Numpy Fundamentals

- **Date:** November 7, 2023
- **Author:** Engr Muhammad Saqlain
- **Institute:** Department of Computer System Engineering, IUB

```
# import numpy
import numpy as np
```

Arrays Indexing

Array attributes

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

```
x.shape
```

```
x.ndim
```

```
x.nbytes
```

```
x.dtype
```

Array operations

```
arr1 = np.arange(5)
arr1
```

```
arr1**0.5
```

```
arr2 = np.full((2,5),arr1)
arr2
```

```
arr3 = arr2
arr3
```

```
arr3[0,:] = 1 + arr2[0,:]
arr3
```

```
arr2
```

```
arr3 == arr2
```

Slicing

```
a = np.arange(8)
a
```

```
a[4:7]

a[4:7]=9

a

a_slice = a[4:7]
a_slice

a_slice[1] = 11
a_slice

a

a_slice[:] = 13

a

a_slice_copy = a_slice.copy()

a_slice_copy

a_slice_copy[:]=15

a

a_slice[:]=17

a

a_slice=19

a

a_slice_copy

a_slice
```

Slicing Example

```
a = np.array([np.arange(i, i + 6) for i in range(0, 60, 10)])
a

b = a[1:5,1:5]
b

b[:,:]=0
b

a
```

Slicing expressions

```
a = np.arange(80).reshape(8,10)
a

a.shape

a[3]    # row selection

a[3,:]

a[:,3]    # column selection

a[::2]  # every 2nd row

a[3::]   # from 3rd row onwards

a[:,::2]  # every 2nd column

a[:,3::] # 3rd column onwards
```

Indexing 3D arrays

```
a = np.arange(60).reshape(3,4,5)   # 3 blocks of 4x5 size
a

a[0,:,:] # block 0

a[0,...]

a[:,2,:] # 2nd row from each block

a[:,:,3] # every column is converted to a row

a[...,3]
```

3D array indexing example 2

```
import numpy as np
arr = np.arange(24).reshape(2,3,4)
arr

arr.shape

arr[:,:,:]

arr[:,:]

arr[:]

arr[:,]
```

Boolean indexing

```
names = np.array(['Ali', 'Kashif', 'Sohail', 'Anam', 'Saba', 'Rida'])
names

names == 'Saba'

names == 'Ali'

mask = (names=='Saba')|(names=='Ali')
mask

data = np.random.randn(6,3)
data

data[mask, :] # mask is applied on all columns

data[~mask, :]   # Slicing on the basis of False

data

data[data<0]=0

data
```

Integer lists as indices

```
arr = np.empty((6,4))
for i in range(6):
    arr[i]=i*i
arr

arr[[2, 0, 4, 2, 0]]

arr[[-4, -1, -4, -3]]  # row counting backwards
```

Differences between Python Lists and Numpy Arrays

```
a = [1, 2, 3]
a

b=a
b

a == b

a is b

c = a[:]    # copy
c

a is c

a==c
```

```
d = a.copy()
d
```

```
a is d
```

```
a == d
```

```
a
```

```
a[1:2] = [5,6]
a
```

```
a1 = np.arange(1,4)
a1
```

```
a1[1:2] = [5,6]
```

Broadcasting

```
a = np.arange(5).reshape(5,1)
a
```

```
a.shape
```

```
b = np.arange(6).reshape(1,6)
b
```

```
b.shape
```

```
a+b   # a will be broadcasted to 5,6 and b will be broadcasted tp 5,6
```

```
(a+b).shape
```

```
c=np.arange(6)
c
```

```
c.shape
```

```
a.shape
```

```
a+c  # c will be broadcasted to 5,6
```

```
d = 10
d
```

```
a+d  # d will be broadcasted to 1,1 and then 5,1
```

Broadcasting example

```
arr = np.arange(6).reshape(3,2)
arr
```

```
arr.shape
```

```
col_mean = arr.mean(0)
col_mean

col_mean.shape

col_demean = arr - col_mean
col_demean

col_demean.mean(0)
```

Expanding dimensions

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

x.shape

# We want to expand it to 1,3
y = np.expand_dims(x, axis=0)
y

y.shape

z=np.expand_dims(x, axis=1)
z

z.shape

y=np.expand_dims(x, axis=(0,1))
y

y.shape

z = np.expand_dims(x,axis=(1,0))
z

z.shape

y=np.expand_dims(x,axis=(0,2))
y

y.shape

y=np.expand_dims(x,axis=(1,2))
y

y.shape

y=np.expand_dims(x,axis=(0,2,3))
y

y.shape
```

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

x.shape

y=np.expand_dims(x,axis=0)
y

y.shape

y=np.expand_dims(x,axis=(0,3))
y

y.shape
```

Transpose

```
arr = np.arange(15).reshape(3,5)
arr

arr.shape

arr.T

# transpose of a vector
a = np.arange(6)
a

a.T

a.shape

a.T.shape

arr

arr.transpose(1,0)

x = np.arange(24).reshape(3,2,4)
x

x.transpose(0,1,2)

x.transpose(0,2,1)  # 2nd and 3rd dimensions swapped

x.shape

x

x.transpose(1,0,2)  # dimension 0 and 1 swapped

x.transpose(2,1,0)
```

Array manipulations

```
arr = np.arange(15).reshape(5,3)
arr

np.fliplr(arr)

np.flipud(arr)

np.rot90(arr)

arr

np.rot90(np.fliplr(arr))
```