Welcome to DATA2001: Data Science: Big Data and Data Diversity

Welcome to the first tutorial of DATA2001. We expect that all the students have learned at least one programming language before, preferably Python. However we understand that some students might lack the skills in Python and/or other programming languages. Therefore we would like to request you to complete “Programming Experience Survey” available in the Canvas. The survey will guide us to customise the contents of this course.

We will be using Python and Jupyter for the majority of the lectures/tutorials, therefore, the first tutorial introduces you the basics of Python and Jupyter.

You will have access to the following Jupyter servers.

```
https://soit-ucpu-pro-1.ucc.usyd.edu.au
https://soit-ucpu-pro-2.ucc.usyd.edu.au
https://soit-ucpu-pro-3.ucc.usyd.edu.au
https://ucpu1.ug.it.usyd.edu.au
```

EXERCISE 1: Python Refresher

Data structures

Python list, dictionary, tuple and set are the most used data structures in data science applications.

In [1]:

```
list = [1, 2, 3]
dictionary = {1: "one", 2: "two", 3: "three"}
tuple = (1, 2, 3)
set = {1, 2, 3}

print("list:", list)
print("dictionary:", dictionary)
print("tuple:", tuple)
print("set:", set)
```

```
list: [1, 2, 3]
dictionary: {1: 'one', 2: 'two', 3: 'three'}
tuple: (1, 2, 3)
set: {1, 2, 3}
```

Loops and conditions in Python

There are two types of loops in Python, for and while. The code within a loop gets executed as long as the condition is true.

In [2]:

```python
for i in range(3):
    print(i)

# To do: produce the same output using while loop
```

```
0
1
2
```

In [3]:

```python
# Conditions if and else are employed to nest conditions
x = 5;
if x%2==0 :
    print ("This is an even number.")
else :
    print ("This is an odd number.")
```

```
This is an odd number.
```

Functions

Python code could be made modular as block of reuseable code, called function. A function can accept parameters and can return values.

In [4]:

```python
def addTwoNumbers (x, y):
    return x+y
z = addTwoNumbers(5,6)

print("Addition function output: ", z)

# An example of a recursive function
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)

print("Recursive function output: ", factorial(4))
```

```
Addition function output:  11
Recursive function output:  24
```

List comprehension

List comprehension is a compact code form for quick and easy population of a new list by processing an existing list.

In [5]:

```
x = [1,2,3,4,5,6,7,8,9,10]

y = [ i**2 for i in x]

print("squared list of x: ",y)

# conditions could be included in list comprehension
number_list = [i ** 2 for i in x if i % 2 == 0]
print("Square of only even numbers: ", number_list)


```

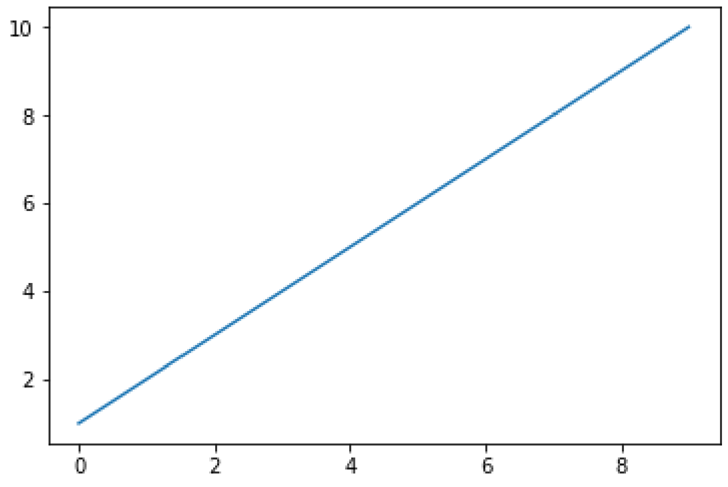
```
squared list of x:  [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
Square of only even numbers:  [4, 16, 36, 64, 100]
```

EXERCISE 2: Python Visualisation

In this UoS we will be using matplotlib library to plot our visualsations. Here are some demonstrations.

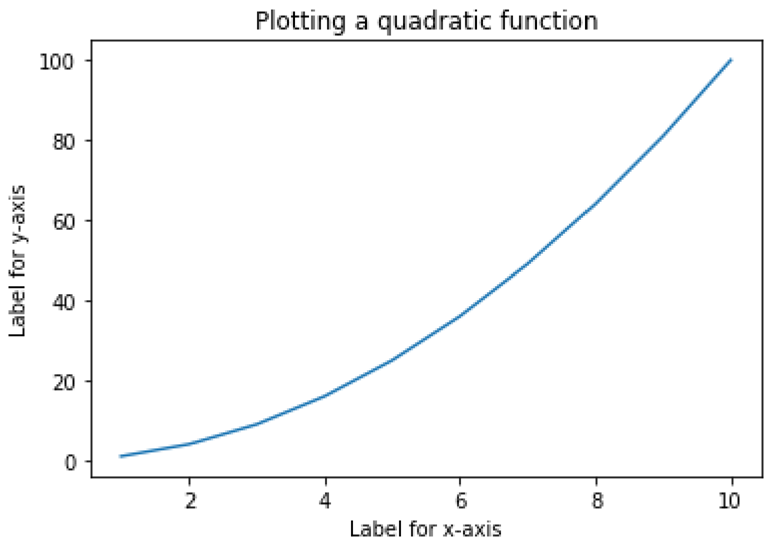
In [6]:

```
import matplotlib.pyplot as plt
x = [1,2,3,4,5,6,7,8,9,10]
plt.plot(x)
plt.show()

