Contents:

Creating NumPy arrays Basic Indexing and Slicing Boolean Indexing Fancy Indexing Matrix operations Broadcasting

In [ ]:

```
#Creating arrays

import numpy as np
a=np.zeros(10,dtype=int)
print(a)

a=np.ones((2,3),dtype=float)
print(a)

a=np.full((3,4),22)
print(a)

a=np.eye(4,4)
print(a)

a=np.arange(0,20,3)
print(a)

a=np.linspace(0,1,5)
print(a)

#uniformly distrbuted random values
a=np.random.random((2,2))
print(a)

#normally distributed random values
a=np.random.normal(0,1,(2,3))
print(a)

#random integers
a=np.random.randint(2,20,(2,2))
print(a)

#create an uninitialized array of 3 elements
a=np.empty(3)
print(a)

l=[3,6,9,1]
a=np.array(l)
print(a)

l=[1,2.,4,7,9]
a=np.array(l)
print(a)

#Nested sequences

l1=[2,3,4]
l2=[1,2,3]
a=np.array((l1,l2))
print(a)
```

In [ ]:

```
#Creating similar arrays
a1 = np.array([1, 2, 3], dtype=np.float64)
print(a1)
print(a1.dtype)

a2 = np.array([1., 2, 3], dtype=np.int32)
print(a2,a2.dtype)

#Explicitly convert or cast an array from one dtype to another using ndarray’s astype m
a3=a2.astype(np.float64)
print(a3,a3.dtype)

numeric_strings = np.array(['1.25', '-9.6', '42'], dtype=np.string_)
numeric_strings.astype(float)
print(numeric_strings)

print(a3.astype(a2.dtype))
```

In [ ]:

```
#Numpy array attributes
a=np.random.randint(20,size=(2,4))
print(a)
print('ndim',a.ndim,'shape=',a.shape,a.size)
```

In [ ]:

```
#Arithmetic operations with NumPy arrays
a = np.array([[1., 2., 3.], [4., 5., 6.]])
print(a+a)
print(a*a)
print(1/a)
print(a-a)
print(a+10)
print(a*10)
```

In [ ]:

```
import numpy as np
a = np.array([[1., 2., 3.], [4., 5., 6.]])
b = np.array([[1., 2.], [3.,4.],[4., 5.]])

print(a@b)
print(np.dot(a,b))
print(np.matmul(a,b))
```

In [ ]:

```
#Basic Indexing and Slicing#
#if you assign a scalar value to a slice, as in arr[5:8] = 12, the value is
#propagated to the entire selection
# array slices are views on the original array

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

temp=a[0].copy()
a[0]=66
```

```
print(a)

a[0]=temp
print(a)
```

In [ ]:

```
#slicing
a=np.arange(1,20)
print(a)
print(a[3:8])

a2=a[3:8]

print(a,a2)

a2[:]=88
print(a2,a)

print("_____________")
a=np.array([[[1,2,3],[4,5,6]],
             [[7,8,9],[9,8,7]]])
print(a)
print('*****')
print(a[0])
print(a[0,1])
print(a[0,1,2])
print('*****')
print(a[1])

a=np.array([[1, 2, 3],
            [4, 5, 6],
            [7, 8, 9]])
print(a)
print(a[:2,:1])
print(a[1,:2])
print(a[2,:1])
###
b=a[:,1]
print("columns")
print(b,b.size, b.shape)
c=a[:,:2]
print(c,c.size, c.shape,c.ndim)
c=a[:,2]
print(c,c.size, c.shape,c.ndim)
d=a[:,2:]
print(d,d.size,d.shape,d.ndim)
```

In [ ]:

```
#Boolean Indexing

import numpy as np
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will'])
data = np.random.randn(5, 4)
print(names,names.dtype,data,data.dtype)

print(names == 'Bob')
print(data[names=='Bob'])

print(names != 'Bob')
```

```
mask = (names == 'Bob') | (names == 'Will')
print('mask=    ',mask)

print(data[data<0])
#set all of the negative values in data to 0 we need only do
data[data<0]=0
print(data)
#The boolean array must be of the same length as the axis it's indexing
```

