

于 False，整个表达式就等价于 True。

```
>>> 3 and 5          #整个表达式的值是最后一个计算的子表达式的值
5
>>> 3 or 5
3
>>> 0 and 5          #0 等价于 False
0
>>> 0 or 5
5
>>> not [1, 2, 3]    #非空列表等价于 True
False
>>> not {}           #空字典等价于 False
True
```

4.2 选择结构

4.2.1 单分支选择结构

单分支选择结构语法如下所示，其中表达式后面的冒号“:”是不可缺少的，表示一个语句块的开始，并且语句块必须做相应的缩进，一般是以 4 个空格为缩进单位。

```
if 表达式:
    语句块
```

当表达式值为 True 或其他与 True 等价的值时，表示条件满足，语句块被执行，否则该语句块不被执行，而是继续执行后面的代码（如果有的话），如图 4-1 所示。

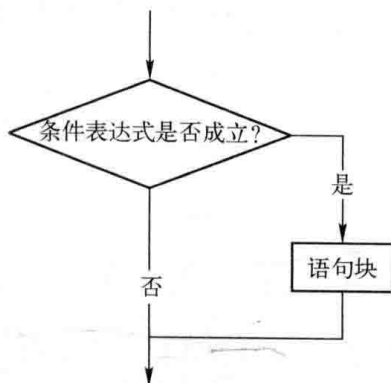


图 4-1 单分支选择结构

例 4-1 编写程序，输入使用空格分隔的两个整数，然后按升序输出。

```
1. x = input('Input two numbers:')    #input()函数返回字符串
2. a, b = map(int, x.split())          #split()方法使用空格对字符串进行切分
```

```

3. if a>b:
4.     a, b = b, a           #序列解包，交换两个变量的值
5. print(a, b)

```

4.2.2 双分支选择结构

双分支选择结构的语法如下。

```

if 表达式:
    语句块 1
else:
    语句块 2

```

当表达式值为 `True` 或其他等价值时，执行语句块 1，否则执行语句块 2。语句块 1 或语句块 2 总有一个会执行，然后再执行后面的代码（如果有的话），如图 4-2 所示。

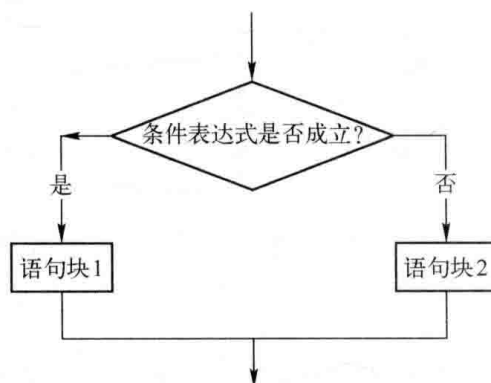


图 4-2 双分支选择结构

例 4-2 编写程序，使用双分支结构计算鸡兔同笼问题。

问题描述：鸡兔同笼问题是指已知鸡、兔总数量和腿的总数量，求解鸡、兔各多少只，这实际上是一个二元一次方程组的求解问题。根据数学知识容易知道，二元一次方程组如果有解应该只有唯一解。

基本思路：本例代码模拟的是下面的二元一次方程组求解过程，其中 `ji` 表示鸡的数量，`tu` 表示兔子的数量，`jitu` 表示鸡和兔子的总数量，`tui` 表示腿的总数量。

$$\begin{cases} ji + tu = jitu \\ 2ji + 4tu = tui \end{cases}$$

```

1. jitu, tui = map(int, input('请输入鸡兔总数和腿总数: ').split())
2. tu = (tui - jitu*2) / 2
3. if int(tu) == tu:
4.     print('鸡: {0},兔: {1}'.format(int(jitu-tu), int(tu)))
5. else:
6.     print('数据不正确，无解')

```

另外，Python 还提供了一个三元运算符，并且在三元运算符构成的表达式中还可以

嵌套三元运算符，可以实现与双分支选择结构相似的效果。语法如下。

```
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`。

```
>>> a = 5
>>> print(6 if a>3 else 5)
6
>>> b = 6 if a>13 else 9          #赋值运算符优先级非常低
>>> b
9
```

4.2.3 多分支选择结构

多分支选择结构的语法如下。

```
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 n
```

其中，关键字 `elif` 是 `else if` 的缩写。

例 4-3 编写程序，输入一个百分制考试成绩，然后输出对应的等级制成绩，要求使用多分支选择结构。

基本思路：考试成绩的百分制和等级制之间的对应关系为：[90, 100]区间上的分数对应 A，[80, 89]区间上的分数对应 B，[70, 79]区间上的分数对应 C，[60, 69]区间上的分数对应 D，小于 60 分的成绩对应 F。

```
1. score = int(input('请输入一个整数: '))
2. if score > 100 or score < 0:
3.     print('wrong score.must between 0 and 100.')
4. elif score >= 90:
5.     print('A')
6. elif score >= 80:
7.     print('B')
8. elif score >= 70:
9.     print('C')
10. elif score >= 60:
```



```
11.     print('D')
12. else:
13.     print('F')
```

4.2.4 选择结构的嵌套

选择结构可以进行嵌套，示例语法如下所示。

```
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
```

使用嵌套选择结构时，一定要严格控制好不同级别代码块的缩进量，这决定了不同代码块的从属关系和业务逻辑是否被正确实现，以及代码是否能够被解释器正确理解和执行。

例 4-4 编写程序，输入一个百分制考试成绩，然后输出对应的等级制成绩，要求使用嵌套的选择结构。

基本思路：首先检查输入的成绩是否介于 0~100，如果是的话再进一步计算其对应的字母等级。

```
1. score = int(input('请输入一个整数: '))
2. degree = 'DCBAAF' # [90, 99]区间和 100 都对应 A
3. if score > 100 or score < 0:
4.     print('wrong score.must between 0 and 100.')
5. else:
6.     index = (score - 60) // 10
7.     if index >= 0: # 这里对应 60 分以上的成绩
8.         print(degree[index])
9.     else:
10.         print(degree[-1]) # 60 分以下，对应 F
```

4.3 循环结构

4.3.1 for 循环与 while 循环

Python 主要有 for 循环和 while 循环两种形式的循环结构，多个循环可以嵌套使用，也可以和选择结构嵌套使用来实现复杂的业务逻辑。

在 Python 中，循环结构可以带 else 子句，其执行过程为：如果循环因为条件表达式