



《大数据分析导论》——程序的控制结构



上次课回顾：数字与字符串

- Python中的数字
 - 整数
 - 浮点数
 - math库
 - 数值运算
 - 逻辑运算
- Python中的字符串
 - 字符串索引
 - 字符串+和*
 - 字符串format

操作符	描述
$x + y$	x与y之和
$x - y$	x与y之差
$x * y$	x与y之积
x / y	x与y之商
$x // y$	x与y之整数商，即：不大于x与y之商的最大整数
$x \% y$	x与y之商的余数，也称为模运算
$-x$	x的负值，即： $x*(-1)$
$+x$	x本身
$x**y$	x的y次幂，即： x^y

`"{1}：计算机{0}的CPU占用率为{2}%。".format("2016-12-31", "PYTHON", 10)`



提纲

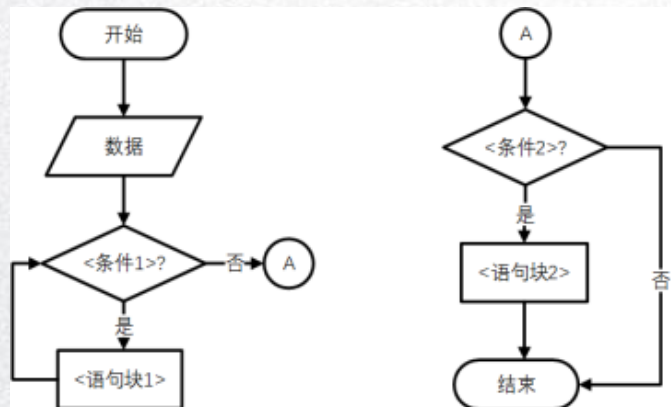
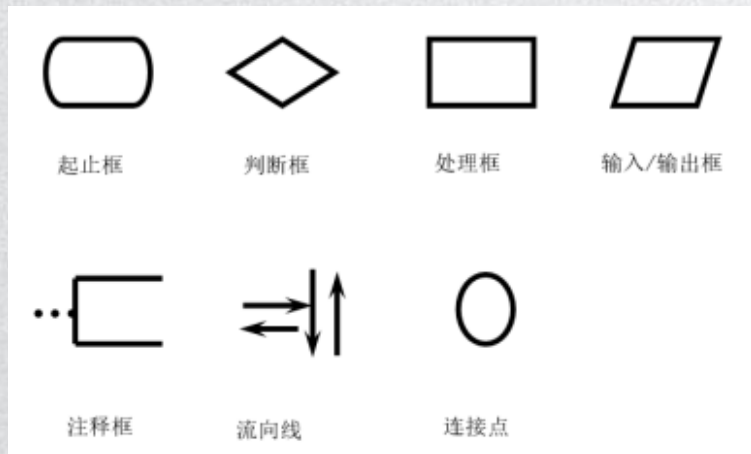


Python 03 程序的控制结构

- ☐ 程序的基本结构
- ☐ 分支结构
- ☐ 循环结构
-

程序过程描述的基本方式：流程图

- 程序流程图用一系列图形、流程线和文字说明描述程序的基本操作和控制流程
- 流程图的基本元素包括7种



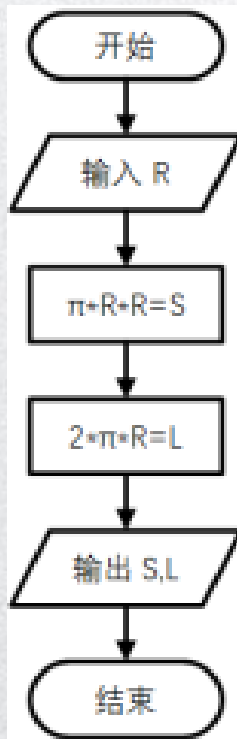


程序基本结构

- 顺序结构是程序的基础，但功能极为有限
- 三种基本结构组成
 - 顺序结构
 - 分支结构
 - 循环结构
- 任何程序都由这三种基本结构组合而成

顺序结构示例：圆的面积与周长计算

- 输入：圆半径R
- 处理：
 - 圆面积： $S = \pi * R * R$
 - 圆周长： $L = 2 * \pi * R$
- 输出：圆面积S、周长L



```
▶ import math
r = float(input("圆半径: "))
S = math.pi * r * r
L = math.pi * 2 * r

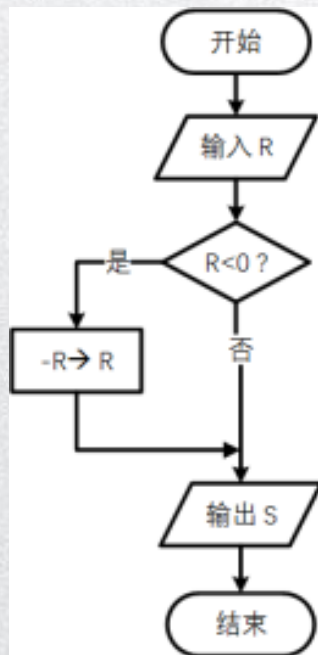
print("面积: {0}, 周长: {1}".format(S, L))
```

圆半径: 2

面积: 12.566370614359172, 周长: 12.566370614359172

分支结构示例：实数绝对值计算

- 输入：实数R
- 处理：
 - 如果 $R \geq 0$, $|R| = R$
 - 如果 $R < 0$, $|R| = -R$
- 输出: $|R|$

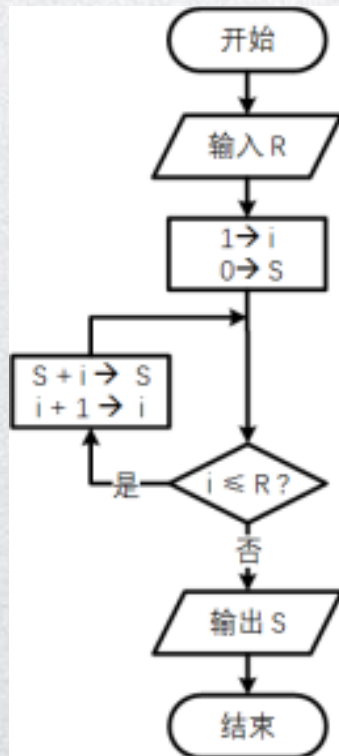


```
R = float(input("实数值: "))  
if R < 0:  
    R = -R  
  
print("绝对值: {0}".format(R))
```

实数值: -0.7
绝对值: 0.7

循环结构示例：整数累加

- 输入：整数R
- 处理：
 - $S = 1 + 2 + \dots + R$
- 输出: S



```
▶ R = int(input("输入整数: "))
i, S = 1, 0
while i <= R:
    S = S + i
    i = i + 1

print("1 + ... + {0} = {1}".format(R, S))
```

输入整数: 100

1 + ... + 100 = 5050



提纲



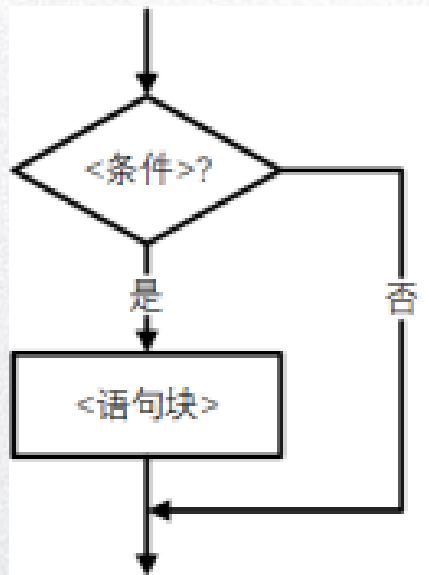
Python 03 程序的控制结构

- ☐ 程序的基本结构
- ☐ 分支结构
- ☐ 循环结构
-

简单情况：单分支结构if语句

if <条件>:
 语句块

- 语句块是if条件满足后执行的一个或多个语句序列
- 语句块中语句通过与if所在行形成**缩进**表达包含关系
- if语句首先评估<条件>的结果值
 - 如果结果为True，则执行语句块里的语句序列，然后控制转向程序的下一条语句
 - 如果结果为False，语句块里的语句会被跳过。
- 注意：
 - if语句中语句块执行与否依赖于条件判断
 - 无论什么情况，控制都会转到if语句后与该语句同级别的下一条语句



