Python Random Module

Python has a built-in module that you can use to make random numbers.

The `random` module has a set of methods:

Method	Description
seed()	Initialize the random number generator
getstate()	Returns the current internal state of the random number generator
setstate()	Restores the internal state of the random number generator
getrandbits()	Returns a number representing the random bits
randrange()	Returns a random number between the given range
randint()	Returns a random number between the given range
choice()	Returns a random element from the given sequence
choices()	Returns a list with a random selection from the given sequence

shuffle()	Takes a sequence and returns the sequence in a random order
sample()	Returns a given sample of a sequence
random()	Returns a random float number between 0 and 1
uniform()	Returns a random float number between two given parameters
triangular()	Returns a random float number between two given parameters, you can also set a mode parameter to specify the midpoint between the two other parameters

Python Random seed() Method

Example

Set the seed value to 10 and see what happens:

```
import random

random.seed(10)
print(random.random())
```

Definition and Usage

The `seed()` method is used to initialize the random number generator.

The random number generator needs a number to start with (a seed value), to be able to generate a random number.

By default the random number generator uses the **current system time**.

Use the seed() method to customize the start number of the random number generator.

Note: If you use the same seed value twice you will get the same random number twice. See example below

Syntax

```
random.seed(a, version)
```

Parameter Values

Parameter	Description
a	Optional. The seed value needed to generate a random number. If it is an integer it is used directly, if not it has to be converted into an integer. Default value is None, and if None, the generator uses the current system time.
version	An integer specifying how to convert the a parameter into a integer. Default value is 2

More Examples

Example

Demonstrate that if you use the same seed value twice, you will get the same random number twice:

```
import random

random.seed(10)
print(random.random())

random.seed(10)
print(random.random())
```

Python
Random getstate() Method

Example[Get your own Python Server]

Return the current state of the random generator:

```
import random

print(random.getstate())
```

Definition and Usage

The `getstate()` method returns an object with the current state of the random number generator.

Use this method to capture the state, and use the `setstate()` method, with the captured state, to restore the state

Syntax

```
random.getstate()
```

Parameter Values

No parameter values

Python
Random setstate() Method

Example

Capture and restore the state of the random number generator:

```
import random

#print a random number:
print(random.random())

#capture the state:
state = random.getstate()

#print another random number:
print(random.random())

#restore the state:
random.setstate(state)

#and the next random number should be the same as when you captured the
state:
print(random.random())
```

Definition and Usage

The `setstate()` method is used to restore the state of the random number generator back to the specified state

Use the `getstate()` method to capture the state

Syntax

```
random.setstate(state)
```

Parameter Values

Parameter	Description
state	Required. A state object. the setstate() method will restore the state of the random number generator back to this sate.

Python
Random getrandbits() Method

Example Get your own Python Server

Return an 8 bits sized integer:

```
import random

print(random.getrandbits(8))
```

Definition and Usage

The `getrandbits()` method returns an integer in the specified size (in bits).

Syntax

```
random.getrandbits(n)
```

Parameter Values

Parameter	Description
n	Required. A number specifying the size, in bits, of the returned integer.

Python
Random randrange() Method

Example

Return a number between 3 and 9:

```
import random

print(random.randrange(3, 9))
```

Definition and Usage

The `randrange()` method returns a randomly selected element from the specified range.

Syntax

`random.randrange(start, stop, step)`

Parameter Values

Parameter	Description
start	Optional. An integer specifying at which position to start. Default 0
stop	Required. An integer specifying at which position to end.
step	Optional. An integer specifying the incrementation. Default 1

Python
Random randint() Method

Example

Return a number between 3 and 9 (both included):

```
import random

print(random.randint(3, 9))
```

Definition and Usage

The `randint()` method returns an integer number selected element from the specified range.

Note: This method is an alias for `randrange(start, stop+1)`.

Syntax

`random.randint(start, stop)`

Parameter Values

Parameter	Description
start	Required. An integer specifying at which position to start.
stop	Required. An integer specifying at which position to end.

Python
Random choice() Method

Example

Return a random element from a list:

```
import random

mylist = ["apple", "banana", "cherry"]

print(random.choice(mylist))
```

Definition and Usage

The `choice()` method returns a randomly selected element from the specified sequence.

The sequence can be a string, a range, a list, a tuple or any other kind of sequence.

Syntax

`random.choice(`*sequence*`)`

Parameter Values

Parameter	Description
sequence	Required. A sequence like a list, a tuple, a range of numbers etc.

More Examples

Example

Return a random character from a string:

```
import random

x = "WELCOME"

print(random.choice(x))
```

Python
Random choices() Method

Example

Return a list with 14 items.
The list should contain a randomly selection of the values from a specified list, and there should be 10 times higher possibility to select "apple" than the other two:

```
import random

mylist = ["apple", "banana", "cherry"]

print(random.choices(mylist, weights = [10, 1, 1], k = 14))
```

Definition and Usage

The `choices()` method returns a list with the randomly selected element from the specified sequence.

You can weigh the possibility of each result with the weights parameter or the cum_weights parameter.

The sequence can be a string, a range, a list, a tuple or any other kind of sequence.

Syntax

```
random.choices(sequence, weights=None, cum_weights=None, k=1)
```

Parameter Values

Parameter	Description
sequence	Required. A sequence like a list, a tuple, a range of numbers etc.
weights	Optional. A list were you can weigh the possibility for each value. Default None
cum_weights	Optional. A list were you can weigh the possibility for each value, only this time the possibility is accumulated. Example: normal weights list: [2, 1, 1] is the same as this cum_weights list; [2, 3, 4]. Default None
k	Optional. An integer defining the length of the returned list

---

Python
Random shuffle() Method

Example

Shuffle a list (reorganize the order of the list items):

```
import random

mylist = ["apple", "banana", "cherry"]
random.shuffle(mylist)

print(mylist)
```

Definition and Usage

The `shuffle()` method takes a sequence, like a list, and reorganize the order of the items.

Note: This method changes the original list, it does not return a new list.

Syntax

`random.shuffle(`*sequence*`)`

Parameter Values

Parameter	Description
sequence	Required. A sequence.
function	Deprecated since Python 3.9. Removed in Python 3.11. Optional. The name of a function that returns a number between 0.0 and 1.0. If not specified, the function `random()` will be used

More Examples

Example

This example uses the *function* parameter, which is deprecated since Python 3.9 and removed in Python 3.11.

You can define your own function to weigh or specify the result.

If the function returns the same number each time, the result will be in the same order each time:

```
import random

def myfunction():
  return 0.1

mylist = ["apple", "banana", "cherry"]
random.shuffle(mylist, myfunction)

print(mylist)
```

Python
Random sample() Method

Example[Get your own Python Server]

Return a list that contains any 2 of the items from a list:

```
import random

mylist = ["apple", "banana", "cherry"]

print(random.sample(mylist, k=2))
```

Definition and Usage

The `sample()` method returns a list with a randomly selection of a specified number of items from a sequnce.

Note: This method does not change the original sequence.

Syntax

```
random.sample(sequence, k)
```

Parameter Values

Parameter	Description
sequence	Required. A sequence. Can be any sequence: list, set, range etc.
k	Required. The size of the returned list

Python
Random random() Method

Example[Get your own Python Server]

Return random number between 0.0 and 1.0:

```
import random

print(random.random())
```

Definition and Usage

The `random()` method returns a random floating number between 0 and 1.

Syntax

```
random.random()
```

Parameter Values

No parameters

Python
Random uniform() Method

Example

Return a random number between, and included, 20 and 60:

```
import random

print(random.uniform(20, 60))
```

Definition and Usage

The `uniform()` method returns a random floating number between the two specified numbers (both included).

Syntax

```
random.uniform(a, b)
```

Parameter Values

Parameter	Description
a	Required. A number specifying the lowest possible outcome
b	Required. A number specifying the highest possible outcome

Python
Random triangular() Method

Example

Return a random number between, and included, 20 and 60, but most likely closer to 20:

```
import random

print(random.triangular(20, 60, 30))
```

Definition and Usage

The `triangular()` method returns a random floating number between the two specified numbers (both included), but you can also specify a third parameter, the `mode` parameter.

The `mode` parameter gives you the opportunity to weigh the possible outcome closer to one of the other two parameter values.

The `mode` parameter defaults to the midpoint between the two other parameter values, which will not weigh the possible outcome in any direction.

Syntax

```
random.triangular(low, high, mode)
```

Parameter Values

Parameter	Description
low	Optional. A number specifying the lowest possible outcome. Default 0
high	Optional. A number specifying the highest possible outcome. Default 1
mode	Optional. A number used to weigh the result in any direction. Default the midpoint between the *low* and *high* values