第3章 Python程序的基本流程控制

- 掌握赋值语句和顺序结构
- 运用 if语句实现选择结构
- 运用 while语句和 for语句实现循环结构
- 掌握 break、continue 和 pass语句的用法
- 学会使用异常处理机制，完善程序。

3.1 基本语句及顺序结构

❖ 赋值语句

◆ 使用赋值符号 =，将右边的值（表达式）赋给左边变量的语句。

```
>>> name = "张三"
>>> age = 18
>>> score = 82.5
>>> value = 3 + 2j
```

◆ 变量的数据类型和初值在赋值时被初始化。

3.1 基本语句及顺序结构

❖ 复合赋值语句

◆ 复合赋值——利用复合赋值运算符

- 算术复合赋值运算符： +=　-=　*=　/=　//=　%=　**=
- 位复合运算符： <<=（左移等于）　>>=（右移等于）
  &=（与等于）　|=（或等于）　^=（异或等于）

```
>>> #位复合赋值运算
>>> age = 3
>>> age <<= 2 # age=age<<2, 将3左移2位, age变成12
>>> age &= 4  # age=age&4, 将12按位和4进行与运算, age变成4
>>> age
4
```

3.1 基本语句及顺序结构

❖ 复合赋值语句

◆ 多目标赋值—— 将同一个值赋给多个变量。

变量1 = 变量2 = … = 变量n = 表达式

如：>>> a = b = 10

赋值语句执行时，创建一个值为10的整型对象，将对象的同一个引用赋值给a和b，即a和b均指向数据对象10。

```
>>> #多目标赋值
>>> first = second = third = "welcome"
```

```
>>> #用序列赋值语句实现两个变量值的交换
>>> a, b = 10, 20
>>> a, b = b, a
>>> print(a, b)
20 10
```

3.1 基本语

❖ 复合赋值语句

◆ 序列赋值——同时为多个变量赋不同的值，变量之间用逗号隔开。

变量1，变量2，…，变量n = 表达式1，表达式2，…，表达式n

如：>>> a, b, c = 10, 20, 30

首先计算右边n个表达式的值，然后同时将表达式的值赋给左边的n个变量。

```
>>> #序列赋值
>>> name, age, score = "张三", 18, 82.5
```

3.1 基本语句及顺序结构

❖ 数据的输入

【例】从键盘输入语文、数学、英语三门功课的成绩，均成绩，结果保留1位。

输入样例:

```
80
85
90
```

输出样例:

```
85.0
```

```
# 逐行输入三个成绩
chin = float(input())
math = float(input())
engl = float(input())
ave = (chin + math + engl) / 3
print('%.1f' % ave )  # 输出保留1位小数
```

问题：针对不同格式的输入，如何正确接收数据？

输入样例:

```
80 85 90
```

输出样例:

```
85.0
```

输入样例:

```
80, 85, 90
```

输出样例:

```
85.0
```

3.1 基本语句及顺序结构

- 多个数据的输入——利用序列赋值
  - 字符串的split()方法

```
>>> x = input().split()
80 85 90
>>> x
['80', '85', '90']
```

```
>>> a, b, c = input().split()
80 85 90
>>> a
'80'
>>> b
'85'
>>> c
'90'
```

输入样例:

80 85 90

输出样例:

85.0

  - map( )函数

map(func, *iterables)

```
>>> y = map(float, x)
>>> y
<map object at 0x029C6A30>
>>> list(y)
[80.0, 85.0, 90.0]
```

```
>>> a, b, c = input().split(',')
80, 85, 90
>>> a
'80'
>>> b
' 85'
>>> c
' 90'
```

输入样例:

80, 85, 90

输出样例:

85.0

3.1 基本语句及顺序结构

❖ 多个数据的输入——利用序列赋值

◆ map( )函数　　map(func, *iterables)

将func作用于序列*iterebale，生成一个新的序列。

```
>>> a, b, c = map(float, input().split())
80 85 90
>>> a
80.0
>>> b
85.0
>>> c
90.0
```

输入样例:

```
80 85 90
```

输出样例:

```
85.0
```

```
>>> a, b, c = map(float, input().split(','))
80, 85, 90
>>> a
80.0
>>> b
85.0
>>> c
90.0
```

输入样例:

```
80, 85, 90
```

输出样例:

```
85.0
```

3.1 基本语句及顺序结构

- 多个数据的输入——利用序列赋值
  - 内置函数eval( )　　eval(source_s　　g)