if语句的条件判断

- if语中<条件>部分可以使用任何能够产生True或False的语句
- 形成判断条件最常见的方式是采用关系操作符
- Python语言共有6个关系操作符

操作符	数学符号	操作符含义
<	<	小于
<=	≤	小于等于
>=	≥	大于等于
>	>	大于
==	=	等于
!=	≠	不等于



举例：空气PM2.5预警

- 输入：接收外部输入PM2.5值
- 处理：
 - If PM2.5值 ≥ 75 ，打印空气污染警告
 - If $35 \leq$ PM2.5值 < 75 ，打印空气污染警告
 - If PM2.5值 < 35 ，打印空气质量优，建议户外运动
- 输出：打印空气质量提醒

```
PM=int(input("请输入PM2.5数值:"))
if 0 <= PM and PM < 35:
    print("空气优质，快去户外运动!")
if 35 <= PM and PM < 75:
    print("空气良好，适度户外活动!")
if 75 <= PM:
    print("空气污染，请小心!")
```

请输入PM2.5数值:75
空气污染，请小心!



如图所示的程序，输入75时，程序的输出是什么？

- ☐ A 空气优质，快去户外运动！
- ☐ B 空气良好，适度户外活动！
- ☐ C 空气污染，请小心！
- ☒ D 空气良好，适度户外活动！
空气污染，请小心！

```
PM=int(input("请输入PM2.5数值:"))
if 0 <= PM and PM < 35:
    print("空气优质，快去户外运动!")
if 35 <= PM and PM <= 75:
    print("空气良好，适度户外活动!")
if 75 <= PM:
    print("空气污染，请小心!")
```

请输入PM2.5数值:75

提交

二分支结构:if – else语句

- 语法格式如下：

if<条件>:

 <语句块1>

else:

 <语句块2>

- <语句块1>是在if条件满足后执行的一个或多个语句序列
- 语句块2>是if条件不满足后执行的语句序列
- 二分支语句用于区分<条件>的两种可能True或者False，分别形成执行路径

```
PM=int(input("请输入PM2.5数值:"))  
if 0 <= PM and PM < 35:  
    print("空气优质，快去户外运动!")  
if 35 <= PM and PM <= 75:  
    print("空气良好，适度户外活动!")  
if 75 <= PM:  
    print("空气污染，请小心!")
```

请输入PM2.5数值:75
空气良好，适度户外活动!
空气污染，请小心!

```
PM=int(input("请输入PM2.5数值:"))  
if PM >= 75:  
    print("空气存在污染，请小心!")  
else:  
    print("空气没有污染，可以开展户外运动!")
```

请输入PM2.5数值:75
空气存在污染，请小心!



if – else语句紧凑结构表示

<表达式1> if <条件> else <表达式2>

```
PM=int(input("请输入PM2.5数值:"))  
print("空气存在污染, 请小心!") if PM >= 75 else print("空气没有污染, 可以开展户外运动!")
```

请输入PM2.5数值:75
空气存在污染, 请小心!

```
PM=int(input("请输入PM2.5数值:"))  
if PM >= 75:  
    print("空气存在污染, 请小心!")  
else:  
    print("空气没有污染, 可以开展户外运动!")
```

请输入PM2.5数值:75
空气存在污染, 请小心!

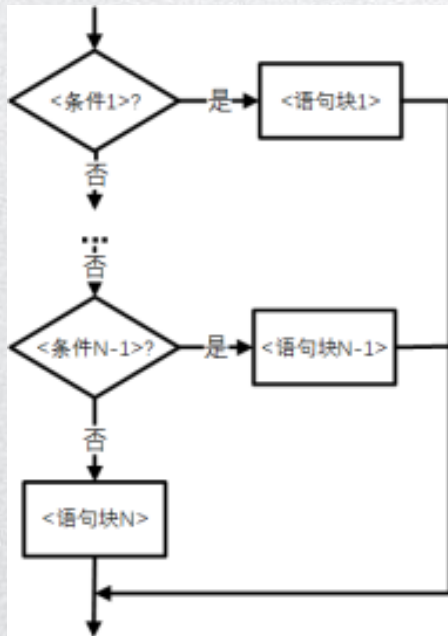
- 适合对出现的特殊值进行处理

```
PM=int(input("请输入PM2.5数值:"))  
y=PM if PM >= 75 else 0  
  
print(y)
```

请输入PM2.5数值:60
0

多分支结构：if-elif-else语句

if <条件1>:
 <语句块1>
elif <条件2>:
 <语句块2>
...
else:
 <语句块N>



- 用于设置同一个判断条件的多条执行路径
 - 依次评估寻找第一个结果为True的条件，执行该条件下的语句块，同时结束后跳过整个if-elif-else结构
 - 如果没有任何条件成立，else下面的语句块被执行
 - else子句可选



多分支结构示例：PM2.5预警

```
▶ PM=int(input("请输入PM2.5数值:"))
if PM < 35:
    print("空气优质，快去户外运动!")
elif PM <= 75:
    print("空气良好，适度户外活动!")
else:
    print("空气污染，请小心!")
```

请输入PM2.5数值:75
空气良好，适度户外活动!

多分支结构

```
▶ PM=int(input("请输入PM2.5数值:"))
if 0 <= PM and PM < 35:
    print("空气优质，快去户外运动!")
if 35 <= PM and PM <= 75:
    print("空气良好，适度户外活动!")
if 75 < PM:
    print("空气污染，请小心!")
```

请输入PM2.5数值:75
空气良好，适度户外活动!

单分支结构



示例2：BMI计算

- 编写一个根据体重和身高计算BMI值的程序，并同时输出国际和国内的BMI指标建议值

分类	国际BMI值 (kg/m ²)	国内BMI值 (kg/m ²)
偏瘦	< 18.5	< 18.5
正常	18.5 ~ 25	18.5 ~ 24
偏胖	25 ~ 30	24 ~ 28
肥胖	>= 30	>= 28

```
height, weight=eval(input("请输入身高(米)和体重(公斤)[逗号隔开]:"))
bmi= weight / pow(height, 2)
print("BMI数值为: {:.2f}".format(bmi))

wto, dom="", ""

if bmi < 18.5: #WHO标准
    wto="偏瘦"
elif bmi < 25: #18.5<=bmi<25
    wto="正常"
elif bmi < 30: #25<=bmi<30
    wto="偏胖"
else:
    wto="肥胖"

if bmi < 18.5: #我国卫生部标准
    dom="偏瘦"
elif bmi < 24: #18.5<=bmi<24
    dom="正常"
elif bmi < 28: #24<=bmi<28
    dom="偏胖"
else:
    dom="肥胖"

print("BMI指标为:国际' {0}', 国内' {1}'".format(wto, dom))
```

请输入身高(米)和体重(公斤)[逗号隔开]:1.7, 70

BMI数值为: 24.22

BMI指标为:国际' 正常', 国内' 偏胖'

单选题 1分



```
number = 30
if number % 2 == 0:
    print (number, 'is even')
elif number % 3 == 0:
    print (number, 'is multiple of 3')
```

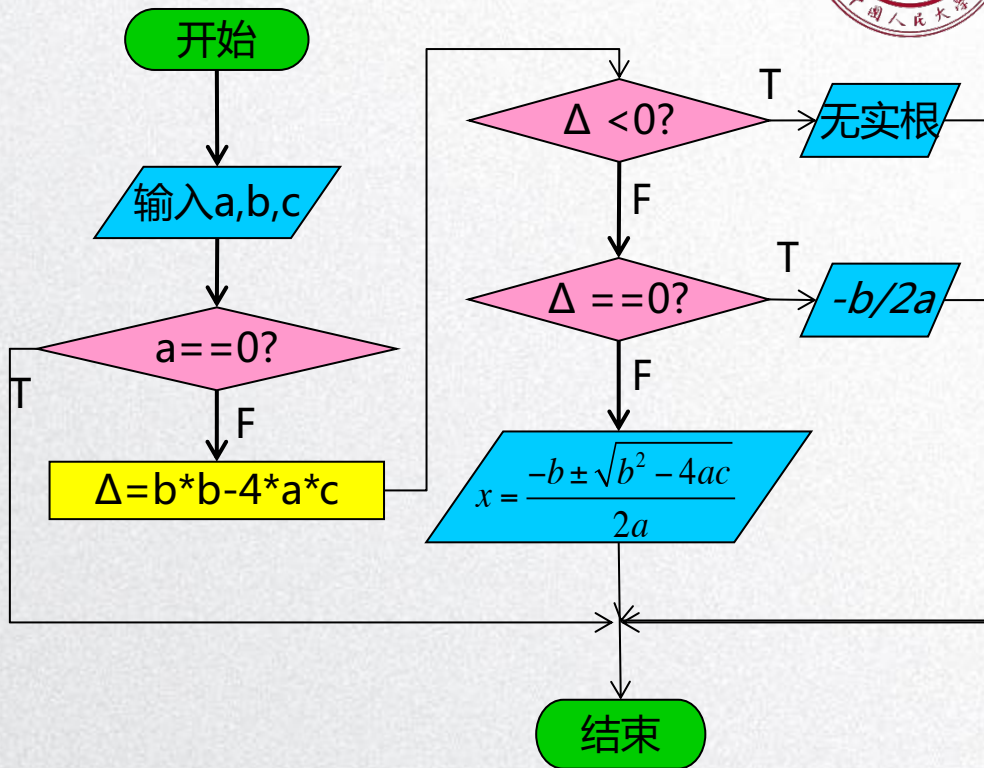
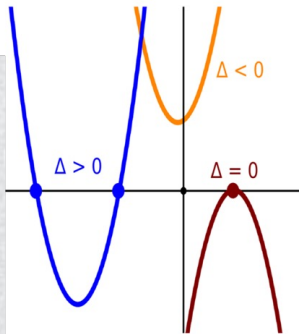
- ☐ A 程序出错
- ☐ B 30 is even
30 is multiple of 3
- ☒ C 30 is even
- ☐ D 30 is multiple of 3

