Python程序设计

第四讲 程序流程控制

选择结构

张 华
WHU

选择结构

- 条件表达式
- 选择结构
  - if
  - if-else
  - if-elif-else
- 条件运算
- 选择结构嵌套
- 程序设计举例

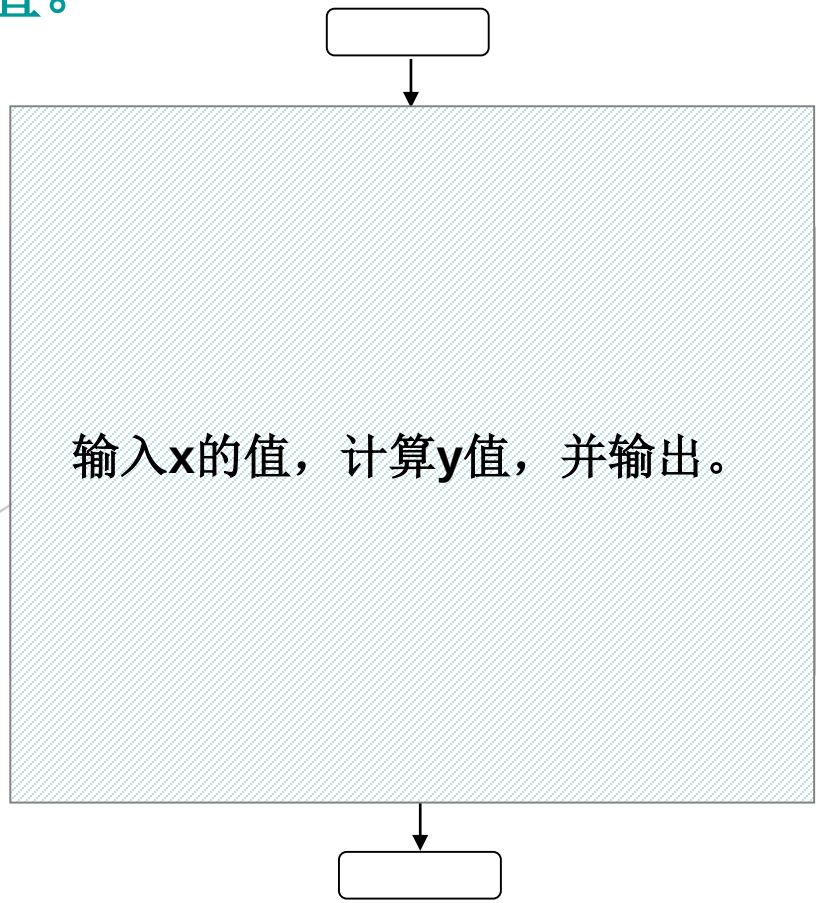
引例：程序流程控制

引例

- 问题：根据输入的x值，计算y值。

$$y = \begin{cases} x^2+1 & (x \le 2.5) \\ x^2-1 & (x > 2.5) \end{cases}$$

有了合适的数据类型和数据结构之后，还要依赖于选择和循环结构来实现特定的业务逻辑。



引例：程序流程控制

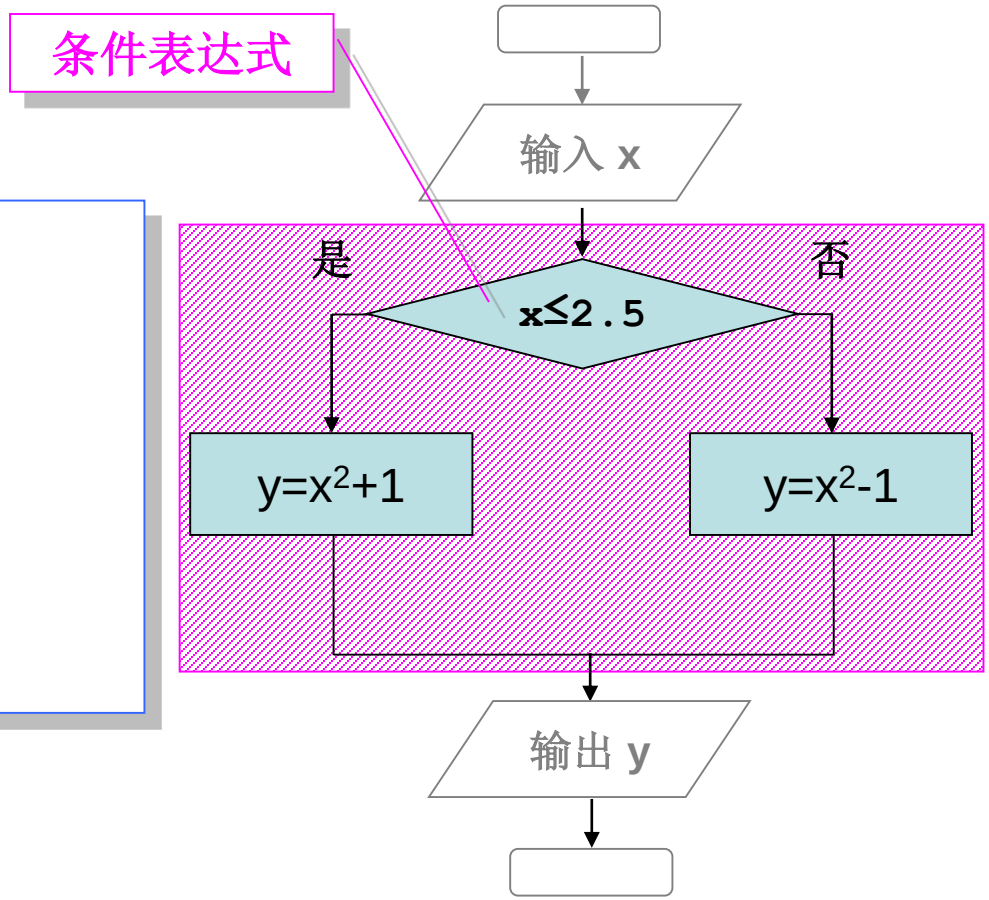
引例

源代码

```
x = float(input("x = "))

if x<=2.5:
    y = x**2+1
else:
    y = x**2-1

print("y =", y)
```



条件表达式

- **条件表达式的值**
  - 在选择和循环结构中，条件表达式的值只要不是False、0（或0.0、0j等）、空值None、空列表、空元组、空集合、空字典、空字符串、空range对象或其他空迭代对象，Python解释器均认为与True等价。
- **构造条件表达式主要使用的运算符**
  - 关系运算符
  - 逻辑运算符

条件表达式

关系运算符

< <= > >= == !=

- Python中的关系运算符可以连续使用，这样不仅可以减少代码量，也比较符合人类的思维方式。（从左到右）

```
>>> print(1<2<3)          #等价于1<2 and 2<3
True
>>> print(1<2>3)          #等价于1<2  and  2>3
False
>>> print(1<3>2)          #等价于1<3  and  3>2
True
```

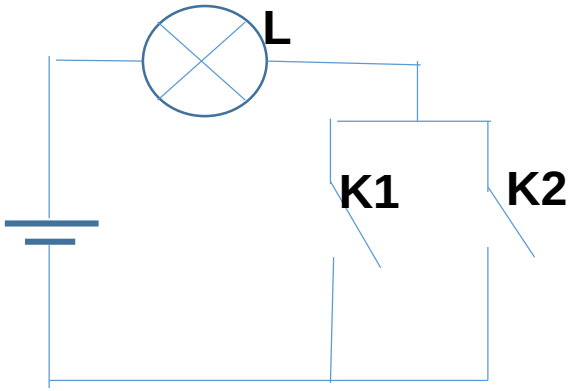
条件表达式

逻辑运算符

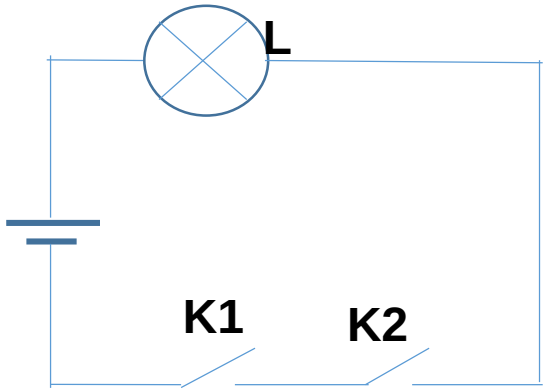
逻辑与：and

逻辑或：or

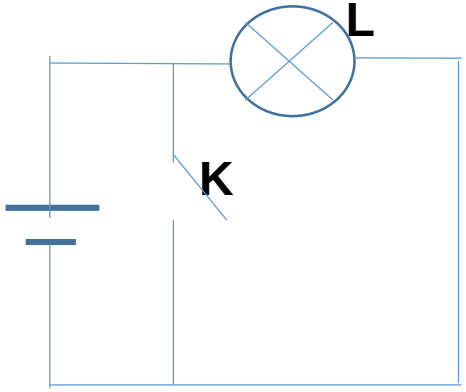
逻辑非：not



（1）or，并联电路



（2）and，串联电路



（3）not，短路

条件表达式

逻辑运算

- 逻辑运算符and和or具有短路求值或惰性求值的特点
  - 可能不会对所有表达式进行求值，而是只计算必须计算的表达式的值。
  - 以“and”为例，对于表达式“表达式1 and 表达式2”而言，如果“表达式1”的值为“False”或其他等价值时，不论“表达式2”的值是什么，整个表达式的值都是“False”，丝毫不受“表达式2”的影响，因此“表达式2”不会被计算。
  - 例如：gender=='F' and age>=60→FALSE
- 优化代码 a=1,b=2,c=3 a>b,b>c→FALSE
  - 在设计包含多个条件的条件表达式时，如果能够大概预测不同条件失败的概率，并将多个条件根据“and”和“or”运算符的短路求值特性来组织顺序，可以大幅度提高程序运行效率。

条件表达式

条件表达式举例

```
>>> 3<4 and 5<6
True
>>> 3 and 5
5
>>> 3 or 5
3
>>> 0 and 5
0
>>> 0 or 5
5
>>> not 3
False
>>> not 0
True
```

这时，结果是最后一个
被计算的表达式的值

选择结构

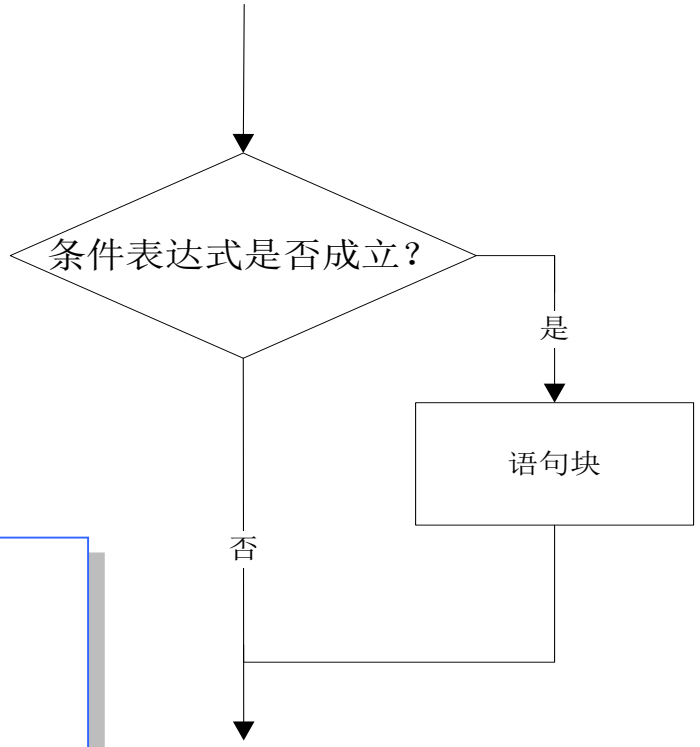
常见的选择结构

- 单分支选择结构
- 双分支选择结构
- 多分支选择结构
- 嵌套的分支结构
- 也可以构造跳转表来实现类似的逻辑
- 循环结构和异常处理结构中也可以带有“else”子句，可以看作是特殊形式的选择结构。

单分支选择结构

单分支选择结构

if 表达式:
    语句块



split：字符串的拆分（默认为所有的空字符）

```
# 按从小到大的顺序输出两个数
x = input('Input two number:')
a, b = map(int, x.split())

if a > b:
    a, b = b, a   #序列解包，交换两个变量的值

print(a, b)
```

```
Input two number:12 7
7 12
```

双分支选择结构

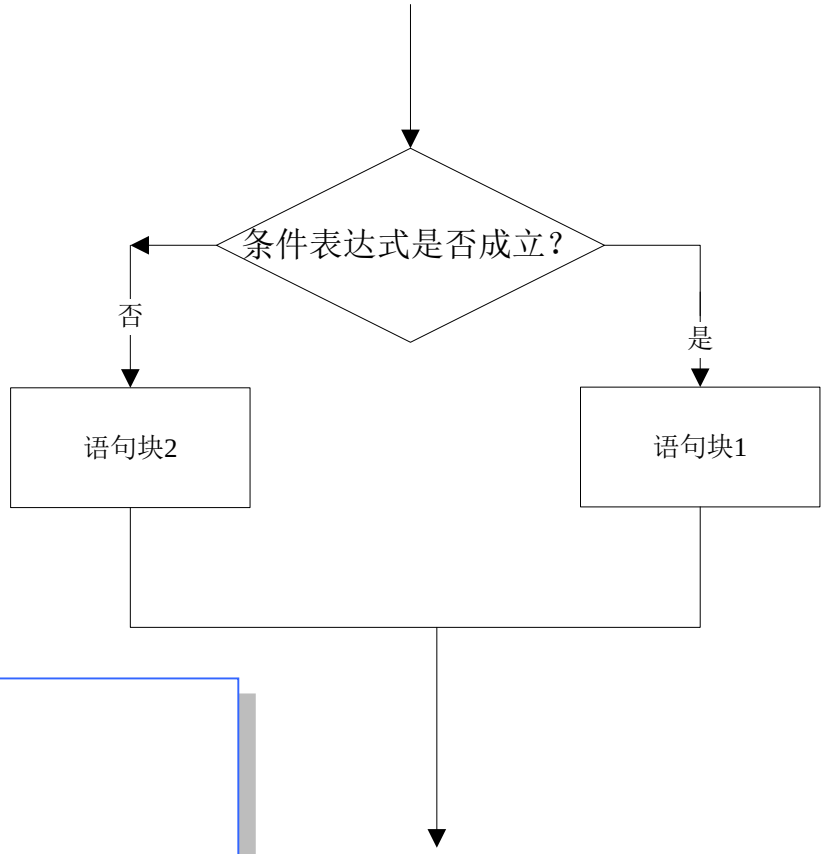
双分支选择结构

if 表达式:

语句块1

else:

语句块2




```
x = float(input("x = "))
if x<=2.5:
    y = x**2+1
else:
    y = x**2-1
print("y =", y)
```

双分支选择结构

案例：鸡兔同笼问题

- 输入一个笼子中鸡和兔的总数，以及腿的总数，计算鸡和兔的数量。

判断兔是否是整数

```
jitu, tui = map(int, input('请输入鸡兔总数和腿总数：').split())

tu = (tui - jitu*2) / 2
if tu>0 and int(tu) == tu:
    print('鸡：{0},兔：{1}'.format(int(jitu-tu), int(tu)))
else:
    print('数据不正确，无解')
```

```
请输入鸡兔总数和腿总数：5 14
鸡：3,兔：2
```

```
请输入鸡兔总数和腿总数：9 16
数据不正确，无解
```

条件运算

条件运算符

- 一个三元运算符，连接3个操作数构造条件表达式

  value1 if condition else value2

- 当条件表达式condition的值与True等价时，表达式的值为value1，否则表达式的值为value2。
- 例如：

```
>>> b = 6 if 5>13 else 9        #赋值运算符优先级非常低
>>> b
9
```

多分支选择结构

多分支选择结构

```
if 表达式1:
    语句块1
elif 表达式2:
    语句块2
elif 表达式3:
    语句块3
else:
    语句块4
```

其中，关键字elif是else if的缩写。

多分支选择结构

案例：

- 使用多分支选择结构将成绩从百分制变换到等级制。

```
if score > 100 or score < 0:
    print('wrong score.must between 0 and 100.')
elif score >= 90:
    print('A')
elif score >= 80:
    print('B')
elif score >= 70:
    print('C')
elif score >= 60:
    print('D')
else:
    print('E')
```

注意这里隐含的情况：

成绩小于90

多分支选择结构

案例：判断坐标点象限

- 已知坐标点(x,y)，判断其所在的象限。

```
x = int(input("请输入x坐标："))
y = int(input("请输入y坐标："))

if (x == 0 and y == 0): print("位于原点")
elif (x == 0): print("位于y轴")
elif (y == 0): print("位于x轴")
elif (x > 0 and y > 0): print("位于第一象限")
elif (x < 0 and y > 0): print("位于第二象限")
elif (x < 0 and y < 0): print("位于第三象限")
else: print("位于第四象限")
```

多分支选择结构

练习：出租车计价程序

- 下面是某城市出租车收费标准：
  - 起步里程为3公里，起步费10元；
  - 超起步里程后10公里以内，每公里2元；
  - 超过10公里以上的部分加收50%的回空补贴费，即每公里3元。
- 请根据该标准编写出租车计费程序，输入总里程数，输出应付费用。

多分支选择结构

出租车计价程序的参考实现

```
'''
    出租车计价程序
    某城市…
'''
import math

rawdis = float(input('输入总里程数（公里）：'))

dis = math.ceil(rawdis) # 不足1公里的算1公里，提醒用round函数存在的问题

fare = 10
if dis>10:
    fare += (dis-10)*3 + 7*2
elif dis>3:
    fare += (dis-3)*2

print('总里程数{:.1f}公里，应付{:.2f}元'.format(rawdis, fare))
```

```
输入总里程数（公里）：4.6
总里程数4.6公里，应付14.00元
```

```
输入总里程数（公里）：12.1
总里程数12.1公里，应付33.00元
```

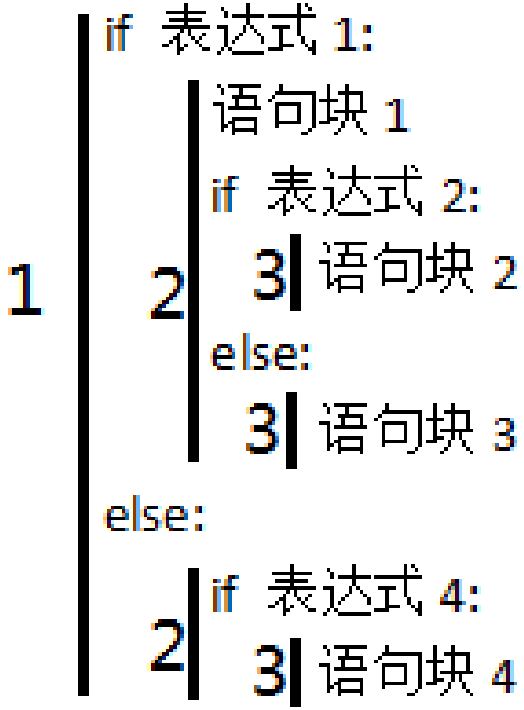
```
输入总里程数（公里）：3
总里程数3.0公里，应付10.00元
```

```
输入总里程数（公里）：3.1
总里程数3.1公里，应付12.00元
```

选择结构的嵌套

嵌套选择结构

```
if 表达式1:
    语句块1
    if 表达式2:
        语句块2
    else:
        语句块3
else:
    if 表达式4:
        语句块4
```



注意：缩进必须要正确并且一致。

选择结构的嵌套

思考：

- 下面三个if语句的区别

```
if (i>0):
    if (j>0): n=1
else: n=2
```

```
if (i>0):
    if (j>0): n=1
    else: n=2
```

```
if (i>0): n=1
else:
    if (j>0): n=2
```

选择结构的嵌套

案例：

- 使用嵌套选择结构将成绩从百分制变换到等级制。

```
score = int(input('输入百分制考试成绩：'))

if score > 100 or score < 0:
    print('输入无效。分数必须在0和100之间。')
else:
    degree = 'DCBAAE'
    print('对应的等级为', end=' ')

    index = (score - 60) // 10
    if index >= 0:
        print(degree[index])
    else:
        print(degree[-1])
```

end的作用：提示不换行

// 取整　% 取余

[x] list（可变序列）

degree[index]寻找index在degree对应的索引号
避免了使用多分支

程序设计举例

案例：

- 编写程序，计算今天是今年的第几天。

```
import time

date = time.localtime() #获取当前日期时间
year, month, day = date[:3]

day_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
if year%400==0 or (year%4==0 and year%100!=0): #判断是否为闰年
    day_month[1] = 29

if month==1:
    print(day)
else:
    print(sum(day_month[:month-1])+day)
```

小结

- 构造选择结构的流程控制语句
  - if
  - if-else
  - if-elif-else
- 智能的一个体现方面是根据实际环境来调节反应的能力，所以构造选择结构的语句是开发具有智能行为程序的基础。