NumPy

NumPy is a Python library.

NumPy is used for working with arrays.

NumPy is short for "Numerical Python".

Installation of NumPy

If you have Python and PIP already installed on a system, then installation of NumPy is very easy.

Install it using this command:

C:\Users\Your Name>pip install numpy

Import NumPy

Once NumPy is installed, import it in your applications by adding the import keyword:

Create a NumPy ndarray Object

NumPy is used to work with arrays. The array object in NumPy is called ndarray.

We can create a NumPy ndarray object by using the array() function

```
import numpy as np

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

print(arr)

print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

Access Array Elements

Array indexing is the same as accessing an array element.

You can access an array element by referring to its index number.

The indexes in NumPy arrays start with 0, meaning that the first element has index 0, and the second has index 1 etc.

```
import numpy as np

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

print(arr[0])
```

NumPy Array Slicing

Slicing arrays Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [start:end].

Example1

Slice elements from index 1 to index 5 from the following array:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
```

```
[2 3 4 5]
```

Example2

Slice elements from index 4 to the end of the array:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[4:])
```

Example3

Slice elements from the beginning to index 4 (not included):

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[:4])
```

```
[1 2 3 4]
```

Example4

Negative Slicing Slice from the index 3 from the end to index 1 from the end:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[-3:-1])
```

```
[5 6]
```

Example5

STEP Use the step value to determine the step of the slicing:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5:2])
```

```
[2 4]
```

Example 6

Slicing 2-D Arrays From the second element, slice elements from index 1 to index 4 (not included):

```
import numpy as np
```

```
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(arr[1, 1:4])
```

```
[7 8 9]
```

Example7

From both elements, slice index 1 to index 4 (not included), this will return a 2-D array:

```
import numpy as np

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

print(arr[0:2, 1:4])
```

```
[[2 3 4]]
```

Checking the Data Type of an Array

The NumPy array object has a property called dtype that returns the data type of the array:

```
import numpy as np

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

print(arr.dtype)
```

```
int64
```

```
import numpy as np

arr = np.array(['apple', 'banana', 'cherry'])

print(arr.dtype)
```

```
<U6
```

COPY:

The copy SHOULD NOT be affected by the changes made to the original array.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42

print(arr)
print(x)
```

```
[42  2  3  4  5]
[1 2 3 4 5]
```

VIEW:

The view SHOULD be affected by the changes made to the original array.

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42

print(arr)
print(x)
```

```
[42  2  3  4  5]
[42  2  3  4  5]
```

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
x[0] = 31

print(arr)
print(x)
```

Shape of an Array

The shape of an array is the number of elements in each dimension.

```
import numpy as np

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

print(arr.shape)
```

```
(2, 4)
```

NumPy Joining Array

Joining means putting contents of two or more arrays in a single array.

```
import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```

NumPy Splitting Array

Splitting is reverse operation of Joining.

```
import numpy as np

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

newarr = np.array_split(arr, 3)

print(newarr)
```

Searching Arrays

You can search an array for a certain value, and return the indexes that get a match.

To search an array, use the where() method.

Example1

```
import numpy as np

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

x = np.where(arr == 4)

print(x)
```

```
(array([3, 5, 6]),)
```

Example2 Find the indexes where the values are even:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])

x = np.where(arr%1 == 0)

print(x)
```

```
(array([0, 1, 2, 3, 4, 5, 6, 7]),)
```

Example

Find the indexes where the values are odd:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])

x = np.where(arr%2 == 1)

print(x)
```

```
(array([0, 2, 4, 6]),)
```

Sorting Arrays

The NumPy ndarray object has a function called sort(), that will sort a specified array.

```
import numpy as np

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

print(np.sort(arr))
```

Random Numbers in NumPy

What is a Random Number? Random number does NOT mean a different number every time. Random means something that can not be predicted logically.

