# random模块
import random

- random　random.ranndom() 随机生成0-1之间的随机浮点数
- uniform　random.uniform(a,b)　随机生成a到b内的随机小数
- randint　random.randint(a,b) 返回a-b之间的随机数，既包括a也包括b
- randrange　random.randrange(a,b) 返回a-b之间的 随机数包括a不包括b
- choise　random.choise(可迭代对象) 随机选择一个元素
- choises　random.choises(可迭代对象) 随机选择一个元素并存入列表返回
- sample　random.sample(可迭代对象，个数）随机从可迭代对象中选取指定个数元素，并存入列表中
- shuffle　random.shuffle(列表名) 将列表中元素随机打乱

```python
import random

print(random.random())  # 返回0-1之间的浮点数（不包括1包括0）

print(random.uniform(1, 6))  # 返回1-6之间的小数（不包括6包括1）

print(random.randrange(1, 10))  # 返回1-10之间的一个随机数，不包括10

print(random.randint(1, 10))  # 返回1-10之间的随机数，包括10

print(random.random())  # 生成0-1的随机数，随机浮点数

print(random.choice('hello word!'))  # 随机选取字符串中的一个字符
print(random.choices('hello word!'))  # 随机选取字符串中的一个字符，以列表形式返回

print(random.sample('hello word', 3))  # 从多个字符中选出特定数量的字符，以列表形式返回

lis = [1, 2, 3, 4, 5, 6, 7]
random.shuffle(lis)  # 洗牌，将列表内容随机打乱
print(lis)

import string

# 生成随机字符串
s = ''.join(random.sample(string.ascii_lowercase + string.digits, 6))
print(s)

结果:
0.30321223999484015
4.532463664001345
5
5
0.9902494775660357
h
[' ']
['h', 'e', 'd']
[4, 6, 1, 3, 2, 5, 7]
ibya08
```

```python
def random(self): # real signature unknown; restored from __doc__
    """ random() -> x in the interval [0, 1). """
    pass
```

```python
def uniform(self, a, b):
    "Get a random number in the range [a, b) or [a, b] depending on rounding."
    return a + (b-a) * self.random()
```

```python
def randint(self, a, b):
    """Return random integer in range [a, b], including both end
points.
    """
```

```python
def randrange(self, start, stop=None, step=1, _int=int):
    """Choose a random item from range(start, stop[, step]).

    This fixes the problem with randint() which includes the
    endpoint; in Python this is usually not what you want.

    """
```

```python
def choice(self, seq):
    """Choose a random element from a non-empty sequence."""
```

```python
def choices(self, population, weights=None, *, cum_weights=None, k=1):
    """Return a k sized list of population elements chosen with replacement.

    If the relative weights or cumulative weights are not specified,
    the selections are made with equal probability.

    """
```

```python
def sample(self, population, k):
    """Chooses k unique random elements from a population sequence or set.

    Returns a new list containing elements from the population while
    leaving the original population unchanged.  The resulting list is
    in selection order so that all sub-slices will also be valid random
    samples.  This allows raffle winners (the sample) to be partitioned
    into grand prize and second place winners (the subslices).

    Members of the population need not be hashable or unique.  If the
    population contains repeats, then each occurrence is a possible
    selection in the sample.

    To choose a sample in a range of integers, use range as an argument.
    This is especially fast and space efficient for sampling from a
    large population:
```

```python
def shuffle(self, x, random=None):
    """Shuffle list x in place, and return None.

    Optional argument random is a 0-argument function returning a
    random float in [0.0, 1.0); if it is the default None, the
    standard random.random will be used.

    """
```