提交

示例：求一元二次方程的解

一元二次方程 $ax^2+bx+c=0$ ($a \neq 0$) 的根与 $\Delta=b^2-4ac$ 有如下关系：

- ①当 $\Delta > 0$ 时，方程有两个不相等的两个实数根；
- ②当 $\Delta = 0$ 时，方程有两个相等的两个实数根；
- ③当 $\Delta < 0$ 时，方程无实数根。





示例：源代码

```
▶ import math
a = float(input("Input a:"))
b = float(input("Input b:"))
c = float(input("Input c:"))

if a == 0:
    print("The equation is linear.")
else:
    delta = b**2 - 4 * a * c
    if delta < 0:
        print("No real root")
    elif delta == 0:
        print("One root:", -b/(2*a))
    else:
        root = math.sqrt(delta)
        s1 = (-b + root)/(2*a)
        s2 = (-b - root)/(2*a)
        print("Two roots:", s1, s2)
```

Input a:1
Input b:-2
Input c:1
One root: 1.0



提纲



Python 03 程序的控制结构

- ☐ 程序的基本结构
- ☐ 分支结构
- ☐ 循环结构
-



有限循环：for语句遍历循环

for <循环变量> in <遍历结构>:
 <语句块>

- 遍历结构可以是字符串、文件、组合数据类型或range()函数

```
▶ R=int(input("请输入正整数:"))  
s = 0  
for i in range(R + 1):  
    s = s + i  
  
print("s = {0}".format(s))
```

请输入正整数:100
s = 5050

```
▶ for c in "abcdefg":  
    print(c)
```

a
b
c
d
e
f
g

循环N次

for i in range(N):

<语句块>

遍历文件fi的每一行

for line in fi:

<语句块>

遍历字符串s

for c in s:

<语句块>

遍历列表ls

for item in ls:

<语句块>

无限循环while语句

while <条件>:
 <语句块> 语句块

- 无限循环一直保持循环操作直到特定循环条件不被满足才结束，不需要提前知道确定循环次数。

```
▶ R=int(input("请输入正整数:"))
s = 0
i = 0
while s < R:
    s = s + i
    i = i + 1

print("s = {0}, i = {1}".format(s, i))
```

