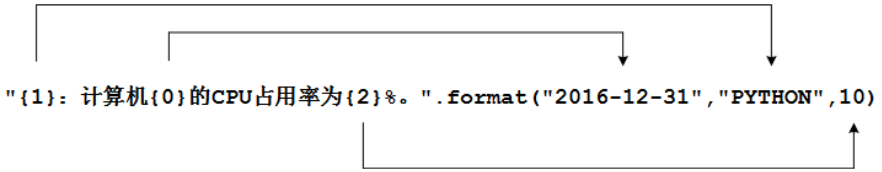
《人工智能与Python程序设计》——程序的控制结构

人工智能与Python程序设计 教研组

上次课回顾：数字与字符串

- Python中的数字
  - 整数，浮点数，复数
  - math库
  - 数值运算
  - 关系运算
  - 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



程序设计中的重要概念：表达式

- 由字面值、变量、函数调用、索引和切片、属性引用、运算符、括号等元素组成
  - 字面值(Literals)：整数、浮点数、复数、字符串
  - 变量(Variables)：由赋值语句”创建“
  - 函数调用：input()、pow()
    - 注意：print()也是一个函数
  - 索引和切片：[]
  - 属性引用：math.pi，<模板字符串>.format()
  - 运算符：算数运算符、关系运算符、逻辑运算符
  - 括号：改变计算优先级
- 表达式求值：
  - 表达式可以出现在赋值语句中：a = 10
  - 表达式本身也是一条语句（反过来语句并不一定是表达式）
  - 当表达式所在的语句被**执行**时，Python解释器会对其进行**求值**

字符串使用实例1

- 问题：输入一个月份数字，返回对应月份名称缩写
  - 输入：输入一个表示月份的数字(1-12)
  - 处理：利用字符串基本操作实现该功能
  - 输出：输入数字对应月份名称的缩写 (如：1 -> Jan)

- 设计思路
  - 1. 将所有的月份名称缩写存储在一个字符串中

```
months = "JanFebMarAprMayJunJulAugSepOctNovDec"
```

  - 2. 在字符串中截取适当的子串来查找特定月份
  - 3. 找出在哪里切割子串
    - 每个月份的缩写都由3个字母组成
    - 如果pos表示一个月份的第一个字母，则months[pos:pos+3]表示这个月份的缩写
    - monthAbbrev = months[pos:pos+3]

字符串使用实例1

- 月份与字符串初始位置关系： (n – 1) * 3

	月份	字符串中位置
Jan	1	0
Feb	2	3
Mar	3	6
Apr	4	9

- 编写程序

```
months = "JanFebMarAprMayJunJulAugSepOctNovDec"
n = int(input("输入月份数(1-12)"))
if n <= 0 or n > 12:
    print("输入1-12的月份数。")
else:
    pos = (n-1)*3
    print(months[pos:pos+3])
```

```
输入月份数(1-12)5
May
```

字符串使用实例2

- 任务：简单的非刷新文本进度条
- 设计思路：
  - 利用print()函数实现
  - 按照任务执行百分比将整个任务划分为100个单位，每执行N%输出一次进度条。
  - 每一行输出包含进度百分比，代表已完成的部分(**)和未完成的部分(..)的两种字符，以及一个跟随完成度前进的小箭头

```
%10 [*****->.....................................]
```

字符串使用实例

- scale=10：分成10段执行，每一段提升10%
- 每一个循环画出一个进度条
  - i：当前的轮数的进度，0，1，2, ..., 10
  - a: 把字符串“**”重复i次，表示完成的进度
  - b: 把字符串 “..”重复scale –i 次，表示未完成的进度
  - c: 当前进度度百分比，scale = 10时候，就是i*10
  - 通过字符串格式化画得到进度条字符串

```
import time
scale = 10
print("------执行开始------")
for i in range(scale + 1):
    a = "**" * i
    b = ".." * (scale - i)
    c=(i / scale) * 100
    print("%{0:^3.0f}[{1}->{2}]".format(c,a,b))
    time.sleep(0.2)

print("------执行结束------")
```

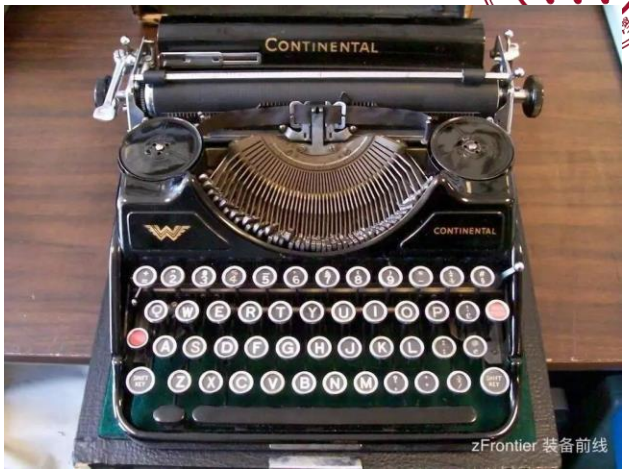
```
------执行开始------
% 0 [->....................]
%10 [**->..................]
%20 [****->................]
%30 [******->..............]
%40 [********->............]
%50 [**********->..........]
%60 [************->........]
%70 [**************->......]
%80 [****************->....]
%90 [******************->..]
%100[********************->]
------执行结束------
```

进度条单行动态刷新

- 了解“回车”和“换行”字符
  - 回车<\r>(carriage return)：把打印头定位到左边界（打印头重新放在这一行的开始）。
  - 换行<\n>(line feed)：把纸向下移一行。

- Unix系统：每行结尾只有"<换行>"，即"\n"；
- Windows：每行结尾是"<回车><换行>"，即"\r\n"；
- Mac：每行结尾是"<回车>"。

- 如何单行刷新?
  - 只回车，不换行
  - 用后打印的内容覆盖原来的内容



试一下效果

- print("\r{:3}%".format(i), end="")
  - \r：回到最左列开始打印
  - end= “” ：print函数不打印\n

- 注意：如果后打印的字符比之前的字符段，不能覆盖的部分将仍然保留在屏幕上

```
import time
for i in range (101):
    print("\r{:3}%".format(i),end="")
    time.sleep(0.1)
```

```
100%
```

```
import time
for i in range (101):
    print("\r{:}%".format(100-i),end="")
    time.sleep(0.1)
```

```
0%%%
```

主观题 10分

设置

改动图示的程序，实现进度条单行动态刷新

```
import time
scale = 10
print("------执行开始------")
for i in range(scale + 1):
    a = "**" * i
    b = ".." * (scale - i)
    c=(i / scale) * 100
    print("%{0:^3.0f}[{1}->{2}]".format(c,a,b))
    time.sleep(0.2)

print("------执行结束------")
```

```
------执行开始------
% 0 [->....................]
%10 [**->..................]
%20 [****->................]
%30 [******->..............]
%40 [********->............]
%50 [**********->..........]
%60 [************->........]
%70 [**************->......]
%80 [****************->....]
%90 [******************->..]
%100[********************->]
------执行结束------
```

