Name: Asad Haroon

Numpy Tutorials

- It is fast and require less space as compared to Python Lists.
- It is written in C language.
- It is used to do mathematical operations like Linear Algebra.
- It is used to do many operations on Array.
- It provides a high-performance multidimensional array object

Functions:

- **arange():** It works like range function. And it will create an array from 0 value to inputted value.
- **ones():**It will create an matrix of all ones of specific dimension.
- **repeat(array,x):**It will repeat the elements of array each x times.
- **tile(array,x):**It will repeat the whole array x times.
- **zeros():** It will create an array of all zeros of specific dimension.
- **shape():**It will return the order of the matrix.
- **full():**It will make an matrix of all input value in specific order.
- **sum(axis):**It will sum up col-wise elements of matrix if **axis=0. else if axis=1,** then it will sum up row-wise elements.
- **where(condition):**It will tell the location of elements of given condition.

In [1]:
```
import numpy as np
```

In [2]:

```
#arange function():

#Example1 arange()
arr=np.arange(10)
print("Array of range till 10: {}".format(arr))
#Example2 arange()
arr3=np.arange(10,50,5) # it will create an array of value starting from 10 and ending at 25 with step-iteration
#Syntax of arange is arange(start=10,stop=50,step=5)
print("Array of arrange function [10:25,5] is {}".format(arr3))

#ones function():
arr2=np.ones((2,3),dtype=int)
print("\nArray of 2x3 of ones: {}".format(arr2))

#repeat function():
arr=np.array([1,2,3])
arr=np.repeat(arr,3)
print("The contents of repeating arr,3 are {}".format(arr))

#tile function():
a=np.array([1,2,3])
arr=np.tile(a,4)
print("The contents of tile a,4 are {}".format(arr))
```

```
Array of range till 10: [0 1 2 3 4 5 6 7 8 9]
Array of arrange function [10:25,5] is [10 15 20 25 30 35 40 45]

Array of 2x3 of ones: [[1 1 1]
 [1 1 1]]
The contents of repeating arr,3 are [1 1 1 2 2 2 3 3 3]
The contents of tile a,4 are [1 2 3 1 2 3 1 2 3 1 2 3]
```

In [3]:

```
#zeros function():
zero=np.zeros((2,2),dtype=int)
print("Array of 2x2 of all zeros is {}\n".format(zero))

#shape function():
arr=np.zeros((3,4))
row,col=np.shape(arr) # shape function return tuple of order of matrix which is (rows,cols)
print("rows are {} and cols are {}\n".format(row,col))

#full function():
arrfull=np.full((3,4),10) # it will create an matrix of all values 10 with dimension of 3x4.
print("Matrix full with 10 is {}".format(arrfull))
#axis function():
arr1=np.array(([1,2,3],[4,5,6]))
print("Contents of arr1 is {}\n".format(arr1))
a=arr1.sum(axis=0)
print("Sum of arr1 on axis=0 is {}".format(a))
a=arr1.sum(axis=1)
print("Sum of arr1 on axis=1 is {}".format(a))

#where function():
arr=np.array([1,2,10,22,21])
a=np.where(arr>10)
print("The locations of arr>10 are {}".format(a))
```

```
Array of 2x2 of all zeros is [[0 0]
 [0 0]]

rows are 3 and cols are 4

Matrix full with 10 is [[10 10 10 10]
 [10 10 10 10]
 [10 10 10 10]]
Contents of arr1 is [[1 2 3]
 [4 5 6]]

Sum of arr1 on axis=0 is [5 7 9]
Sum of arr1 on axis=1 is [ 6 15]
The locations of arr>10 are (array([3, 4], dtype=int64),)
```

Apply Function on Array

In [4]:
```
def f(x):
    if x>=1:
        return -1
    return 100

x=np.array([0,0,0,1,0,-1,1,1,2,3,4,5,6,8,-1,0]).reshape(4,4)
print("Original array:\n",x)
func=np.vectorize(f)

output_arr=func(x)
print("Output Array:\n",output_arr)
```

```
Original array:
 [[ 0  0  0  1]
 [ 0 -1  1  1]
 [ 2  3  4  5]
 [ 6  8 -1  0]]
Output Array:
 [[100 100 100  -1]
 [100 100  -1  -1]
 [ -1  -1  -1  -1]
 [ -1  -1 100 100]]
```

Method to change values without Function

In [5]:
```
output_arr[output_arr>=1]=2000
```

In [6]:
```
output_arr
```

Out[6]:
```
array([[2000, 2000, 2000,   -1],
       [2000, 2000,   -1,   -1],
       [  -1,   -1,   -1,   -1],
       [  -1,   -1, 2000, 2000]])
```

In [7]:
```
newvalues=[0,1,2]
values,counts=np.unique(output_arr, return_counts=True)
print("Unique values found in array are \n",values)
d=dict(zip(values,newvalues))
print("Dictionary of prev,new values are\n",d)
# now changing values of numpy array with dictionary
newarray=np.vectorize(d.get)(output_arr)
print("Before:\n",output_arr)
print("Now Values in array are\n",newarray)
```

```
Unique values found in array are
 [  -1 2000]
Dictionary of prev,new values are
 {-1: 0, 2000: 1}
Before:
 [[2000 2000 2000   -1]
 [2000 2000   -1   -1]
 [  -1   -1   -1   -1]
 [  -1   -1 2000 2000]]
Now Values in array are
 [[1 1 1 0]
 [1 1 0 0]
 [0 0 0 0]
 [0 0 1 1]]
```

In [ ]:
```
```