程序控制结构

内容

- 条件表达式
- 选择结构
- 循环结构

Python条件表达式

单个常量、变量或者任意合法的表达式都可以作为条件表达式

5　　“abc”　　[1, 2, 3]　　None

x+y　　x == y**5　　x and False

条件表达式的“值”

- 对于其实际值并非布尔型的条件表达式，当用于选择和循环结构时，需确定一个“布尔值”
- 等价于False的条件表达式的值
  - ✓False
  - ✓0、0.0、0j
  - ✓None
  - ✓值为空的数据：空字符串、空列表、空集合、……

```
>>> 5 and True
True
```

```
>>> True and 5
5
```

```
>>> not("")
True
```

```
>>> bool([1,2,3])
True
```

“等价于”不是“等于”

```
>>> [1,2,3] == True
False
```

```
>>> "" == False
False
```

```
>>> not [1,2,3] == False
True
```

判断以下语句输出为True的有：

print([] == False) ×

print(not []) √

print(1 == True) √

print(False == 0) √

内容

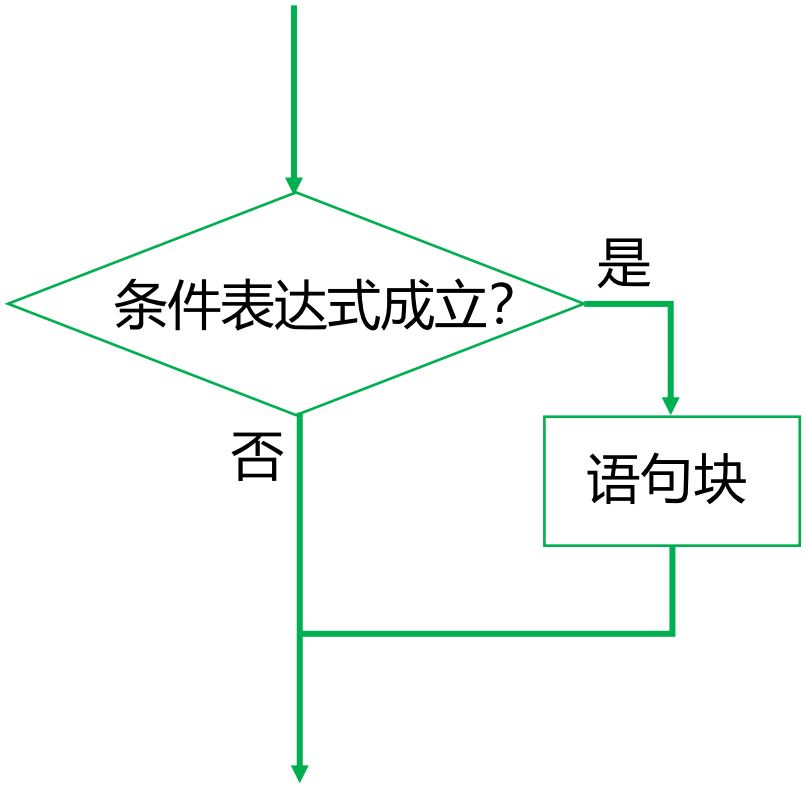
- 条件表达式
- 选择结构
- 循环结构

选择结构：单分支选择结构

if 语句

if <条件表达式>：

<语句块>



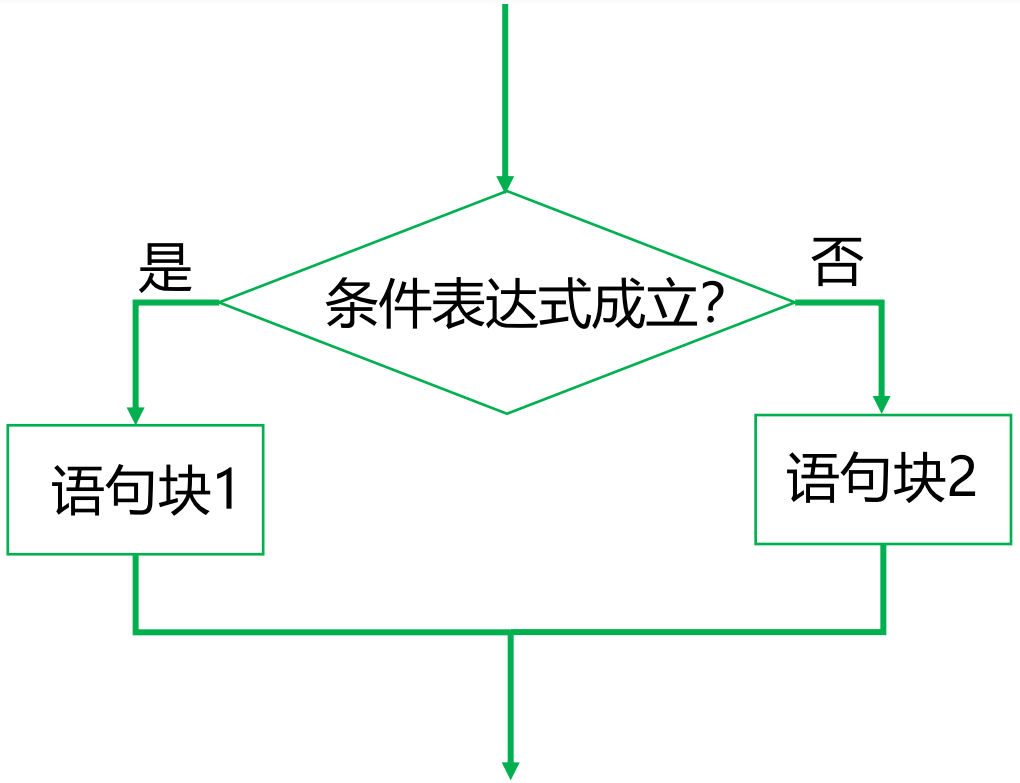
选择结构：单分支选择结构

```
score = int(input("请输入百分制成绩："))
if score >= 60:
    print(“及格”)
if score < 60:
    print(“不及格”)
```

选择结构：双分支选择结构