正常使用主观题需2.0以上版本雨课堂

作答

Python 03
程序的控制结构

提纲

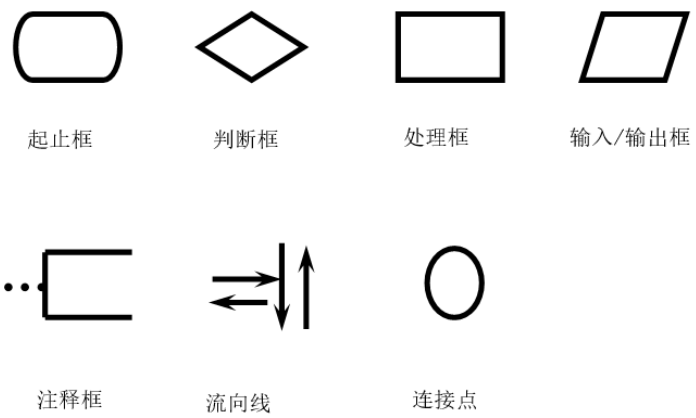
- **程序的基本结构**
- 分支结构
- 循环结构

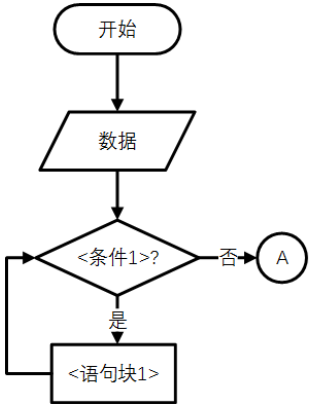
教学目标

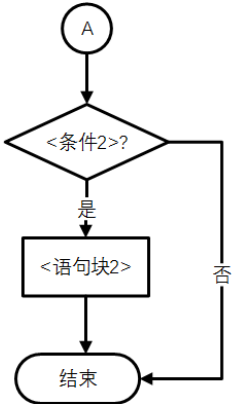
- 了解和掌握使用流程图表示程序基本结构和执行过程
- 熟练掌握Python语言中的分支语句
  - if
  - if-else
  - if-elif-else
- 了解和掌握Python语言中的异常处理
  - try-except
  - *Raise*
- 熟练掌握Python语言中的循环语句
  - while
  - for
  - break/continue
- 实现一个采用随机模拟计算圆周率的程序

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





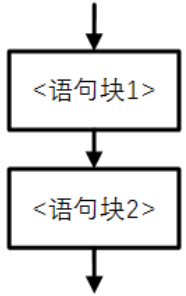


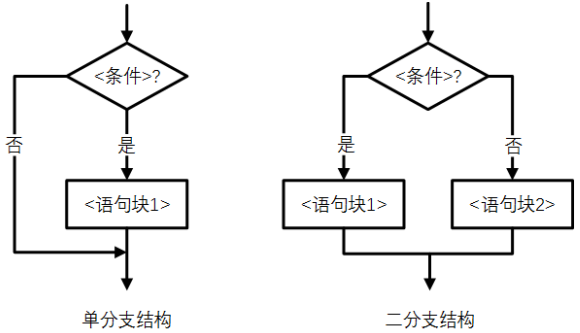
程序基本结构

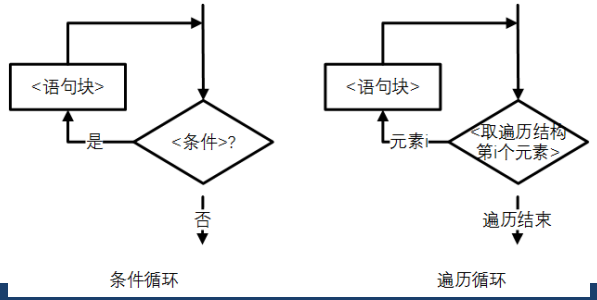
- 顺序结构是程序的基础，但功能极为有限
- 三种基本结构组成
  - **顺序结构**
  - **分支结构**
  - **循环结构**
- 任何程序都由这三种基本结构组合而成

程序的基本结构

- 顺序结构：程序按照线性顺序依次执行
  - 语句块1和语句块2表示一个或一组顺序执行的语句
- 分支结构：程序根据条件判断结果而选择不同向前执行路径
  - 包括单分支结构和二分支结构
  - 由二分支结构会组合形成多分支结构
- 循环结构：根据条件判断结果向后反复执行
  - 包括条件循环和遍历循环结构

