26. Random một số trong python

Random và “secrets” module, bộ tạo số giả ngẫu nhiên mặc định của random module được thiết kế với trọng tâm vào mô phỏng chứ không phải về bảo mật. Vì vậy, bạn không nên tạo ra các thông tin nhạy cảm như mật khẩu, secure tokens, khóa phiên và những thứ tương tự bằng cách sử dụng các hàm ngẫu nhiên đó, ví dụ các hàm cơ bản như "randint", "random", "choice" và những hàm tương tự. Có một ngoại lệ và tôi sẽ đề cập trong đoạn tiếp theo: SystemRandom.

Cảnh báo trong tài liệu python mà tôi muốn trích dẫn ở đây là: “Warning

The pseudo-random generators of this module should not be used for security purposes. For security or cryptographic uses, see the secrets module.”

Lớp SystemRandom cung cấp một cách thích hợp để vượt qua vấn đề bảo mật này. Các phương thức của lớp này sử dụng một bộ tạo số ngẫu nhiên xen kẽ, sử dụng các công cụ được cung cấp bởi hệ điều hành (như /dev/urandom trên Unix hoặc CryptGenRandom trên Windows.

Mối quan tâm lớn mà các nhà phát triển Python vô tình có thể làm cho lỗi bảo mật nghiêm trọng, mặc dù các cảnh báo được chỉ rõ trong documentation, Python 3.6 đi kèm với một mô-đun mới "secrets" với một CSPRNG (dùng Pseudo Random Number Generator).

Hãy bắt đầu bằng việc tạo các số float ngẫu nhiên với hàm random của random module. Hãy nhớ rằng nó không nên được sử dụng để tạo ra các thông tin nhạy cảm:

Trong bài viết này tôi sẽ dùng giao diện terminal(CMD).

```
>>> import random
>>> random_number = random.random()
>>> print(random_number)
0.698359573093
```

Chúng ta sẽ đưa ra một cách tiếp cận khác và an toàn trong ví dụ sau, khi đó chúng ta sẽ sử dụng lớp SystemRandom của random module. Nó sẽ sử dụng một bộ tạo số ngẫu nhiên khác, được cung cấp bởi hệ điều hành. Đây sẽ là /dev/urandom trên Unix và CryptGenRandom trên window. Phương pháp random của lớp SystemRandom tạo ra một số float trong khoảng từ [0,1.0).

```
>>> from random import SystemRandom
>>> crypto = SystemRandom()
>>> print(crypto.random())
0.741553529075
>>>
```

Sinh random một list các số trong python

Không quan tâm đến vấn đề security ta có thể sử dụng random trong random package. Để có nhiều tùy chọn, tôi sẽ minh họa với hàm random trong thư viện numpy. Cú pháp như sau:

numpy.random.randint (low, high=None, size=None, dtype=’l’)

Trong đó:

Size là shape của mảng kết quả có kiểu int hoặc tuple of ints. Default là None tương ứng với kết quả trả về có 1 phần tử.

Dtype là kiểu của kết quả trả về (ví dụ: ‘int64’, ‘int’,…). Giá trị default là ‘np.int’

Hàm trên sẽ trả về một list các số nguyên random trong nửa khoảng [low,high) có chiều là size. Nếu higt là None thì khoảng random là [0, low).

Các bạn có thể theo dõi ví dụ bên dưới để hiểu rõ hơn.

```
>>> import numpy as np
>>> np.random.random(10)
array([ 0.38512197,  0.01572542,  0.25035755,  0.53453149,  0.60234844,
        0.10175073,  0.44585687,  0.84354704,  0.01282373,  0.87865167])
>>> #sinh danh sach so nguyen trong khoang nho hon 100
...
>>> np.random.randint(100,size=10)
array([86, 17,  2, 77,  9, 20, 68,  0, 81, 53])
>>> np.random.randint(100,size=(10,2))
array([[34, 94],
       [ 9, 85],
       [51, 32],
       [58, 90],
       [20, 93],
       [30, 86],
       [ 2, 95],
       [10, 42],
       [75, 64],
```

```
       [74, 22]])
```

Khi quan tâm đến vấn đề security ta cần gọi liên tục SystemRandom().

```
>>> list([random.SystemRandom().random() for i in range(10)])
[0.10045004100519428, 0.8105229864919358, 0.5116013716283684, 0.6125839506294763, 0.4936838066464031, 0.7739209793710309, 0.5981480525512537, 0.9298128234257795, 0.6
>>>
```

Sinh ra một danh sách các số ngẫu nhiên thỏa mãn điệu kiện tổng

Ví dụ như tổng các số đó bằng 1 như sau:

```
>>> x = np.random.random(100)
>>> sum_of_x = sum(x)
>>> x = x/sum_of_x
>>> x
array([ 0.00681564,  0.00046032,  0.01555603,  0.01224856,  0.00824919,
        0.01844947,  0.00083868,  0.0187569 ,  0.01903362,  0.00095173,
        0.01801133,  0.01398661,  0.00564896,  0.00453737,  0.00762683,
        0.00639701,  0.01631088,  0.017837  ,  0.01903103,  0.01525394,
        0.00484442,  0.00264082,  0.01540946,  0.00562404,  0.00364663,
        0.01544763,  0.01822115,  0.00781301,  0.00603914,  0.00109898,
        0.00931398,  0.00764734,  0.00931861,  0.01629682,  0.01827744,
        0.01945222,  0.01059581,  0.00463923,  0.00955128,  0.00209377,
        0.00388371,  0.00695361,  0.00980641,  0.01647755,  0.01483753,
        0.00036168,  0.00897731,  0.00761304,  0.00397424,  0.01893571,
        0.00254276,  0.00374991,  0.01700657,  0.00504006,  0.00858757,
        0.01841522,  0.015517  ,  0.01980629,  0.01366108,  0.00049385,
        0.01395694,  0.00255102,  0.00427743,  0.01510116,  0.01188987,
        0.01068046,  0.00502589,  0.00328158,  0.01415927,  0.00665739,
        0.01296565,  0.0183652 ,  0.00731404,  0.00619696,  0.01836427,
        0.00142202,  0.01575556,  0.00044663,  0.01250287,  0.00770227,
        0.01655119,  0.00016539,  0.01225387,  0.01818291,  0.00024227,
        0.01566063,  0.01052128,  0.01125321,  0.01673902,  0.01606259,
        0.00946126,  0.01469521,  0.00483952,  0.01250304,  0.00371621,
        0.002107  ,  0.0114111 ,  0.0089728 ,  0.00191371,  0.01351734])
>>> sum(x)
1.0000000000000004
>>>
```

Lựa chọn ngẫu nhiên các phần tử trong một list

Chúng ta có thể chọn một ký tự ngẫu nhiên từ một chuỗi hoặc một phần tử ngẫu nhiên từ danh sách hoặc một tuple, chúng ta có thể thấy trong các ví dụ sau đây. Ví dụ: bạn muốn có một chuyến đi trong thành phố ở Châu Âu và bạn không thể quyết định đi đâu? Hãy để Python giúp bạn:

```
>>> from random import choice
>>> possible_destinations = ["Berlin", "Hamburg", "Munich",
...                          "Amsterdam", "London", "Paris",
...                          "Zurich", "Heidelberg", "Strasbourg",
...                          "Augsburg", "Milan", "Rome"]
>>> print(choice(possible_destinations))
Zurich
```

Hàm choice() của random package thuộc mô đun numpy là thuận tiện hơn, bởi vì nó cung cấp thêm nhiều options.

Khi gọi hàm mặc định, tức là không có các tham số bổ sung được sử dụng, hoạt động giống như choice() của mô-đun ngẫu nhiên:

```
>>> from numpy.random import choice
>>> print(choice(possible_destinations))
Rome
```

Dùng tham số size cho phép chúng ta có thể chọn ra ngẫu nhiên một numpy.ndarray từ tập giá trị có shape mong muốn.

```
>>> print(choice(possible_destinations,size=(3,3)))
```

```
[['Zurich' 'Paris' 'Hamburg']

 ['Amsterdam' 'Hamburg' 'Strasbourg']

 ['Milan' 'Milan' 'Rome']]

>>>
```