```



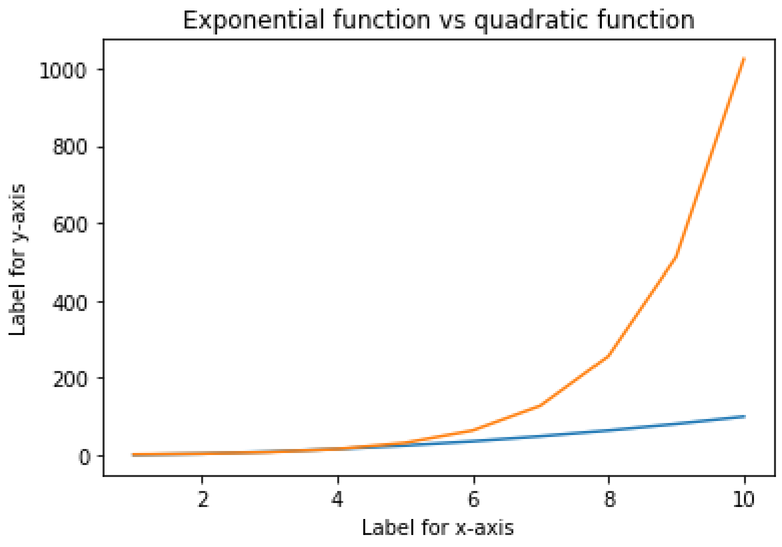
In [7]:

```
#Setting labels and plotting two variables x and y
y = [x**2 for x in x]
plt.plot(x,y)
plt.xlabel("Label for x-axis")
plt.ylabel("Label for y-axis")
plt.title('Plotting a quadratic function')
plt.show()
```



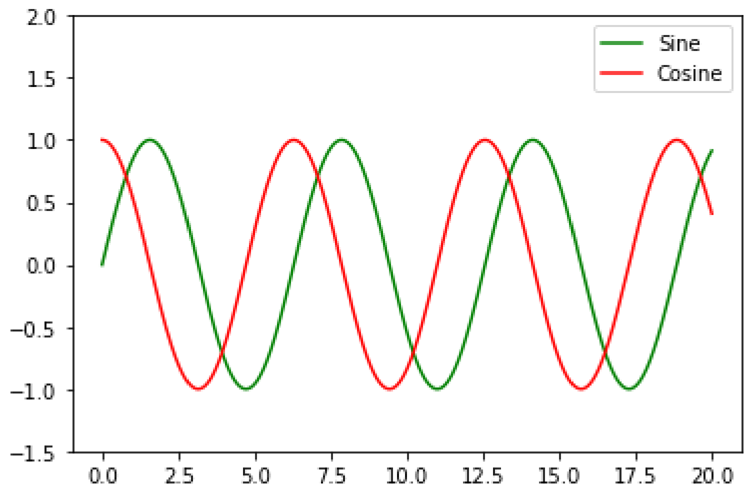
In [8]:

```
#Setting labels and plotting two variables x and y

z = [2**i for i in x]
plt.plot(x,y)
plt.plot(x,z)
plt.xlabel("Label for x-axis")
plt.ylabel("Label for y-axis")
plt.title('Exponential function vs quadratic function')
plt.show()
```



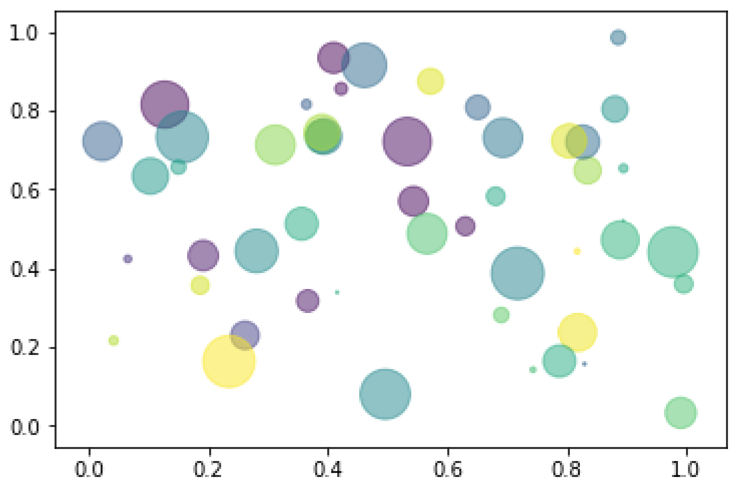
In [9]:

```
# Legend in the cornor
import numpy as np
x = np.linspace(0, 20, 1000)
y1 = np.sin(x)
y2 = np.cos(x)

plt.plot(x, y1, '-g', label='Sine')
plt.plot(x, y2, '-r', label='Cosine')
plt.legend(loc='upper right')
plt.ylim(-1.5, 2.0)
plt.show()
```



In [10]:

```
# Scatter plot
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2  # 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0