In [ ]:

```

```

In [ ]:

```
#Fancy Indexing - describes indexing using integer arrays
a=np.empty((8,4))
for i in range(8):
    a[i]=i
print(a)
print('\n*******\n')
print(a[[4,1,6,0]])

#using negative indices
print(a[[-1,-3,-6]])

#Multiple index arrays - selects a one dimensional array of elements corresponding to e
#regardless of no of dimensions of the array, the result is one-dimensional
a= np.arange(32).reshape(8,4)

print(a)
print('\n\n -----')
print(a[[1,5,7,2],[0,3,1,2]])
print('/n******\n\n')
#slicing
a1=a[[1,5,7,2]][:,[0,3,1,2]]
print(a1)

#Fancy indexing, unlike slicing, always copies the data into a new array.
```

In [ ]:

```
#transposing arrays and swapping Axes
#transposing is a special form of reshaping that similarly returns a view on the underl
#Arrays have the transpose method and also the special T attribute. used for inner matr
import numpy as np
a=np.arange(15).reshape((3,5))
print('matrix= \n',a)

print('transpose=\n ',a.T)

a1=np.arange(18).reshape((3,6))
print('new mat= ',a1)
print('dot product\n')
print(np.dot(a1.T,a1))
print('product')
print(a1@a1.T)
```

In [ ]:

```
#Matrix operations
a=np.array([[1,2],[3,4],[5,6]])
print('insert operation axis=0  \n')
print(np.insert(a,1,[4,9],axis=0))
print('\ninsert operation axis=1 \n')
print(np.insert(a,1,[9],axis=1))
```

In [ ]:

```
#Matrix operations
a=np.array([[1,2],[3,4],[5,6]])
b=np.array([[11,22],[33,44],[55,66]])
c=a+b
d=np.append(a,b)
print(c,type(c),np.shape(c))
print(d, type(d),np.shape(d))
z=np.concatenate((a,b))
print(z)
```

#Universal functions - ufunc, is a function that performs element-wise operations on data in ndarrays #think of them as fast vectorized wrappers for simple functions that take one or more scalar #values and produce one or more scalar results

In [ ]:

```
#unary ufunc
a=np.arange(10)
print(a)
print(np.sqrt(a))

print(np.exp(a))

#binary ufunc
#np.maximum computes element wise maximum
#fmax ignores NaN
a=np.random.randn(8)
b=np.random.randn(8)
print(a)
print(b)
print(np.maximum(a,b))

#modf returns the fractional and integral parts of floating point array
a=np.random.randn(5)*5
print(a)
rem,whole=np.modf(a)
print(rem)
print(whole)
```

In [ ]:

```
#mathematical and statistical methods
#use aggregations like sum, mean and std either by calling the array instance method or

a=np.random.randn(5,4)
print(a.mean())
print(np.mean(a))
print(np.sum(a))

# mean and sum take an optyional axis argument that computes the statistic over the giv
a.mean(axis=1)
a.sum(axis=0)

a = np.array([1,0,2,-3,6,8,4,7])
b = np.array([[3,6],[4,2]])
print(a.max())
print(a.min())
print(a.sum())
print(a.mean())
print(b.mean(axis=0))
print(a.std())
print(b.std(axis=0))
```

In [ ]:

```
#mathematical and statistical methods
#use aggregations like sum, mean and std either by calling the array instance method or

a=np.random.randn(5,4)
print(a.mean())
print(np.mean(a))
print(np.sum(a))

# mean and sum take an optyional axis argument that computes the statistic over the giv
a.mean(axis=1)
a.sum(axis=0)
```

In [ ]:

In [ ]:

```
#Broadcasting
import numpy as np
a=np.array([[1,2],[3,4],[5,6]])
b=np.array([1,2,3,4])
print(a.shape,b.shape)
print(a+b)
```

In [ ]:

```
#Broadcasting
import numpy as np
a=np.array([[1,2],[3,4],[5,6]])
b=np.array([1,2,3,4])
c=7
d=np.array([8])
print(a.shape,b.shape,d.shape)
#print(a+b)
print(a+c)
print(a+d)
```

In [ ]:

```
#Broadcasting
import numpy as np
#a=np.array([[1,2],[3,4],[5,6]])
a=np.array([1,2,3,4])
#b=np.array([[1,2,3,4],[5,6,7,8],[2,4,6,8]])
#b=np.array([[1],[2],[3]])
b=np.array([[1],[2],[3],[4]])
print(a.ndim,b.ndim)
print(a.shape,b.shape)
print(a+b)
```

In [ ]:

```
a = np.array([1, 2, 3, 4, 5, 6])
np.save('filename', a)
```

```
b = np.load('filename.npy')
```

In [ ]:

```
#Save a NumPy array as a plain text file like a .csv or .txt file with np.savetxt.
a = np.array([1, 2, 3, 4, 5, 6, 7, 8])
np.savetxt('new_file.csv', a)
b=np.loadtxt('new_file.csv')
print(b,type(b))
```

In [ ]:

```
#Loading Arrays from Files
import numpy as np
sdata = np.loadtxt('student.txt', skiprows=1, delimiter=' ')
print(sdata)
```

In [ ]:

```
#Loading Arrays from Files
import numpy as np
sdata = np.loadtxt('st.txt', skiprows=1, delimiter=',')
print(sdata, type(sdata))
```

In [ ]:

```
#18/11/22


import numpy as np
a=np.array([1,2,3,4,5])
print(a)
print('\n No of dimensions = ',a.ndim,'\n',
       'shape =', a.shape,'\n',
       'size = ', a.size,'\n',
       'data type = ',a.dtype,'\n',
       'size of element = ',a.itemsize,'\n',
       'total size= ',a.nbytes)
for i in range(a.size):
    print(id(a[i]))
```

In [ ]:

```
import numpy as np
a=np.array([[1,2],[3,4],[5,6]])
for i in range(3):
    for j in range(2):
        print(id(a[i]))
```

In [ ]:

```
#splitting arrays
import numpy as np
a=np.arange(1,25).reshape(2,12)
print(a, a.ndim,a.size,a.shape)
print('\n  hsplit  \n')
#hsplit -  split into equally shaped arrays

print(np.hsplit(a,3))
print(np.hsplit(a,(3,4)))
```

In [ ]:

```
#splitting arrays
import numpy as np
```

```
a=np.arange(1,25).reshape(12,2)
print(a, a.ndim,a.size,a.shape)
print('\n  vsplit  \n')
#hsplit -  split into equally shaped arrays
print(np.vsplit(a,3))

print('\n  vsplit  \n')
print(np.vsplit(a,(3,4)))
```

In [ ]:

```
#stacking arrays
a1=np.array([[1,1],[2,2]])
a2=np.array([[3,3],[4,4]])
print(a1,a2)
print('\n vstack \n')
print(np.vstack((a1,a2)))
print('\n hstack \n')
print(np.hstack((a1,a2)))
```

In [ ]:

```
#sorting
a=np.array([10,6,3,8,2,1,7,5])
print(np.sort(a))
a=np.array([[1,4],[3,2]])
print(a)
print('\n  axis = None \n')

print(np.sort(a,axis=None))

print('\n  axis = 0 \n')
print(np.sort(a))

print('\n  axis = 1 \n')
b=np.array([[9,4],[13,2]])
print('\n array \n')
print(b,'\n')
print(np.sort(b,axis=1))
```

In [ ]:

```
#Reverse an array
a=[9,2,5,1,8,4]
print(np.flip(a))
#Reverse only rows or only columns
a=[[3,9,5],[6,2,4]]
print(np.flip(a))
print(np.flip(a,axis=0))
print(np.flip(a,axis=1))
```

In [ ]:

```
#flatten
a=np.array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])
b=a.flatten()

print(a)
print(b)
```

In [ ]:

```
#Searching arrays
#linear search
```

```
a=np.array([5,8,9,3,6,1,3,7])
b=np.where(a==3)
print(b)
print('\n', np.where(a>7))

#Searchsorted - performs a binary search and returns the index
#where the specified value would be inserted
a=[2,4,7,9,12,15,17]
b=np.searchsorted(a,7)
print(b)
b=np.searchsorted(a,7,side='right')
print(b)
```

In [ ]:

```
#how to make numpy array immutable
a=np.array([1,2,3,4,5])
a[0]=99
print(a)
a.setflags(write=False)
a[0]=88
print(a)
```

In [ ]:

```
#structured arrays
x=np.array([('Rama',12,6),('sita',10,7)],
           dtype=[('name','<U10'),('age','i4'),('cgpa','f4')])
print(x)
x
print(x['name'])
```

In [ ]: