Day 13

Numpy Part 2

Data Types in Python By default Python have these data types:

strings integer float boolean

In [ ]:

```
Data Types in NumPy
NumPy has some extra data types, and refer to data types with one character, li

i - integer
b - boolean
u - unsigned integer
f - float
c - complex float
m - timedelta
M - datetime
O - object
S - string
U - unicode string
V - fixed chunk of memory for other type ( void )
```

Checking the Data Type of an Array

In [1]:

```
import numpy as np

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

print(arr.dtype)
```

```
int32
```

In [3]:

```
import numpy as np

arr = np.array(['animal', 'bat', 'cat'])

print(arr.dtype)
```

```
<U6
```

Creating Arrays With a Defined Data Type

In [4]:

```
import numpy as np

arr = np.array([1, 2, 3, 4], dtype='S')

print(arr)
print(arr.dtype)
```

```
[b'1' b'2' b'3' b'4']
|S1
```

Create an array with data type 4 bytes integer:

In [5]:

```
arr = np.array([1, 2, 3, 4], dtype='i4')

print(arr)
print(arr.dtype)
```

```
[1 2 3 4]
int32
```

NumPy Array Shape

The shape of an array is the number of elements in each dimension.

Print the shape of a 2-D array:

In [6]:

```
import numpy as np

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

print(arr.shape)
```

```
(2, 4)
```

Create an array with 5 dimensions using ndmin using a vector with values 1,2,3,4 and verify that last dimension has value 4:

In [7]:

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

print(arr)
print('shape of array :', arr.shape)
```

```
[[[[[1 2 3 4]]]]]
shape of array : (1, 1, 1, 1, 4)
```

Joining NumPy Arrays

In [8]:

```
import numpy as np

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

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```

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

Join two 2-D arrays along rows (axis=1): Rows

In [9]:

```
import numpy as np

arr1 = np.array([[1, 2], [3, 4]])

arr2 = np.array([[5, 6], [7, 8]])

arr = np.concatenate((arr1, arr2), axis=1)

print(arr)
```

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

Join two 2-D arrays along rows (axis=1): Columns

In [10]:

```
import numpy as np

arr1 = np.array([[1, 2], [3, 4]])

arr2 = np.array([[5, 6], [7, 8]])

arr = np.concatenate((arr1, arr2), axis=0)

print(arr)
```

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

Splitting NumPy Arrays

Splitting is reverse operation of Joining.

Joining merges multiple arrays into one and Splitting breaks one array into multiple.

Split the array in 3 parts:

In [11]:

```


arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array_split(arr, 3)

print(newarr)
```

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

Split the array in 4 parts

In [12]:

```
arr = np.array([1, 2, 3, 4, 5, 6])

newarr = np.array_split(arr, 4)

print(newarr)
```

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

ravel & flatten - converts multidimensional array into 1 d array

In [14]:

```
m = np.array([[[1,2,3],[4,5,6],[7,8,9]]])
print(m)
print("dimension is", m.ndim)
print()
n = m.ravel()
print(n)
print("now the new dimesnion is", n.ndim)
```

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

[1 2 3 4 5 6 7 8 9]
now the new dimesnion is 1
```

In [15]:

```
c = np.array([[[[1,2,3],[78,89,25],[45,26,84]]]])
print(c)
print("so the dimension is ",c.ndim)
print()
d = c.ravel()
print(d)

print("the dimension is",d.ndim)
```

```
[[[[ 1  2  3]
   [78 89 25]
   [45 26 84]]]]
so the dimension is  4

[ 1  2  3 78 89 25 45 26 84]
the dimension is 1
```

arithmetic operations in numpy arrays

Addition

In [16]:

```
x = np.array([1,2,3,4,5])
y = x+3
print(y)
```

```
[4 5 6 7 8]
```

In [17]:

```
x = np.array([1,2,3,4,5])
m = np.array([5,6,7,8,9])
y = x+m
print(y)
```

```
[ 6  8 10 12 14]
```

In [18]:

```
x = np.array([1,2,3,4,5])
m = np.array([5,6,7,8,9])
y = np.add(x,m)
print(y)
```

```
[ 6  8 10 12 14]
```

Subtraction

In [19]:

```
x = np.array([1,2,3,4,5])
y = x-3
print(y)
```

```
[-2 -1  0  1  2]
```

In [20]:

```
x = np.array([1,2,3,4,5])
m = np.array([5,6,7,8,9])
y = np.subtract(x,m)
print(y)
```

```
[-4 -4 -4 -4 -4]
```

In [21]:

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

```
[-4 -4 -4 -4 -4]
```

Multiplication

In [22]:

```
x = np.array([1,2,3,4])
y = x*3
print(y)
```

```
[ 3  6  9 12]
```

In [23]:

```
x = np.array([1,2,3,4,5])
m = np.array([5,6,7,8,9])
y =x*m
print(y)
```

```
[ 5 12 21 32 45]
```

In [24]:

```
x = np.array([1,2,3,4,5])
m = np.array([5,6,7,8,9])
y = np.multiply(x,m)
print(y)
```

```
[ 5 12 21 32 45]
```

Divison

In [25]:

