Python 程序设计基础

江西财经大学-朱文强

第三章

流程控制

流程控制

任何一种编程语言编写的**程序都是由顺序结构、条件结构、循环结构这3种基本流程结构组成**，实际上这也是我们解决任何复杂问题的基本结构，本章将介绍Python的条件结构、循环结构和循环控制语句。

1 条件结构

Python语言的语句默认是按书写顺序依次执行的，但有些情况下，需要根据条件情况，有选择性的执行某些语句。

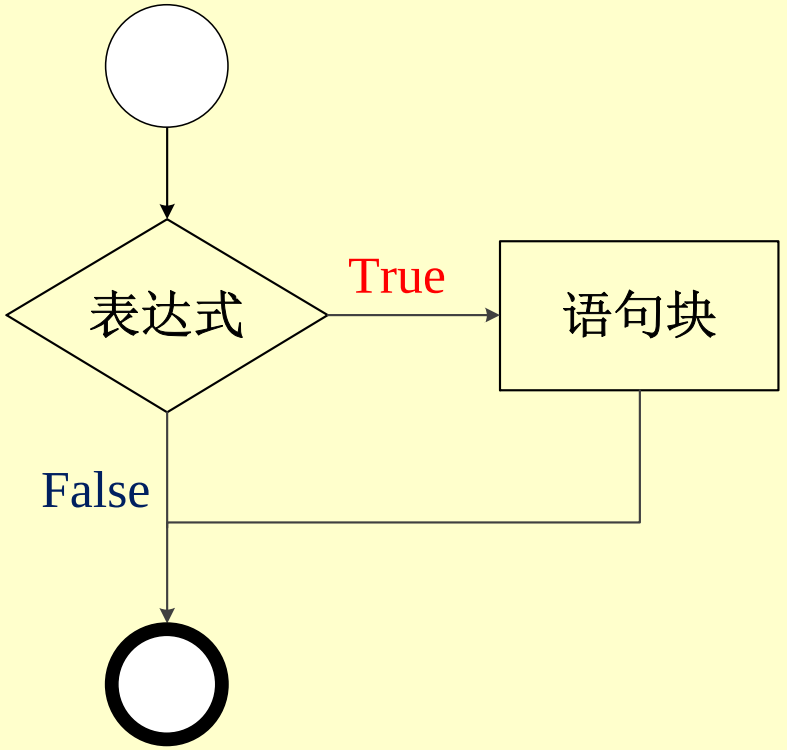
在Python语言中，主要有**3类条件结构：单项if语句、双向if-else语句、多分支if-elif-else语句**。以下依次对它们进行讲解。

1.1 单向if语句

单向if语句只有if没有else语句，只针对满足条件的情况做一些额外的操作，如果不满足条件则什么也不做。其执行流程如下图所示。语法格式如下。

if 布尔表达式:

语句块



1.1 单向if语句

```
age = int(input('请输入您的年龄:'))
#  如果语句块有多条语句, 要保持统一缩进格式
if age < 0 or age > 120:      #   表达式为True, 则执行语句块
    print('输入不合法, 采用默认值!')
    age = 20
print(age)
```

1.2 双向if-else语句

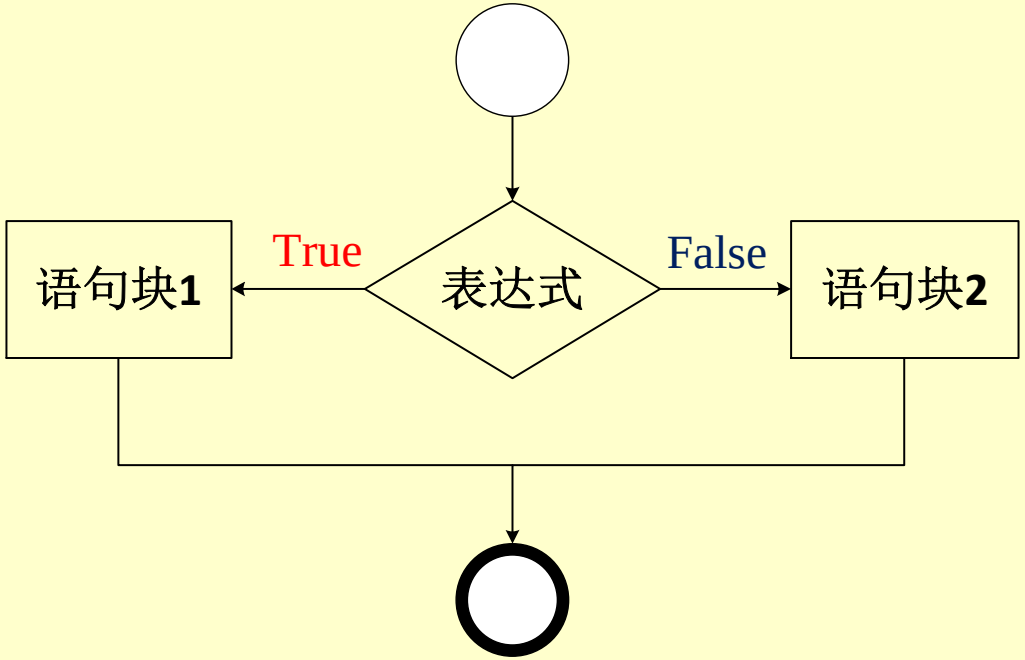
if-else语句是一种双向结构，**是对单向if语句的扩展，如果表达式结果为True，则执行语句块1，否则执行语句块2**。其执行流程如下图所示。语法格式如下。

if 布尔表达式:

　　语句块1

else:

　　语句块2

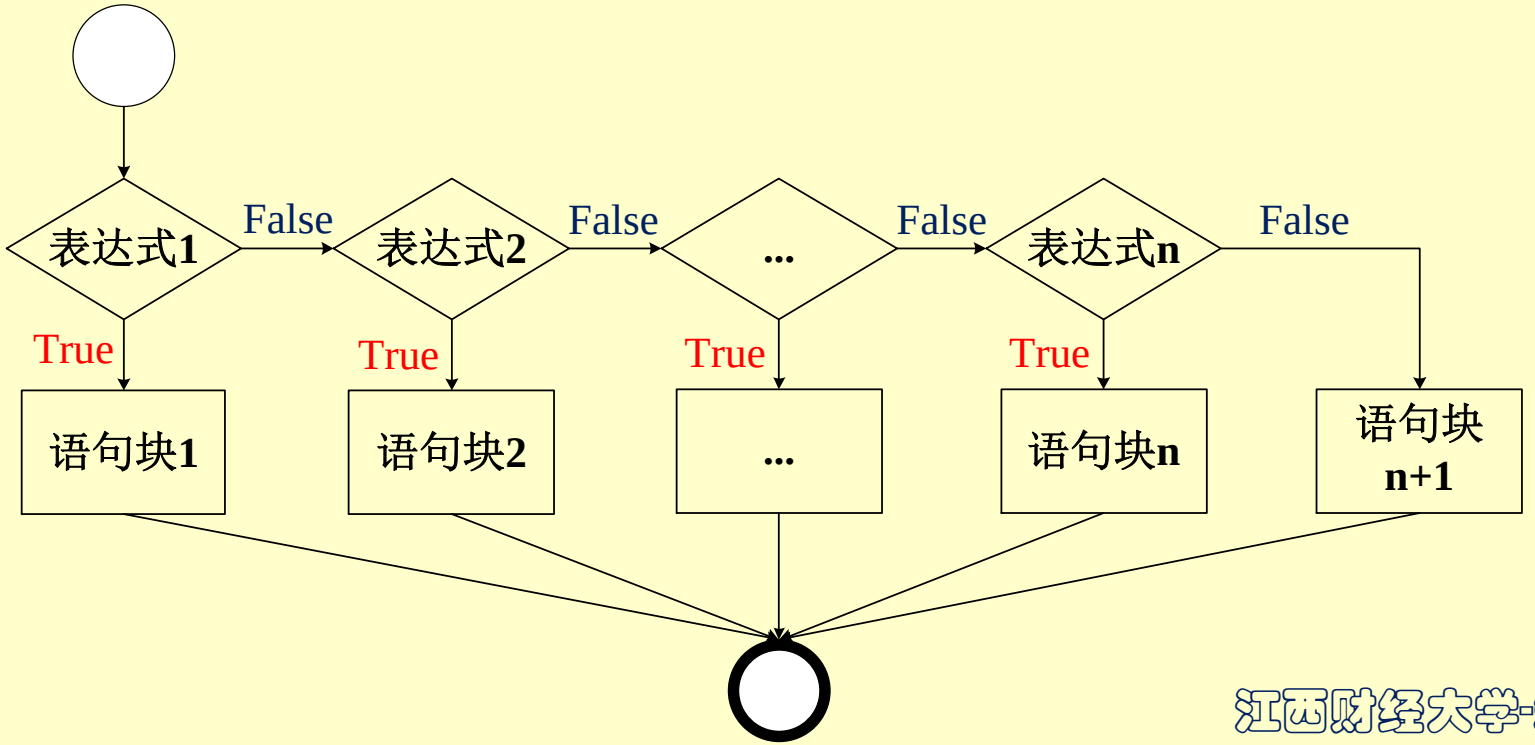


1.2 双向if-else语句

```
num = int(input("请输入一个整数:"))
if  num % 2 == 0:    #  判断num模2是否为0
    print("这是一个偶数！")
else:          #  else语句块的缩进要和if语句块缩进保持一致
    print(“这是一个奇数！”)
```

1.3 多分支if-elif-else语句

if-elif-else语句是一种多分支结构，**该语句利用一系列布尔表达式进行检查，并在某个表达式为真的情况下执行相应的一组代码**。if-elif-else语句的备选操作较多，但是**有且只有一组操作会被执行**。其执行流程如下图所示。



1.3 多分支if-elif-else语句

if-elif-else语句的语法结构如下。

```
if 布尔表达式1:
    语句块1
elif 布尔表达式2:
    语句块2
…
elif 布尔表达式n:
    语句块n
else:                          #  else语句块可选
    语句块n+1
```

1.3 多分支if-elif-else语句

if-elif-else语句的功能完全可以使用多层嵌套的if else语句等价实现，但代码会更加繁琐，且容易影响程序的可读性。并且通常来讲，**不建议if else语句的嵌套超过3层**。

```
score = float(input("请输入你的分数:"))
if score >= 90:
    grade = "优秀"
elif score >= 80:
    grade = "良好"
```

1.3 多分支if-elif-else语句

```
elif score >= 70:
    grade = "中等"
elif score >= 60:
    grade = "及格"
else:
    grade = "不及格"
print(score, "对应的等级为:",grade)
```

1.3 多分支if-elif-else语句

Python并不要求if-elif-else结构后面必须有else代码块，有时省略else语句反而逻辑更清晰，代码更安全**(else是一条不安全的语句，只要不满足if或elif中的条件测试，其代码就会执行，这可能会引入恶意的代码执行)**。

```
…… # __import__('os').startfile(r'c:\windows\notepad.exe')
elif score <= 60:     #  省略else, 代码却更安全(进一步优化?)
    grade = "不及格" #
print(score, "对应的等级为:",grade)
```

1.4 简化版if语句

条件判断的使用频率很高，为了简化条件判断语句的书写，Python中提供了简化版的if语句。语法结构如下。

表达式1 if 布尔表达式 else 表达式2

语法含义：**如果布尔表达式结果为True，那么整个语句的返回结果就是表达式1；否则，整个语句的返回结果就是表达式2**。

```
num1 = eval(input("请输入第一个数:"))
num2 = eval(input("请输入第二个数:"))
min = num1 if num1 < num2 else num2
```

1.4 简化版if语句

```
>>> age = 18
>>> vote = 'Adult' if age >= 18 else 'Youth'
>>> vote          #Adult
>>> PM = 76
>>> print(“轻度污染” if PM >= 76 else “良好")
#   轻度污染
```

1.4 简化版if语句

```
>>> def add(a, b):
        return a + b
>>> def subtract(a, b):
        return a - b
>>> a, b = 4, 5
print((subtract if a > b else add)(a, b))   #  9
```

2 循环结构

循环结构就是在一定条件下重复执行某些操作。和其他语言类似，Python提供了两种类型的循环语句：while条件式循环语句和for遍历式循环语句。

2.1 while循环

while循环是在条件满足的情况下，重复执行某段程序。基本形式为:

while 循环继续条件:

循环体

在while语句中，程序先判断循环继续条件，条件满足则执行循环体，执行完循环体后，再继续判断循环继续条件，满足条件再执行循环体，依次往复，直到循环继续条件不满足，才跳出循环。因此，需要在循环体中会对循环继续条件进行改变，以避免程序进入死循环。

2.1 while循环

```
index = 1

total = 0

while index <= 100:     # 当index小于等于100时, 执行循环体

    total = total + index    #  循环体代码块的缩进要统一

    index = index + 1

print(total)
```

2.2 for循环

for循环是一种遍历型的循环，它会依次对某个序列中全体元素进行遍历，遍历完所有元素之后便终止循环。常用于循环次数确定的场景。其基本形式为:

for 控制变量 in 可遍历序列:

循环体

在for循环语句中，控制变量是一个临时变量，可遍历序列保存了多个元素，将序列中每个元素依次赋值给控制变量后，程序执行循环体，循环体中一般会对控制变量进行操作，以保证循环能正常结束。

2.2 for循环

```
total = 0

for index in range(1, 101):      #  range返回值为[1~100]

    total = total + index      #   语句块必须保持缩进一致

print(total)
```

2.2 for循环

注意事项:

1、**循环内的语句必须缩进必须保持一致**，但不要求整个程序的缩进都统一**(一般都建议统一使用4个空格);**

2、**建议for循环临时变量使用单数, 而列表使用复数词;**

3、for、while语句末尾的冒号不能省略;

4、只要能够使用for循环, 就不要使用while循环;

5、for循环是一种遍历列表的有效方式, 但**在for循环中不应修改列表，否则将导致其难以跟踪列表中的元素；要在遍历列表的同时, 对列表进行修改, 可使用while循环**。

2.3 range()函数

在Python中，for循环语句经常结合range()函数一起使用，用于快速生成整数数字序列。以下是range()函数的语法格式。

range([start,] stop[, step])

函数说明:

(1)start: 计数从start开始, 可以没有, 缺省为0; 如range(0, 6)等价于range(6);

(2)stop: 计数到stop结束, 但不包括stop, 该参数必填。如range(1, 3)函数返回连续整数1, 2的序列;

2.3 range()函数

(3)step: 步长, 表示每次递增或递减的值, 缺省为1, 正数表示递增, 负数表示递减;

(4)start, stop, step 只能为整数, 不能为浮点数;

(5)该函数返回值为range对象，可通过循环遍历其元素或通过下标访问其元素。

2.3 range()函数

```
>>>a = range(1, 10)
>>>print(a)       #  range(1, 10)
>>>a[1]          #   2
>>>print(list(a))      #   [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>b = range(10, 1, -1)
>>>print(list(b))     #    [10, 9, 8, 7, 6, 5, 4, 3, 2]
>>>print(list(range(5)))     #    [0, 1, 2, 3, 4]
```

2.4 循环嵌套

在处理一些较为复杂的问题时, 可能会用到循环的嵌套。如在while循环中可以再嵌入while循环或for循环, 或在for循环中再嵌入for循环或while循环。**一般建议循环嵌套层次不要超过3层, 以保证程序的可读性**。

```
for i in range(1, 10):        #    打印九九乘法表
    for j in range(1, i+1):
        print(str(j)+"*"+str(i)+"="+str(i*j), end=" ")
    print("")
```

2.4 循环嵌套

注意: 在编写循环语句时, **应尽量减少循环内部不必要的计算**, 将与循环无关的代码尽可能放到循环之外。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, 尽可能将计算放在外层**。

2.5 for循环修改列表(进阶)

```
x = [1,2,1,2,1,2,1,2,1]

for i in x:          #   这个for循环不会有问题

    if i == 1:

        x.remove(i)

print(x)

x = [1,2,1,2,1,2,1,2,1,1,1]

for i in x:          #   这个for循环会出现问题

    if i == 1:
```

2.5 for循环修改列表(进阶)

```
        x.remove(i)
print(x)
x = [1,2,1,2,1,2,1,2,1,1,1]
while 1 in x:      #   while循环则完全没有问题
    x.remove(1)
print(x)
```

3 循环控制

3.1 循环控制语句

循环控制语句主要**包括break语句和continue语句**。

break语句用于终止或中断整个循环语句, 即使循环条件没有False或者序列还没被完全遍历完, 也会停止执行循环语句。**如果使用嵌套循环, break语句将跳出当前层次的循环**, 并开始执行当前层次循环语句外的下一行代码。

而**continue语句则是终止当次循环, 忽略continue之后的语句, 提前进入下一次循环过程**。

3.1 循环控制语句

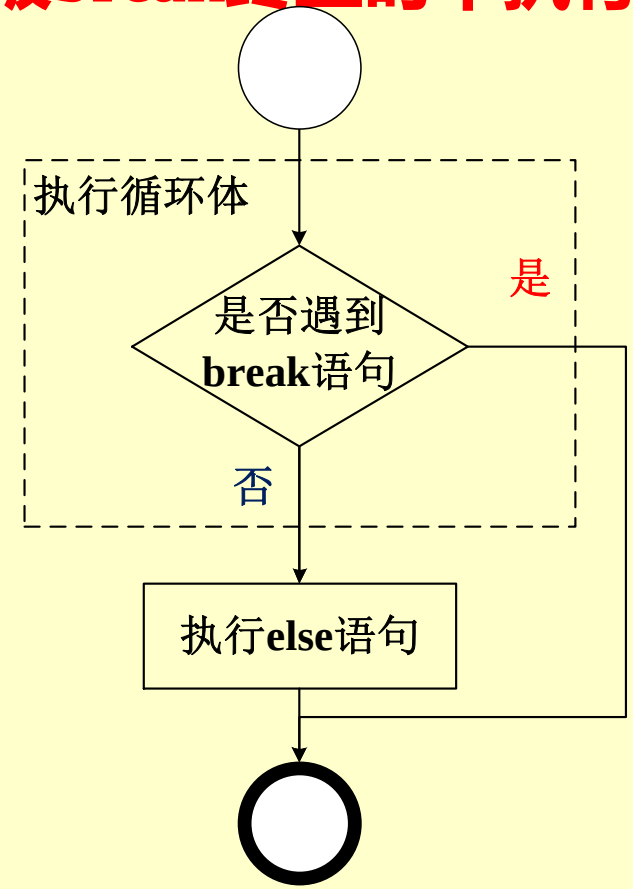
```
total = 0     #  break语句使用示例

for i in range(1, 10):

    if i % 3 == 0:

        break

    total = total + i

print(total)
```

3.1 循环控制语句

```
total = 0       #  continue语句使用示例

for i in range(1, 10):

    if i % 3 == 0:

        continue

    total = total + i

print(total)
```

3.2 循环中的else语句(进阶)

和其他语言不同, Python的循环语句还可以带有else子句, **else子句在序列遍历结束(for语句)或循环条件为假(while语句)时执行, 但循环被break终止时不执行**。执行流程如下图所示。



3.2 循环中的else语句(进阶)

带有else子句的while循环语句完整形式如下:

while 循环继续条件:

    循环体

else:

    语句体

3.2 循环中的else语句(进阶)

带有else子句的for语句完整形式如下:

for 控制变量 in 可遍历序列:

    循环体

else:

    语句体

3.2 循环中的else语句(进阶)

```
total = 0        # 在前述break示例中加入了else语句
for i in range(1, 10):
    if i % 3 == 0:
        break
    total = total + i
else:
    print("i=", i)
print(total)
```

3.2 循环中的else语句(进阶)

```
total = 0       # 在前述continue示例中加入了else语句
for i in range(1, 10):
    if i % 3 == 0:
        continue
    total = total + i
else:
    print("i=", i)
print(total)
```

3.2 循环中的else语句(进阶)

```
n = int(input('Input an integer:')) # 判断整数是否素数
m = math.ceil(math.sqrt(n)+1)
for i in range(2, m):
    if n%i == 0 and i<n:  # i <n(for 2)
        print('No')
        break
else:   # for, else的较好应用
    print('Yes')
```

Answers

 # coding = gbk

注意: 有时候’=’两边的空格不能少

 # ~ ~ 打印数字八

age= 23

message = "happy "+ str(age) +" rd Birtyday!“

练习

01 完成教材上的例子和练习

02 理解不同循环结构的作用

谢 谢