if-else 语句

```
if <条件表达式>:
    <语句块1>
else:
    <语句块2>
```



选择结构：双分支选择结构

```
score = int(input("请输入百分制成绩："))
if score >= 60:
    print(“及格”)
else:
    print(“不及格”)
```

双分支选择结构的简洁表达方式

```
if <条件表达式>:
    <表达式1>
else:
    <表达式2>
```

三元运算符

<表达式1> if <条件表达式> else <表达式2>

```
score = int(input("请输入百分制成绩："))
print("及格") if score >= 60 else print("不及格")
```

```
score = int(input("请输入百分制成绩："))
print("及格" if score >= 60 else "不及格")
```

选择结构：多分支选择结构

if-elif-else语句

```
if <条件表达式1>:
    <语句块1>
elif <条件表达式2>:
    <语句块2>
elif <条件表达式3>:
    <语句块3>
…
else:
    <语句块n>
```

选择结构：多分支选择结构

```
score = int(input("请输入百分制成绩："))
if score > 100:
    print("错误，成绩必须小于等于100")
elif score >= 90:
    print("成绩：A")
elif score >= 80:
    print("成绩：B")
elif score >= 70:
    print("成绩：C")
elif score >= 60:
    print("成绩：D")
elif score >= 0:
    print("成绩：E")
else:
    print("错误，成绩必须大于等于0")
```

