Numpy Randoms:-

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

Pseudo Random and True Random:-

Pseudo Random:-
Computers work on programs, and programs are definitive set of instructions. So it means there must be some algorithm to generate a random number as well.
If there is a program to generate random number it can be predicted, thus it is not truly random.
Random numbers generated through a generation algorithm are called pseudo random.

True Random:-
Can we make truly random numbers?
Yes. In order to generate a truly random number on our computers we need to get the random data from some outside source. This outside source is generally our keystrokes, mouse movements, data on network etc.
We do not need truly random numbers, unless its related to security (e.g. encryption keys) or the basis of application is the randomness (e.g. Digital roulette wheels).
,,
,,

Generate Random Number:-

NumPy offers the "random" module to work with random numbers.

Example:-

1)Generate a random integer from 0 to 100:

```
from numpy import random
x = random.randint(100)                                 ##randint for generating random integers
print(x)
```

,,,,,,,,,,,,,,o/p is [it will produce random integer with all posible values of integer 100]

2)Generate Random Float
The random module's "rand()" method returns a random float between 0 and 1.

Example
Generate a random float from 0 to 1:

```
from numpy import random
x = random.rand()
print(x)
```

,,,,,,,,,,,,,,,,,o/p is [it will produce random flot with all posible values of flot

number with 100 decimal value]
,,
,,

Generate Random Array:-

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

1)Integers

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

Example:-
*)enerate a 1-D array containing 5 random integers from 0 to 100:

```
from numpy import random
x=random.randint(100, size=(5))                 ###it will produce 5 random integers from 100 by chance  with equal chance to all 100
print(x)
```

Example:-
*)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))            ###it will produce 3 rows & 5 columns random integers from 100 integers by chance  with equal chance to all 100
print(x)
print(x)
```

2)Floats

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

Example:-
*)Generate a 1-D array containing 5 random floats:

```
from numpy import random
x = random.rand(5)
print(x)
```

*)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)
```

,,

Generate Random Number From Given 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.

Example
*)Return one of the values in an array:

```
from numpy import random
x = random.choice([3, 5, 7, 9])                         ###it will  genrate random number from specified array [3,5,7,9]
print(x)
```

The "choice()" method also allows you to return an array of values.
Add a size parameter to specify the shape of the array.

Example
*)Generate a 2-D array that consists of the values in the array parameter (3, 5, 7, and 9):

```
from numpy import random
x = random.choice([3, 5, 7, 9], size=(3, 5))           ###It will generate random number in 3 rows & 5 column from  array [3,5,7,9]
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.
The "random" module offer methods that returns randomly generated data distributions.

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.

*We can generate random numbers based on defined probabilities using the "choice()" method of the random module.

*The choice() method allows us to specify the probability for each value.

*The probability is set by a number between 0 and 1, where 0 (0% chance of occurence,it will never come in output) means that the value will never occur and 1 (100% chance of occurence,it will come always comes in input) means that the value will always occur.

Example
*)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   ##10% chance of occurence
The probability for the value to be 5 is set to be 0.3   ##30% chance of occurence
The probability for the value to be 7 is set to be 0.6   ##60% chance of occurence
The probability for the value to be 9 is set to be 0   ##0% chance of occurence, this is bcz summing of all probability should be 1 or 100%(i,e,, 10+30+60=100))

```
from numpy import random
x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(10))
##here 9 will never came bcz of 0% probability
print(x)
```
,,,,,,,,,,,,,,,,,,exact o/p can't predicted [5 3 3 5 5 3 7 5 7 7] this is for only understandind purpose if we execute again we got different result

note:-
The sum of all probability numbers should be 1. or 100%
Even if you run the example above 100 times, the value 9 will never occur.

*)You can return arrays of any shape and size (i,e,,rows & column) by specifying the shape in the size parameter.

Example
Same example as above, but return a 2-D array with 3 rows, each containing 5 values.

```
from numpy import random
x = random.choice([3, 5, 7, 9], p=[0.1, 0.3, 0.6, 0.0], size=(3, 5))
print(x)
```
,,,,,,,,,,,,,,,,,,,,,o/p while executing this program [[5 7 5 5 7]
[7 7 3 7 3]
[7 7 7 7 3]]
,,

Random Permutations:-

read the permutation & combination notebook file before starting this

Random Permutations of Elements:-

A permutation refers to an arrangement of elements. e.g. [3, 2, 1] is a permutation of [1, 2, 3] and vice-versa.

The NumPy "Random" module provides two methods for this: "shuffle()" and "permutation()".

1)Shuffling Arrays:-
        Shuffle means changing arrangement of elements in-place. i.e. in the array itself.
Example:-
Randomly shuffle elements of following array:

```
from numpy import random
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(random.shuffle(arr))                              ##it changes to original array
,,,,,,,,,,,,,,,,,,,,,,,,,,,,,o/p is [4 2 1 5 3],it is not same  for every time
```

The "shuffle()" method makes "changes to the original array".

2)Generating Permutation of Arrays
Example
Generate a random permutation of elements of following array:

```
from numpy import random
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(random.permutation(arr))                          ##it leaves original array un changed
,,,,,,,,,,,,,,,,,,,,,,,,,,,,o/p is [5 4 3 2 1],it is not same  for every time
```

The "permutation()" method returns a re-arranged array (and leaves the original array un-changed).

Application of Random Numbers:-
*Randomness has many uses in science, art, statistics, cryptography, gaming, gambling, and other fields.
*Random numbers are important for computer encryption, lotteries, scientific modelling, and gambling.
*Current methods of generating random numbers can produce predictable results.
*Most cryptographic applications require random numbers, for example: key generation. nonces.
*salts in certain signature schemes, including ECDSA, RSASSA-PSS(These are the Digital Signature).(In cryptography, a salt is random data that is used as an additional input to a one-way function that hashes data, a password or passphrase. Salts are used to safeguard passwords in storage. ... Cryptographic salts are broadly used in many modern computer systems, from Unix system credentials to Internet security.)
,,,

, ,