

学习目标:

1. 了解程序的基本结构并绘制流程图
2. 掌握程序的分支结构
3. 运用if语句实现分支结构
4. 掌握程序的循环结构
5. 运用for语句和while语句实现循环结构
6. 掌握随机库的使用方法
7. 了解程序的异常处理及方法



第四章 程序的控制结构

目

录

CONTENTS

4.1 程序的基本结构

4.2 程序的分支结构

4.3 实例5 身体质量指数BMI

4.4 程序的循环结构

4.5 模块2 random库的使用

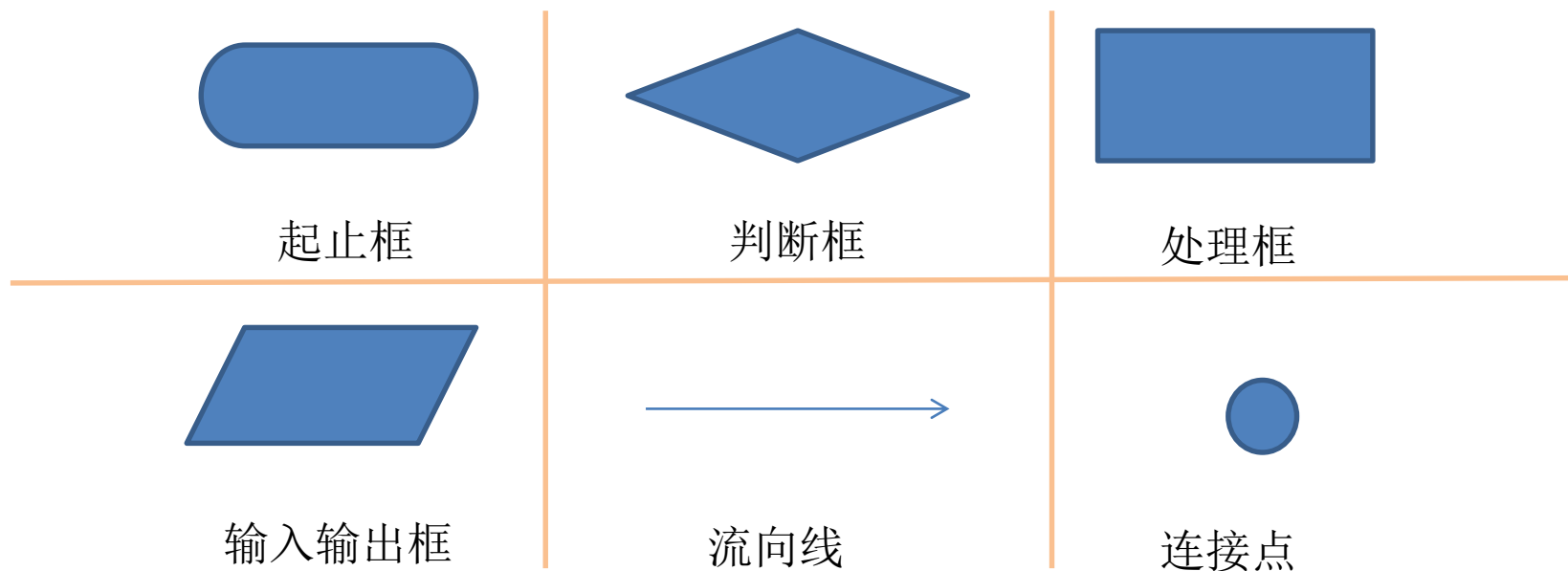
4.6 例6 圆周率的计算

4.7 程序的异常处理

程序流程图

流程图是一种描述程序执行流程的工具，是表达算法的常用工具。

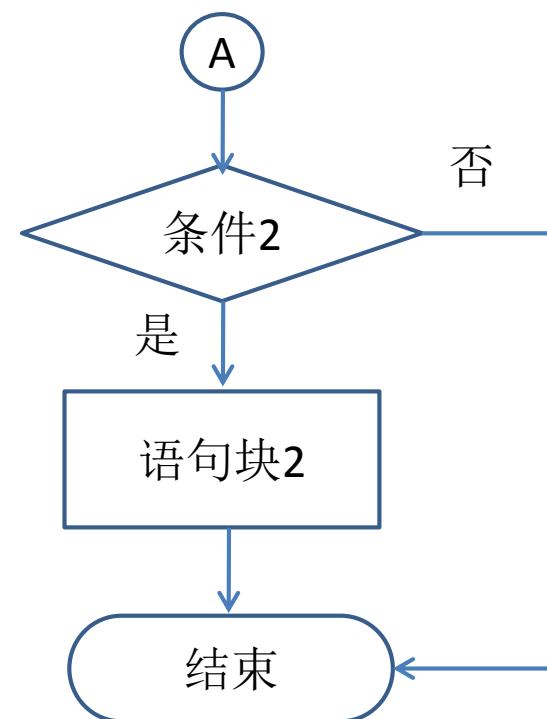
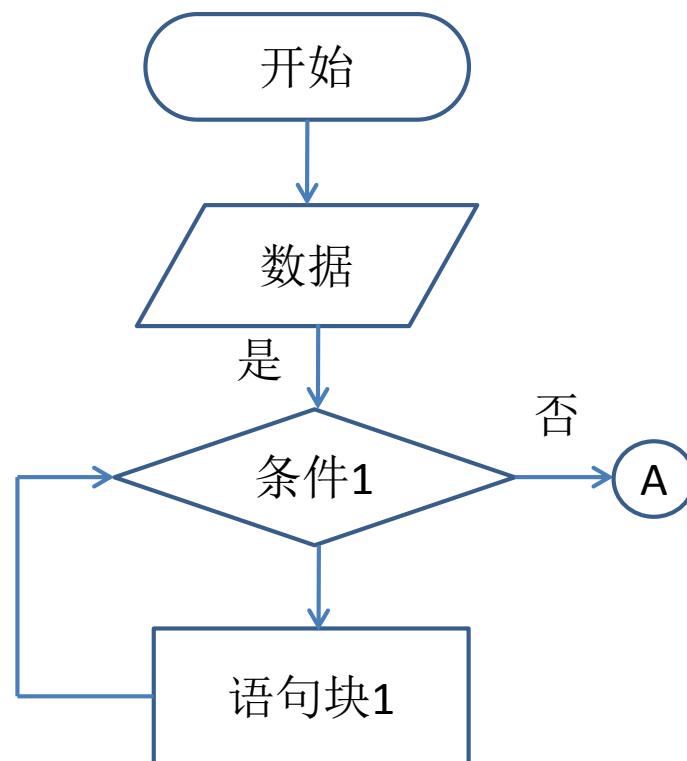