选择结构：多分支选择结构

```
if <条件表达式1>:
    <语句块1>
elif <条件表达式2>:
    <语句块2>
elif <条件表达式3>:
    <语句块3>
…
else:
    <语句块n>
```

if-elif-else

```
if <条件表达式1>:
    <语句块1>
elif <条件表达式2>:
    <语句块2>
elif <条件表达式3>:
    <语句块3>
…
elif:
    <语句块n>
```

if-elif

选择结构中的缩进

Python程序的语句前面不能加空格或制表符，除非它：

- 在选择结构中的某个“语句块”里
- 在循环结构中的某个“语句块”里
- 在函数体里
- ……

选择结构中的缩进

if/else/elif语句中的语句块，相对于其条件表达式句必须缩进，且同一语句块中的语句缩进必须一致（对齐）

```
if <条件表达式>:
    <语句1>
    <语句2>
    ...
```

同一语句块

Python IDE中，都会通过识别条件表达式后的冒号，在换行后自动缩进

选择结构中的缩进

```
if int(input()) == 5:
    print("a",end="")
    print("b")
```

```
4          #输入
>>>        #无输出

5          #输入
ab         #输出
```

```
if int(input()) == 5:
    print("a",end="")
print("b")
```

```
4          #输入
b          #输出

5          #输入
ab         #输出
```

选择结构的嵌套

```
score = int(input("请输入百分制成绩："))
if 0 <= score <= 100:
    if score >= 60:
        print(“及格”)
    else:
        print(“不及格”)
else:
    print(“成绩必须在0到100之间！”)
```

内容

- 条件表达式
- 选择结构
- 循环结构

for循环和while循环

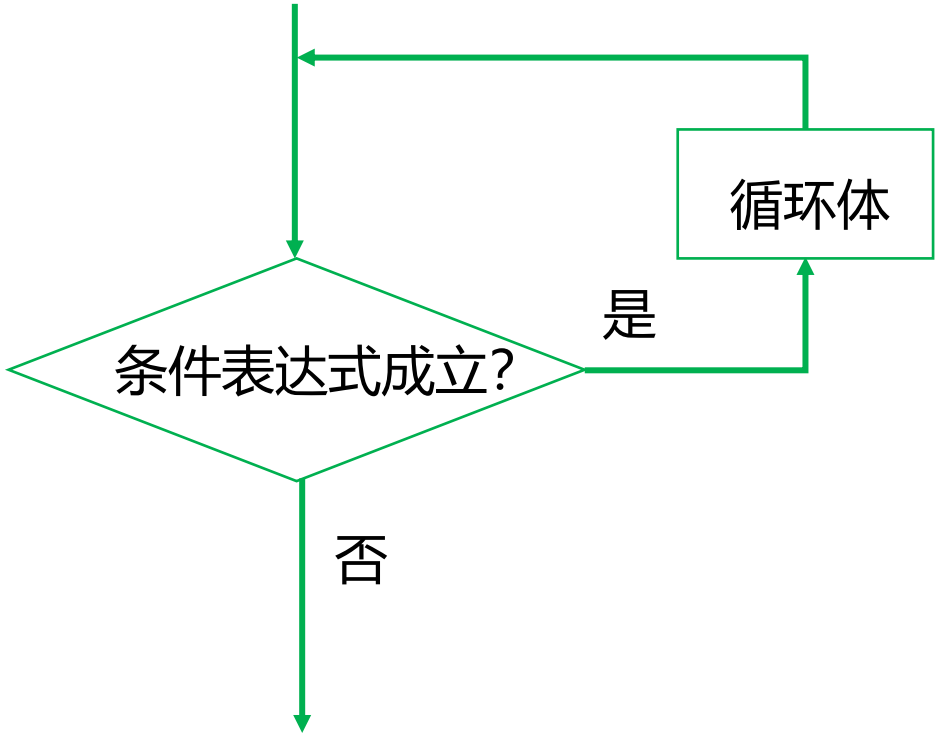
- for循环属于遍历循环，一般用于循环次数可以提前确定的情况，尤其适用于依次处理枚举序列或迭代数据
- while循环属于当型循环，一般用于循环次数难以提前确定的情况
- 一般优先考虑使用for循环
- 相同或不同的循环结构之间都可以互相嵌套，实现更为复杂的逻辑
- for循环和while循环都可以带else语句

