Piaic Assignment # 1: Q2 _Section # 1

In [53]:
```
import numpy as np
np.zeros(10, dtype = 'i')
```

Out[53]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int32)
```

In [56]:
```
print(np.arange(10,50))
print("This is the shape {shape} \nAnd this is the type {type}".format(shape = np.shape(vect
or), type = vector.dtype))
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
This is the shape (40,)
And this is the type int32
```

In [27]:
```
print(np.__version__)
print(np.show_config())
```

```
1.18.5
blas_mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/User1/anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.11
7\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.117
\\windows\\mkl\\include', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_201
9.0.117\\windows\\mkl\\lib', 'C:/Users/User1/anaconda3\\Library\\include']
blas_opt_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/User1/anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.11
7\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.117
\\windows\\mkl\\include', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_201
9.0.117\\windows\\mkl\\lib', 'C:/Users/User1/anaconda3\\Library\\include']
lapack_mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/User1/anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.11
7\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.117
\\windows\\mkl\\include', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_201
9.0.117\\windows\\mkl\\lib', 'C:/Users/User1/anaconda3\\Library\\include']
lapack_opt_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/Users/User1/anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.11
7\\windows\\mkl', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_2019.0.117
\\windows\\mkl\\include', 'C:\\Program Files (x86)\\IntelSWTools\\compilers_and_libraries_201
9.0.117\\windows\\mkl\\lib', 'C:/Users/User1/anaconda3\\Library\\include']
None
```

In [28]:
```
print(vector.ndim) #How many Dimentions?
```

```
1
```

In [47]:
```
arr = np.arange(8)
arr_bool = arr < 50
print(arr_bool)
```

```
[ True  True  True  True  True  True  True  True]
```

In [48]:
```
arr2d = arr_bool.reshape(2,4)
print(arr2d)
```

```
[[ True  True  True  True]
 [ True  True  True  True]]
```

In [50]:
```
arr3d = arr2d.reshape(2,2,2)
print(arr3d)
```

```
[[[ True  True]
  [ True  True]]

 [[ True  True]
  [ True  True]]]
```

Section # 2 _Difficulty= Easy

In [69]:
```
arr1 = np.arange(10)
rvs_arr1 = arr1[::-1]
print(rvs_arr1)   #Reversaing a Vectoer
```

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

In [62]:
```
arr_2= np.zeros(10, dtype='i')
arr2[4]=1
print(arr_2)
```

```
[0 0 0 0 1 0 0 0 0 0]
```

In [65]:
```
iden_mat = np.ones((3,3))
print(iden_mat)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

In [70]:
```
arr_1 = np.array([1, 2, 3, 4, 5], dtype='f')
print(arr_1)
```

```
[1. 2. 3. 4. 5.]
```

In [81]:
```
array1 = np.array([[1., 2., 3.],[4., 5., 6.]])
array2 = np.array([[0., 4., 1.],[7., 2., 12.]])
product = array1*array2
print(product)
comparing = array1 == array2
print("Resulting array after comparing both\n",comparing)
```

```
[[ 0.  8.  3.]
 [28. 10. 72.]]
Resulting array after comparing both
 [[False False False]
 [False False False]]
```

In [98]:
```
odd_arr = np.arange(10)
odd = []
for i in odd_arr:
    if i%2 == 1:
        odd.append(i)
odd_val =np.array(odd)
print(odd_val)   #Extracting Odd values
```

```
[1 3 5 7 9]
```

In [101]:
```
b = np.where(odd_val==14, odd_val, -1)
print(b)   #Replacing above array with -1
```

```
[-1 -1 -1 -1 -1]
```

In [107]:
```
odd_arr[(odd_arr > 4) & (odd_arr < 9)] = 12
print(odd_arr)     #Replacing 5,6,7,8 with 12
```

```
[ 0  1  2  3  4 12 12 12 12  9]
```

In [115]:
```
box = np.ones(4,4), dtype='i')
box[1:-1, 1:-1] = 0
print(box)
```

```
[[1 1 1 1]
 [1 0 0 1]
 [1 0 0 1]
 [1 1 1 1]]
```

Section # 3 _Difficulty=Medium

In [119]:
```
arr_2d = np.array([[1, 2, 3],[4, 5, 6], [7, 8, 9]])
arr_2d[1:1] =12
print(arr_2d)
```

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

In [129]:
```
arr_3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
arr_3d[0:1, 0:2] =64
print(arr_3d)  #Changing values of 1st array to 64
```

```
[[[64 64 64]
  [64 64 64]]

 [[ 7  8  9]
  [10 11 12]]]
```

In [138]:
```
sliced2d= np.arange(10)
slice_2d = sliced.reshape(2,5)
a = slice_2d[0:1,0:5]
print(a)   #Slicing 1st array
```

```
[[0 1 2 3 4]]
```

In [141]:
```
c = slice_2d[1,1]
print(c)   #slice out the 2nd value from 2nd 1-D array
```

```
6
```

In [144]:
```
d = slice_2d[0:2,2]
print(d)       #slice out the third column but only the first two rows
```

```
[2 7]
```

In [159]:
```
farr = np.random.randint((1,101))
print(farr.min())
farr.max()    #find the minimum and maximum values
```

```
0
```

Out[159]:
```
94
```

In [160]:
```
a1 = np.array([1,2,3,2,3,4,3,4,5,6])
b1= np.array([7,2,10,2,7,4,9,4,9,8])
print(np.intersect1d(a1,b1))  #common items between a and b
```

```
[2 4]
```

In [161]:
```
a_1 = np.array([1,2,3,2,3,4,3,4,5,6])
b_1 = np.array([7,2,10,2,7,4,9,4,9,8])
np.where( a_1 == b_1)  # positions where elements of a and b match
```

Out[161]:
```
(array([1, 3, 5, 7], dtype=int64),)
```

In [171]:
```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
data = np.random.randn(7, 4)
e = []
for i in names:
    if i != 'Will':
        e.append(True)
    else:
        e.append(False)
filar = data[e]
print(e)
print(filar) # all the values from array data where the values from array names are not equa
```

```
[True, True, False, True, False, True, True]
[[-0.03560971  0.10243188 -0.82482451  1.95104768]
 [ 1.37835574 -0.02703835 -1.55229039 -0.89914178]
 [-1.68246311  2.39209654  1.03784597  0.70965806]
 [-0.57976241  0.94160015 -0.4094964  -0.13807723]
 [ 0.1172901  -1.5238749   0.5624968   0.24032881]]
```

In [176]:
```
names1 = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
data1 = np.random.randn(7, 4)
e1 = []
for nam in names1:
    if nam == 'Bob':
        e1.append(True)
    else:
        e1.append(False)
filar1 = data[e1]
print(e1)
print(filar1) #all the values from array data where the values from array names are not equa
```

```
[True, False, False, True, False, False, False]
[[-0.03560971  0.10243188 -0.82482451  1.95104768]
 [-1.68246311  2.39209654  1.03784597  0.70965806]]
```

Section # 4 _Difficulty = Hard

In [184]:
```
np.arange(1.0,16.0).reshape(5,3)
```

Out[184]:
```
array([[ 1.,  2.,  3.],
       [ 4.,  5.,  6.],
       [ 7.,  8.,  9.],
       [10., 11., 12.],
       [13., 14., 15.]])
```

In [185]:
```
np.arange(1.0,17.0).reshape(2,2,4)
```

Out[185]:
```
array([[[ 1.,  2.,  3.,  4.],
        [ 5.,  6.,  7.,  8.]],

       [[ 9., 10., 11., 12.],
        [13., 14., 15., 16.]]])
```

In [200]:
```
x1 = np.arange(1.0,17.0).reshape(2,2,4)
np.swapaxes(x1,1,2) #Swap axes of the array
```

Out[200]:
```
array([[[ 1.,  5.],
        [ 2.,  6.],
        [ 3.,  7.],
        [ 4.,  8.]],

       [[ 9., 13.],
        [10., 14.],
        [11., 15.],
        [12., 16.]]])
```

In [204]:
```
r = np.arange(10)
np.sqrt(r)   #The square root of every element in the array, if the values less than 0.5, re
```

Out[204]:
```
array([0.  , 1.  , 2.  , 3.  , 4.  , 5.  , 6.  , 7.  , 8.  , 9.  ])
```

In [239]:
```
r1 = np.random.randint(12, size=(4,3))
r2 =  np.random.randint(12, size=(4,3))
maxi = np.maximum(r1,r2)
print(maxi)              # two random arrays of range 12 and array with the maximum values
```

```
[[ 6  7  6]
 [11  7 11]
 [ 3  2 10]
 [ 4  4  4]]
```

In [226]:
```
namesid = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
u = np.unique(namesid)
np.sort(u)  # unique names and sort
```

Out[226]:
```
array(['Bob', 'Joe', 'Will'], dtype='<U4')
```

In [227]:
```
a2 = np.array([1,2,3,4,5])
b2 = np.array([5,6,7,8,9])
np.setdiff1d(a2,b2) #From array a remove all items present in array b
```

Out[227]:
```
array([1, 2, 3, 4])
```

In [232]:
```
sampleArray = np.array([[34,43,73],[82,22,12],[53,94,66]])
newColumn = np.array([[10,10,10]])
delarr = np.delete(sampleArray,1,1)
insarr = np.insert(delarr,1,newColumn,1)
print(insarr) #NumPy array delete column two and insert following new column in its place
```

```
[[34 10 73]
 [82 10 12]
 [53 10 66]]
```

In [233]:
```
xx = np.array([[1., 2., 3.], [4., 5., 6.]])
yy = np.array([[6., 23.], [-1, 7], [8, 9]])
np.dot(xx,yy) # the dot product of the above two matrix
```

Out[233]:
```
array([[ 28.,  64.],
       [ 67., 181.]])
```

In [240]:
```
cumarr = np.random.randint(20 , size=(5,4))
np.cumsum(cumarr) #Generate a matrix of 20 random values and find its cumulative sum
```

Out[240]:
```
array([  2,  20,  26,  28,  39,  46,  51,  69,  70,  71,  73,  85,  97,
       116, 134, 147, 152, 155, 173, 189], dtype=int32)
```

Finished= Assignment:1