流程图的基本元素：



程序流程图

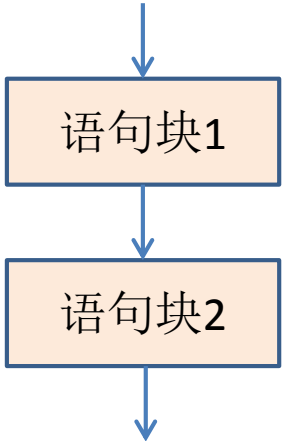
流程图是一种描述程序执行流程的工具，是表达算法的常用工具。

流程图的基本元素：

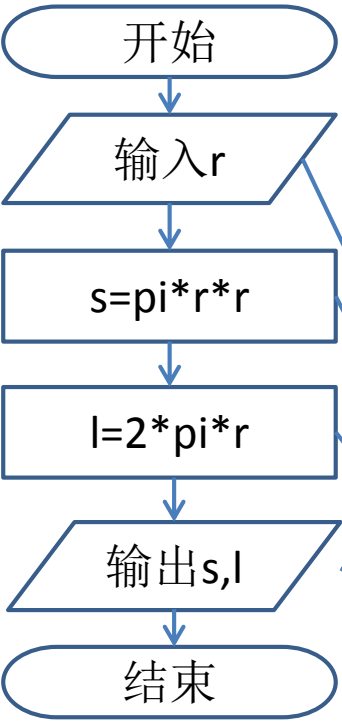


程序的基本结构

程序有三种基本结构组成：顺序结构、分支结构、循环结构



顺序结构

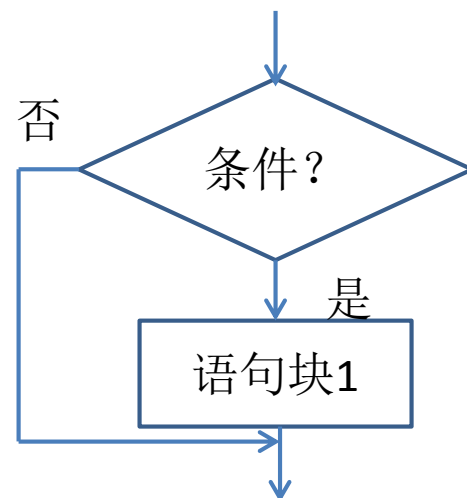


实例4.1：求圆面积和周长
输入：圆半径r
处理： $s=\pi*r*r$
 $l=2*\pi*r$
输出：圆面积、周长

```
import math
r=eval(input("输入半径:"))
s=math.pi*r*r
l=2*math.pi*r
print("圆的面积={}, 周长={}".format(s, l))
```