将字符串string对象转化为有效的表达式，参与求值运算，返回计算结果。

```
>>> x = eval('2+3')
>>> x
5
>>> x = eval('80, 85, 90')
>>> x
(80, 85, 90)
```

```
>>> a, b, c = eval(input())
80, 85, 90
>>> a
80
>>> b
85
>>> c
90
```

注意：输入时要用逗号作为数据项的间隔

输入样例:

```
80, 85, 90
```

输出样例:

```
85.0
```

3.1 基本语句及顺序结构

❖ 格式化的输出

◆ 方法一：使用格式字符串

格式字符串 % (表达式1,…,表达式n)

格式字符串用于控制(表达式1,…,表达式n)的输出格式，格式字符串包括普通字符和格式字符。

```
>>> print('我的名字是%s' % '张三')
我的名字是张三
>>> print('%s的年龄是%d' % ('张三', 20))
张三的年龄是20
>>> print('共修了%d门课，平均成绩是%.2f分' % (5, 81.2378))
共修了5门课，平均成绩是81.24分
```

3.1 基本语句及顺序结构

❖ 格式化的输出

◆ 方法一：使用格式字符串

格式字符串 % (表达式1,…,表达式n)

常见的格式字符	含义	备注
%s	输出字符串	
%d	输出整数	
%f	输出小数	%.2f 保留2位小数
%c	输出字符chr(num)	
%o	以八进制格式输出	
%x 或 %X	以十六进制格式输出	
%e 或 %E	以科学计数法格式输出	

可进一步规范输出格式

3.1 基本语句及顺序结构

%[m.n]格式字符

格式化的辅助符号

辅助符号	
m	指定输出的宽度
-	在指定的宽度内输出
+	在输出的正数前显示
#	在输出的八进制数前 加'0x'
0	在指定宽度内输出值
.n	对于浮点数，输出时 入） 对于字符串，输出字符串的前n位

```
>>> test = 5000
>>> print('%6d' % test)
  5000
>>> print('%2d' % test)
5000
>>> print('%-6d%d' % (test, 123))
5000  123
>>> print('%+6d' % test)
 +5000
>>> print('%06d' % test)
005000
>>> print('%#o' % 64)
0o100
>>> print('%#x' % 64)
0x40
```

3.1 基本语句及顺序结构

❖ 格式化的输出

◆ 方法二：使用format( )方法

格式字符串.format(表达式1,…,表达式n)

格式字符串包括普通字符和格式字符。在形式上相当于通过{}来代替%。

格式字符的格式：{[序号][:格式字符]}

```
>>> print('%s的年龄是%d' % ('张三', 20))
张三的年龄是20
>>> print('{}的年龄是{}'.format('张三', 20))
张三的年龄是20
>>> print('{1}的年龄是{0}'.format(20, '张三'))
张三的年龄是20
>>> print('{}的成绩是{:.2f}'.format('张三', 80.567))
张三的成绩是80.57
```