```
x = np.array([1,2,3,4])
y = x/3
print(y)
```

```
[0.33333333 0.66666667 1.         1.33333333]
```

In [26]:

```
x = np.array([1,2,3,4,5])
m = np.array([5,6,7,8,9])
y = np.divide(x,m)
print(y)
```

```
[0.2        0.33333333 0.42857143 0.5        0.55555556]
```

Modulous

In [27]:

```
x = np.array([10,15,25,30])
y = x%3
print(y)
```

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

In [28]:

```
x = np.array([12,16,24,32])
m = np.array([7,4,8,23])
y = np.mod(x,m)
print(y)
```

```
[5 0 0 9]
```

In [29]:

```
x = np.array([12,16,24,32])
m = np.array([7,4,8,23])
y = x%m
print(y)
```

```
[5 0 0 9]
```

arithmetic operations in 2 d array

In [30]:

```
n1 = np.array([[1,2,3,4],[5,6,7,8]])
n2 = np.array([[1,2,3,4],[5,6,7,8]])
x = n1+n2
print(x)
```

```
[[ 2  4  6  8]
 [10 12 14 16]]
```

In [31]:

```
n1 = np.array([[1,2,3,4],[5,6,7,8]])
n2 = np.array([[1,2,3,4],[5,6,7,8]])
x = n1-n2
print(x)
```

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

In [32]:

```
n1 = np.array([[1,2,3,4],[5,6,7,8]])
n2 = np.array([[1,2,3,4],[5,6,7,8]])
x = n1*n2
print(x)
```

```
[[ 1  4  9 16]
 [25 36 49 64]]
```

In [33]:

```
n1 = np.array([[1,2,3,4],[5,6,7,8]])
n2 = np.array([[1,2,3,4],[5,6,7,8]])
x = n1/n2
print(x)
```

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

arithmetic opr - 3d array

In [34]:

```
n1 = np.array([[[1,2,3,4],[5,6,7,8]]])
n2 = np.array([[[1,2,3,4],[5,6,7,8]]])
x = n1+n2
print(x)
print(x.ndim)
```

```
[[[ 2  4  6  8]
  [10 12 14 16]]]
3
```

In [35]:

```
n1 = np.array([[[1,2,3,4],[5,6,7,8]]])
n2 = np.array([[[1,2,3,4],[5,6,7,8]]])
x = n1-n2
print(x)
print(x.ndim)
```

```
[[[0 0 0 0]
  [0 0 0 0]]]
3
```

In [36]:

```
n1 = np.array([[[1,2,3,4],[5,6,7,8]]])
n2 = np.array([[[1,2,3,4],[5,6,7,8]]])
x = n1*n2
print(x)
print(x.ndim)
```

```
[[[ 1  4  9 16]
  [25 36 49 64]]]
3
```

In [37]:

```
n1 = np.array([[[1,2,3,4],[5,6,7,8]]])
n2 = np.array([[[1,2,3,4],[5,6,7,8]]])
x = n1/n2
print(x)
print(x.ndim)
```

```
[[[1. 1. 1. 1.]
  [1. 1. 1. 1.]]]
3
```

Unique function

In [38]:

```
k = np.array([12,14,6,7,9,3,45,5,33,2,33,2,33,14,12,7,9])
print(k)
x = np.unique(k)
print(x)
```

```
[12 14  6  7  9  3 45  5 33  2 33  2 33 14 12  7  9]
[ 2  3  5  6  7  9 12 14 33 45]
```

In [39]:

```
k = np.array([12,14,6,7,9,3,45,5,33,2,33,2,33,14,12,7,9])
print(k)
x = np.unique(k, return_index = True)
print(x)
```

```
[12 14  6  7  9  3 45  5 33  2 33  2 33 14 12  7  9]
(array([ 2,  3,  5,  6,  7,  9, 12, 14, 33, 45]), array([9, 5, 7, 2, 3, 4, 0,
1, 8, 6], dtype=int64))
```

In [40]:

```
k = np.array([12,14,6,7,9,3,45,5,33,2,33,2,33,14,12,7,9])
print(k)
print()
x = np.unique(k, return_index = True, return_counts = True)
print(x)
```

```
[12 14  6  7  9  3 45  5 33  2 33  2 33 14 12  7  9]

(array([ 2,  3,  5,  6,  7,  9, 12, 14, 33, 45]), array([9, 5, 7, 2, 3, 4, 0,
1, 8, 6], dtype=int64), array([2, 1, 1, 1, 2, 2, 2, 2, 3, 1], dtype=int64))
```

Delete

In [41]:

```
a = np.array([12,13,14,15])
print(a)
print()
d = np.delete(a,[1])
print(d)
```

```
[12 13 14 15]

[12 14 15]
```

In [42]:

```
x = np.array([[2,7,9,6,8],[4,5,7,1,2],[50,0,65,6,7]])
print(x)
print()
m = np.delete(x, 1, axis = 0)
print(m)
```

```
[[ 2  7  9  6  8]
 [ 4  5  7  1  2]
 [50  0 65  6  7]]

[[ 2  7  9  6  8]
 [50  0 65  6  7]]
```

In [ ]: