Assignment For Numpy

Difficulty Level **Beginner**

1. Import the numpy package under the name np

In [4]:

```
import numpy as np
```

2. Create a null vector of size 10

In [22]:

```
v = np.zeros(10)
v
```

Out[22]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

3. Create a vector with values ranging from 10 to 49

In [23]:

```
a = np.arange(10,49)
a
```

Out[23]:

```
array([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])
```

4. Find the shape of previous array in question 3

In [8]:

```
a.shape
```

Out[8]:

```
(39,)
```

5. Print the type of the previous array in question 3

In [11]:

```
a.dtype
```

Out[11]:

```
dtype('int32')
```

6. Print the numpy version and the configuration

In [15]:

```
 print(np.__version__)
print(np.show_config())
```

```
1.19.2
blas_mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/ProgramData/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:/ProgramData/Anaconda3\\Library\\include']
blas_opt_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/ProgramData/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:/ProgramData/Anaconda3\\Library\\include']
lapack_mkl_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/ProgramData/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:/ProgramData/Anaconda3\\Library\\include']
lapack_opt_info:
    libraries = ['mkl_rt']
    library_dirs = ['C:/ProgramData/Anaconda3\\Library\\lib']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['C:/ProgramData/Anaconda3\\Library\\include']
None
```

7. Print the dimension of the array in question 3

In [16]:

```
a.ndim
```

Out[16]:

```
1
```

8. Create a boolean array with all the True values

In [29]:

```
b = np.random.randn(10)
b>0
```

Out[29]:

```
array([False, False, False, False, False, False, False,  True,  True,
        True])
```

9. Create a two dimensional array

In [33]:

```
c = np.zeros((3,4))
c.ndim
```

Out[33]:

```
2
```

10. Create a three dimensional array

In [35]:

```
c = np.empty((2,4,6))
c.ndim
```

Out[35]:

```
3
```

Difficulty Level **Easy**

11. Reverse a vector (first element becomes last)

In [37]:

```
a = np.array(range(1,8))
print(a)
a[0],a[len(a)-1]=a[len(a)-1],a[0]
a
```

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

Out[37]:

```
array([7, 2, 3, 4, 5, 6, 1])
```

12. Create a null vector of size 10 but the fifth value which is 1

In [38]:

```
null_vec = np.zeros(10)
null_vec[5]=1
null_vec
```

Out[38]:

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.])
```

13. Create a 3x3 identity matrix

In [40]:

```
m = np.identity(3)
m
```

Out[40]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

14. arr = np.array([1, 2, 3, 4, 5])

Convert the data type of the given array from int to float

In [42]:

```
arr = np.array([1,2,3,4,5])
arr = arr.astype(np.float32)
arr
```

Out[42]:

```
array([1., 2., 3., 4., 5.], dtype=float32)
```

15. arr1 = np.array([[1., 2., 3.],

```
                   [4., 5., 6.]])
```

arr2 = np.array([[0., 4., 1.],

```
                  [7., 2., 12.]])
```

Multiply arr1 with arr2

In [44]:

```
arr1 = np.array([[1., 2., 3.],[4., 5., 6.]])
arr2 = np.array([[0., 4., 1.],[7., 2., 12.]])
a = arr1*arr2
a
```

Out[44]:

```
array([[ 0.,  8.,  3.],
       [28., 10., 72.]])
```

16. arr1 = np.array([[1., 2., 3.],

```
                    [4., 5., 6.]])
```

    arr2 = np.array([[0., 4., 1.],

```
                    [7., 2., 12.]])
```

Make an array by comparing both the arrays provided above

In [45]:

```
arr1 = np.array([[1., 2., 3.],[4., 5., 6.]])
arr2 = np.array([[0., 4., 1.],[7., 2., 12.]])
b = arr1>arr2
b
```

Out[45]:

```
array([[ True, False,  True],
       [False,  True, False]])
```

17. Extract all odd numbers from arr with values(0-9)

In [49]:

```
d = np.arange(9)
x = filter(lambda i: i%2 != 0, d)
list(x)
```

Out[49]:

```
[1, 3, 5, 7]
```

18. Replace all odd numbers to -1 from previous array

In [50]:

```
d[d%2 !=0] = -1
d
```

Out[50]:

```
array([ 0, -1,  2, -1,  4, -1,  6, -1,  8])
```

19. arr = np.arange(10)

Replace the values of indexes 5,6,7 and 8 to **12**

In [54]:

```
arr = np.arange(10)
arr[5:9]= 12
arr
```

Out[54]:

```
array([ 0,  1,  2,  3,  4, 12, 12, 12, 12,  9])
```

20. Create a 2d array with 1 on the border and 0 inside

In [64]:

```
a = np.ones((5,8))
a[1:-1, 1:-1] = 0
a
```

Out[64]:

```
array([[1., 1., 1., 1., 1., 1., 1., 1.],
       [1., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1.]])
```

Difficulty Level **Medium**

21. arr2d = np.array([[1, 2, 3],

```
                    [4, 5, 6],

                    [7, 8, 9]])
```

Replace the value 5 to 12

In [72]:

```
arr2d = np.array([[1, 2, 3],[4, 5, 6],[7, 8, 9]])
arr2d [1, 1]= 12
arr2d
```

Out[72]:

```
array([[ 1,  2,  3],
       [ 4, 12,  6],
       [ 7,  8,  9]])
```

22. arr3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])

Convert all the values of 1st array to 64

In [75]:

```
arr3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
arr3d[0][:] = 64
arr3d
```

Out[75]:

```
array([[[64, 64, 64],
        [64, 64, 64]],

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

23. Make a 2-Dimensional array with values 0-9 and slice out the first 1st 1-D array from it

In [82]:

```
a = np.arange(9)
b = a.reshape(3,3)
print(b)
c = b[0]
c
```

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

Out[82]:

```
array([0, 1, 2])
```

24. Make a 2-Dimensional array with values 0-9 and slice out the 2nd value from 2nd 1-D array from it

In [80]:

```
c = b[1][1]
c
```

Out[80]:

```
4
```

25. Make a 2-Dimensional array with values 0-9 and slice out the third column but only the first two rows

In [85]:

```
c = b [0:-1,-1]
c
```

Out[85]:

```
array([2, 5])
```

26. Create a 10x10 array with random values and find the minimum and maximum values

In [92]:

```
x = np.random.randn(10)
y = np.random.randn(10)
a = np.maximum(x,y)
print(a)
b = np.minimum(x,y)
print(b)
```

```
[ 0.32489469  1.47737755 -0.20214492 -0.42561296  0.19838726 -0.11353063
  0.73815921 -0.14864081  0.00415652  0.30609905]
[-0.41645288 -0.45415636 -0.47754711 -1.1475709  -0.67672663 -0.57729826
 -0.83704347 -0.53091542 -0.11940108 -0.04550748]
```

27. a = np.array([1,2,3,2,3,4,3,4,5,6]) b = np.array([7,2,10,2,7,4,9,4,9,8])

Find the common items between a and b

In [94]:

```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
np.intersect1d(a, b)
```

Out[94]:

```
array([2, 4])
```

28. a = np.array([1,2,3,2,3,4,3,4,5,6]) b = np.array([7,2,10,2,7,4,9,4,9,8])

Find the positions where elements of a and b match

In [95]:

```
np.where(a == b)
```

Out[95]:

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

29. names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe']) data = np.random.randn(7, 4)

Find all the values from array **data** where the values from array **names** are not equal to **Will**

In [100]:

```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
data = np.random.randn(7, 4)
data [names != 'Will']
```

Out[100]:

```
array([[ 0.07417387,  0.61780779,  0.28641796,  0.16559453],
       [ 1.12387455, -0.85762767, -0.21607959,  0.14085558],
       [-1.72523903, -1.92268221,  0.11661072, -1.0999373 ],
       [-0.07408603, -0.51811919, -0.59171114, -1.85771244],
       [ 0.91342748, -2.04779742, -0.43525572, -0.2150221 ]])
```

30. names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe']) data = np.random.randn(7, 4)

Find all the values from array **data** where the values from array **names** are not equal to **Will** and **Joe**

In [103]:

```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
data = np.random.randn(7, 4)
mask = (names != 'Will') & (names != 'Joe')
data[mask]
```

Out[103]:

```
array([[-0.75267251,  0.39106928,  0.816148  ,  0.84104313],
       [ 0.72292912,  0.05726951,  0.85272857, -1.49951081]])
```

Difficulty Level **Hard**

31. Create a 2D array of shape 5x3 to contain decimal numbers between 1 and 15.

In [104]:

```
a = np.random.uniform(1, 15, size =(5,3))
a
```

Out[104]:

```
array([[ 1.59350255, 14.873676  ,  3.83986136],
       [ 3.12071658,  2.90730622,  4.76617713],
       [12.24598107,  7.53101656,  5.71642467],
       [ 1.91960697,  8.40757832, 13.42511627],
       [ 5.47435609, 12.57550826,  5.76133226]])
```

32. Create an array of shape (2, 2, 4) with decimal numbers between 1 to 16.

In [106]:

```
a = np.random.uniform(1, 16, size =(2, 2, 4))
a
```

Out[106]:

```
array([[[13.80198702,  6.41782173,  5.33830782,  1.80275588],
        [ 5.14803192,  9.868016  ,  4.10416329,  5.06015751]],

       [[13.94924221, 13.98079277,  6.88830949, 10.56992696],
        [ 6.2408857 , 11.51266935,  9.15204427, 15.89297728]]])
```

33. Swap axes of the array you created in Question 32

In [109]:

```
np.swapaxes(a, 0, 1)
a
```

Out[109]:

```
array([[[13.80198702,  6.41782173,  5.33830782,  1.80275588],
        [ 5.14803192,  9.868016  ,  4.10416329,  5.06015751]],

       [[13.94924221, 13.98079277,  6.88830949, 10.56992696],
        [ 6.2408857 , 11.51266935,  9.15204427, 15.89297728]]])
```

34. Create an array of size 10, and find the square root of every element in the array, if the values less than 0.5, replace them with 0

In [110]:

```
import math
arr = np.arange(10)
print(arr)
b = filter(lambda x : x if math.sqrt(x)>0.5 else 0, arr)
np.asarray(list(b))
```

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

Out[110]:

```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

35. Create two random arrays of range 12 and make an array with the maximum values between each element of the two arrays

In [117]:

```
a = np.random.randint(12, size=random.randrange(12))
b = np.random.randint(12, size=random.randrange(12))
np.where(a == b)
```

```
<ipython-input-117-210de27e7f1a>:3: DeprecationWarning: elementwise comparis
on failed; this will raise an error in the future.
  np.where(a == b)
```

Out[117]:

```
(array([], dtype=int64),)
```

36. names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])

Find the unique names and sort them out!

In [118]:

```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
np.unique(names)
```

Out[118]:

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

37. a = np.array([1,2,3,4,5]) b = np.array([5,6,7,8,9])

From array a remove all items present in array b

In [119]:

```
a = np.array([1,2,3,4,5])
b = np.array([5,6,7,8,9])
c =np.intersect1d(a,b)
a = filter(lambda x : x if x not in c else None, a)
np.asarray(list(a))
```

Out[119]:

```
array([1, 2, 3, 4])
```

38. Following is the input NumPy array delete column two and insert following new column in its place.

---

sampleArray = numpy.array([[34,43,73],[82,22,12],[53,94,66]])

---

newColumn = numpy.array([[10,10,10]])

In [121]:

```
sampleArray = np.array([[34,43,73],[82,22,12],[53,94,66]])
sampleArray[:, 1] = 10
sampleArray
```

Out[121]:

```
array([[34, 10, 73],
       [82, 10, 12],
       [53, 10, 66]])
```

39. x = np.array([[1., 2., 3.], [4., 5., 6.]]) y = np.array([[6., 23.], [-1, 7], [8, 9]])

Find the dot product of the above two matrix

In [122]:

```
x = np.array([[1., 2., 3.], [4., 5., 6.]])
y = np.array([[6., 23.], [-1, 7], [8, 9]])
np.dot(x,y)
```

Out[122]:

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

40. Generate a matrix of 20 random values and find its cumulative sum

In [123]:

```
m = np.random.randint(20)
np.cumsum(m)
```

Out[123]:

```
array([6], dtype=int32)
```

In [ ]:

```

```