Numpy Fundamentals

- **Date:** November 15, 2023
- **Author:** Engr Muhammad Saqlain
- **Institute:** Department of Computer System Engineering, IUB

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

Python Sorting

1D array sorting

```
a = np.random.randn(6)
a

a.sort()

a
```

Multi-dimensional array sorting

```
b=np.random.randn(5,3)
b

b.sort(0)   #column-wise sorting
b

b.sort(1)
b

c = np.random.randn(5,3)
c

c.sort()  # by default row-wise sorting
c
```

Slice sorting

```
a = np.random.randn(3,5)
a

# sort first column only
a[:,0].sort()
a

a_copy = a.copy()
a_copy
```

```
a[0,:].sort()
a

a_copy

a_copy[0].sort()
a_copy

a_copy==a
```

Sorting in reverse order

```
a = np.random.randn(3,5)
a

a.sort(axis=1) # sort row-wise
a

a[:,::-1]  # for all rows, start from last column
```

Numpy Sorting

np.sort() function

```
a = np.random.randn(6)
a

np.sort(a)

a  # a is not affected
```

argsort()

```
values = np.array([5,0,1,3,2])
values

indexer = values.argsort()
indexer

values[indexer]
```

Sorting set elements in 1D arrays

```
import numpy as np
ints = np.array([3,3,3,2,2,1,1,4,4])
ints

np.unique(ints)   # sorted unique elements
```

```
x = np.array([2,3,0,0,4,2,3,1])
y = np.array([1,1,0,2,2,2,3,5])
x

y

np.intersect1d(x,y)  # sorted common elements

np.union1d(x,y)  # sorted all elements of x and y

np.in1d(x,y)

np.in1d(y,x)

x

np.setdiff1d(x,y) # elements of x that are not in y

np.setdiff1d(y,x) # elements of x that are not in y

np.setxor1d(x,y) # elements that are either in x or y but not in both
```

Random numbers

```
np.random.randint(2,10,3)  # randomly generate 3 numbers between 2 and 10

np.random.randn(3)   # randomly generate 3 points from normal distribution u=0, sigma=1

np.random.standard_normal(3)  # alternate to above

np.random.rand(3) # uniform distribution [0,1]

np.random.uniform(0,1,3) # alternate to above but can change start and finish values

np.random.binomial(10,0.5,size=5) # a coin is tossed 10 times, probability of heads is 0.5, in first batch we received 3 heads
```

Using seeds

```
# Global seed
np.random.seed(42)
np.random.randint(2,10,3)

# class instnace of seed or local seed
rng = np.random.RandomState(42)
rng.randint(2,10,3)
```

Reshaping arrays

```
a = np.arange(15)
a

a.shape

x =a.reshape(5,-1) # five rows and remaining columns
x

x.shape

a.reshape(6,-1)

y=a.reshape(3,-1)
y

y.shape

y=y+1  # This is different from z+=1 which does not create another
memory ocation
y

# above does not affect a
a

z=a.reshape(5,-1)
z

z.shape

z+=1
z

# above affects a
a

z[0,0]=-100

z

a

a[1]=200
a

z
```

Reshaping in row or column vector

```
a = np.arange(1,8)
a

a.reshape(1,-1)
```

```
a.reshape(-1,1)

b = np.arange(8)
b

b.shape

c = np.expand_dims(b, axis=0)
c

c.shape

d = np.squeeze(c)
d

d.shape
```