实验课 03

实验3-1 Python中random模块的使用

很多时候我们需要产生一些随机数。比如在从统计总体中抽取有代表性的样本的时候，或者在将实验样本分配到不同的试验组的过程中，或者在进行蒙特卡罗模拟法计算的时候。

真正的随机数是使用物理现象产生的：比如掷钱币、骰子、转轮、使用电子组件的噪音、核裂变等等。但是这些方法要么花费时间太多；要么成本太高，不太适合需要大规模随机数的场景。

幸好计算机中可以在短时间内产生大量的随机数，不过计算机产生的随机数叫做“伪随机数”。这些数列是“似乎”随机的数，实际上它们是通过一个固定的、可以重复的计算方法产生的。它们不真正地随机，因为它们实际上是可以计算出来的，但是它们具有类似于随机数的统计特征。这样的生成器叫做伪随机数生成器。

Python自带了一个随机数生成库random，可提供多种服从不同分布的随机数。

步骤1 产生随机整数

如果我们需要产生在[a,b]范围内的整数，可以使用`randint()`方法

不要忘了第一次调用随机数函数需要导入random库

```
import random
num_1 = random.randint(1,10)  # 生成一个整数N，介于1-10（包含1和10）之间

print(num_1)  # 可以多运行几次，看看产生的随机数是多少
```

步骤2 产生随机浮点数

如果我们需要产生一个浮点数，可使用以下方法

```
num_2 = random.random()  # 生成一个服从均匀分布的浮点随机数，介于[0.0,1.0)之间
print(num_2)
```

```
num_3 = random.uniform(3.5,7.2)  # 生成一个服从均匀分布的随机浮点数，介于[a,b]之间
print(num_3)
```

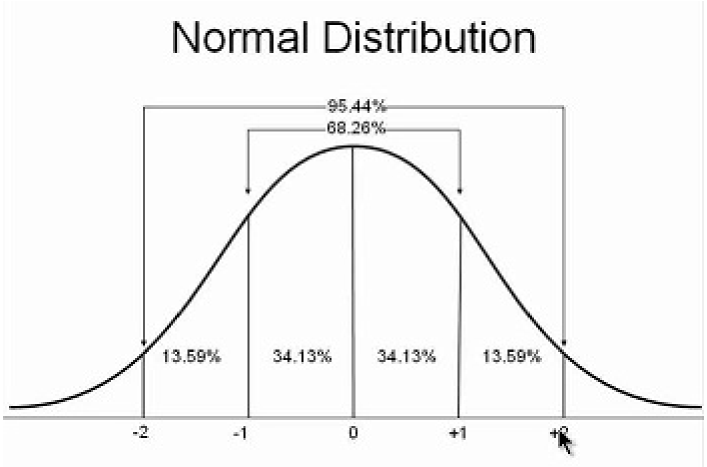
当然我们可以使用`round(var,bit)`方法保留小数点后bit位

```
print(round(num_3,2))
```

正态分布是常见的概率分布模型，我们可以产生服从正态分布的随机数

```
num_4 = random.normalvariate(500,70)  # 产生服从均值为500，标准差为70的正态分布的随机数
print(num_4)
```

实际上这是目前“大学英语考试”（College English Test）的成绩分布，最终报告成绩以整数呈现，所以我们可以使用`int()`方法将浮点数转换成整数。




```
print(int(num_4))
```

步骤3 产生多个随机数

商场年终促销，少不了抽奖环节吸引顾客。

一般一等奖只有1名幸运顾客抽中，而"参与奖"就很多了，如果按照上述方法一个个选取随机数，那太麻烦了，我们可以从候选列表中随机选出一批数据，作为抽奖结果。

```
list_1 = ["0135","0247","1248","2536","0014","3725","5102","0003"]  # 假设这是候选顾客的奖券号码，我们要从中选出3位中奖者
win_list = random.choices(list_1, k = 3)  # 我们从list_1列表中选出3位中奖者，结果赋值给win_list变量
print(win_list)
```

有的时候我们需要像洗牌那样打乱列表中元素的原有顺序，可以使用`shuffle()`方法

```
list_2 = [1,2,3,4,5,6,7,8,9,10]
random.shuffle(list_2)
print(list_2)
```

步骤4 设置随机种子

在做机器学习的相关实验时，我们需要将数据集随机分为训练集和测试集，在调试的情况下我们需要重复运行程序，但不想让数据集被重新随机分配，我们可以设置随机数种子，以达到产生的随机数不再随机（或可理解为固定）

```
random.seed(0)
print(random.random())  # 多运行几次，是不是发现生成的随机数不再变化了
```

实验3-2 Python中math模块的使用

计算机最擅长的就是数学计算，不过我们在C语言和Python语言中只会写一些简单的四则运算。幸好Python中自带math模块，为我们提供了许多数学函数供使用。

步骤1 数学常数

我们在《高等数学》用到最频繁的两个常数就是 π 和 e 了，现在我们调用math模块中的这两个数学常数。

别忘了首先要导入`math`模块。

```
import math
print(math.pi)  # 圆周率
print(math.e)  # 自然对数
```

步骤2 下界与上界

去机场停车，入口处都会放着一块停车费价格表。比如虹桥机场的停车费如下图所示：

虹桥枢纽东交通中心停车收费标准

停车场（库）	计费时间	小型车收费标准
虹桥T2航站楼	首2小时	10元/小时
	超2小时后	5元/小时
	24小时内计费 上限10小时	最高60元
	24-48小时计费 上限14小时	最高80元
	48小时及以上每24小时计费 上限20小时	最高110元

1、免费停放时间00:00-24:00入库车辆可享受一次20分钟内免费出库；
2、停车按小时计费，不足1小时按1小时计，超过1小时按1小时递进；
3、上述连续停放实行每24小时累计收费；
4、表中收费标准为小型车，大型车收费标准是小型车收费标准的两倍；
5、上述收费标准自2015年2月12日启用。

规定**不足1小时按1小时计**，所以停了1小时零1分钟也要收取2小时的停车费......

math库中有`floor()`和`ceil()`两个方法可以求出一个数的下界（不大于该数的最大整数）和上界（不小于该数的最小整数）。

```
print(math.floor(3.5))
print(math.floor(3.49))
print(math.floor(3))
print(math.floor(-2.7))
```

```
print(math.ceil(3.49))
print(math.ceil(3.5))
print(math.ceil(5))
print(math.ceil(-1.2))
```

步骤3 基本初等函数

在高中数学课上我们学过了指数函数、对数函数、阶乘；初中也学习过绝对值和最大公约数。

```
print(math.exp(2))  # 求e的2次方
print(math.pow(2,3))  # 求2的3次方
print(math.sqrt(2))  # 求2的算术平方根
```

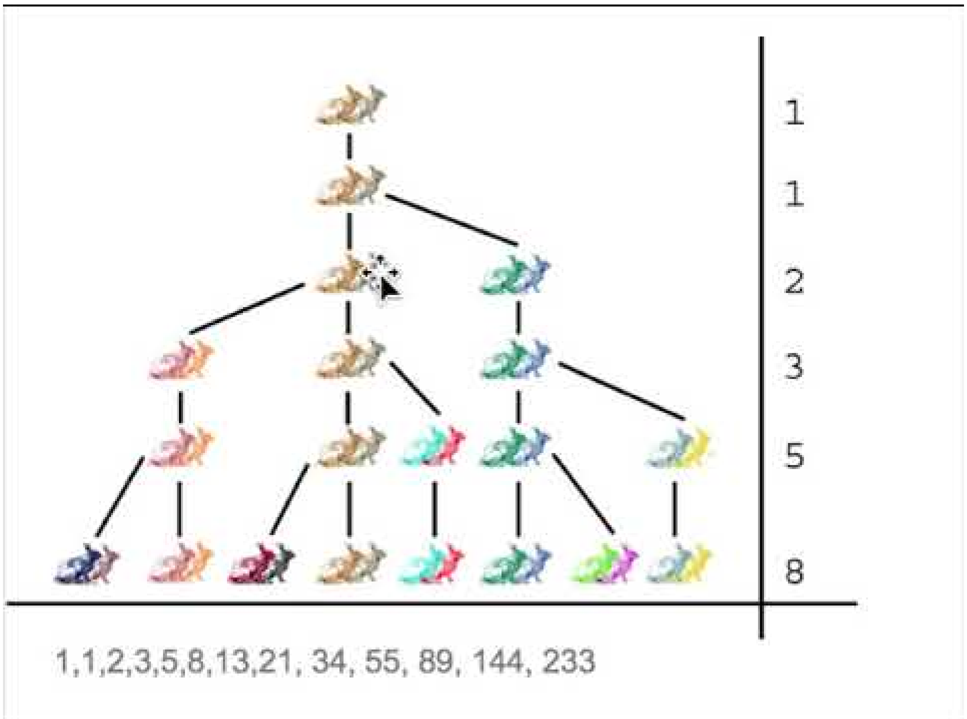
```
print(math.log2(1024))  # 求以2为底1024的对数
print(math.log10(1000))  # 求以10为底1000的对数
print(math.log(27,3))  # 求以3为底27的对数
print(math.log(math.e))  # 求e的自然对数
```

```
print(math.fabs(-5))  # 求-5的绝对值
print(math.factorial(5))  # 求5的阶乘
print(math.gcd(27,9))  # 求27和9的最大公约数
```

实验3-3 计算思维实例1：递归

公元1150年印度数学家Gopala和金月在研究箱子包装对象长宽刚好为1和2的可行方法数目时，首先描述这个数列。在西方，最先研究这个数列的人是比萨的列奥那多（意大利人斐波那契Leonardo Fibonacci），他描述兔子生长的数目时用上了这数列。

- 第一个月初有一对刚诞生的兔子
- 第二个月之后（第三个月初）它们可以生育
- 每月每对可生育的兔子会诞生下一对新兔子
- 兔子永不死去



从数学角度来看，斐波那契数列如下表述

$$
\begin{aligned}
F_0 &= 0 \\
F_1 &= 1 \\
F_n &= F_{n-1} + F_{n-2}, n \geq 2
\end{aligned}
$$

我们编写一个递归程序模拟一下斐波那契数列。

```
def Fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return Fibonacci(n-1) + Fibonacci(n-2)
```

编写递归程序时，务必注意在**函数开始处**编写递归的结束条件，否则可能会发生无限调用导致内存溢出。

现在我们验证一下函数是否可以正常运行。

```
result_list = []  # 用于存放结果的列表
for i in range(15):
    result_list.append(Fibonacci(i))  # 使用循环分别求出当i等于0到15时的斐波那契数，并加入列表中
print(result_list)
```