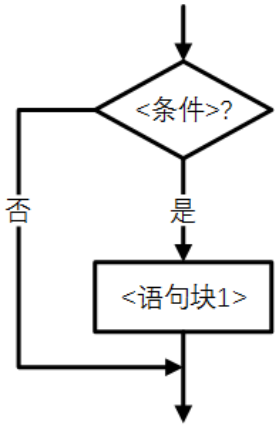




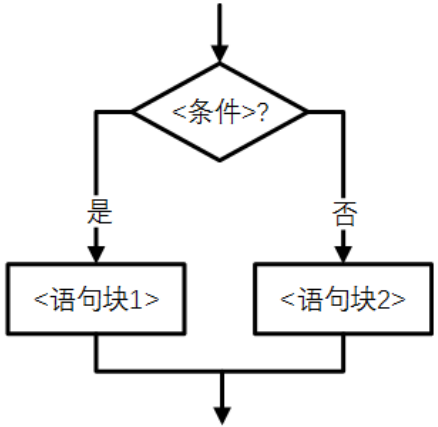


分支结构

- 分支结构是程序根据条件判断结果而选择不同向前执行路径的一种运行方式，包括单分支结构和二分支结构。由二分支结构会组合形成多分支结构



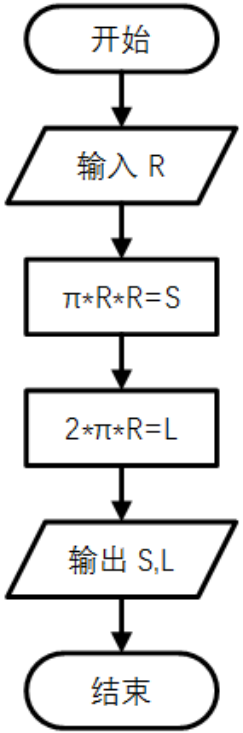
单分支结构



二分支结构

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=π*R*R
  - 圆周长：L=2*π*R
- 输出：圆面积S、周长L

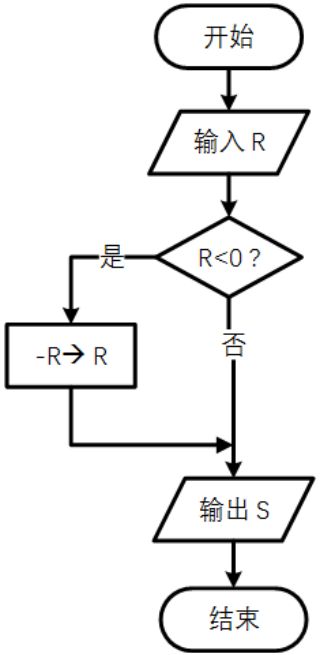



```
import math
r = float(input("圆半径："))
S = math.pi * r * r
L = math.pi * 2 * r

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>=0，|R|=R
  - 如果R< 0, |R| = -R
- 输出: |R|

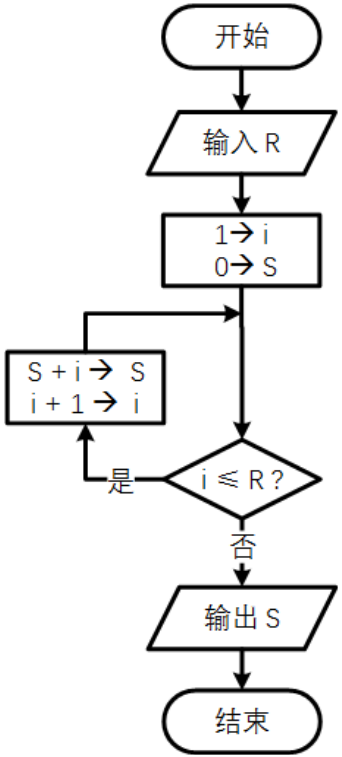



```
R = float(input("实数值："))
if R < 0:
    R = -R

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 + …+ 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
```

Python 03
程序的控制结构

提纲

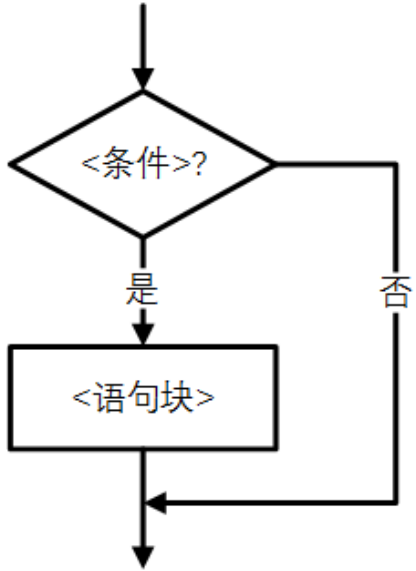
- 程序的基本结构
- **分支结构**
- 循环结构

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<=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！
```

单选题 1分

设置

如图所示的程序，输入75时，程序的输出是什么？

```
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
```

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!

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！
```

条件表达式

<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！
```

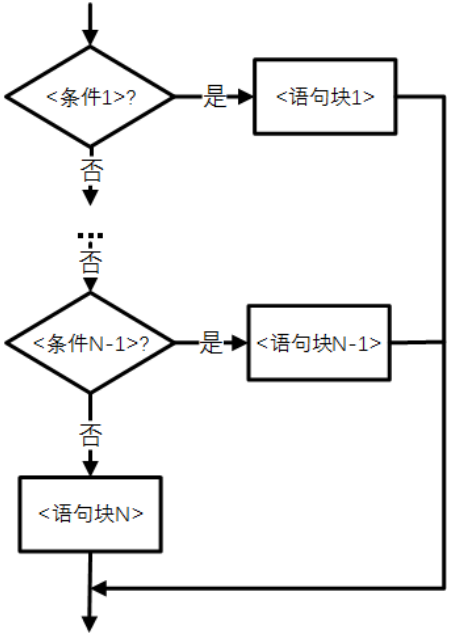
- 适合对出现的特殊值进行处理

```
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: #WTO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

设置

```
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

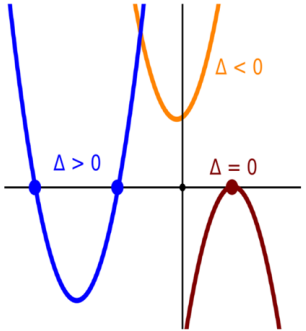
示例：求一元二次方程的解

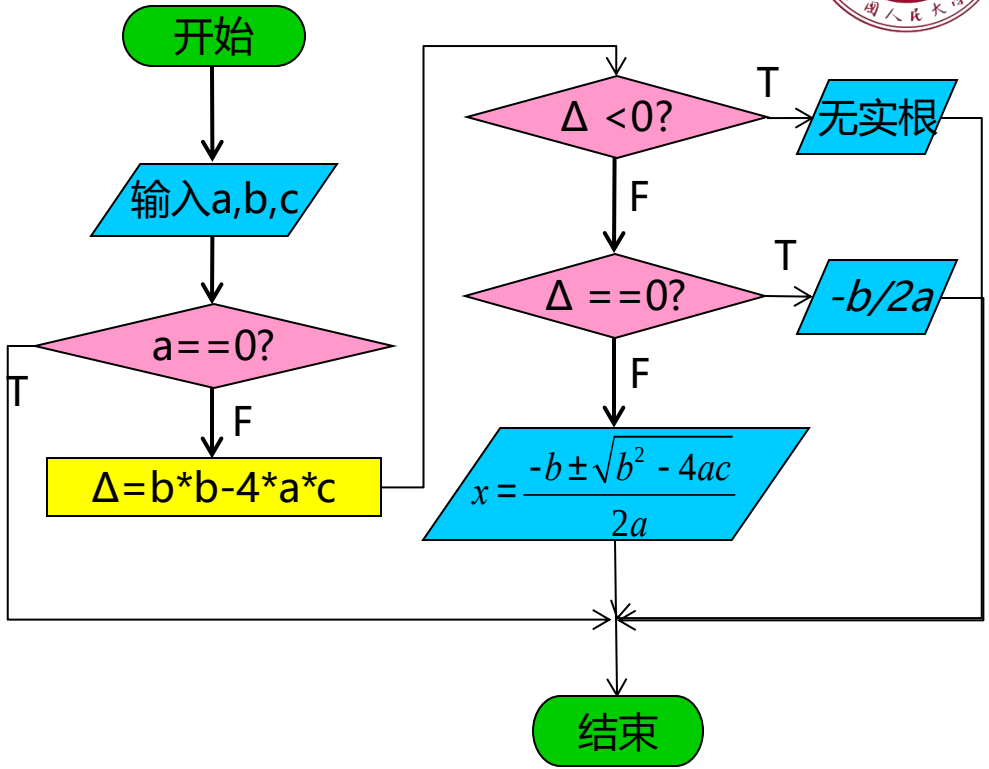
一元二次方程$ax^2+bx+c=0$（$a\neq0$）的根与$\triangle=b^2-4ac$有如下关系：

①当$\triangle>0$时，方程有两个不相等的两个实数根；

②当$\triangle=0$时，方程有两个相等的两个实数根；

③当$\triangle<0$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
```

程序异常处理

- 用户输入类型与预期不符合，产生了异常

  行号：1 num = eval(.....

  NameError: name ‘a’ is not defined

  – NameError：异常类型
  – name ‘a’ is ...：内容提示

- 问题：如何捕捉上述异常，避免程序直接退出

```
num = eval(input("请输入一个整数"))
print(num ** 2)
```

```
请输入一个整数a
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-2c78da7a8114> in <module>
----> 1 num = eval(input("请输入一个整数"))
      2 print(num ** 2)

<string> in <module>

NameError: name 'a' is not defined
```

异常处理

```
try :
    <语句块1>
except :
    <语句块2>
```

```
try :
    <语句块1>
except <异常类型> :
    <语句块2>
```

```
try:
    num = eval(input("请输入一个整数"))
    print(num ** 2)
except:
    print("输入不是整数")
```

```
请输入一个整数a
输入不是整数
```

用法1：捕捉所有异常

```
try:
    num = eval(input("请输入一个整数"))
    print(num ** 2)
except NameError:
    print("输入不是整数")
```

```
请输入一个整数a
输入不是整数
```

用法2：捕捉特定类型的异常

异常处理的高级使用

try:
    <正常程序块>
except:
    <发生异常执行>
else:
    <不发生异常执行>
finally:
    <一定执行>

```
try:
    num = eval(input("请输入一个整数"))
except NameError:
    print("输入不是整数")
else:
     print(num ** 2)
finally:
    print("done!")
```

```
请输入一个整数a
输入不是整数
done!
```

```
try:
    num = eval(input("请输入一个整数"))
except NameError:
    print("输入不是整数")
else:
     print(num ** 2)
finally:
    print("done!")
```

```
请输入一个整数2
4
done!
```

Python 03
程序的控制结构

提纲

- 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

循环N次	遍历文件fi的每一行	遍历字符串s	遍历列表ls
for i in range(N):	for line in fi:	for c in s:	for item in ls:
<语句块>	<语句块>	<语句块>	<语句块>

