1. 程序的控制结构

1、程序的分支结构

单分支结构：根据判断条件结果而选择不同向前路径的运行方式。

If <条件>:

<语句块>

例如：

guess=eval(input())

if guess==99:

print("猜对了")

二分支结构：根据判断条件结果而选择不同向前路径的运行方式。

If <条件>:

<语句块1>

Else:

<语句块2>

例如：

guess=eval(input())

if guess==99:

print("猜对了")

else:

print("猜错了")

紧凑形式：适用于简单表达式的二分支结构

<表达式1>if<条件>else<表达式2>

例：

guess=eval(input())

print("猜{}了".format("对" if guess==99 else "错"))

多分支结构：

If<条件>:

<语句块1>

elif:

<语句块2>

…

else:

<语句块2>

条件判断及组合：

用于条件组合的三个保留字

|  |  |
| --- | --- |
| 操作符及使用 | 描述 |
| x and y | 两个条件x和y的逻辑与 |
| x or y | 两个条件x和y的逻辑或 |
| nor x | 条件x的逻辑非 |

程序的异常处理：

异常处理的基本使用：

try:

<语句块1>

except:

<语句块2>

增加一个异常标记

try：

<语句块1>

except<异常类型>:

<语句块2>

异常处理的高级使用

try:

<语句块1>

except:

<语句块2>

else:

<语句块3>

finally:

<语句块4>

//finally对应语句块4一定执行

//else 对引发语句块3在不发生异常时执行

实例：身体质量指数BMI

定义;BMI=体重（kg）/身高的平方（m的平方）

国际：世界卫生组织 国内：国家卫生健康委员会

|  |  |  |
| --- | --- | --- |
| 分类 | 国际BMI值 | 国内BMI值 |
| 偏瘦 | <18．5 | <18.5 |
| 正常 | 18.5~25 | 18.5~24 |
| 偏胖 | 25~30 | 25~28 |
| 肥胖 | >=30 | >=28 |

问题需求：

输入：给定体重和身高值

输出：BMI指标分类信息（国际和国内）

思路方法：

难点在于同时输出国际和国内对应的分类

思路1：分别计算并给出国际和国内BMI分类

思路2：混合计算并给出国际和国内BMI分类

例：

#BMI1.py

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))

who,nat="",""

if bmi < 18.5:

who,nat= "偏瘦","偏瘦"

elif 18.5 <= bmi < 24:

who,nat= "正常","正常"

elif 24 <=bmi <25:

who,nat= "正常","偏胖"

elif 25 <=bmi < 28:

who,nat= "偏胖","偏胖"

elif 28 <=bmi < 30:

who,nat= "肥胖","肥胖"

2、程序的循环结构

遍历循环：遍历某个结构形成的循环运行方式。

for <循环变量·> in <遍历结构>:

<语句块>

从遍历结构中逐一提取元素，放在循环变量中。

由保留字for和in组成，完整便利所有元素后结束。

每次循环，所获得元素放入循环变量，并执行一次语句块。

应用;

1. 计数循环（N次）

for I in range(N):

<语句块>

遍历由range()函数产生的数字序列，产生循环。

for i in range(M,N,K):

<语句块>

遍历由range()函数产生的数字序列，产生循环

从M开始以K为步长到N结束。

1. 字符串遍历循环

for c in s:

<语句块>

S是字符串，遍历字符串每个字符，产生循环。

1. 列表循环遍历

for item in ls

<语句块>

ls是一个列表，遍历其每个元素，产生循环。

1. 文件遍历循环

for line in fi

<语句块>

fi是一个文件标识符，遍历其每行，产生循环

无限循环：由条件控制的循环运行方式

while <条件>:

<语句块>

是反复执行语句块，直到条件不满足时结束。

Ctrl+c组合键退出执行

循环控制保留字

break和 continue

break跳出并结束当前整个循环，执行循环后的语句。

continue结束当次循环，继续执行后续次数循环。

break和continue可以与for和while循环搭配使用

循环的高级用法

循环的扩展

循环与else

for <循环变量> in <遍历结构>:

<语句块1>

else：

<语句块2>

当循环没有被break语句退出时，执行else语句块。

这里的else与异常捕获的else类似

模块3：random库的使用

1、random库基本介绍

random库是使用随机数的python标准库。

伪随机数：采用梅森旋转算法生成的（伪）随机序列中元素

random库主要用于生成随机数。

使用random库：impor random

random库包括两类函数，常用的共8个

基本随机数函数：seed()，random()

扩展随机数函数:randint(), getrandbits(), uniform(), randrange(), choice(), shuffle()

2、基本随机数函数：

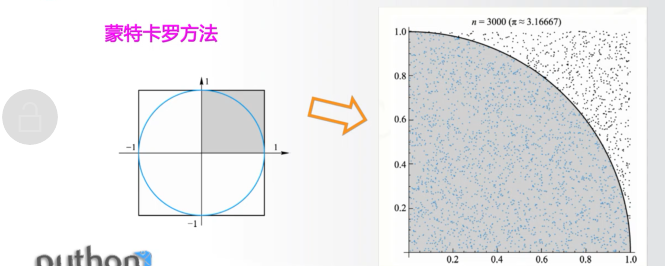
产生随机数需要一个随机数种子——然后通过梅森旋转算法——产生随机数列

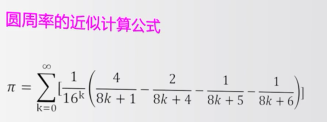
|  |  |
| --- | --- |
| 函数 | 描述 |
| seed(a=None) | 初始化给定的随机数种子，默认为当前系统时间 |
| random() | 生成一个0.1~1.0之间的随机小数 |

3、扩展随机数函数

|  |  |
| --- | --- |
| 函数 | 描述 |
| randint(a,b) | 生成一个[a，b]之间的整数 |
| randrange(m,n[,k]) | 生成一个[m,n]之间以k为步长的随机整数 |
| getrandbits(K) | 生成一个K比特长的随机整数 |
| uniform(a,b), | 生成一个[a,b]之间的随机小数 |
| choice(seq) | 从序列seq中随机选择一个元素 |
| shuffle(seq) | 将序列seq中元素随机排列，返回打乱后的序列 |

实例：圆周率的计算  
蒙特卡罗方法：





例：

#yuanzhoulv.py

pi=0

N=100

for k in range(N):

pi += 1/pow(16,k)\*( \

4/(8\*k+1) - 2/(8\*k+4) - \

1/(8\*k+5) - 1/(8\*k+6))

print("圆周率值是:{}".format(pi))

例：

#yuanzhoulv2.py

from random import random

from time import perf\_counter

DARTS=1000\*1000

hits=0.0

start=perf\_counter()

for i in range(1,DARTS+1):

x,y=random(), random()

dist=pow(x \*\* 2 + y \*\* 2,0.5)

if dist <=1.0:

hits=hits+1

pi=4\*(hits/DARTS)

print("圆周率值是:{}".format(pi))

print("运行时间是:{:.5f}s".format(perf\_counter()-start))