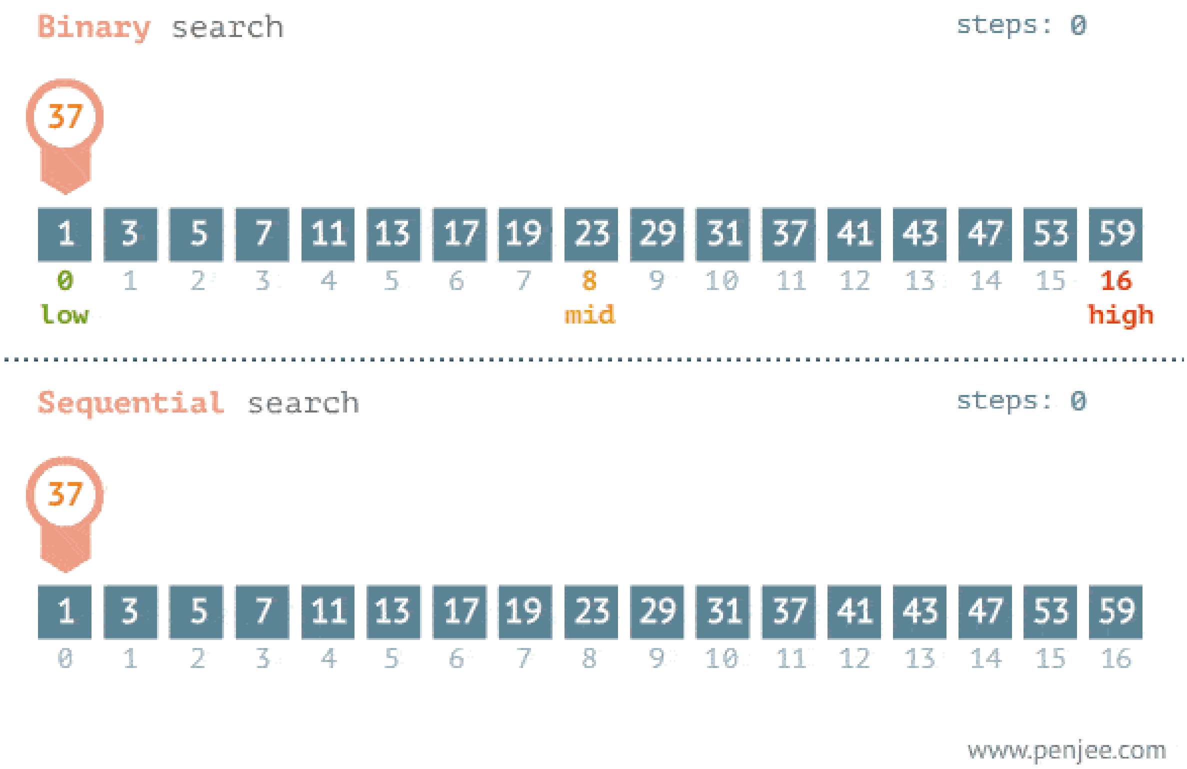
实验3-4 计算思维实例2：分治

《武林外传》中有一集，白三娘为了为难湘玉，让她从早到晚做苦力，晚饭时白三娘想递给湘玉一个馒头，结果却**掰开一半又一半**，最后剩下一小块馒头递给湘玉作为晚饭。

分治的思想与其类似，将一个大问题不断分解成若干个相同或相互独立的小问题，小问题继续划分为更小的问题，最终“最小”的问题可以直接求解，再将小问题的解层层合并，最终计算出原问题的解。

我们以著名的**二分查找**为例介绍分治算法的程序编写。

在1946年，约翰·莫奇利在摩尔学院讲座上第一次提出二分搜索的概念。1957年，威廉·皮特逊发表了第一个应用插值搜索的算法。在此时，每个发表的二分搜索算法只对长度为2的幂减一的数组有用。直到1960年，德里克·亨利·莱默发表了一个对于所有长度的数组都适用的算法。1962年，赫尔曼·博滕布鲁赫发表了一个用ALGOL 60写的二分搜索，将判断相等的步骤放到算法末尾。虽然将平均迭代次数增加一，但是每次迭代中的比较次数减少了1次。均匀二分搜索则是史丹佛大学的A. K.钱德拉在1971年发明的。1986年，伯纳德·查泽尔和列奥尼达斯·吉巴斯引入了分散层叠来解决计算几何中大量存在的搜索问题。




```python
def binary_search(arr, start, end, hkey):
    if start > end:
        return -1  # 如果首尾界标交叉，说明未找到该数
    mid = start + (end - start) // 2  # 确定中心元素下标
    if arr[mid] == hkey:
        return mid  # 如果中心元素就是目标元素，则直接返回下标
    elif arr[mid] > hkey:
        return binary_search(arr, start, mid - 1, hkey)  # 如果中心元素大于目标元素，则目标元素在左半子列表中
    elif arr[mid] < hkey:
        return binary_search(arr, mid + 1, end, hkey)  # 相反，则在右半子列表中
```

我们以上述动画所示测试这个函数是否可用

```python
arr=[1,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59]
target_index=binary_search(arr,0,16,37)
print(target_index)
```

实验3-5 数据思维实例：蒙特卡洛法

大家在高中数学课上学过定积分，通过牛顿——莱布尼茨公式可以计算一些简单函数的积分，例如

$$\int_0^{\pi} sin(x)\ dx = 2$$

如果你没有学过积分，那么怎么知道sin(x)与x=0、x=π和y轴围成的面积呢？我们可以用蒙特卡洛法，向面积为 π×π 的正方形中投入很多的随机点，计算落入sin(x)函数下点的数量与全部点数量之比，则该比值就近似于目标积分值与矩形面积之比。

```python
import random
import math
S = math.pi*math.pi
N = 10000000
C = 0
for i in range(N):
    x = random.uniform(0.0,math.pi)
    y = random.uniform(0.0,math.pi)
    if y <= math.sin(x):
        C += 1
I = C / N * S
print(I)
```

实验练习03

1. 请编写一个函数，使用递归方法计算 n 的阶乘，并进行测试
2. 请编写程序，使用`math`库计算27的立方根
3. 有装有15个硬币的袋子。15个硬币中有一个是伪造的，并且那个伪造的硬币比真的硬币要轻一些。我们要找出这个伪造的硬币。我们有一台可用来比较两组硬币重量的仪器，利用这台仪器，可以知道两组硬币的重量是否相同。请编写程序使用分治方法模拟上述过程，假设硬币的重量为列表[2,2,2,2,2,2,1,2,2,2,2,2,2,2,2]找出假币的序号（序号从0开始，假币的序号为6）
4. 使用蒙特卡洛法计算

$$\int_2^3 (x^2 + 4xsin(x))\,dx$$

的值*（参考答案：11.8114）*