This is a walk-through of some of the basic python packages and jupyter functioning

Follow this tutorial step-by-step before completing other labs, particularly if you have no prior python or jupyter experience.

Using Jupyter notebook

First, let's understand how this notebook works. Notebooks are written by cells that can be evaluated individually. In the cell below, the text "This is a Markdown cell" has been typed. Double click on the words to view the cell.

This is a Markdown cell

Now try to create your own cell. First, click once on this cell, then add a new markdown cell by clicking the "+" button above. This should create a new cell beneath this one. Before we can type markdown text, you need to click on the dropdown menu above that says "Code" and instead choose "Markdown". Type whatever text you want! Then either click the "Run" button above or type Shift+Return to evaluate the cell. Once you have done so, the text should look just like it does here!

Now let's create text a bit larger by typing out "#" in front of the text. In the 3 cells below, double click on the words to see how the larger fontsize is created.

Largest header

Slightly smaller header

Smaller header

These markdown cells are useful for writing comments in your Jupyter notebook - a very important part of writing reproducible code!

Notice the buttons near the top of this window - these are very useful!

- To add new cells, click the "+" button above.
- To move cells up and down, use the up and down arrow buttons.
- To switch cells between markdown (for writing text) and code (for writing python code), select the appropriate word from the menu that says "Markdown".
- Hover over the other buttons to see what they do!

On to some python coding!

We can use basic python to, e.g., do some simple math! Comments (i.e. text that is not evaluated as python code) are denoted by a "#". Evaluate the examples in the cells below.

In [1]:
```
5+7 # addition
```

Out[1]:
```
12
```

In [2]:
```
7**2 # squaring numbers
```

Out[2]:
```
49
```

In [3]:
```
(4+5)/3 # mathematical expressions
```

Out[3]:
```
3.0
```

In [4]:
```
print('this will print out text from a coding cell')
```

```
this will print out text from a coding cell
```

In [5]:
```
# Assigning variables
x = 3 + 7
print('The variable x has the value',x)
```

```
The variable x has the value 10
```

You can see that the answers or outputs are printed below each cell. To do more interesting things, we need to import libraries. Let's start by importing one called numpy, and let's denote it "np".

In [6]:
```
import numpy as np
```

The package numpy allows us to do some basic array manipulation and more complicated math. Evaluate the following cells for some examples.

In [7]:
```
# Calculate sine and cosine, and use the number pi
np.sin(np.pi)
```

Out[7]:
```
1.2246467991473532e-16
```

In [8]:
```
# Assign arrays, or matrices, of numbers
a = np.array((1,3,6,11))
b = np.arange(10)
print(a,b)
```

```
[ 1  3  6 11] [0 1 2 3 4 5 6 7 8 9]
```

There are many more useful functions in the numpy library that you will likely see in other python labs this week.

Let's move on to another library. This one is for plotting! Note that we need to include the first line "%matplotlib inline" in order to view the plots in this notebook.

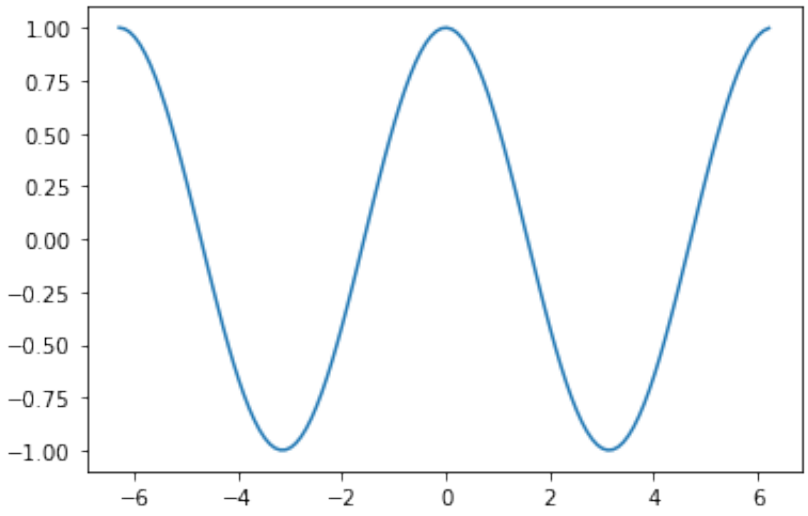
In [9]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

Let's make a simple plot.