Tuy nhiên bạn có thể nhận thấy, nếu chỉ dùng tham số “size” thì các phần tử của numpy.ndarray trả về có thể bị lập lại như ví dụ phía trên (giá trị 'Hamburg' bị trả về hai lần), và có một tham số có thể giúp giải quyết vấn đề đó. Sử dụng replace=False, như ví dụ dưới đây:

```
>>> print(choice(possible_destinations,size=(3,3),replace=False))
[['Strasbourg' 'Hamburg' 'London']
 ['Paris' 'Rome' 'Augsburg']
 ['Heidelberg' 'Zurich' 'Amsterdam']]
>>>
```

MORE POWERFUL!!!!

Tạo mẫu (sample) ngẫu nhiên trong python

Một mẫu có thể được hiểu là một phần đại diện từ một nhóm lớn hơn, thường được gọi là "population".
Mô-đun numpy.random chứa một hàm random_sample(), nó trả về các float ngẫu nhiên trong khoảng mở nửa [0.0, 1.0). Kết quả tuân theo phân bố đều trong khoảng đã nêu. Hàm này chỉ lấy một tham số "size", định nghĩa shape đầu ra. ví dụ, nếu chúng ta đặt size = (3, 4),chúng ta sẽ nhận được một mảng có shape (3, 4) là các phần tử ngẫu nhiên:

```
>>> import numpy as np
>>> x = np.random.random_sample((3, 4))
>>> print(x)
[[ 0.13138299  0.24141636  0.86548418  0.17989193]
 [ 0.30837552  0.06581604  0.28407039  0.43604533]
 [ 0.07316762  0.15726525  0.42304019  0.00151701]]
```

Chúng ta cũng có thể sample mẫu từ tập gốc (
population) qua hàm sample với cú pháp sau: sample(population, k). Nó sẽ tạo một list, chứa “k” phần tử của tập “population”. Ví dụ:

```
>>> import random

>>> random.sample(range(100),10)

[45, 84, 32, 97, 94, 83, 56, 18, 39, 92]

>>>
```

Random Seed

Một random seed, còn được gọi là "seed state", hoặc chỉ là "seed" - là một số được sử dụng để khởi tạo một bộ sinh giả mã ngẫu nhiên. Khi chúng ta gọi random.random(), chúng ta đã mong đợi và có một số ngẫu nhiên giữa 0 và 1. random.random() tính một số ngẫu nhiên mới bằng cách sử dụng số ngẫu nhiên được sinh ra trước đó. Lần đầu tiên chúng ta sử dụng ngẫu nhiên trong chương trình của chúng ta là gì? Vâng, không có số ngẫu nhiên đã tạo trước đó. Nếu một máy bộ sinh ngẫu nhiên được gọi cho lần đầu tiên, nó sẽ phải tạo ra một số "ngẫu nhiên" đầu tiên.

Nếu chúng ta gieobộ sinh số ngẫu nhiên, chúng ta cung cấp một giá trị trước giá trị đầu tiên. Giá trị seed tương ứng với một dãy các giá trị được tạo ra cho một bộ tạo số ngẫu nhiên nhất định. Nếu bạn sử dụng cùng một giá trị seed một lần nữa, bạn sẽ có được hay nhận được cùng một chuỗi số một lần nữa.

Bản thân seed, nó không cần phải được chọn ngẫu nhiên sao cho thuật toán tạo ra các giá trị theo sau một phân bố xác suất theo cách ngẫu nhiên. Tuy nhiên, seed lại là vấn đề ở kía cạnh security. Nếu bạn biết hạt giống, ví dụ bạn có thể tạo ra khóa mật mã bí mật dựa trên seed này.

Các seed ngẫu nhiên được tạo ra từ trạng thái của hệ thống máy tính trong ngôn ngữ lập trình,nhiều trường hợp lại chính là thời gian hệ thống.

Điều này đúng với Python. Trợ giúp về random.seed nói rằng nếu bạn gọi hàm với đối số “none” hoặc không có đối số nó sẽ sinh seed "từ thời điểm hiện tại hoặc từ một nguồn ngẫu nhiên riêng biệt của hệ điều hành nếu có."

```
>>> import random
>>> help(random.seed)
Help on method seed in module random:

seed(self, a=None) method of random.Random instance
    Initialize internal state from hashable object.


    None or no argument seeds from current time or from an operating
    system specific randomness source if available.


    If a is not None or an int or long, hash(a) is used instead.

>>>
```

Các chức năng seed cho phép bạn nhận được một chuỗi xác định các số ngẫu nhiên.
Bạn có thể lặp lại chuỗi này, bất cứ khi nào bạn cần nó, ví dụ: Cho mục đích gỡ lỗi.

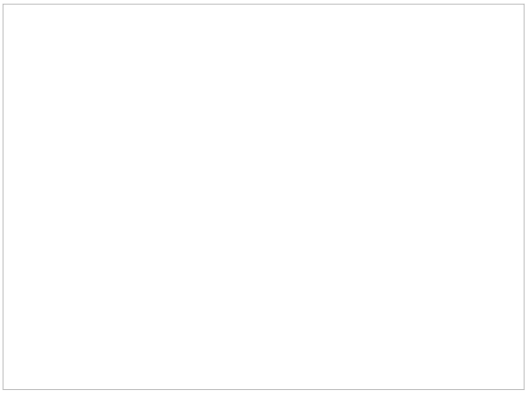
```
>>> import random
>>> random.seed(42)
>>> [random.random() for i in range(10)]
[0.6394267984578837, 0.025010755222666936, 0.27502931836911926, 0.22321073814882275, 0.7364712141640124, 0.6766994874229113, 0.8921795677048454, 0.08693883262941615
>>> #su dung cung seed = 42 de sinh ra mot danh sach 10 so
...
>>> random.seed(42)
>>> [random.random() for i in range(10)]
[0.6394267984578837, 0.025010755222666936, 0.27502931836911926, 0.22321073814882275, 0.7364712141640124, 0.6766994874229113, 0.8921795677048454, 0.08693883262941615
>>>
```

Sinh số ngẫu nhiên tuân theo phân bố xác suất Gaussian and Normalvariate Distribution

Ví dụ: Chúng ta muốn tạo 1000 số ngẫu nhiên giữa 130 và 230 có phân bố gaussian với giá trị trung bình mu thiết lập là 180 và độ lệch chuẩn sigma là 30.

```
>>> from random import gauss
>>> n = 1000
>>> values = []
>>> frequencies = {}
>>> while len(values) < n:
...     value = gauss(180, 30)
...     if 130 < value < 230:
...         frequencies[int(value)] = frequencies.get(int(value), 0) + 1
...         values.append(value)
...
>>> print(values[:10])
[186.96893210720162, 214.90676059797428, 199.69909520396007, 183.31521532331496, 157.85035192965537, 149.56012897536849, 187.39026585633607, 219.33242481612143, 181.
>>> # visualization of the result by graph

  ...
>>> import matplotlib.pyplot as plt
>>> freq = list(frequencies.items())
>>> freq.sort()
>>> plt.plot(*list(zip(*freq)))
[<matplotlib.lines.Line2D object at 0x0000000004AD4C88>]
>>> plt.show()
>>>
```



Với bài toán phía trên hãy sử dụng normalvariate để giải quyết, phần các bạn tự hoàn thành.

Kết luận

Như vậy chúng tôi đã giới thiệu khá chi tiết về module Random trong Python. Module Random có chức năng chính là sinh ra các số ngẫu nhiên phục vụ cho việc mô phỏng. Nếu bạn quan tâm đến vấn đề bảo mật thì nên sử dụng lớp SystemRandom của random module. Một số phương thức cơ bản, thường dùng trong Random là randint() – tạo các số nguyên ngẫu nhiên; random() – tạo các số float ngẫu nhiên; sample() – tạo mẫu ngẫu nhiên; choice() – lựa chọn một phần tử ngẫu nhiên từ danh sách; seed() – khởi tạo một bộ sinh mã giả ngẫu nhiên, với cùng 1 giá trị seed bạn sẽ nhận được cùng một chuỗi số. Nếu muốn sử dụng nhiều tùy chọn hơn thì các bạn nên sử dụng module Random trong thư viện Numpy.