NumPy

reference link : [https://cs231n.github.io/python-numpy-tutorial/#numpy (https://cs231n.github.io/python-numpy-tutorial/#numpy)](https://cs231n.github.io/python-numpy-tutorial/#numpy)

ARRAY CREATION

In [9]:

```
import numpy as np
a = np.array([1, 2, 3])
for i in range(3):
    print(a[i])
# shape of an np array is a tuple
print(f"Shape : {a.shape}")
```

```
1
2
3
Shape : (3,)
```

In [19]:

```
import numpy as np
a = np.array([[1, 2, 3], [4,5,6,7]], dtype='object')
print(f"Shape : {a.shape}")
for i in range(2):
    print(a[i])
    for j in range(3):
        print(a[i][j])
```

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

```
In [21]:
```

```
#some other types of standard matrices

import numpy as np

a = np.zeros((2,2))   # Create an array of all zeros

b = np.ones((1,2))    # Create an array of all ones

c = np.full((2,2), 7)  # Create a constant array

d = np.eye(2)         # Create a 2x2 identity matrix

e = np.random.random((2,2))  # Create an array filled with random values

print(a)
print(b)
print(c)
print(d)
print(e)
```

```
[[0. 0.]
 [0. 0.]]
[[1. 1.]]
[[7 7]
 [7 7]]
[[1. 0.]
 [0. 1.]]
[[0.19838158 0.43820667]
 [0.01598001 0.2706866 ]]
```

INDEXING

In [32]:

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

#Slicing
# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
b = a[:2, 1:3]
print(f"b:\n {b}")
c = a[1:3, :]
print(f"c:\n {c}")
print(type(c))
```

```
b:
 [[2 3]
 [6 7]]
c:
 [[ 5  6  7  8]
 [ 9 10 11 12]]
<class 'numpy.ndarray'>
```

In [36]:

```
#INTEGER ARRAY INDEXING - allows you to construct arbitrary arrays using the data from ano
ther array,
#                          unlike the previous case where the created array was a view of t
heoriginal
import numpy as np

a = np.array([[1,2], [3, 4], [5, 6]])
# Consider:
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))

# This can be written using Integer array indexing as:
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))
```

```
[1 4 5]
[1 4 5]
```

In [38]:

```
# Boolean array indexing - lets you pick out arbitrary elements of an array.
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
print(a>2)
print(a[a>2])
```

```
[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
```

In [41]:

```
# MATHEMATICAL OPERATIONS
# Basic mathematical functions operate elementwise on arrays

import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# Elementwise sum; both produce the array
# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))
# same holds for other operators
# * perfroms element wise multiplication, not dot product

#FOR DOT PRODUCT - either one of the below
print("\nDOT PRODUCT\n")
print(a.dot(b))
print(np.dot(a, b))
```

```
[[ 6.  8.]
 [10. 12.]]
[[ 6.  8.]
 [10. 12.]]

DOT PRODUCT

[[14 17]
 [30 37]
 [46 57]]
[[14 17]
 [30 37]
 [46 57]]
```

In [42]:

```
# AXIS WISE OPERATION (ROW, COL WISE)
import numpy as np

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

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # axis is 0 for column
print(np.sum(x, axis=1))  # axis is 1 for row

# transpose
print(x.T)
```

```
10
[4 6]
[3 7]
[[1 3]
 [2 4]]
```

In [ ]:

```

```