```
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
```

无限循环while语句

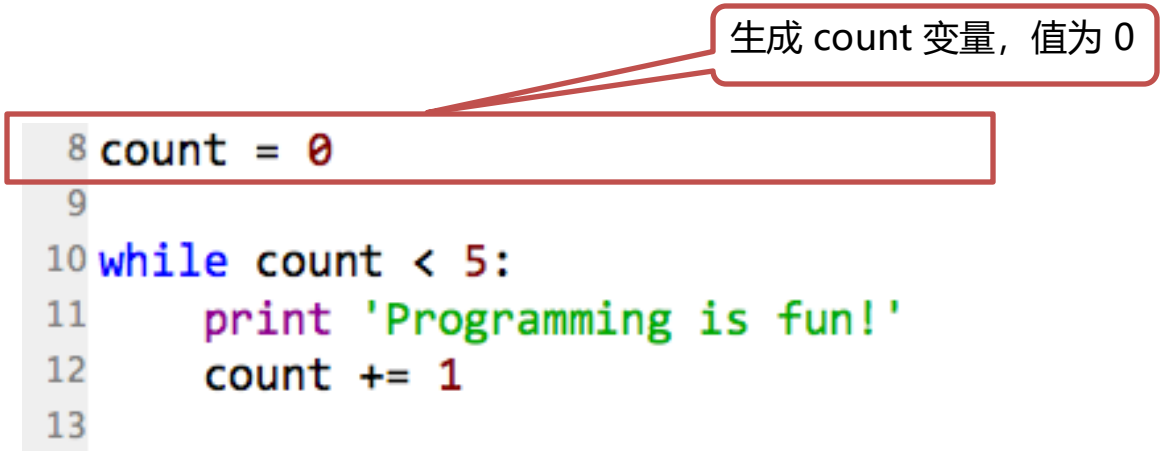
while <条件>:
    <语句块>

- 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。

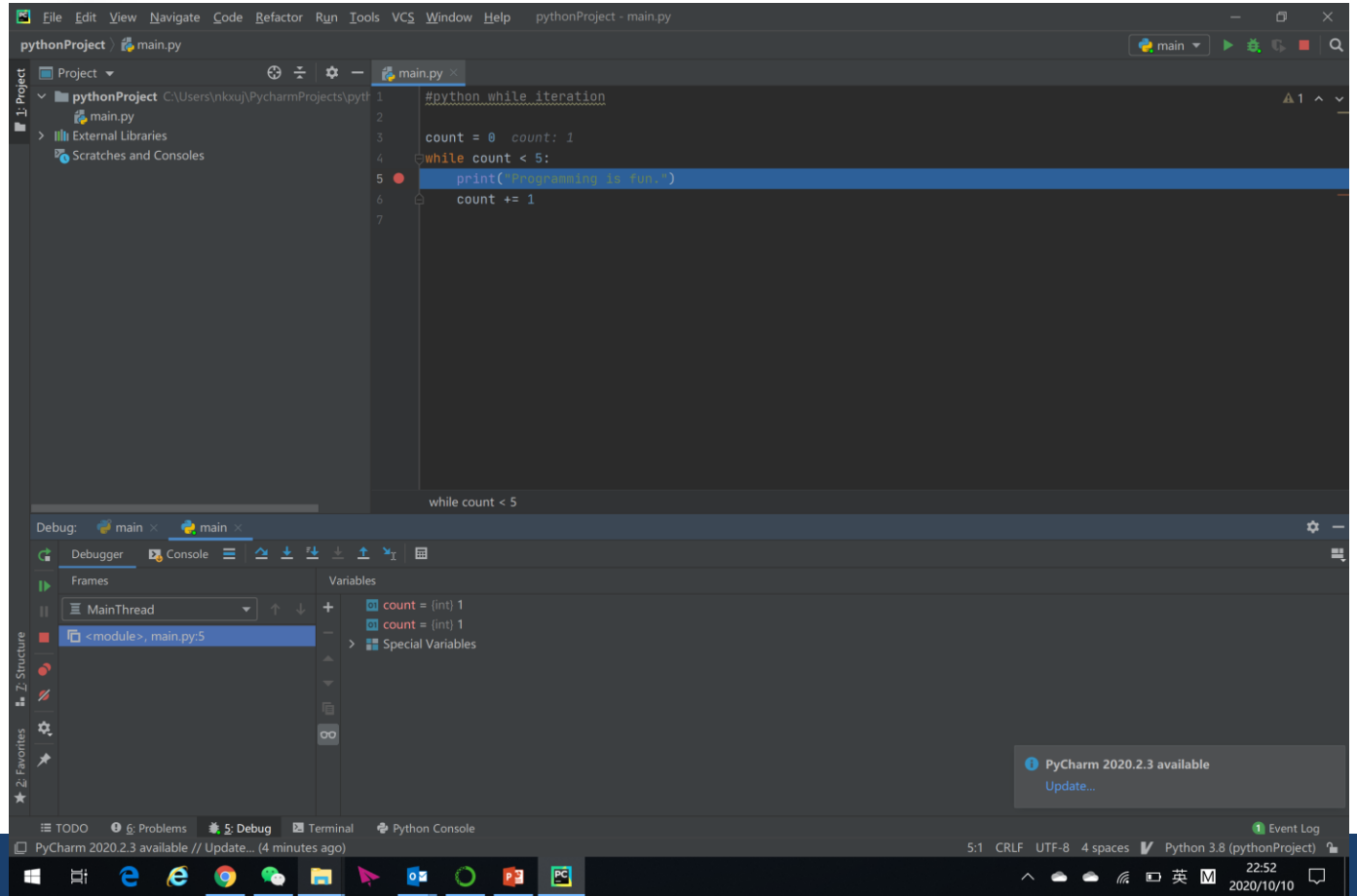
```
R=int(input("请输入正整数:"))
s = 0
i = 0
while s < R:
    s = s + i
    i = i + 1