程序的基本结构

程序有三种基本结构组成：顺序结构、分支结构、循环结构



单分支结构

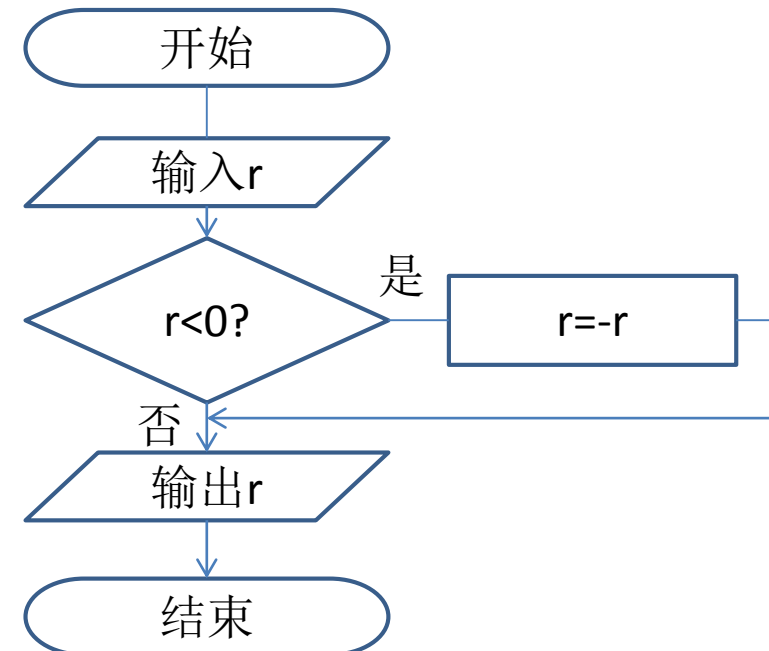
实例4.2：求实数的绝对值

输入：实数 r

处理： $|r| = \begin{cases} r & r \geq 0 \\ -r & r < 0 \end{cases}$

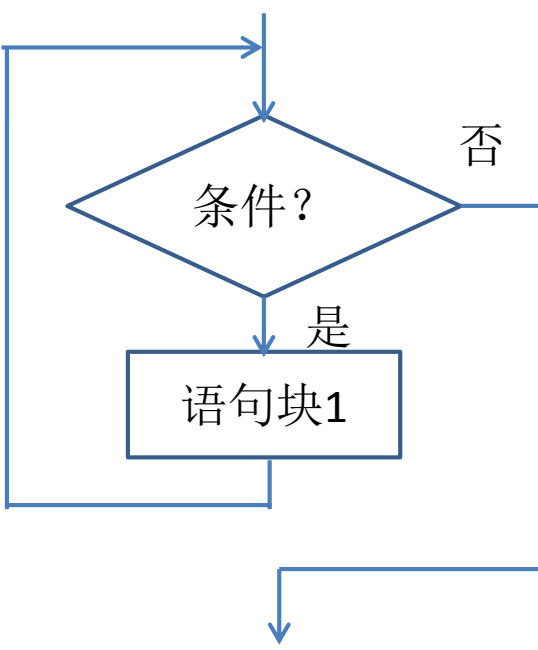
输出：输出 $|r|$

```
r=eval(input("输入实数:"))
if (r<0):
    r=-r
print("绝对值={} ".format(r))
```



程序的基本结构

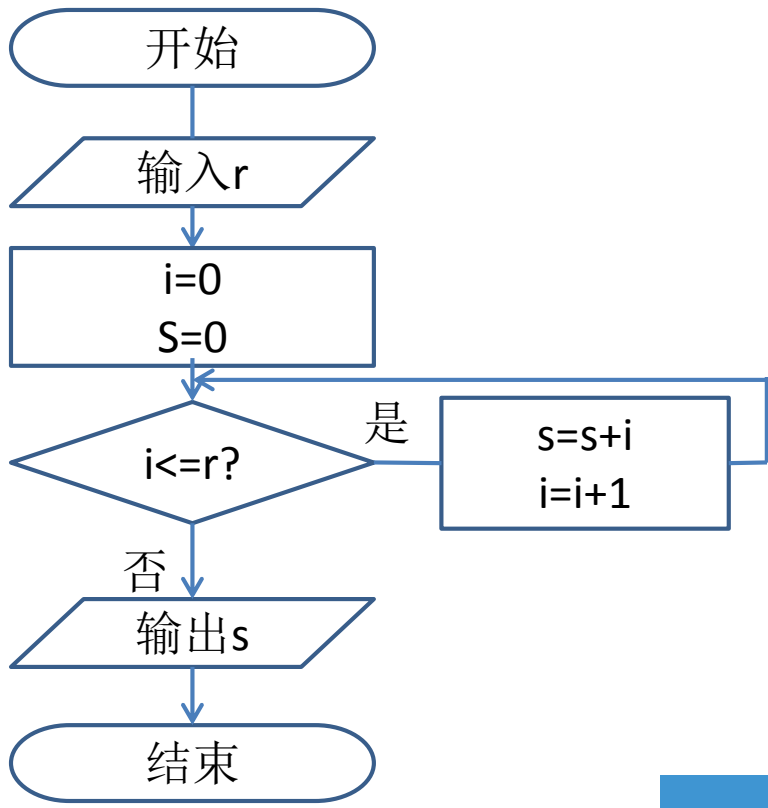
程序有三种基本结构组成：顺序结构、分支结构、循环结构



循环结构

实例4.3：求累加和
输入：正整数r
处理： $s=1+2+3+...+r$
输出：输出s

```
r=eval(input("输入r:"))  
i, s=0, 0  
while(i<=r):  
    s=s+i  
    i+=1  
print("累加求和=", s)
```