Generate Random Number NumPy offers the random module to work with random numbers.

```
from numpy import random

x = random.randint(100)

print(x)
```

```
72
```

Example2

Generate a random float from 0 to 1:

```
from numpy import random

x = random.rand()

print(x)
```

```
0.7839172400775848
```

Example3

Generate a 1-D array containing 5 random integers from 0 to 100:

```
from numpy import random
```

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

print(x)
```

Example4

Generate a 2-D array with 3 rows, each row containing 5 random integers from 0 to 100:

```
from numpy import random

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

print(x)
```

Example5 Generate a 1-D array containing 5 random floats:

```
from numpy import random

x = random.rand(5)

print(x)
```

```
[0.72250016 0.06230072 0.35584658 0.5670206  0.54500234]
```

Example Generate a 2-D array with 3 rows, each row containing 5 random numbers:

```
from numpy import random

x = random.rand(3, 5)

print(x)
```

Random Data Distribution

What is Data Distribution? Data Distribution is a list of all possible values, and how often each value occurs.

Such lists are important when working with statistics and data science.

Random Distribution A random distribution is a set of random numbers that follow a certain probability density function.

Probability Density Function: A function that describes a continuous probability. i.e. probability of all values in an array.

Example1

Generate a 1-D array containing 100 values, where each value has to be 3, 5, 7 or 9.

The probability for the value to be 3 is set to be 0.1

The probability for the value to be 5 is set to be 0.3

The probability for the value to be 7 is set to be 0.6

The probability for the value to be 9 is set to be 0

```
rom numpy import random

x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(1

print(x)
```

Normal (Gaussian) Distribution

Normal Distribution The Normal Distribution is one of the most important distributions.

It is also called the Gaussian Distribution after the German mathematician Carl Friedrich Gauss.

It fits the probability distribution of many events, eg. IQ Scores, Heartbeat etc.

Use the random.normal() method to get a Normal Data Distribution.

It has three parameters:

loc - (Mean) where the peak of the bell exists.

scale - (Standard Deviation) how flat the graph distribution should be.

size - The shape of the returned array.

Example1

Generate a random normal distribution of size 2x3:

```
from numpy import random
```

```
x = random.normal(size=(2, 3))

print(x)
```

```
[[ 0.27171102  0.24972395 -1.44649751]
 [-0.15025974 -0.76295156 -0.73433465]]
```

Example2

Generate a random normal distribution of size 2x3 with mean at 1 and standard deviation of 2:

```
from numpy import random

x = random.normal(loc=1, scale=2, size=(2, 3))

print(x)
```

```
[[-0.15084103  2.52862166 -1.44337486]
 [ 4.11796733 -0.6706324   0.82362355]]
```

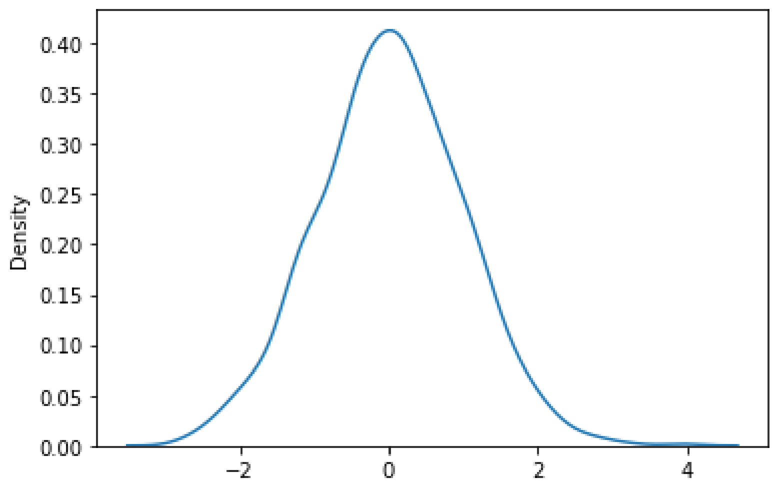
Visualization of Normal Distribution

```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.normal(size=1000), hist=False)

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning:
  warnings.warn(msg, FutureWarning)
```



✓ 0s completed at 9:01 PM