Lecture 9

Random Numbers in NumPy

NumPy offers the random module to work with random numbers.

In [2]:
```
from numpy import random

x = random.randint(100)

print(x)
```

```
86
```

The random module's rand() method returns a random float between 0 and 1.

In [3]:
```
from numpy import random

x = random.rand()

print(x)
```

```
0.11613224081156859
```

Generate Random Array

In NumPy we work with arrays, and you can use the two methods from the above examples to make random arrays.

Integers

The randint() method takes a size parameter where you can specify the shape of an array.

In [4]:
```
from numpy import random

x=random.randint(100, size=(5))

print(x)
```

```
[48 33 45 84 50]
```

In [5]:
```
from numpy import random

x = random.randint(100, size=(3, 5))

print(x)
```

```
[[64 15 92 93 12]
 [20 95 86  9 60]
 [23 53 24 43 62]]
```

Floats

The rand() method also allows you to specify the shape of the array.

In [6]:
```
from numpy import random

x = random.rand(5)

print(x)
```

```
[0.35592263 0.0930416  0.00804877 0.42294096 0.42511472]
```

In [7]:
```
from numpy import random

x = random.rand(3, 5)

print(x)
```

```
[[0.58778487 0.06054826 0.74392998 0.54122775 0.59848055]
 [0.09540357 0.59056713 0.54620292 0.82977807 0.73034845]
 [0.3547389  0.46213795 0.71762793 0.11305376 0.87802588]]
```

Generate Random Number From Array

The choice() method allows you to generate a random value based on an array of values.

The choice() method takes an array as a parameter and randomly returns one of the values.

In [10]:
```
from numpy import random

x = random.choice([3, 5, 7, 9])

print(x)
```

```
5
```

The choice() method also allows you to return an array of values.

Add a size parameter to specify the shape of the array.

In [11]:
```
from numpy import random

x = random.choice([3, 5, 7, 9], size=(3, 5))

print(x)
```

```
[[5 7 9 9 3]
 [3 3 9 3 7]
 [9 9 9 9 5]]
```

NumPy Matrix Operations

Here are some of the basic matrix operations provided by NumPy: |function|Description| |:-|:-| |dot()| performs matrix multiplication| |transpose()| transposes a matrix| |linalg.inv()| calculates the inverse of a matrix| |linalg.det()| calculates the determinant of a matrix| |flatten()| transforms a matrix into 1D array|

Perform Matrix Multiplication in NumPy

We use the np.dot() function to perform multiplication between two matrices.

Let's see an example.

In [14]:
```
import numpy as np

# create two matrices
matrix1 = np.array([[1, 3],
                    [5, 7]])

matrix2 = np.array([[2, 6],
                    [4, 8]])

# calculate the dot product of the two matrices
result = np.dot(matrix1, matrix2)

print("matrix1 x matrix2: \n",result)
```

```
matrix1 x matrix2:
 [[14 30]
 [38 86]]
```

Note: We can only take a dot product of matrices when they have a common dimension size. For example, For A = (M x N) and B = (N x K) when we take a dot product of C = A . B the resulting matrix is of size C = (M x K).

Transpose NumPy Matrix

The transpose of a matrix is a new matrix that is obtained by exchanging the rows and columns. For 2x2 matrix,

Matrix:

a11 a12
a21 a22

Transposed Matrix:

a11 a21
a12 a22

In NumPy, we can obtain the transpose of a matrix using the np.transpose() function. For example,

In [15]:
```
import numpy as np

# create a matrix
matrix1 = np.array([[1, 3],
                    [5, 7]])

# get transpose of matrix1
result = np.transpose(matrix1)
```

```
print(result)
```

```
[[1 5]
 [3 7]]
```

Note: Alternatively, we can use the .T attribute to get the transpose of a matrix. For example, if we used matrix1.T in our previous example, the result would be the same.

Calculate Inverse of a Matrix in NumPy

In NumPy, we use the np.linalg.inv() function to calculate the inverse of the given matrix.

However, it is important to note that not all matrices have an inverse. Only square matrices that have a non-zero determinant have an inverse.

Now, let's use np.linalg.inv() to calculate the inverse of a square matrix.

In [16]:
```
import numpy as np

# create a 3x3 square matrix
matrix1 = np.array([[1, 3, 5],
                    [7, 9, 2],
                    [4, 6, 8]])

# find inverse of matrix1
result = np.linalg.inv(matrix1)

print(result)
```

```
[[-1.11111111 -0.11111111  0.72222222]
 [ 0.88888889  0.22222222 -0.61111111]
 [-0.11111111 -0.11111111  0.22222222]]
```

Note: If we try to find the inverse of a non-square matrix, we will get an error message: numpy.linalg.linalgerror: Last 2 dimensions of the array must be square

Find Determinant of a Matrix in NumPy

We can find the determinant of a square matrix using the np.linalg.det() function to calculate the determinant of the given matrix.

Suppose we have a 2x2 matrix A:

a b

c d

So, the determinant of a 2x2 matrix will be:

det(A) = ad - bc

where a, b, c, and d are the elements of the matrix.

Let's see an example.

In [17]:
```
import numpy as np

# create a matrix
matrix1 = np.array([[1, 2, 3],
                    [4, 5, 1],
                    [2, 3, 4]])

# find determinant of matrix1
result = np.linalg.det(matrix1)

print(result)
```

```
-4.999999999999999
```

Flatten Matrix in NumPy

Flattening a matrix simply means converting a matrix into a 1D array.

To flatten a matrix into a 1-D array we use the array.flatten() function. Let's see an example.

In [18]:
```
import numpy as np

# create a 2x3 matrix
matrix1 = np.array([[1, 2, 3],
                    [4, 5, 7]])

result = matrix1.flatten()
print("Flattened 2x3 matrix:", result)
```

```
Flattened 2x3 matrix: [1 2 3 4 5 7]
```

Common NumPy Array Functions

There are many NumPy array functions available but here are some of the most commonly used ones.

Array Operations	Functions
Array Creation Functions	np.array(), np.zeros(), np.ones(), np.eye(), etc.
Array Mathematical Functions	np.add(), np.subtract(), np.sqrt(), np.power(), etc.
Array Statistical Functions	np.median(), np.mean(), np.std(), and np.var().
Array Input and Output Functions	np.save(), np.load(), np.loadtxt(), etc.

NumPy Array Creation Functions

Array creation functions allow us to create new NumPy arrays. For example,

In [20]:
```
import numpy as np

# create an array using np.array()
array1 = np.array([1, 3, 5])
```

```
print("np.array():\n", array1)

# create an array filled with zeros using np.zeros()
array2 = np.zeros((3, 3))
print("\nnp.zeros():\n", array2)

# create an array filled with ones using np.ones()
array3 = np.ones((2, 4))
print("\nnp.ones():\n", array3)

# create an eye array
array4 = np.eye(4)
print("\nnp.eye():\n", array4)
```

```
np.array():
 [1 3 5]

np.zeros():
 [[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]

np.ones():
 [[1. 1. 1. 1.]
 [1. 1. 1. 1.]]

np.eye():
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

NumPy Array Mathematical Functions

In NumPy, there are tons of mathematical functions to perform on arrays. For example,

In [21]:
```
import numpy as np

# create two arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([4, 9, 16, 25, 36])

# add the two arrays element-wise
arr_sum = np.add(array1, array2)

# subtract the array2 from array1 element-wise
arr_diff = np.subtract(array1, array2)

# compute square root of array2 element-wise
arr_sqrt = np.sqrt(array2)


print("\nSum of arrays:\n", arr_sum)
print("\nDifference of arrays:\n", arr_diff)
print("\nSquare root of first array:\n", arr_sqrt)
```

```
Sum of arrays:
 [ 5 11 19 29 41]

Difference of arrays:
 [ -3  -7 -13 -21 -31]

Square root of first array:
 [2. 3. 4. 5. 6.]
```

NumPy Array Statistical Functions

NumPy provides us with various statistical functions to perform statistical data analysis.

These statistical functions are useful to find basic statistical concepts like mean, median, variance, etc. It is also used to find the maximum or the minimum element in an array.

Let's see an example.

In [22]:
```
import numpy as np

# create a numpy array
marks = np.array([76, 78, 81, 66, 85])

# compute the mean of marks
mean_marks = np.mean(marks)
print("Mean:",mean_marks)

# compute the median of marks
median_marks = np.median(marks)
print("Median:",median_marks)

# find the minimum and maximum marks
min_marks = np.min(marks)
print("Minimum marks:", min_marks)

max_marks = np.max(marks)
print("Maximum marks:", max_marks)
```

```
Mean: 77.2
Median: 78.0
Minimum marks: 66
Maximum marks: 85
```

NumPy Array Input/Output Functions

NumPy offers several input/output (I/O) functions for loading and saving data to and from files. For example,

In [23]:
```
import numpy as np

# create an array
array1 = np.array([[1, 3, 5], [2, 4, 6]])

# save the array to a text file
np.savetxt('data.txt', array1)
```

```
# load the data from the text file
loaded_data = np.loadtxt('data.txt')

# print the loaded data
print(loaded_data)
```

```
[[1. 3. 5.]
 [2. 4. 6.]]
```

In [ ]: