Numpy Fundamentals

- **Date:** November 5, 2023
- **Author:** Engr Muhammad Saqlain
- **Institute:** Department of Computer System Engineering, IUB

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

Arrays

Arrays are fundamental data structures used to represent and manipulate data in various forms, such as:

- Column Vector
- Row Vector
- Matrix
- 2D Array
- 3D Array

Creating Arrays

Creating arrays from a list

```
list1 = [1,2,3,4,5]
list1

arr1 = np.array(list1)
arr1

# finding shape
arr1.shape     # a tuple of 1 element

# finding data type
arr1.dtype

# finding number of dimensions
arr1.ndim

# Given list with string data
list2 = ["apple", "banana", "cherry", "date"]

# Create an array from the list
arr2 = np.array(list2)
```

```
# Print the resulting array
print(arr2)

arr2.shape

arr2.dtype    # Unicode characters of maximum length 6

arr2.ndim

# Using mixed data types
list3 = [1, 2.5, "apple", True, [3, 4]]
arr3 = np.array(list3)
print(my_arr3)

# If mixed data types, use dtypes = objects
# Using mixed data types
list3 = [1, 2.5, "apple", True, [3, 4]]
arr3 = np.array(list3, dtype=object)
print(arr3)

arr3.shape

arr3.dtype    # Object data type

arr3.ndim
```

Creating arrays from list of lists

```
list1 = [[1, 2, 3, 4], [5, 6, 7, 8]]
list1

arr1 = np.array(list1)
arr1

arr1.shape   # tuple of 2,4

arr1.dtype

arr1.ndim

list2 = [[1, 2, 3], [4, 5], [6, 7, 8, 9]]
arr2 = np.array(list2, dtype=object)
print(arr2)

arr2.shape

arr2.dtype

arr2.ndim
```

Referencing arrays

```
arr1
```

```
# Creating new reference for an existing array
arr3=arr1
arr3

# referring to a row
arr1[0]

# referring to elements
arr1[0][0]

# reference to elements
arr3[1,3]

arr1[:,2]  # referring to column

arr3 is arr1   # checking equality of arrays. True because both refer
to same object

arr3 is not arr2

arr3 == arr1

arr4 = np.array([[0,1,2,3,4,5],[6,7,8,9,10,11]])
arr4

# Creating two identical objects
arr5 = arr4.copy()
arr5

arr4 is arr5  # False because they refer to two different memory
locations

arr5 == arr4
```

Creating special arrays

```
np.zeros(3)

np.zeros((3,4))

np.ones(4)

np.ones((4,4))

np.full((3,5),8)

np.full((3,5),[1,2,3,4,5])

arr = np.array([0,1,2,3,4,5])
arr

arr.tolist()
```

```
arr6 = np.full((2,6), arr.tolist())
arr6

np.eye(3)

np.eye(5)

np.eye(5,k=1)

np.eye(5,k=2)

np.eye(5,k=-1)

np.empty((4,5,2))  # 3D array with 4 blocks of 5x2 each

range(4)

np.array(range(4))

np.array(range(4))

# Use arange
np.arange(4)
```

Creating arrays with sequences

```
np.arange(8)  # starts at 0

np.arange(2,8) # starts at 2 and ends at 7

np.arange(1,8,2) # step size 2

# for non-integer sequences
np.linspace(0,1,10)

np.linspace(-np.pi, np.pi, 20)

T=10
dt=0.1
t=np.linspace(0,T,int(T/dt+1))
t

np.logspace(1,5,5)
```

Creating Meshgrids

```
points1 = np.arange(0,5)
points1

points1.shape

xs, ys = np.meshgrid(points1, points1)
xs
```

```
ys

xs.shape

ys.shape

points2 = np.arange(10, 18)
points2

points2.shape

xs, ys = np.meshgrid(points1,points2)
xs

ys

xs.shape

ys.shape

x = np.array([-1, 0, 1])
x

y = np.array([-2,0,2])
y

X, Y = np.meshgrid(x,y)
X

Y

Z = X**2+Y**2
Z

import matplotlib.pyplot as plt

# Create a meshgrid
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)

# Define a function to plot (in this example, a simple quadratic
function)
Z = (X-Y)**2 #+ Y**2

# Create a 3D surface plot
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z)

# Label the axes
ax.set_xlabel('X')
ax.set_ylabel('Y')
```

```
ax.set_zlabel('Z')

# Show the plot
plt.show()
```

Creating array elements conditionally

```
a=np.array([[0,1,2,3],[14,15,16,17]])
a

b = np.resize(a,(10,2))
b

x = np.arange(1,6)
x

y=np.arange(11,16)
y

cond = np.array([True, False, True, True, False])
cond

z = np.where(cond,x,y)
z

x = np.random.randn(4,4)
x

x>0

y = np.where(x>0, 2, -2)
y

z = np.where(x>0, 2, x)
z
```

Creating arrays from a function of indices

```
f = lambda m,n: 2*m+n
f

a = np.fromfunction(f, (4,3))
a
```