In [10]:
```
# Define an array x
x = np.arange(-2*np.pi, 2*np.pi, 0.1) # this defines an array that spa
ns numbers from -2pi to 2pi with an interval of 0.1
y = np.cos(x)

plt.plot(x,y)
```

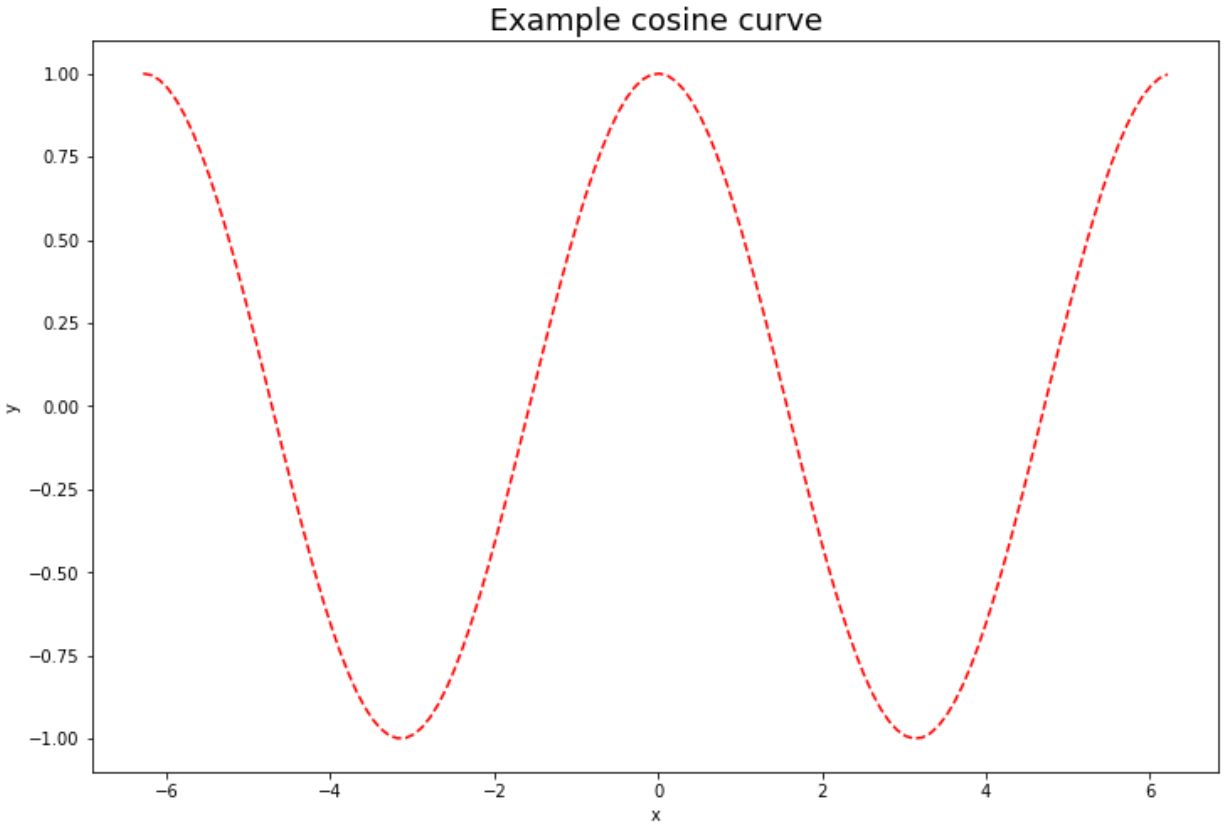
Out[10]:
```
[<matplotlib.lines.Line2D at 0x11e7d6da0>]
```



It's always good practice to label the axes and give your plot a title. Let's also make the plot a bit bigger and change the color and linestyle, just for fun.

