3.11 随机选择¶

问题¶

你想从一个序列中随机抽取若干元素，或者想生成几个随机数。

解决方案¶

`random` 模块有大量的函数用来产生随机数和随机选择元素。 比如，要想从一个序列中随机的抽取一个元素，可以使用 `random.choice()`：

```
>>> import random
>>> values = [1, 2, 3, 4, 5, 6]
>>> random.choice(values)
2
>>> random.choice(values)
3
>>> random.choice(values)
1
>>> random.choice(values)
4
>>> random.choice(values)
6
>>>
```

为了提取出N个不同元素的样本用来做进一步的操作，可以使用 `random.sample()`：

```
>>> random.sample(values, 2)
[6, 2]
>>> random.sample(values, 2)
[4, 3]
>>> random.sample(values, 3)
[4, 3, 1]
>>> random.sample(values, 3)
[5, 4, 1]
>>>
```

如果你仅仅只是想打乱序列中元素的顺序，可以使用 `random.shuffle()`：

```
>>> random.shuffle(values)
>>> values
[2, 4, 6, 5, 3, 1]
>>> random.shuffle(values)
>>> values
[3, 5, 2, 1, 6, 4]
>>>
```

生成随机整数，请使用 `random.randint()`：

```
>>> random.randint(0,10)
2
>>> random.randint(0,10)
5
>>> random.randint(0,10)
0
>>> random.randint(0,10)
7
>>> random.randint(0,10)
10
>>> random.randint(0,10)
3
>>>
```

为了生成0到1范围内均匀分布的浮点数，使用 `random.random()`：

```
>>> random.random()
0.9406677561675867
>>> random.random()
0.133129581343897
>>> random.random()
0.4144991136919316
>>>
```

如果要获取N位随机位(二进制)的整数，使用 `random.getrandbits()`：

```
>>> random.getrandbits(200)
335837000776573622800628485064121869519521710558559406913275
>>>
```

讨论¶

`random` 模块使用 *Mersenne Twister* 算法来计算生成随机数。这是一个确定性算法， 但是你可以通过 `random.seed()` 函数修改初始化种子。比如：

```
random.seed() # Seed based on system time or os.urandom()
random.seed(12345) # Seed based on integer given
random.seed(b'bytedata') # Seed based on byte data
```

除了上述介绍的功能，random模块还包含基于均匀分布、高斯分布和其他分布的随机数生成函数。 比如， `random.uniform()` 计算均匀分布随机数， `random.gauss()` 计算正态分布随机数。 对于其他的分布情况请参考在线文档。

在 `random` 模块中的函数不应该用在和密码学相关的程序中。 如果你确实需要类似的功能，可以使用ssl模块中相应的函数。 比如， `ssl.RAND_bytes()` 可以用来生成一个安全的随机字节序列。