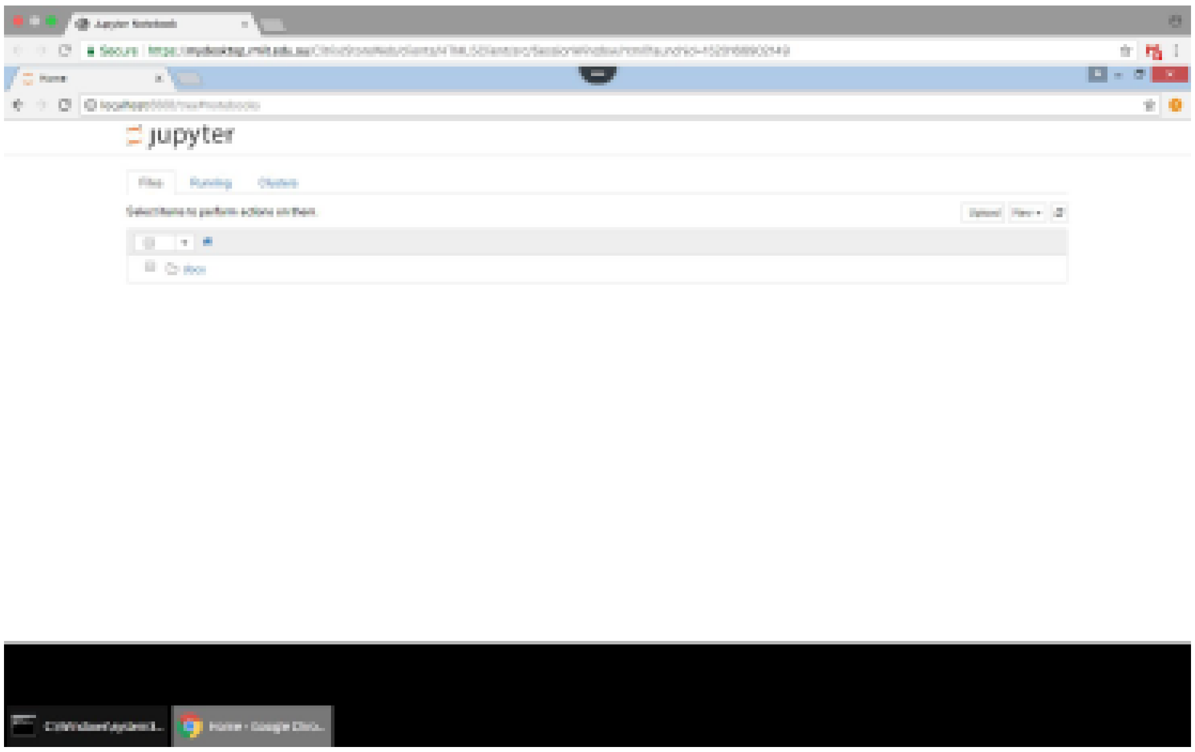
# Practical exercise 1: Accessing Jupyter Notebook

In this course, we will be making extensive use of the iPython interactive environment in Jupyter Notebook. You can access it via:

- `Lab Computer`

  `Start -> All Programs -> Anaconda2 (64-bit) -> Jupyter Notebook`

`Then, you will see the following (or a similar) screenshot:`



Then,

- Select New -> Python 2
- The new created ‘*.ipynd’ is created at the following location:
    - C:\Users\sXXXXXXX
    - where sXXXXXXX should be replaced with a string consisting of the letter "s" followed by your student number.

More materials for Jupyter Notebook:

- A tutorial:
    - Jupyter Notebook Tutorial: The Definitive Guide (https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook)

More materials for Anaconda:

- Install Install Anaconda on your PC:
    - https://www.anaconda.com/download
    - (please select Python version at least 3.7)

More materials for Python:

- https://docs.python.org/2/tutorial/

**If you are new to Python, it is suggested that you go through some Python tutorials**, e.g.

- https://www.codecademy.com/tracks/python
- (Need to register)

# Practical Exercise 2: A First iPython Session

Often, when writing a python program, a set of statements are written into a single text file, and then executed together to carry out some purpose. For data exploration, it is often convenient to work in an interactive environment. iPython offers such an environment for the python language.

First, let's create a directory for practical data science, and this tutorial.

After you log in to Lab computer, you should start in your home directory. You can create a new folder named “`pds`” for Practical Data Science.

Then create a new directory under it, called "tute01", and save your *.ipynb files (from Jupyter Notebook) here for this tutorial.

Now enter the following commands

```
In [1]: import matplotlib.pyplot as plt

In [2]: x = range(100)

In [3]: y = [val**2 for val in x]

In [4]: plt.plot(x,y)
Out[4]: [<matplotlib.lines.Line2D at 0x7f8e6008ced0>]

In [5]: plt.show()
```

Examine the plot window.

iPython prompts are numbered, and each time you enter a command, this number will increase by 1. This makes it easy to repeat commands without typing them again.

iPython includes "magic" commands, which start with a % sign.

```
In [6]: %history -n
```

will print the **history** of commands that you have entered in the session, the -n flag indicates that the corresponding line numbers should also be shown. You can now **re-run a command**, for example

```
In [7]: %rerun 4
```

will cause item number 4 from your session history to be executed again. You can also **scroll through your input history** using the up and down arrow keys.

Input is also simplified through **tab-completion**. For example type

```
In [8]: name = "Noctis Lucis Caelum"
In [9]: job = "Crown Prince of Lucis"
In [10]: na<tab>
```

Notice that the interpreter automagically completes the string "na" to "name", since this is a previously defined variable with the same prefix. Tab completion is also handy for scrolling through available methods. Try:

```
In [11]: plt.<tab>
```

You will be offered a list of all available methods for the matplotlib object with you imported in line 1.

While it's often convenient to use the iPython shell interactively, you might also want to **run code from a file**. You can create a text file called *my_first_script.py.*

Type the following content into the file:

```
my_string = "hello, world"
print(my_string)
print(my_string[0:5])
print(my_string[7:])
```

Now, in your ipython session, type

```
In [12]: %run my_first_script.py
```

(In fact, you only need to type %run my_<tab> and iPython will complete the name for you!)

Interactive help is available in iPython through the ? and ?? operators. This can display information on functions and defined objects. For example

```
In [13]: ?range
```

will display information about the range() function that we used earlier, while

```
In [14]: ?x
```

shows information about the variable x that was previously defined. Using ?? instead of ? can display extra information for some objects.

You can learn more about iPython from the official documentation. There are also many tutorials available online.