随机数

在Python中，随机数有许多有趣的应用。例如，我们可以使用随机数生成器来模拟掷骰子、抽奖等事件。或者在游戏中，我们可以使用随机数生成器来随机生成敌人的位置、攻击力等属性。在密码学领域中，随机数也是非常重要的，因为随机数可以用于生成加密密钥，从而保证密码的安全性。

random模块

在Python中，我们可以使用random模块的函数生成伪随机数，这些随机数看起来是随机的，但其实是按照一定的算法生成的。这些算法通常使用当前时间作为种子，以保证每次生成的随机数序列都是不同的。

生成0-1之间的随机数:random

In [5]:

```
import random

a=random.random()

print(a)
```

0.6332147178199808

生成一个a到b之间的整数，包括a和b : randint

In [9]:

```
import random

a=random.randint(1,10) #生成一个1-10之间的整数

print(a)
```

10

生成一个a到b之间的随机浮点数: uniform

In [13]:

```
import random

a=random.uniform(1,10) #生成一个1-10之间的整数

print(a)
```

1.2814973538563046

choice(seq): 从序列seq中随机选择一个元素返回。

In [14]:

```
import random

a=random.choice([0,1,2]) #在0，1，2之间随机抽一个数出来

print(a)
```

1

小练习：编写一个Python程序，生成一个随机密码。要求如下：

密码长度为8位
密码由大小写字母、数字以及特殊字符组成
密码第一位由大写字母组成
密码最后一位使用特殊字符，特殊字符可以使用以下任意一个：!@#$%^&*()。

提示： 可以使用str()函数将数字类型转换为字符串；使用+运算符实现字符串拼接。

In [42]:

```
import random

Letter=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','

PartiChar=['!','@','#','$','%','^','&','*','(',')']

Code=[] #随机密码

for i in range(8):
    #如果是末位
    if i==7:
        Code.append(random.choice(PartiChar))  #从PartiChar列表中随机选一个元素添加到Code中

    #如果是首位
    elif i==0:
        Code.append(random.choice(Letter))  #从字母列表中选择一位数作为密码

    else: #如果是其他位：

        #有50%的概率从数字中选
        r=random.random()
        if r<0.5:
            Code.append(random.randint(0,9))  #从数字列表中选择一位数作为密码
        else:
            Code.append(random.choice(Letter))  #从字母列表中选择一位数作为密码

Code[0]=Code[0].upper()

Code
```

Out[42]:

['Q', 'i', 2, 5, 5, 0, 3, ')']

In [ ]:

```
```

小练习：用蒙特卡洛方法计算圆周率

In [2]:

```
import random

n=0 #计数器
simTime=100000 #模拟投点的次数
for i in range(simTime):

    #生成一组坐标,位于0-2之间
    x=random.uniform(0,2)
    y=random.uniform(0,2)

    #判断这组坐标在圆内还是圆外
    if (x-1)**2+(y-1)**2<=1:
        n=n+1 #计数器+1

pi=4*(n/simTime)

print(pi)
```

```
3.13772
```

In [3]:

```
import random
import math
import tkinter as tk

# 创建画布
canvas_width = 400
canvas_height = 400
canvas = tk.Canvas(width=canvas_width, height=canvas_height)
canvas.pack()

# 绘制正方形和内接圆
square_size = 300
square_center = (canvas_width/2, canvas_height/2)
square_coords = (square_center[0]-square_size/2, square_center[1]-square_size/2,
                 square_center[0]+square_size/2, square_center[1]+square_size/2)
canvas.create_rectangle(square_coords, outline='black')
canvas.create_oval(square_coords, outline='black')

# 计算正方形和内接圆的面积和半径
square_area = square_size**2
circle_radius = square_size/2

# 生成随机点并绘制
num_points = 2000
num_points_in_circle = 0
for i in range(num_points):
    x = random.uniform(square_coords[0], square_coords[2])
    y = random.uniform(square_coords[1], square_coords[3])
    distance = math.sqrt((x-square_center[0])**2 + (y-square_center[1])**2)
    if distance <= circle_radius:
        canvas.create_oval(x, y, x+1, y+1, fill='red')
        num_points_in_circle += 1
    else:
        canvas.create_oval(x, y, x+1, y+1, fill='blue')
    canvas.update()
    canvas.after(3)

    if i%50==1:
        # 估算圆周率
        pi_estimate = 4 * num_points_in_circle / i
        print('估算的圆周率为：', pi_estimate)

# 进入主循环
tk.mainloop()
```

```
估算的圆周率为： 8.0
估算的圆周率为： 3.2941176470588234
估算的圆周率为： 3.287128712871287
估算的圆周率为： 3.3642384105960264
估算的圆周率为： 3.283582089552239
估算的圆周率为： 3.250996015936255
估算的圆周率为： 3.1627906976744184
估算的圆周率为： 3.156695156695157
估算的圆周率为： 3.162094763092269
估算的圆周率为： 3.1485587583148558
估算的圆周率为： 3.1457085828343314
估算的圆周率为： 3.1361161524500907
估算的圆周率为： 3.1613976705490847
```

```
---------------------------------------------------------------------------
TclError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7588\3188935993.py in <module>
     29     distance = math.sqrt((x-square_center[0])**2 + (y-square_center
[1])**2)
     30     if distance <= circle_radius:
---> 31         canvas.create_oval(x, y, x+1, y+1, fill='red')
     32         num_points_in_circle += 1
     33     else:

D:\anaconda\lib\tkinter\__init__.py in create_oval(self, *args, **kw)
   2796     def create_oval(self, *args, **kw):
   2797         """Create oval with coordinates x1,y1,x2,y2."""
-> 2798         return self._create('oval', args, kw)
   2799
   2800     def create_polygon(self, *args, **kw):

D:\anaconda\lib\tkinter\__init__.py in _create(self, itemType, args, kw)
   2774         else:
   2775             cnf = {}
-> 2776         return self.tk.getint(self.tk.call(
   2777             self._w, 'create', itemType,
   2778             *(args + self._options(cnf, kw))))

TclError: invalid command name ".!canvas"
```

本福德定律

In [ ]: