This is a "Hello World" example

In [ ]:

```
def hello_world():
  print("Hellow, World! ")

hello_world()
```

```
Hellow, World!
```

Convert the file to HTML using this explanation

StackOverflow link

In [ ]:

```
%%shell
jupyter nbconvert --to html /content/GSJenv.ipynb
```

```
[NbConvertApp] Converting notebook /content/GSJenv.ipynb to html
[NbConvertApp] Writing 575859 bytes to /content/GSJenv.html
```

Out[ ]:

In [ ]:

```
import nltk
import numpy
```

In [ ]:

```
%lsmagic
```

Out[ ]:

```
Available line magics:
%alias  %alias_magic  %autoawait  %autocall  %automagic  %autosave  %bookmark  %cat  %cd  %clear  %colors  %conda  %config  %connect_info  %cp  %debug  %dhist  %dirs  %doctest_mode  %ed  %edit  %env  %gui  %hist  %history  %killb
gscripts  %ldir  %less  %lf  %lk  %ll  %load  %load_ext  %loadpy  %logoff  %logon  %logstart  %logstate  %logstop  %ls  %lsmagic  %lx  %macro  %magic  %man  %matplotlib  %mkdir  %more  %mv  %notebook  %page  %pastebin  %pdb  %pde
f  %pdoc  %pfile  %pinfo  %pinfo2  %pip  %popd  %pprint  %precision  %prun  %psearch  %psource  %pushd  %pwd  %pycat  %pylab  %qtconsole  %quickref  %recall  %rehashx  %reload_ext  %rep  %rerun  %reset  %reset_selective  %rm  %rm
dir  %run  %save  %sc  %set_env  %shell  %store  %sx  %system  %tb  %tensorflow_version  %time  %timeit  %unalias  %unload_ext  %who  %who_ls  %whos  %xdel  %xmode

Available cell magics:
%%!  %%HTML  %%SVG  %%bash  %%bigquery  %%capture  %%debug  %%file  %%html  %%javascript  %%js  %%latex  %%markdown  %%perl  %%prun  %%pypy  %%python  %%python2  %%python3  %%ruby  %%script  %%sh  %%shell  %%svg  %%sx  %%system
%%time  %%timeit  %%writefile

Automagic is ON, % prefix IS NOT needed for line magics.
```

LaTeX inside text cell

$(a^2 + b^2) = c^2$

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i. \tag{1}$$

LaTex in code cell using Magic

In [ ]:

```
%%latex
$$(a^2 + b^2) = c^2$
```

$$(a^2 + b^2) = c^2$

Calculate Factorial using a for loop

In [ ]:

```
def factorialA(n):
  fact = 1
  for i in range(1, n+1):
    fact = fact * i
  return fact
```

Calculate Factorial using recursion

In [ ]:

```
def factorialB(n):
  if n == 1:
    fact = 1
  else:
    fact = n * factorialB(n-1)
  return fact
```

In [ ]:

```
%timeit factorialA(100)
```

```
4.69 µs ± 36.7 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

In [ ]:

```
%timeit factorialB(100)
```

```
12.5 µs ± 575 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Embed a link

XG Boost link

Basic KNN Milestone

In [1]:

```
from numpy import *
import operator

def createDataSet():
    group = array([[1.0,1.1],[1.0,1.0],[0,0],[0,0.1]])
    labels = ['A','A','B','B']
    return group, labels
```

In [2]:

```
group,labels = createDataSet()
```

In [3]:

```
type(group)
```

Out[3]:

```
numpy.ndarray
```

In [4]:

```
type(labels)
```

Out[4]:

```
list
```

In [5]:

```
from sklearn.neighbors import KNeighborsClassifier as kNN
```

In [6]:

```
t = reshape([0,0],(1,-1))
```

Create a trained model and predict class of one point

In [ ]:

```
model = kNN(n_neighbors=3)
model.fit(group, labels)
model.predict(t)
```

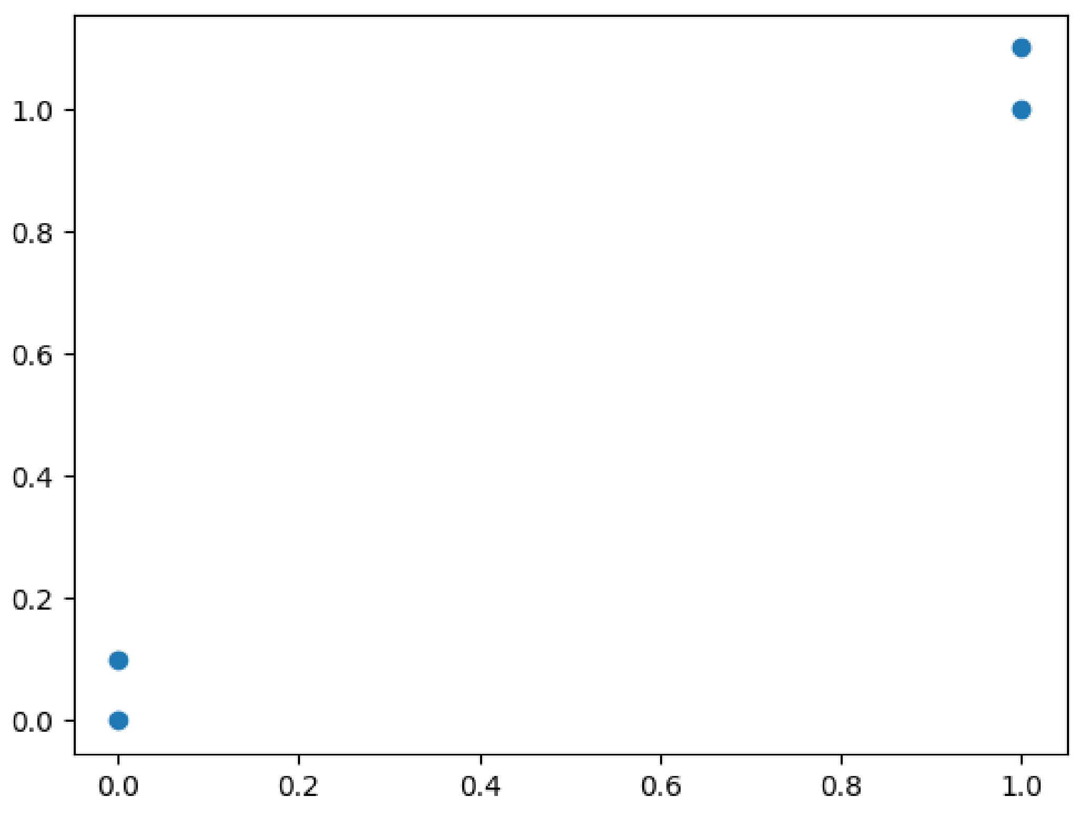
Out[ ]:

```
array(['B'], dtype='<U1')
```

Plot the data using matplotlib

In [15]:

```
import matplotlib
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)
ax.scatter(group[:,0],group[:,1])
plt.show()
```



In [16]:

```
%%shell
jupyter nbconvert --to pdf /content/GSJenv.ipynb
```