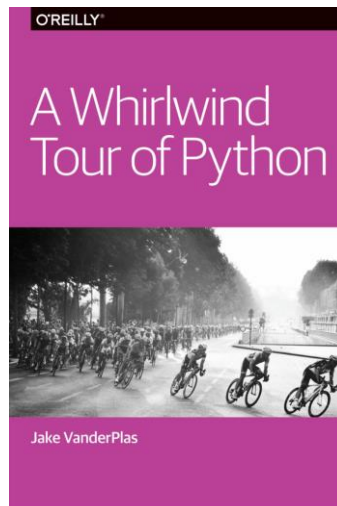
Return a list containing an arithmetic progression of integers.
range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0.
When step is given, it specifies the increment (or decrement).
For example, range(4) returns [0, 1, 2, 3]. The end point is omitted!
These are exactly the valid indices for a list of 4 elements.
Type:      builtin_function_or_method

In [2]: |
```

Note: if you ever lose your cursor in the QtConsole when accessing help, press 'q' to exit help and regain control.

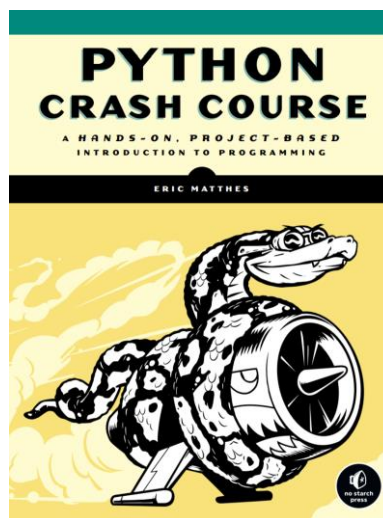
2. Access and Download Free Python eBooks

In addition to online communities, user guides and references, there is the availability of a wide range of high quality, free eBooks which cover all aspects of Python and Data Science. These range from introductions for complete beginners, to in-depth guidance on specialist areas. It is highly recommended that you download these today and bring them with you to your practical classes and tutorials. They are also fantastic resources for your independent learning.



This book provides a very useful light introduction to the key concepts of Python. It is a great starting point. Accessible from:

<http://www.oreilly.com/programming/free/files/a-whirlwind-tour-of-python.pdf>



This book provides a more detailed overview of the key concepts of Python. Accessible from:

<https://github.com/MrAlex6204/Books/blob/master/python-crash-course.pdf>



This book expands beyond the core concepts of Python, by offering a detailed overview of data science techniques including data manipulation, visualisation, and machine learning. This will be very useful for this module and the other modules within the course. Accessible from: <https://github.com/jakevdp/PythonDataScienceHandbook> (Unfortunately, no PDF version exists)

Part 2 – Problem Solving

This section will give you experience in the application of the various techniques you have heard about in today's lecture.

Operators

1. Use Python's operators to obtain the correct result for each of the following:

Operation	Result	
Addition of 2 and 3	5	
Subtraction of 3 from 2	-1	
Multiplication of 3 by 2	6	
Division of 3 by 2	1.5	3/2
Floor division of 3 by 2	1	3//2
Modulus of 3 by 2	1	
Exponentiation of 3 by 2	9	3**2
6 ²	36	pow(6,2)
2 + (5 x 6) ²	902	2+pow((5*6),2)
6 is less than 7	True	6<7
10 is equal to 13	False	10==13
0.5 is less than or equal to 20	True	0.5<=20
20 is greater than or equal to 5	True	
4 is less than 5 AND 4 is greater than 6	False	4<5&4>6
4 is less than 5 OR 4 is greater than or equal to 6	True	

Strings

2. Create a string with the contents "this is a test string", then:
 - 2.1. Calculate the length of the string `len(sentence)`
 - 2.2. Capitalise the first letter of the string `sentence.capitalize()`
 - 2.3. Make the whole string upper case `sentence.upper()`
 - 2.4. Access the 3rd character of the string (i) – remember that we are using a zero-based index.
`sentence[2]`
 - 2.5. Using negative indexing, access the second last character of the string (n).
`sentence[-2]`

Lists

3. Create a list with the strings "Red", "Green", and "Blue", then:


```
colours=["Red", "Green", "Blue"]
```

 - 3.1. Find out if "Purple" is in the list.
`"Purple" in colours`
 - 3.2. Use **Negative Indexing** to access "Blue"
`colours[-1]`
 - 3.3. Sort the list in ascending order (based on the first character of each string)
`colours.sort()`
 - 3.4. Sort the list in descending order (based on the first character of each string)

```
colours.sort(reverse=True)
```


3.5. Sort the list from shortest string to longest.

```
colours.sort(key=len)
```

3.6. Access the first element in the list

```
colours[0]
```

3.7. Access every element in the list except the last

```
colours[0:-1]
```

3.8. Add the colour “Yellow” to the end of list

```
colours.append("Yellow")
```

3.9. Add the colour “Purple” to the second position in the list – make sure you are not replacing the existing second colour. The second colour should now become the third, etc.

```
colours.insert(1,"Purple")
```

Dictionaries

4. Create a dictionary called ‘animalSounds’. This dictionary should contain animal names as keys, and corresponding animal sounds as values. Insert 3 key-value pairs.

4.1. Output the sound one animal makes

```
animalSounds={'cow':'moo','cat':'meow','dog':'bark'}  
animalSounds['cow']
```

4.2. Add a new key-value pair to the dictionary.

```
animalSounds['mouse'] = 'squeak'
```

Sets

5. Create two sets manually. One containing odd numbers from 1 to 19, the other containing all numbers from 1 to 10, then:

5.1. Output one set which is a union of the two sets.

```
oddNumbers.union(allNumbers)
```

5.2. Output only the items that appear in **both** sets.

```
oddNumbers.intersection(allNumbers)
```

5.3. Output only the items that appear in the odd number set, and not the all number set.

```
oddNumbers.difference(allNumbers)
```

5.4. Output only the items that appear in either one set or the other, not both.

```
oddNumbers.symmetric_difference(allNumbers)
```

Control Statements

6. Write a For Loop that will iterate through a list which contains several colours as strings (all lowercase). The loop should capitalise the first letter each string, and output each colour with the capitalised first letter.

```
colours = ["red","green","blue"]
```

```
for colour in colours:  
    print(colour.capitalize())
```

7. Write a series of statements that will remove any duplicates from the following list, and return them in order: 1,1,1,1,5,4,3,6,7,3,2,1. Note: use a For Loop for this.

```
numbers = [1,1,1,1,5,4,3,6,7,3,2,1]
temp = []
for num in numbers:
    if num not in temp:
        temp.append(num)
temp.sort()
numbers = temp
numbers
```

8. Take the sentence: sentence = "The quick brown fox jumps over the lazy dog"

8.1. Write one statement which will reverse the order of all characters in the sentence as follows: 'god yzal eht revo spmuj xof nworb kciuq ehT'

```
sentence[::-1]
```

8.2. Write a series of statements that will reverse the order of the words. Hint: use the string.split() function to separate the sentence into words.

```
sentence = "The quick brown fox jumps over the lazy dog"
words = sentence.split(' ')
words[::-1]
```

8.3. Write a series of statements that will take all of the characters in the original sentence, remove any duplicates, and return them in descending (Z -> A) order.

```
sentence = "The quick brown fox jumps over the lazy dog"
temp = []
for c in sentence:
    if c not in temp:
        temp.append(c.lower())
temp.sort(reverse=True)
temp
```

8.4. Finally, take the list of characters you generated, and output another list with all vowels capitalised. Hint: you may wish to use the "enumerate" function on your list.

```
sentence = "The quick brown fox jumps over the lazy dog"
temp = []
for c in sentence:
    if c not in temp:
        temp.append(c.lower())
temp.sort(reverse=True)

vowels = ['a','e','i','o','u']
for index, c in enumerate(temp):
    if c.islower() and c in vowels:
        temp[index] = c.upper()
temp
```

9. The collection "sounds" contains some errors. Animal names have accidentally been inserted in some instances, instead of the sound they make. Iterate through the sounds collection and use the animal_sounds collection to automatically replace the animal names in the sounds list with the correct sound.

```
animals_sounds =  
{ "horse": "neigh", "pig": "oink", "dog": "bark", "cow": "moo", "duck": "quack",  
  "cat": "meow" }  
sounds = ["meow", "moo", "bark", "pig", "quack", "horse"]
```

```
for i, sound in enumerate(sounds):  
    if sound in animals_sounds.keys():  
        sounds[i] = animals_sounds[sound]  
sounds
```

User Input

10. A topic we did not cover this week is the process of accepting user input, such as a string or integer through a text box. Try to find online the correct approach to accepting user input, and attempt to mimic the output shown in the following screenshot. Text entered by the user has been highlighted in **bold** to make the example clearer:

```
What is your name?  
Jonathan  
Hello Jonathan!  
Please create a list of values by entering a value and pressing  
enter. When you have added all of the values you wish to enter,  
enter '#STOP'  
  
Value:  
cow  
Value:  
cat  
Value:  
dog  
Value:  
#STOP  
The complete list of values entered is: ['cow', 'cat', 'dog']  
What would you like to do? Options are: 'a' to add, press 'd' to  
delete, or press 'r' to retrieve. Enter any other character to exit.  
a  
What value would you like to add?  
mouse  
What would you like to do? Options are: 'a' to add, press 'd' to  
delete, or press 'r' to retrieve. Enter any other character to exit.  
d  
What value would you like to remove?  
cow  
What would you like to do? Options are: 'a' to add, press 'd' to  
delete, or press 'r' to retrieve. Enter any other character to exit.  
r  
What index would you like to retrieve?  
0  
cat
```

What would you like to do? Options are: 'a' to add, press 'd' to delete, or press 'r' to retrieve. Enter any other character to exit.
exit

```
name = input("What is your name?\n")

print("Hello {0}! \nPlease create a list of values by entering a
value and pressing enter. When you have added all of the values you
wish to enter, enter '#STOP'\n".format(name))

values = []
while True:
    value = input("Value:\n")
    if value == '#STOP':
        break
    else:
        values.append(value)

print("The complete list of values entered is:", values)

while True:
    action = input("What would you like to do? Options are: 'a' to
add, press 'd' to delete, or press 'r' to retrieve. Enter any other
character to exit.\n")
    if action == "a":
        value = input("What value would you like to add?\n")
        values.append(value)
    elif action == "d":
        value = input("What value would you like to remove?\n")
        values.remove(value)
    elif action == "r":
        value = int(input("What index would you like to
retrieve?\n"))
        print(values[value])
    else:
        break
```

Preparing for Next Week

Finally, it is highly recommended that if you have time left over in the practical, you should take time to read through the theory and practice each of the concepts covered in today's lecture.