Python → Math → Random module

# Theory: Random module

16 minutes 10 / 10 problems solved

Start practicing

6825 users solved this topic. Latest completion was about 1 hour ago.

Sometimes it happens that we lack data and need to make up a bunch of new examples rather quickly. Of course, you can spend some time writing those examples yourself, but it's not so efficient, right? It would make more sense to shift the responsibility to your computer, namely, the Python's built-in module **random**. In this module, a random search is used to generate elements and is performed using an algorithm whose starting point is a **seed**. Therefore, the results given aren't random at all and, technically, this module should have been called **pseudo-random**. Nevertheless, it may be useful for a large number of applications, such as modeling and simulation.

## §1. Random method: first steps

First of all, we need to import the module:

```
import random
```

After we've managed to do the previous task, it's possible to try `random.random()` function that will provide us with a pseudo-random number from 0 to 1:

```
print(random.random())  # 0.5557276751294531
```

We can also control the pseudo-random behavior by specifying the seed manually, i.e. configure the new sequence of pseudo-random numbers using `random.seed(x)` function. You can set your own number or omit the optional argument x and consequently current system time would be used by default.

```
random.seed()
print(random.random())  # 0.956177930864557
```

Now try to set the x argument. Haven't you noticed the change of the result? If you choose `5`, you'll get `0.6229016948897019` as a result, if `20` – `0.9056396761745207`, etc. Thus, the seed controls the behavior of pseudo-random in Python and can be used with any other function of the random module.

## §2. Random basic functions

Moving forward, other useful functions are:

- `random.uniform(a, b)` – returns a pseudo-random float number in the range between a and b:

```
print(random.uniform(3, 100))  # 35.94079523197162
```

- `random.randint(a, b)` – returns a pseudo-random integer number in the range between a and b:

```
print(random.randint(35, 53))  # 52
```

- `random.choice(seq)` – returns pseudo-random element from *non-empty* sequences:

```
print(random.choice('Voldemort'))  # m
```

- `random.randrange(a, b, c)` – returns a pseudo-random number from a range between a and b with a step c. Just like with the `range()` function, the *start* and *step* arguments may be omitted with the default values 0 and 1 respectively. It means that the function can take one, two, or three parameters:

Current topic:

✓ Random module Stage 4 4★

Topic depends on:

✓ List Stage 1 13★

✓ Load module Stage 1 5★

Table of contents:

```
print(random.randrange(3, 100, 5))  # 18
print(random.randrange(1, 5))       # 2
print(random.randrange(100))        # 44
```

- `random.shuffle(seq, [random])` – shuffles a sequence. Attention: it doesn't work with immutable datatypes!

```
tiny_list = ['a', 'apple', 'b', 'banana', 'c', 'cat']
random.shuffle(tiny_list)
print(tiny_list)  # ['apple', 'banana', 'a', 'cat', 'b', 'c']
```

- `random.sample(population, k)` – returns a pseudo-random k length list from a population sequence. This function is used for random sampling without replacement:

```
print(random.sample(range(100), 3))  # [24, 33, 91]
```

Furthermore, there are plenty of other functions that are used in common mathematical practice, e.g. `random.gammavariate(alpha, beta)` that is used for gamma distribution or `random.gauss(mu, sigma)` that returns Gaussian distribution. If you need such narrow-specialized function, you can address the Python documentation.

The pseudo-random generators of the `random` module should *NOT* be used for security purposes. If you are intending to work with passwords, security tokens and other sensitive data, check out the secrets module. It's considered more reliable since it generates secure random numbers.

Report a typo

**489** users liked this theory. **12** didn't like it. **What about you?**

Start practicing

Comments (20) Hints (2) Useful links (0) Show discussion