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
8 count = 0
9
10 while count < 5:
11 print 'Programming is fun!'
12 count += 1
13

在PyCharm中执行Debug



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
```

```
RRRRRRRRRRUCCCCCCCCCC
```

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

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

```
Py
End
```

- 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

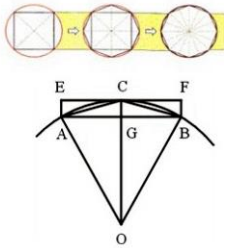
```
for s in "Python":
    if s=="t":
        continue
    print(s, end = "")
print("\nEnd")
```

```
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

《九章算术注》



  - 莱布尼茨级数
  - 分析方法

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

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：

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

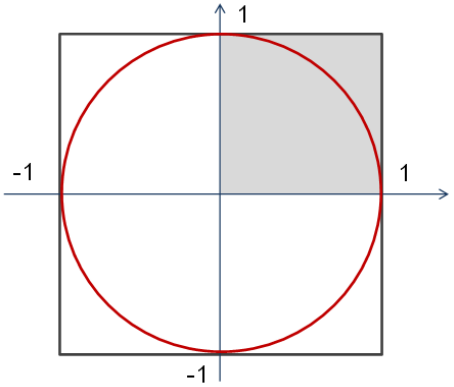
  - **蒲丰投针**
  - 随机抽样或统计试验方法

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≤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：

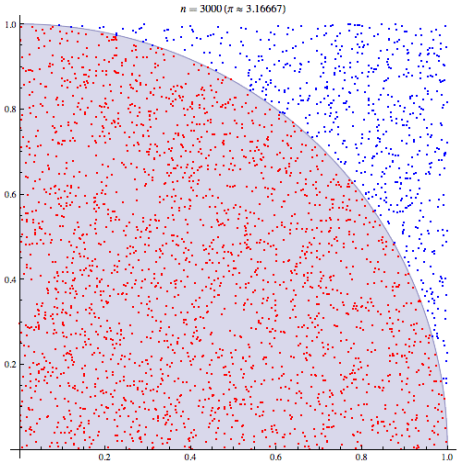
$p = \frac{2l}{\pi a}$（其中π为圆周率）

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**π/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?

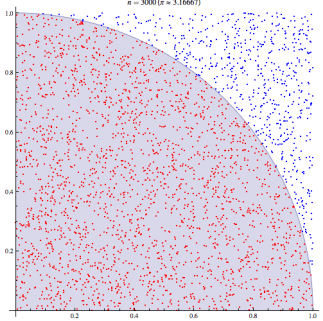
- Python的random库
  - 提供了不同类型的随机数函数，所有函数都是基于最基本的random.random()函数扩展而来

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}	3.1426696777	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.
```

课后作业

- 阅读教材第二章

- 思考：如何使用蒙特卡罗法计算e?

谢谢！