请输入正整数:100

s = 105, i = 15



循环执行过程

生成 count 变量, 值为 0

```
count = 0
```

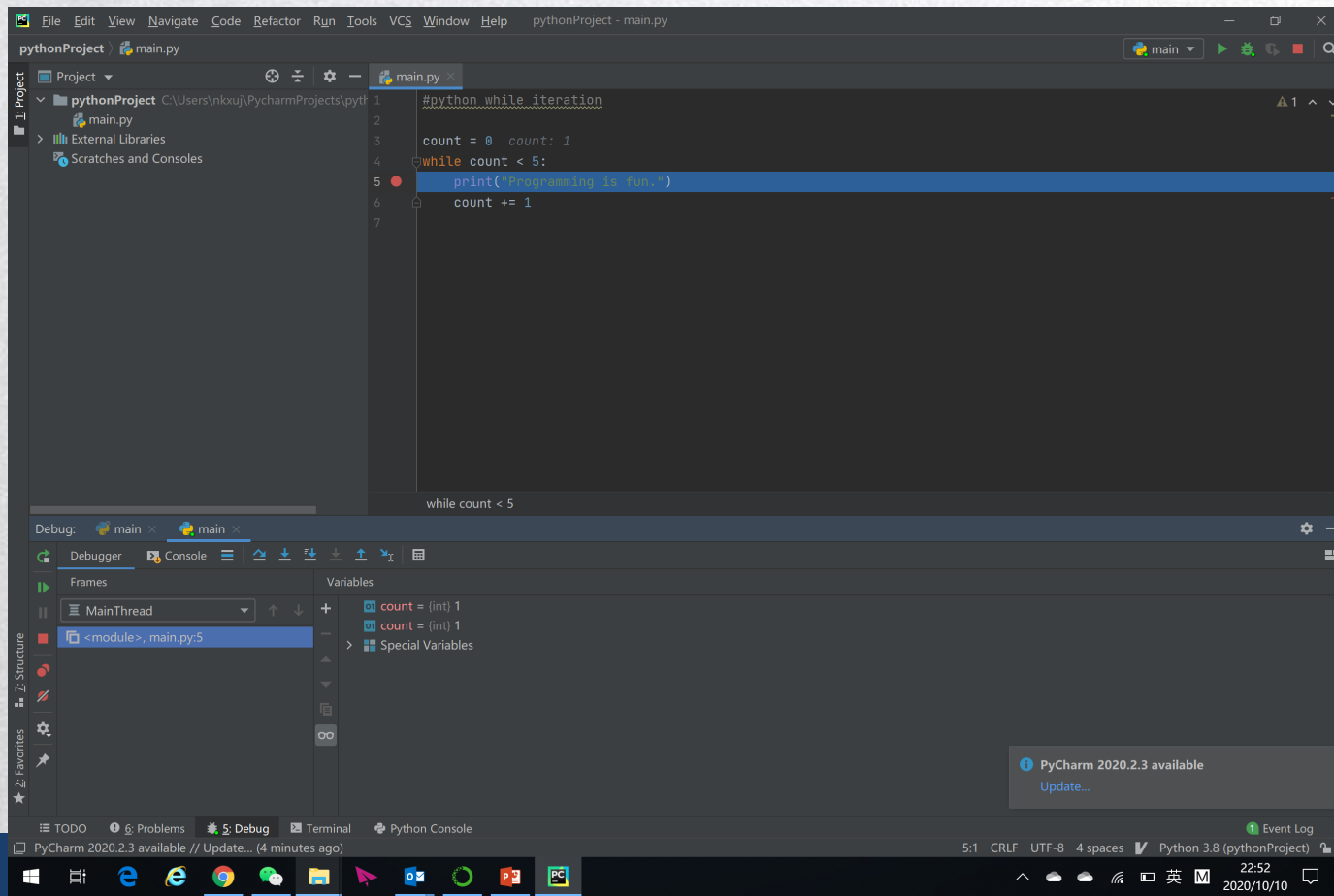
```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```



在PyCharm中执行Debug



循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```

count 值为 0 , 小于 5

循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun!")
    count += 1
```

打印字符串

循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun.")
    count += 1
```

count 值加 1, 结果为 1

循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```

count 值为 1 , 小于 5

循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun!")
    count += 1
```

打印字符串



循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun! ")
    count += 1
```

count 值加 1, 结果为 2

循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```

count 值为 2 , 小于 5

循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun!")
    count += 1
```

打印字符串



循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun! ")
```

```
    count += 1
```

count 值加 1, 结果为 3

循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```

count 值为 3 , 小于 5

循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun!")
    count += 1
```

打印字符串



循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun!")
    count += 1
```

count 值加 1, 结果为 4

循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```

count 值为 4 , 小于 5



循环执行过程

```
count = 0
while count < 5:
    print("Programming is fun!")
    count += 1
```

打印字符串

循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun! ")
```

```
    count += 1
```

count 值加 1, 结果为 5

循环执行过程

```
count = 0
```

```
while count < 5:
```

```
    print("Programming is fun!")
```

```
    count += 1
```

count 值为 5 , 不小于 5

循环执行过程

```
count = 0
while count < 5:
    print("Programming is
    count += 1
```

继续执行 while 后面的语句

循环辅助保留字break

- break用来辅助控制循环执行
- break用来跳出**最内层**for或while循环
 - 脱离该循环后程序**从循环后**代码继续执行
 - 每个break语句只有能力跳出当前层次循环

```
for s in "RUC":  
    print(s, end = "")  
    if s=="U":  
        break
```

RU

```
for s in "RUC":  
    for i in range(10):  
        print(s, end = "")  
        if s=="U":  
            break
```

RRRRRRRRRRRUCCCCCCCCC



循环辅助保留字continue

- continue用来**结束当前当次循环**
 - 跳出循环体中下面尚未执行的语句
 - 但不跳出当前循环
 - 对于while循环，继续求解循环条件。
 - 对于for循环，程序流程接着遍历循环列表
- break与continue对比：
 - continue语句**只结束本次循环**，而不终止整个循环的执行。
 - break语句**结束整个循环过程**，不再判断执行循环的条件是否成立

```
▶ for s in "Python":  
    if s=="t":  
        break  
    print(s, end = "")  
print("\nEnd")
```

Py
End

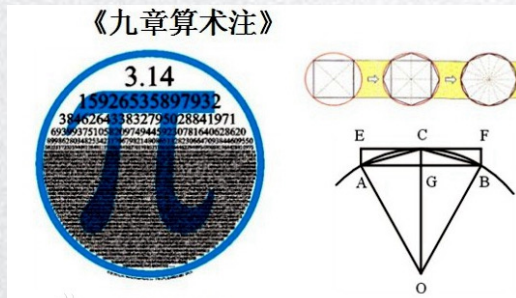
```
▶ for s in "Python":  
    if s=="t":  
        continue  
    print(s, end = "")  
print("\nEnd")
```

Pyhon
End

举例：求圆周率

- 圆周率的求解方法

- 刘徽：割圆术
- 几何方法

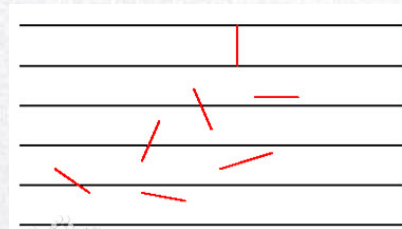


- 莱布尼茨级数
- 分析方法

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$$

右边的展式是一个**无穷级数**，被称为**莱布尼茨级数**，这个级数**收敛**到 $\frac{\pi}{4}$ 。
戈里。使用**求和**符号可记作：

$$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$$



- **蒲丰投针**
- 随机抽样或统计试验方法

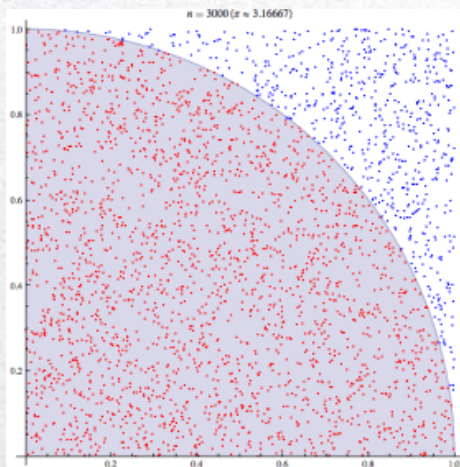
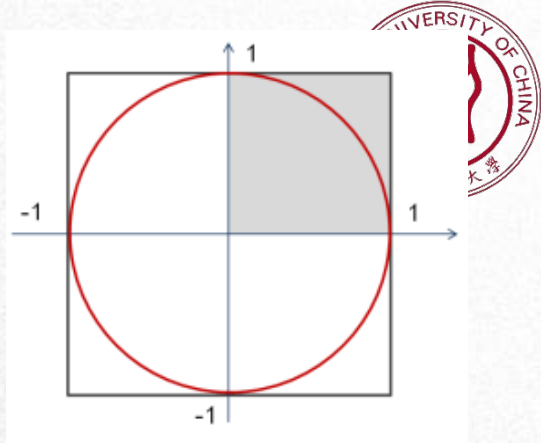
18世纪，法国数学家**布丰**提出的“投针问题”，记载于布丰1777年出版的著作中：“在平面上画有一组间距为**a**的平行线，将一根长度为**l** ($l \leq a$) 的针任意掷在这个平面上，求此针与平行线中任一条相交的**概率**。”

布丰本人证明了，这个概率是：

$$p = \frac{2l}{\pi a} \text{ (其中 } \pi \text{ 为圆周率)}$$

蒙特卡罗法求圆周率

- 基本思想：利用离散点值表示图形的面积，通过面积比例来求解 π 值
- 基本步骤
 - 随机向单位正方形和圆结构，抛洒大量“飞镖”点
 - 计算每个点到圆心的距离，判断该点在圆内或圆外
 - 用圆内的点数除以总点数就是 $\pi/4$ 值
- 随机点数量越大，越充分覆盖整个图形，计算得到的 π 值越精确
 - 前提条件：真正的随机数





如何得到随机点？

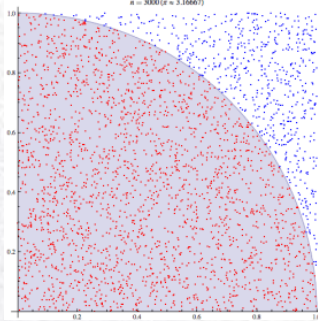
- Python的random库
 - 提供了不同类型的随机数函数，所有函数都是基于最基本的random.random()函数扩展而来

函数	描述
seed(a=None)	初始化随机数种子，默认值为当前系统时间
random()	生成一个[0.0, 1.0)之间的随机小数
randint(a, b)	生成一个[a,b]之间的整数
getrandbits(k)	生成一个k比特长度的随机整数
randrange(start, stop[, step])	生成一个[start, stop)之间以step为步数的随机整数
uniform(a, b)	生成一个[a, b]之间的随机小数
choice(seq)	从序列类型(例如：列表)中随机返回一个元素
shuffle(seq)	将序列类型中元素随机排列，返回打乱后的序列
sample(pop, k)	从pop类型中随机选取k个元素，以列表类型返回

```
▶ import random
random.seed()
print(random.random())
print(random.random())
print(random.random())
print(random.random())
```

```
0.7311103989105229
0.8708195976169583
0.09351451957401158
0.11836133987530661
```


求圆周率



DARTS	π	运行时间
2^{10}	3.109375	0.011s
2^{11}	3.138671	0.012s
2^{12}	3.150390	0.014s
2^{13}	3.143554	0.018s
2^{14}	3.141357	0.030s
2^{15}	3.147827	0.049s
2^{16}	3.141967	0.116s
2^{18}	3.144577	0.363s
2^{20}	3.142696777	1.255s
2^{25}	3.1416978836	40.13s

精度的变化



```
import random
import math

DARTS=100000
hits=0.0

random.seed()
for i in range(1, DARTS+1):
    x,y = random.random(),random.random()
    dist = math.sqrt(x**2+y**2)
    if dist<=1.0:
        hits = hits + 1

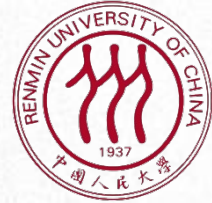
pi= 4 * (hits / DARTS)
print("Pi值是{0}.".format(pi))
```

Pi值是3.13684.



作业1：统计一段文本中关键词出现次数

- 给定一段文本
- 用户输入：
 - 一个关键词
- 输出：
 - 此关键词在文本中出现的次数
- 例如：
 - Text= “高瓴**人工智能**学院是中国人民大学下属学院，承担学校**人工智能**学科的规划与建设”
 - 输入Keyword= “人工智能”
 - 输出：2



● 作业2：不同方法求圆周率（下次上机课交）

- 实现两种不同方法所求得的圆周率，例如：
 - 蒙特卡洛
 - 莱布尼茨
 -
- 比较它们的收敛速度和计算精度

其它圆周率计算公式

- 欧拉公式

$$1 + \frac{1}{2^2} + \frac{1}{3^2} + \dots = \frac{\pi^2}{6}$$

- BBP公式

$$\pi = 3 \sum_{k=0}^{\infty} \frac{1}{(4k+1)(2k+1)(k+1)}$$

$$\pi = \sum_{k=0}^{\infty} \left[\frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \right]$$

- 拉马努金以及Chudnovsky.....

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{99^2} \sum_{k=0}^{\infty} \frac{(4k)!}{(k!)^4} \frac{1103 + 26390k}{396^{4k}}$$

$$\frac{1}{\pi} = \frac{1}{53360\sqrt{640320}} \sum_{k=0}^{\infty} (-1)^k \frac{(6k)!}{(k!)^3 (3k)!} \times \frac{13591409 + 545140134k}{640320^{3k}}$$

$$e^{i\pi} + 1 = 0$$



<https://www.zhihu.com/question/312520105/answer/601728029>



谢谢！