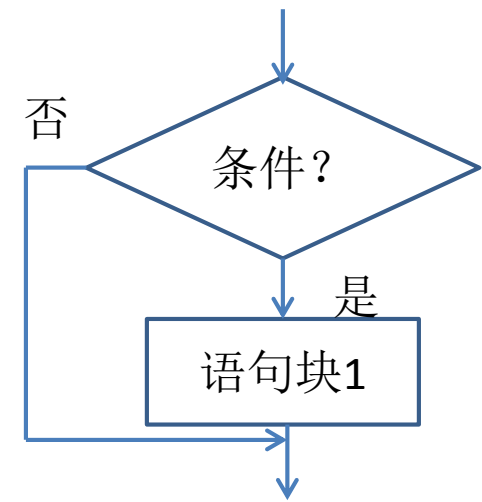
单分支结构

单分支结构的格式：

if <条件> :
 <语句块>

条件是通过6个关系操作符连接的表达式，
6个运算符的运算规则如右表所示：

| 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 |



```
#微实例，PM2.5空气质量提醒
pm=eval(input("请输入PM2.5的数值："))
if 0<=pm<=35:
    print("空气优良，快去户外运动！")
if 35<=pm<=75:
    print("空气良好，适度户外运动！")
if 75<=pm:
    print("空气污染，请小心！")
```


二分支结构：if-else

二分支结构的格式：

if <条件> :

<语句块1>

else:

<语句块2>



if pm>=75:

print("空气存在污染，请小心")

else:

print("空气没有污染，可进行户外运动！")

二分支结构的简洁格式：

<表达式1> if <条件> else <表达式2>



print("空气{}污染！").format("存在" if pm>=75 else "没有")

多分支结构：if-elif-else

二分支结构的格式：

if <条件1> :

 <语句块1>

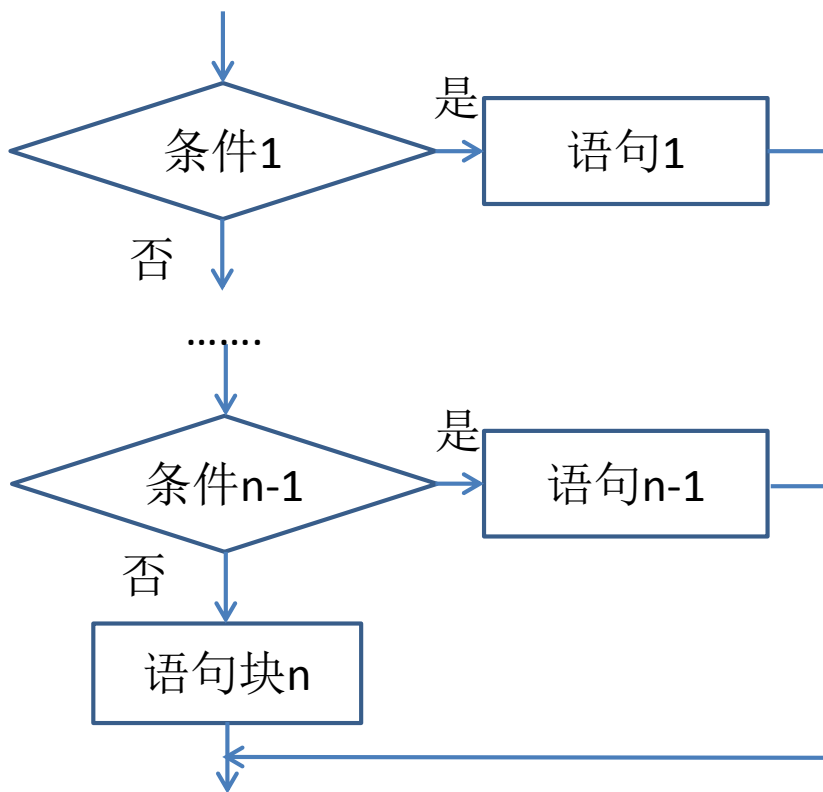
elif <条件2>

 <语句块2>

.....

else:

 <语句块N>



```
pm=eval(input("请输入PM值"))
if 0<pm<35:
    print("空气优质，快去运动")
elif 35<=pm<75:
    print("空气良好，适度运动")
else:
    print("空气污染，请小心！")
```

猜生日：依次询问某人的生日是否在五个集合中，然后就能计算出该生日

| | | | |
|----|----|----|----|
| 1 | 3 | 5 | 7 |
| 9 | 11 | 13 | 15 |
| 17 | 19 | 21 | 23 |
| 25 | 27 | 29 | 31 |

| | | | |
|----|----|----|----|
| 2 | 3 | 6 | 7 |
| 10 | 11 | 14 | 15 |
| 18 | 19 | 22 | 23 |
| 26 | 27 | 30 | 31 |

| | | | |
|----|----|----|----|
| 4 | 5 | 6 | 7 |
| 12 | 13 | 14 | 15 |
| 20 | 21 | 22 | 23 |
| 28 | 29 | 30 | 31 |

+ =19

| | | | |
|----|----|----|----|
| 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 |
| 28 | 29 | 30 | 31 |

```
ls1=[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31]
ls2=[2, 3, 6, 7, 10, 11, 14, 15, 18, 19, 22, 23, 26, 27, 30, 31]
ls3=[4, 5, 6, 7, 12, 13, 14, 15, 20, 21, 22, 23, 28, 29, 30, 31]
ls4=[8, 9, 10, 11, 12, 13, 14, 15, 24, 25, 26, 27, 28, 29, 30, 31]
ls5=[16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]
day=0
```

```
question1= 'Is your birthday in set1?'
question2= 'Is your birthday in set2?'
question3= 'Is your birthday in set3?'
question4= 'Is your birthday in set4?'
question5= 'Is your birthday in set5?'
print(ls1)
answer=eval(input(question1))
if answer==1:
    day+=1
```

```
print(ls2)
answer=eval(input(question2))
if answer==1:
    day+=2
print(ls3)
answer=eval(input(question3))
if answer==1:
    day+=4
print(ls4)
answer=eval(input(question4))
if answer==1:
    day+=8
print(ls5)
answer=eval(input(question5))
if answer==1:
    day+=16
print("your birthday is:"+str(day)+"!")
```

练习4.1 计算属相：

| Year%12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---------|---|---|---|---|---|---|---|---|---|---|----|----|
| 属相 | 猴 | 鸡 | 狗 | 猪 | 鼠 | 牛 | 虎 | 兔 | 龙 | 蛇 | 马 | 羊 |

练习4.2 克拉姆法则求解二元一次方程组：

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} e \\ f \end{bmatrix} \quad x = \frac{ed - bf}{ad - bc} \quad y = \frac{af - ec}{ad - bc}$$

输入a,b,c,d,e,f计算x和y的值，如果ad-bc等于零，则输出“无解”。

遍历循环：for语句

for循环的格式：

for <循环变量> in <遍历结构>:
 <语句块>

for循环的扩展形式：

for <循环变量> in <遍历结构>:
 <语句块1>
else :
 <语句块2>

遍历结构具体的用法如下：

- 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>

```
for s in "BIT":  
    print("循环进行中："+s)  
else:  
    s="循环结束"  
print(s)
```



运行结果为：

循环进行中：B
循环进行中：I
循环进行中：T
循环结束

无限循环：while语句

while循环的格式：

```
while <条件>:  
    <语句块>
```

while循环的扩展形式：

```
while <条件>:  
    <语句块1>  
else :  
    <语句块2>
```

```
s,idx="BIT",0  
while idx<len(s):  
    print("循环进行中："+s[idx])  
    idx+=1  
else:  
    s="循环结束"  
print(s)
```

运行结果为：
循环进行中：B
循环进行中：I
循环进行中：T
循环结束

循环保留字：break和continue

break语句：结束整个循环

如：

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

执行结果为：

Py

continue语句：结束当次循环

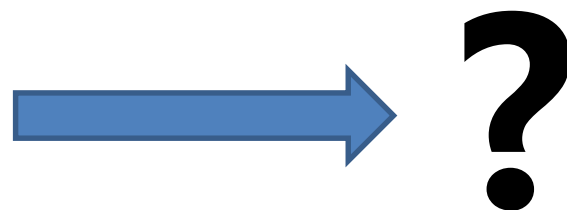
如：

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

执行结果为：

Pyhon

```
for s in "哇哈哈":  
    for l in range(10):  
        print(s,end="")  
        if s=="哇":  
            break
```



random库

random库主要用于产生各种分布的伪随机数，提供了9个随机数生成函数

| 函数 | 描述 |
|------------------------------|--------------------------------|
| seed(a=None) | 初始化随机数种子，默认为当前系统时间 |
| 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比特长度的随机数 |
| randrange(start,stop[,step]) | 生成一个[start,stop]之间以step为步长的随机数 |
| 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) | 将序列类型中的元素随机排列，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 |

使用random库的几个实例

```
>>> from random import *
>>> random()
0.18393265309331686
>>> uniform(1, 10)
7.434119572901974
>>> uniform(1, 20)
2.871294515332796
>>> randrange(0, 100, 4)
72
```

```
>>> choice(range(100))
57
>>> ls=list(range(10))
>>> print(ls)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> shuffle(ls)
>>> print(ls)
[7, 0, 5, 8, 9, 1, 2, 4, 6, 3]
```

生成随机数前，可以用seed()函数生成随机种子，在python中，只要随机种子相同，所产生的随机数是一样的，如：

```
from random import *
seed(125)
print("{}.{}.{}".format(randint(1,10),randint(1,10),randint(1,10)))
print("{}.{}.{}".format(randint(1,10),randint(1,10),randint(1,10)))
seed(125)
print("{}.{}.{}".format(randint(1,10),randint(1,10),randint(1,10)))
```

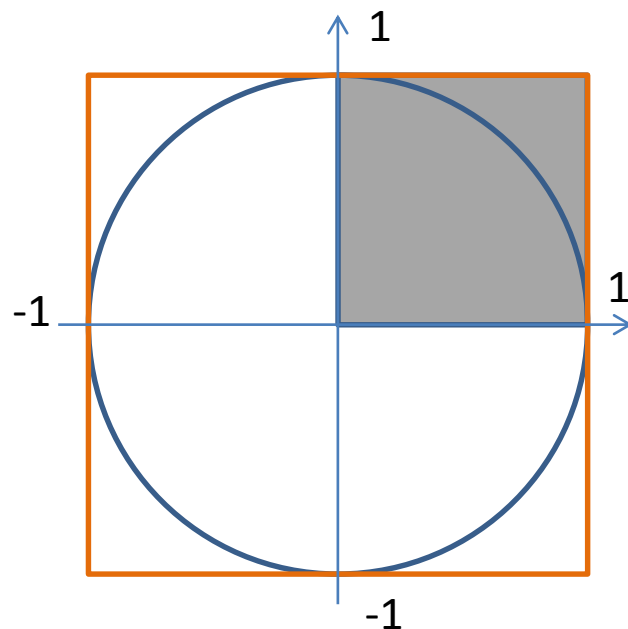
```
C:\compare_homework\venv\Scripts\py
4.4.10
5.10.3
4.4.10

Process finished with exit code 0
```

迄今为止，求解圆周率的最好办法是利用BBP公式求解

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

蒙特卡洛算法是利用计算机求解圆周率的有效方法，其原理如下图：

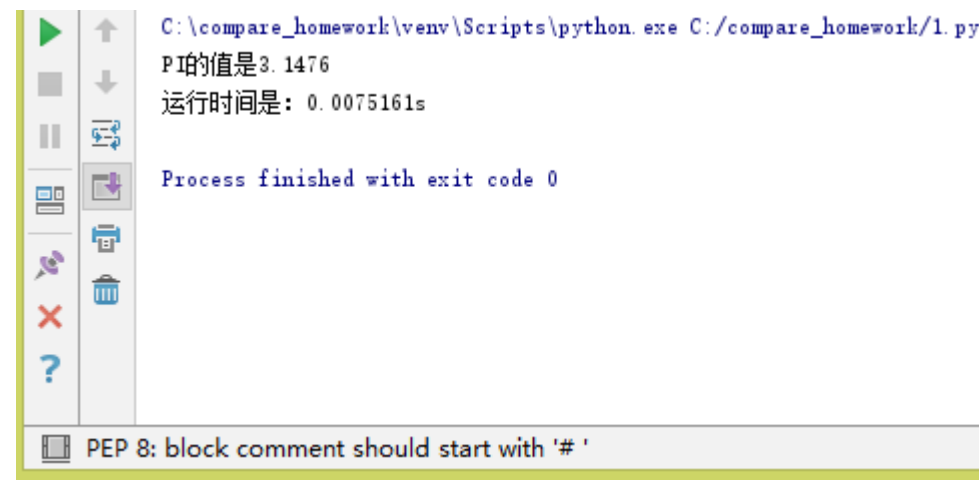


向正方形中随机投入飞镖，落入圆形中的飞镖的个数即为圆周率的值，为计算方便，我们选择正方形的1/4, 计算，小正方形的面积为1，扇形的面积为圆周率的1/4。

蒙特卡洛算法

```
#e6.1calpi.py
from random import random
from math import sqrt
from time import clock
darts=10000
hits=0.0
clock()
for i in range(1,darts+1):
    x,y=random(),random()
    dist=sqrt(x**2+y**2)
    if dist<=1.0:
        hits=hits+1
pi=4*(hits/darts)
print("PI的值是{}".format(pi))
print("运行时间是: {:5.5}s".format(clock()))
```

- ◆ **Clock()**开始计时，第二次启动clock时返回启动计时器后的时间；
- ◆ **Darts**为飞镖个数，**hits**为落在圆内的个数；
- ◆ 判断是否落入圆内：计算飞镖的坐标(x,y)到圆心的距离是否小于1



```
C:\compare_homework\venv\Scripts\python.exe C:/compare_homework/1.py
PI的值是3.1476
运行时间是: 0.0075161s

Process finished with exit code 0
```

PEP 8: block comment should start with '#'

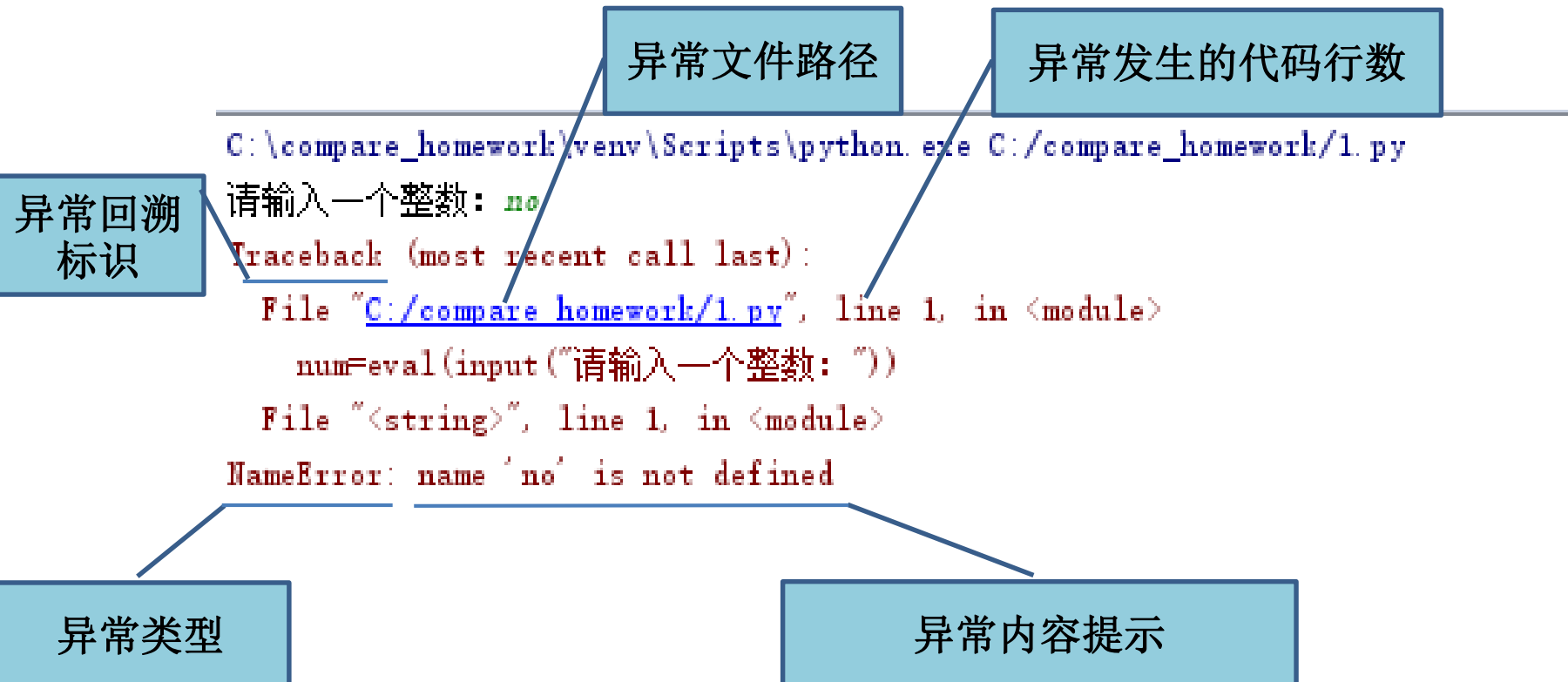
异常处理：try-except语句

有如下程序段：

```
num=eval(input("请输入一个整数："))
```

```
print(num**2)
```

当输入非数值型数据时，会产生如下的异常信息：



异常处理：try-except语句

try-except语句格式：

try:

<语句块1>

except <异常类型>:

<语句块2>

语句块1是正常执行的语句，当发生异常时
执行语句块2，


try:

num=eval(input("请输入一个整数："))

print(num**2)

except NameError:

print("输入错误，请输入一个整数！")



```
C:\compare_homework\venv\Scripts\python.exe C:/compare_homework/1.py
请输入一个整数: No
输入错误，请输入一个整数!

Process finished with exit code 0
```

异常的高级处理

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

```
try:
    <语句块1>
except <异常类型1>:
    <语句块2>
else:
    <语句块3>
finally:
    语句块4
```

- ◆ else后的<语句块3>是没有异常时接着<语句块1>执行的语句;
 - ◆ finally后的<语句块4>是无论异常还是正常都会执行的语句块
- 即: 正常时执行的语句顺序为:
 <语句块1>→<语句块3>→<语句块4>
 异常时执行的语句顺序为:
 <语句块2>→<语句块4>

```
def main():
    try:
        num1,num2=eval(input("enter two number,seperated by comma:"))
        result=num1/num2
        print("result is:",result)
    except ZeroDivisionError:
        print("Division by zero!")
    except SyntaxError:
        print("A comma may be missing in the input")
    except:
        print("Something wrong in the input")
    else:
        print("No exceptions")
    finally:
        print("The finally clause is excuted")
main()
```

```
C:\compare_homework\venv\Scripts\python.exe
enter two numbers, seperated by comma: 36, 24
result is: 1.5
No exceptions!
The finally clause is excuted!

Process finished with exit code 0
```

```
C:\compare_homework\venv\Scripts\python.exe
enter two numbers, seperated by comma: 36 24
A comma may be missing in the input!
The finally clause is excuted!

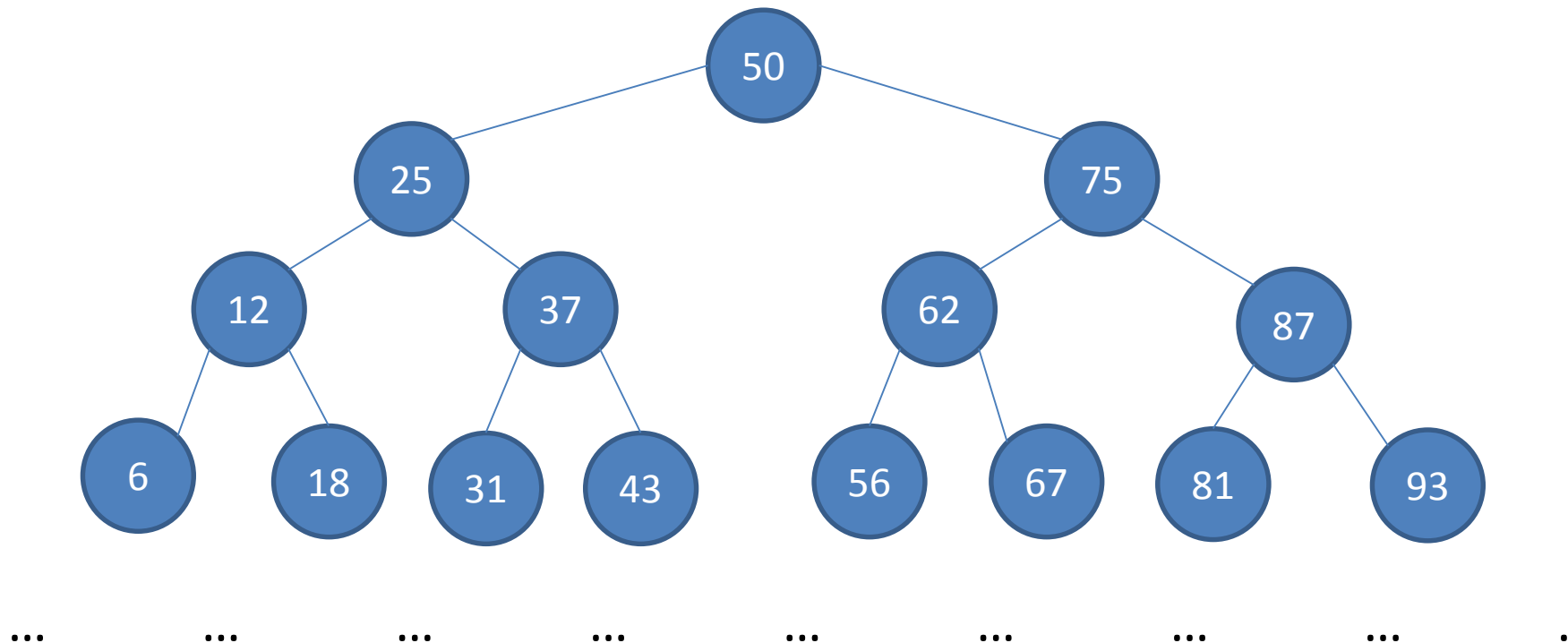
Process finished with exit code 0
```

```
C:\compare_homework\venv\Scripts\python.exe
enter two numbers, seperated by comma: 36, x
Something wrong in the input!
The finally clause is excuted!

Process finished with exit code 0
```

4.3 猜数游戏

生成一个1到100的随机整数`number`，利用循环完成猜数游戏：输入一个整数`guess`，如果`guess`大于`number`，提示“大了”；如果`guess`小于`number`，提示“小了”；如果`guess`等于`number`，输出`number`，并退出循环。



4.4 最大数和次大数

编程输入学生人数和每个学生的分数，利用循环找出最大数和次大数，并输出。