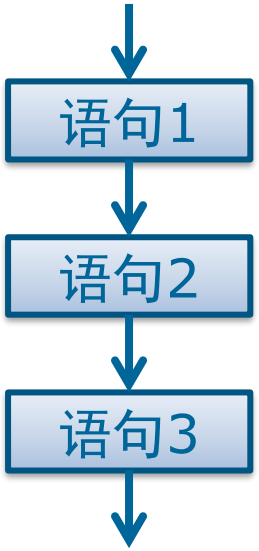
3.1 基本语句及顺序结构

- **程序工作的一般流程为：**

  数据输入→运算处理→结果输出

- **顺序结构**

  自上而下按顺序依次执行各语句。



3.1 基本语句及顺序结构

【课堂练习1】

- 键盘分别输入两点的坐标值（x1, y1），（x2, y2），求这两点之间的距离，结果保留2位小数。

输入样例:

```
1, 2
3, 4
```

输出样例:

```
dist = 1.41
```

输入样例:

```
1, 2, 3, 4
```

输出样例:

```
dist = 1.41
```

输入样例:

```
1  2  3  4
```

输出样例:

```
dist = 1.41
```

3.2 选择结构

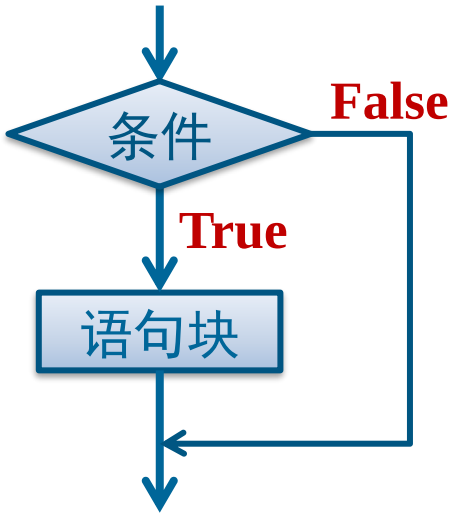
❖ 单分支

任意形式的表达式

```
if <条件>:
    <语句块>
```

【书写格式】

- if与表达式之间用空格隔开
- 表达式后接英文冒号
- 语句块中的全部语句均缩进4个空格



例：

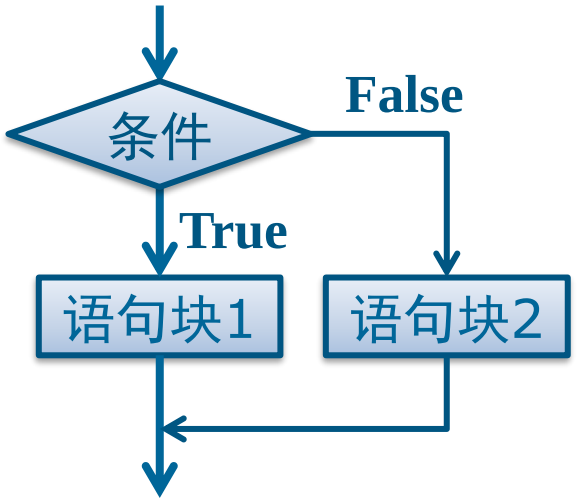
```
name = input("请输入您的姓名：")
age = eval(input("请输入您的年龄："))
if age>=18:
    print(name, "已经成年")
    print("符合驾照考试规定")
```

```
name=input("请输入您的姓名：")
age=eval(input("请输入您的年龄："))

print(name, "已经成年" if age>=18 else "未成年")
print(("" if age>=18 else "不")+"符合驾照考试规定")
```

❖ 双分支

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




```
    print(name,"已经成年")
    print("符合驾照考试规定")
else:
    print(name,"未成年")
    print("不符合驾照考试规定")
```

更简洁的表达：

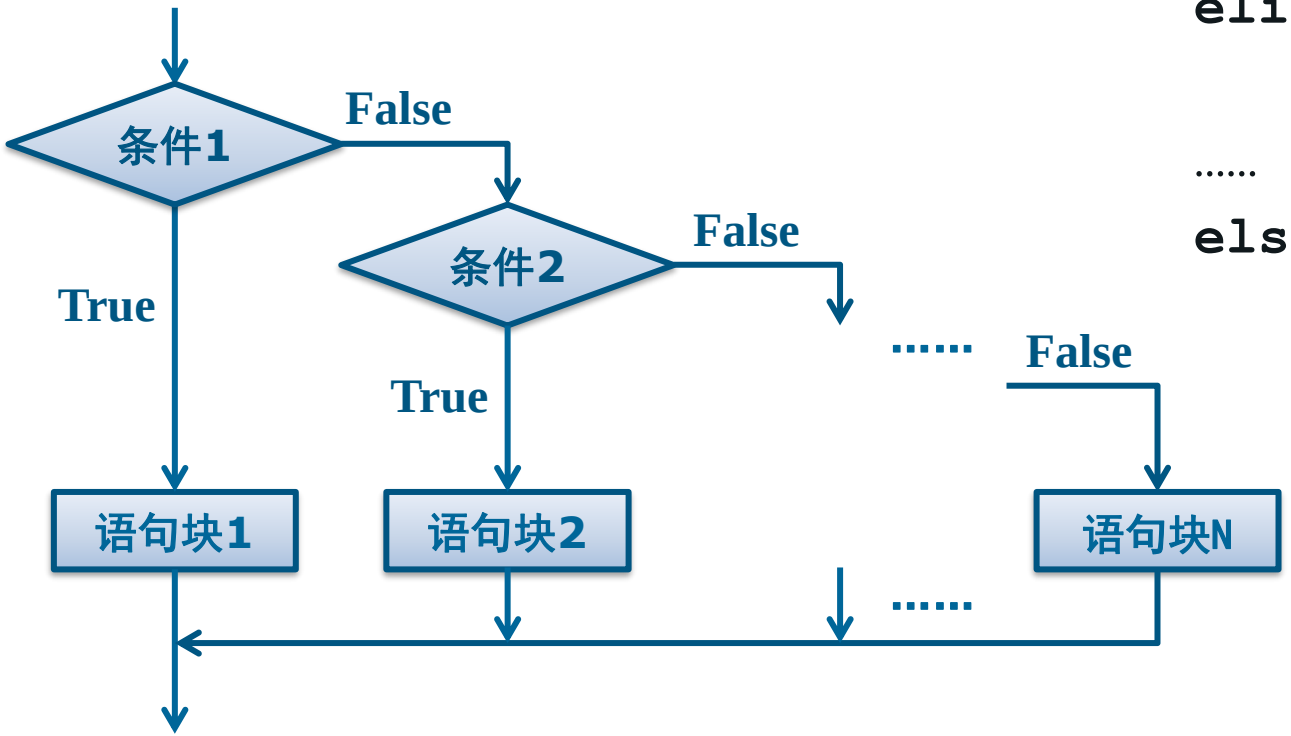
<表达式1> if <条件> else <表达式2>

多分支结构

【书写格式】

- 所有关键词elif均与if左对齐
- elif与后面的表达式之间用空格隔开，表达式后接英文冒号
- 所有语句块左对齐，即均缩进4个空格

```
if <条件1>:
    <语句块1>
elif <条件2>:
    <语句块2>
elif <条件3>:
    <语句块3>
……
else:
    <语句块N>
```



3.2 选择结构

❖ 多分支结构

```
name = input("name: ")
ch, ma, en = eval(input("chinese, math, english = : "))
average = (ch + ma + en )/3
if average >= 90:
    print(name, " 获一等奖")
elif average >= 80:
    print(name, " 获二等奖")
elif average >= 60:
    print(name, " 获三等奖")
else:
    print(name, " 没有获奖")
```

思考：各分支条件的顺序可以随意调换吗？

3.2 选择结构

❖ 分支语句嵌套

- 当有多个条件需要满足并且条件之间有递进关系时，可以使用分支语句的嵌套。
- if子句、elif子句、else子句中都可以嵌套if语句、或者if-else语句、或者if-elif-else语句。

```
if <条件1>:
    if <条件2>:
        <语句块A>
    else:
        <语句块B>
else:
    if <条件3>:
        <语句块C>
    else:
        <语句块D>
```

3.

【例】我
结婚年

【分析】

1. 输入
2. 先用
3. 输出

```
sex = input("请输入您的性别（M或者F）：")
age = int(input("请输入您的年龄（1-120）："))
if sex == 'M':
    if age >= 22:
        print("到达合法结婚年龄")
    else:
        print("未到合法结婚年龄")
else:
    if age >= 20:
        print("到达合法结婚年龄")
    else:
        print("未到合法结婚年龄")
```

3.2 选择结构

【例】 编写程序，从键盘输入用户名和密码。要求先判断用户名再判断密码，如果用户名不正确，则直接提示用户名输入有误；如果用户名正确，则进一步判断密码，并给出判断结果的提示。

【分析】

1. 键盘输入用户名
2. 先判断用户名，
3. 输出结论

```
username = input("请输入您的用户名：")
password = input("请输入您的密码：")
if username == "admin":
    if password == "123456":
        print("输入正确，恭喜进入！")
    else:
        print("密码有误，请重试！")
else:
    print("用户名有误，请重试！")
```

3.2 选择结构

【例】编写程序，开发一个小型计算器。从键盘输入一个简单算式，包括两个数字和一个运算符，根据运算符（+、-、*、/）进行相应的数学运算。如果不是这4种运算符，则给出错误提示。

【分析】

1. 先输入一行算式，得到两个数字及其运算符
2. 判断运算符，如果是+、-、*、/，则进行运算，否则给出错误提示
3. 如果是/，则判断第二个数字是否为0，若是0则出错提示。

输入样例:

```
2 + 3
```

输出样例:

```
2 + 3 = 5
```

3.2 选择结构

【例】编写程序，开发一个小型计算器。从键盘输入一个简单算式，包括两个数字和一个运算符，根据运算符（+、-、*、/）进行相应的数学运算

【分析】

1. 先输入一行算式
2. 判断运算符，如
   否则给出错误提
3. 如果是/，则判
   出错提示。

```
a, op, b = input().split()
a = int(a)
b = int(b)
if op == '+':
    print('{} + {} = {}'.format(a, b, a+b))
elif op == '-':
    print('{} - {} = {}'.format(a, b, a-b))
elif op == '*':
    print('{} * {} = {}'.format(a, b, a*b))
elif op == '/':
    if b!=0:
        print('{} / {} = {:.2f}'.format(a, b, a/b))
    else:
        print('除数不能为0!')
else:
    print('运算符错误! ')
```

【练习】4个人中有一个人做了好事，A说不是我做的，B说是C做的，C说是D做的，D说C胡说。已知3个人说了真话，编写程序判断是谁做的好事。

```
for x in 'ABCD':
    if (x!='A')+(x=='C')+\
       (x=='D')+(x!='D')==3:
        print(x+'做的。')
```

◈练习一

编写程序，计算0~1000内所有能被3和7都整

```
n=0                n=0
for i in ra        i=1
    if i%7     while i <= 1000:
           n +         if i%3 == 0:
           pri             if i%7 == 0:
print('一共                    n += 1
                               print(i,end=',')
                   i += 1
               print('一共有{}个'.format(n))
```

◈搬砖问题

36块砖36人搬，男搬4女搬3，两个小孩抬一砖。男女小孩各若干，刚好一次搬完砖？

【分析】穷举所有可能组合。

```
for m in range(37):
    for f in range(37):
        if 4*m+3*f+0.5*(36-m-f) == 36:
            print('男{},女{},小孩{}'.format(m,f,36-m-f))
```

◈编写程序，输出九九乘法表。


```
for i in range(1,10):
    for j in range(1,i+1):
        print('{}×{}={}'.format(i,j,i*j),end='  ')
    print()
```

【分析】循环的嵌套。

```
#循环的嵌套
for i in range(5):
    for j in range(6):
        print(i*j,end=' ')
```

◈练习二

猜数游戏：在程序中预设一个0~9之间的整数，让用户通过键盘输入所猜的数，如果大于预设的数，显示“Too big!”；小于预设的数，显示“Too small!”；猜对则显示“Succeed after n times!”，n为猜数字的次数。

```
n=5
N=0
while True:
    i=eval(input('Please input a number: '))
    if i > n:
        print('Too big!')
        N += 1
    elif i < n:
        print('Too small!')
        N += 1
    else:
        print('Succeed after %d times!'%N)
        break
```

◈练习

随机产生一个数字1~6，给用户三次机会猜测。程序给出提示（偏大或偏小），如果猜中，

```
import random as r
n = r.randint(1,6)
for i in range(3):
    c = eval(input('请猜测
    if c > n:
        print('偏大')
    elif c < n:
        print('偏小')
    else:
        print('猜对了！')
        break
else:
    print('机会用完。')
```

```
import random as r
n = r.randint(1,6)
for i in range(3):
    c = eval(input('请猜测（1到6）：'))
    if c == n:
        print('猜对了！''')
        break
    else:
        print('偏大'if c>n else '偏小')
else:
    print('机会用完。')
```

◈练习三

统计不同字符个数。用户从键盘输入一行字符，编写一个程序，统计并输出其中英文字符、数字、空格和

```
spa=char=num=other=0
s = input('Please input: ')
for i in s:
    if '0' <= i <='9':
        num += 1
    elif 'a' <= i <='z' or 'A' <= i <='Z':
        char += 1
    elif i == ' ':
        spa += 1
    else:
        other += 1
print('There are %d numbers, %d chars, %d spaces and \
%d other characters.'%(num,char,spa,other))
```

◈练习四

从键盘接收两个整数，编写程序求出这两个整数的最大公约数和最小公倍数。

```
m = int(input('请输入一个整数: '))
n = int(input('请再输入一个整数: '))
if m != n:
    if m > n:
        big = m
        small = n
    else:
        big = n
        small = m
    while m%small or n%small:
        small -= 1
else:
    small = m
print('两个数的最大公约数是{}，最小公倍
      .format(small,int(m*n/small)))
```

```
m = int(input('请输入一个整数: '))
n = int(input('请再输入一个整数: '))
if m != n:
    if m > n:
        big = m
        small = n
    else:
        big = n
        small = m
    while True:
        c = big % small
        if c != 0:
            big = small
            small = c
        else:
            break
else:
    small = m
print('两个数的最大公约数是{}，最小公倍数是{}'\
      .format(small,int(m*n/small)))
```