In [11]:
```
plt.figure(figsize=(12,8))
plt.plot(x,y,color='r',linestyle='dashed')
plt.title('Example cosine curve',fontsize=18)
plt.xlabel('x')
plt.ylabel('y')
```

Out[11]: Text(0, 0.5, 'y')



Let's get to know some other useful libraries

Pandas

Another useful library is called pandas. Pandas is a great library for reading in and analyzing txt and csv files. It is a very convenient way to store data in columns, and each data column can be in very different formats (e.g. numbers, dates, text, etc.). Data is typically loaded in as a "dataframe". Let's look at a simple example.

In [12]:
```
import pandas as pd  # load the pandas library as "pd"

# Create an example dataframe called df (taken from the pandas usergui
de at pandas.pydata.org)
df = pd.DataFrame({'A': 1.,
                   'B': pd.Timestamp('20130102'),
                   'C': pd.Series(1, index=list(range(4)), dtype='floa
t32'),
                   'D': np.array([3] * 4, dtype='int32'),
                   'E': pd.Categorical(["test", "train", "test", "trai
n"]),
                   'F': 'foo'})

df # print out the dataframe
```

Out[12]:

	A	B	C	D	E	F
0	1.0	2013-01-02	1.0	3	test	foo
1	1.0	2013-01-02	1.0	3	train	foo
2	1.0	2013-01-02	1.0	3	test	foo
3	1.0	2013-01-02	1.0	3	train	foo

See how nicely the data is formatted as a pandas dataframe! And you can see that some columns are floats (decimal numbers), integers, dates, and text. If you're not sure what kind of format your data has:

In [13]:
```
df.dtypes
```

Out[13]:
```
A           float64
B    datetime64[ns]
C           float32
D             int32
E          category
F            object
dtype: object
```

Now you can see the format of each column in the dataframe. To select a single column of data:

In [14]:
```
df['E']
```

Out[14]:
```
0     test
1    train
2     test
3    train
Name: E, dtype: category
Categories (2, object): [test, train]
```

There are a LOT more useful functions in pandas, and you will use this library in other labs at this school.

cartopy

The cartopy library is a way to make nice map visualizations! If you have attended previous COESSING schools, you may remember the library called basemap. While basemap still exists, it is no longer being updated and so cartopy is the map plotting package of choice these days. There is an excellent cartopy tutorial made by the amazing Josué Martinez-Moreno, so we won't take time to go over it here.

netCDF4

netCDF4 is a nice package to load netcdf (.nc) files. If you look at the python lab from COESSING 2019 (last year!) you can see examples of using this package.

xarray

The xarray package is very useful particularly when dealing with large amounts of data, e.g. from ocean/climate models. It also streamlines many useful features, and allows you to take averages of data or plot data all in one line. It's pretty nifty!

A few tips:

- Don't be shy to play around with python and Jupyter! If you aren't sure if you did something correctly, the computer will often tell if you did! So try writing some code, try writing some nice text cells - have fun!
- Error messages are (usually) your friends! Don't be upset if you see an error message. Usually they are quite useful. Read the error message and see if you can fix it yourself! Googling error messages is also a great way to find answers on the internet.
- Especially when typing out code, make sure your spelling is exact! The computer can only interpret exact python code, and so (I find) most of my errors are because of a spelling typo. (Which is great - they are easy to fix!) :)

Finally:

It's time to put your knowledge to the test by completing other python notebooks and labs that are part of this school! There are several other specific python-learning notebooks, as well as several labs on other topics that are in python! So you have plenty of chances to practice python this week.

If you have any python-related questions, post them on the Slack channel python-help! This channel will be monitored mostly in the morning in Ghanaian time (since the python instructors are in Australia!)

Of course, you can always Google your questions! Nearly every question about python has been asked before and likely has an answer on the internet. :)

In [ ]: