《Python统计计算》

（秋季学期）

翟祥
北京林业大学

编程基础

第3章 选择与循环

3.1 条件表达式

- 算术运算符：+、-、*、/、//、%、**
- 关系运算符：>、<、==、<=、>=、!=，可以连续使用，如

```
>>> print 1<2<3
True
>>> print 1<2>3
False
>>> print 1<3>2
True
```

- 测试运算符：in、not in、is、is not
- 逻辑运算符：and、or、not，注意短路求值
- 位运算符：~、&、|、^、<<、>>

3.1 条件表达式

- 在选择和循环结构中，条件表达式的值
  - False：False、0、空
    - 0：0、 0.0、 0j等
    - 空：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
  - 其余均认为与True等价
- 从这个意义上来讲，几乎所有的Python合法表达式都可以作为条件表达式，包括含有函数调用的表达式。

3.1 条件表达式

```
>>> if 3: #使用整数作为条件表达式
    print(5)
5
>>> a = [1, 2, 3]
>>> if a: #使用列表作为条件表达式
    print(a)
[1, 2, 3]
>>> a = []
>>> if a:
        print(a)
    else:
        print('empty')

empty
```

3.1 条件表达式

```
>>> i = s = 0
>>> while i <= 10: #使用关系表达式作为条件表达式
    s += i
    i += 1
>>> print(s)
55
>>> i = s = 0
>>> while True: #使用常量True作为条件表达式
        s += i
        i += 1
        if i > 10:
            break
>>>print(s)
55
>>> s = 0
>>> for i in range(0, 11, 1):
    s += i
>>> print(s)
55
```

3.1 条件表达式

- 比较特殊的运算符还有逻辑运算符“and”和“or”，这两个运算符具有短路求值或惰性求值的特点，简单地说，就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”，此时“表达式2”的值无论是什么都不影响整个表达式的值，因此将不会被计算，从而减少不必要的计算和判断。

3.1 条件表达式

- 例如，下面的函数用来使用用户指定的分隔符将多个字符串连接成一个字符串，如果用户没有指定分隔符则使用逗号。

```
>>> def Join(chList, sep=None):
    return (sep or ',').join(chList)  #利用or的短路运算
>>> chTest = ['1', '2', '3', '4', '5']
>>> Join(chTest)
'1,2,3,4,5'
>>> Join(chTest, ':')
'1:2:3:4:5'
>>> Join(chTest, ' ')
'1 2 3 4 5'
```

3.1 条件表达式

- 另外，在Python中，条件表达式中不允许使用赋值运算符“=”，避免了其他语言中误将关系运算符“==”写作赋值运算符“=”带来的麻烦，例如下面的代码，在条件表达式中使用赋值运算符“=”将抛出异常，提示语法错误。

```
>>> if a=3:
SyntaxError: invalid syntax
>>> if (a=3) and (b=4):
SyntaxError: invalid syntax
```

3.2.1 单分支选择结构

```
if 表达式:
    语句块

a,b=input('Input two number:')
if a>b:
   a,b=b,a
print a,b
```

3.2.2 双分支结构

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

```
>>> chTest = ['1', '2', '3', '4', '5']
>>> if chTest:
    print(chTest)
else:
    print('Empty')
['1', '2', '3', '4', '5']
```

3.2.2 双分支结构

- Python还支持如下形式的表达式：

```
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。另外，在value1和value2中还可以使用复杂表达式，包括函数调用和基本输出语句。

```
>>> a = 5
>>> print(6) if a>3 else print(5)
6
>>> print(6 if a>3 else 5)
6
>>> b = 6 if a>13 else 9
>>> b
9
```

下面的代码演示了这个结构的表达式也具有惰性求值的特点。

```
>>> x = math.sqrt(9) if 5>3 else random.randint(1, 100) #此时还没有导入math模块
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    x = math.sqrt(9) if 5>3 else random.randint(1,100)
NameError: name 'math' is not defined
>>> import math
#此时还没有导入random模块，但由于条件表达式5>3的值为True，所以可以正常运行
>>> x = math.sqrt(9) if 5>3 else random.randint(1,100)

#此时还没有导入random模块，由于条件表达式2>3的值为False，需要计算第二个表达式的值，因此出错
>>> x = math.sqrt(9) if 2>3 else random.randint(1, 100)
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    x = math.sqrt(9) if 2>3 else random.randint(1,100)
NameError: name 'random' is not defined
>>> import random
>>> x = math.sqrt(9) if 2>3 else random.randint(1, 100)
```

3.2.3 多分支结构

```
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
```

3.2.3 多分支结构

- 下面的代码演示了利用多分支选择结构将成绩从百分制变换到等级制的用法。

```
>>> def func(score):
if score > 100:
        return 'wrong score.must <= 100.'
elif score >= 90:
        return 'A'
elif score >= 80:
        return 'B'
elif score >= 70:
        return 'C'
elif score >= 60:
        return 'D'
elif score >= 0:
        return 'E'
else:
        return 'wrong score.must >0'
```

3.2.4 选择结构的嵌套

```
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。

3.2.4 选择结构的嵌套

```
>>> def func(score):
    degree = 'DCBAAE'
    if score > 100 or score < 0:
            return 'wrong score.must between 0 and 100.'
    else:
            index = (score - 60)//10
            if index >= 0:
                    return degree[index]
            else:
                    return degree[-1]
```

3.2.5 选择结构应用

- 例1：面试资格确认。

```
age=24
subject="计算机"
college="非重点"
if (age > 25 and subject==“电子信息工程”) or \
    (college=="重点" and subject=="电子信息工程") or \
    (age<=28 and subject=="计算机"):
     print("恭喜，你已获得我公司的面试机会!")
else:
     print("抱歉，你未达到面试要求")
```

3.2.5 选择结构应用

例3：编写程序，判断今天是今年的第几天。

```
import time
date = time.localtime()
year = date[0]
month = date[1]
day = date[2]
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)
```

3.2.5 选择结构应用

```
# 计算今天是今年的第几天
>>> today = datetime.date.today()
>>> today
datetime.date(2015, 7, 27)
>>> firstDay = datetime.date(today.year,1,1)
>>> firstDay
datetime.date(2015, 1, 1)
>>> daysDelta = today-
firstDay+datetime.timedelta(days=1)
>>> daysDelta.days
208
# 或者使用下面的方法
>>> datetime.date.today().timetuple().tm_yday
208
>>> datetime.date(2015,7,25).timetuple().tm_yday
206
```

3.3.1 for循环与while循环

- Python提供了两种基本的循环结构语句——while语句、for语句。
- while循环一般用于循环次数难以提前确定的情况，也可以用于循环次数确定的情况；
- for循环一般用于循环次数可以提前确定的情况，尤其是用于枚举序列或迭代对象中的元素；
- 相同或不同的循环结构之间都可以互相嵌套，实现更为复杂的逻辑。

3.3.1 for循环与while循环

while 表达式:

    循环体

for 取值 in 序列或迭代对象:

    循环体

3.3.1 for循环与while循环

- while循环和for循环都可以带else子句，当循环自然结束时（不是因为执行了break而结束）执行else结构中的语句。

```
while 表达式:
    循环体
else:
    else子句

for 取值 in 序列或迭代对象:
    循环体
else:
    else子句
```

- 例2：用户输入若干个分数，求所有分数的总分。每输入一个分数后询问是否继续输入下一个分数，回答“yes”就继续输入下一个分数，回答“no”就停止输入分数。

3.2.5 选择结构应用

```
endFlag = 'yes'
s = 0

while endFlag.lower() =='yes':
    x = input("请输入一个正整数: ")
    x = eval(x)
    if isinstance(x, int) and 0<=x<=100:
        s = s + x
    else:
        print('不是数字或不符合要求')
    endFlag = input('继续输入? (yes or no)')

print('整数之和=', s)
```

3.3.2 循环结构的优化

- 为了优化程序以获得更高的效率和运行速度，在编写循环语句时，应尽量减少循环内部不必要的计算，将与循环变量无关的代码尽可能地提取到循环之外。对于使用多重循环嵌套的情况，应尽量减少内层循环中不必要的计算，尽可能地向外提。

3.3.2 循环结构的优化

```
import time
digits = (1, 2, 3, 4)
start = time.time()
for i in range(1000):
    result = []
    for i in digits:
        for j in digits:
            for k in digits:
                result.append(i*100+j*10+k)
print(time.time()-start)
print(result)

start = time.time()
for i in range(1000):
    result = []
    for i in digits:
        i = i*100           #下一轮循环，i的值为多少？
        for j in digits:
            j = j*10        #下一轮循环，j的值为多少？
            for k in digits:
                result.append(i+j+k)
print(time.time()-start)
print(result)
```

3.3.2 循环结构的优化

- 另外，在循环中应尽量引用局部变量，因为局部变量的查询和访问速度比全局变量略快，在使用模块中的方法时，可以通过将其转换为局部变量来提高运行速度。例如下面的代码：

```
import time
import math

start = time.time() #获取当前时间
for i in xrange(10000000):
    math.sin(i)
print('Time Used:', time.time()-start) #输出所用时间

loc_sin = math.sin
start = time.time()
for i in xrange(10000000):
    loc_sin(i)
print('Time Used:', time.time()-start)
```

3.4 break和continue语句

- break语句在while循环和for循环中都可以使用，一般放在if选择结构中，一旦break语句被执行，将使得整个循环提前结束。
- continue语句的作用是终止当次循环，并忽略continue之后的语句，然后回到循环的顶端，提前进入下一次循环。
- 除非break语句让代码更简单或更清晰，否则不要轻易使用。

3.4 break和continue语句

- 下面的代码用来计算小于100的最大素数，请注意break语句和else子句的用法。

```
>>> for n in range(100, 1, -1):
        for i in range(2, n):
            if n%i == 0:
                break
        else:
            print(n)
            break
97
```

- 删除上面代码中最后一个break语句，则可以用来输出100以内的所有素数。

```
>>> for n in range(100, 1, -1):
        for i in range(2, n):
            if n%i == 0:
                break
        else:
            print(n, end=' ')
97 89 83 79 73 71 67 61 59 53 47 43 41 37 31 29 23 19 17 13 11 7 5 3 2
```

3.4 break和continue语句

- 警惕continue可能带来的问题：

```
>>> i=0
>>> while i<10:
        if i%2==0:
            continue
        print(i)
        i+=1
```

永不结束的死循环,Ctrl+C强行终止。

3.4 break和continue语句

- 这样子就不会有问题

```
>>> for i in range(10):
        if i%2==0:
            continue
        print(i, end=' ')

1 3 5 7 9
```

3.4 break和continue语句

```
>>> for i in range(10):
        if i%2==0:
            i+=1        #无法改变下一次循环i的值
            continue
        print(i, end=' ')

1 3 5 7 9
```

3.4 break和continue语句

- 每次进入循环时的i已经不再是上一次的i，所以修改其值并不会影响循环的执行。

```
>>> for i in range(7):
        print(id(i),':',i)
10416692 : 0
10416680 : 1
10416668 : 2
10416656 : 3
10416644 : 4
10416632 : 5
10416620 : 6
```

3.5 案例精选

- 例4：计算1+2+3+…+100 的值。

```
s=0
for i in range(1,101):
    s = s + i
print('1+2+3+…+100 = ', s)
print('1+2+3+…+100 = ', sum(range(1,101)))
```

3.5 案例精选

- 例5：输出序列中的元素。

```
a_list=['a', 'b', 'mpilgrim', 'z', 'example']
for i,v in enumerate(a_list):
    print('列表的第', i+1, '个元素是：', v)
```

3.5 案例精选

- 例6：求1~100之间能被7整除，但不能同时被5整除的所有整数 。

```
for i in range(1,101):
    if i % 7 == 0 and i % 5 != 0:
        print(i)
```

3.5 案例精选

- 例7：输出“水仙花数”。所谓水仙花数是指1个3位的十进制数，其各位数字的立方和等于该数本身。例如：153是水仙花数，因为153 = 1^3 + 5^3 + 3^3 。

```
for i in range(100,1000):
    ge =  i % 10
    shi = i // 10 % 10
    bai = i // 100
    if ge**3+shi**3+bai**3 == i:
        print(i)
```

3.5 案例精选

- 例8：求平均分。

```
score = [70, 90, 78, 85, 97, 94, 65, 80]
s = 0
for i in score:
    s += i
print(s/len(score))
print(sum(score)/len(score))
```

3.5 案例精选

- 例10：求200以内能被17整除的最大正整数。

```
for i in range(200,0,-1):
    if i%17 == 0:
        print(i)
        break
```

3.5 案例精选

例12：鸡兔同笼问题。假设共有鸡、兔30只，脚90只，求鸡、兔各有多少只。

```
for ji in range(0, 31):
    if 2*ji + (30-ji)*4 == 90:
        print('ji:', ji, ' tu:', 30-ji)
```

3.5 案例精选

例13：编写程序，输出由1、2、3、4这四个数字组成的每位数都不相同的所有三位数。

```
digits = (1, 2, 3, 4)
for i in digits:
    for j in digits:
        for k in digits:
            if i!=j and j!=k and i!=k:
                print(i*100+j*10+k)
```

3.5 案例精选

从代码优化的角度来讲，上面这段代码并不是很好，其中有些判断完全可以在外层循环来做，从而提高运行效率，即下面形式的代码运行效率比上面的一段代码要高一些。

```
digits = (1, 2, 3, 4)
for i in digits:
    for j in digits:
        if j==i:
            continue
        for k in digits:
            if k==i or k==j:
                continue
            print(i*100+j*10+k)
```

3.5 案例精选

例14：编写程序，生成一个含有20个随机数的列表，要求所有元素不相同，并且每个元素的值介于1到100之间。

```
import random
x = []
while True:
    if len(x)==20:
        break
    n = random.randint(1, 100)
    if n not in x:
        x.append(n)

print(x)
print(len(x))
print(sorted(x))
```

3.5 案例精选

- 例15 编写程序，计算理财产品收益。

理财产品比定期存款的周期短，利息和本金一起滚动，不同于定期存款收益的计算方法。

```
def licai(base, rate, days):
    #初始投资金额
    result = base
    #整除，用来计算一年可以滚动多少期
    times = 365//days
    for i in range(times):
        result = result +result*rate/365*days
    return result


#14天理财，利率0.0385，投资10万
print(licai(100000, 0.0385, 14))
```

3.6 作业

- 作业1：打印九九乘法表。
- 作业2：判断一个数是否为素数

谢谢Q/A