for循环和while循环

- 基本的while循环
- 基本的for循环
- 循环的辅助控制

while循环的基本形式

```
while <条件表达式>:
    <循环体>
```



while循环示例

例题：输入一个正整数n，从小到大输出它的所有因子

```
n = int(input("请输入正整数：  "))
x  = 1
while x <= n
    if n%x == 0:
        print(x, end= " ")
    x += 1
```

```
请输入正整数:96
1 2 3 4 6 8 12 16 24 32 48 96
```

for循环和while循环

- 基本的while循环
- 基本的for循环
- 循环的辅助控制

for循环的基本形式

```
for <变量> in <序列>:
    <循环体>
```

- 变量依次取序列中的每一个值，执行循环体，直至序列取完
- 此过程中变量自动“后移”，不需要写代码控制
- 序列可以是range(...)，也可以是字符串、列表、集合等等

range()函数

Python的内建函数，用于生成一个数字序列

range(*start*, *stop*, *step*)

- *start:* 开始数字，默认为0
- *stop*: 结束数字
- *step:* 步长（增量），默认为1

```
>>> list(range(5))
[0, 1, 2, 3, 4]
```

使用for循环查看range()函数的值

```
for i in range(5):
    print(i, end=' ')
```

```
0 1 2 3 4
```

```
for i in range(2,8):
    print(i, end = ' ')
```

```
2 3 4 5 6 7
```

```
for i in range(2,15,3):
    print(i, end = ' ')
```

```
2 5 8 11 14
```

```
for i in range(11, 3, -2):
    print(i, end = ' ')
```

```
11 9 7 5
```

使用for循环查看range()函数的值

```
for i in range(-2, -9, -3):
    print(i, end = ' ')
```

```
-2 -5 -8
```

```
for i in range(-3):
    print(i, end = ' ')
```

```
for i in range(2,2):
    print(i, end = ' ')
```

```
for i in range(1,5,-1):
    print(i, end = ' ')
```

for循环遍历列表

```
languages = [“C”, “C++”, “Java”, “Python”,1]
for x in languages:
        print (x)
```

```
C
C++
Java
Python
1
```

for循环遍历字符串

```
for x in “Python3”:
    print (x, end = ‘*’)
```

```
P*y*t*h*o*n*3*
```

```
for x in [“Python3”]:
    print (x, end = ‘*’)
```

```
Python3*
```

在for循环体内改变循环变量的值

```
sum = 0
for i in range(1, 7):
    sum = sum + i
    i+=2
print(sum)
```

```
sum= 21
```

```
sum = 0
for i in range(1, 7):
    print(f"i={i}, sum={sum}")
    sum = sum + i
    i+=2
print(sum)
```

```
i=1,  sum=0
i=2,  sum=1
i=3,  sum=3
i=4,  sum=6
i=5,  sum=10
i=6,  sum=15
sum= 21
```

在for循环体内改变循环列表的值

```
languages = [“C”, “C++”, “Java”]
for x in languages:
    print (x)
    languages[0] = “Python”
```

```
C
C++
Java
```

```
languages = [“C”, “C++”, “Java”]
for x in languages:
    print (x)
    languages[1] = “Python”
```

```
C
Python
Java
```

for循环示例

例题：输出以星号组成的菱形，菱形中间一行的星号数为输入的小于10的正整数

```
n = int(input("请输入一个小于10的整数:"))
for i in range(n):
    print(f"{'* '*i:^30}")
for i in range(n, 0, -1):
    print(f"{'* '*i:^30}")
```

```
请输入一个小于10的整数:7
                *
               * *
              * * *
             * * * *
            * * * * *
           * * * * * *
          * * * * * * *
         * * * * * * * *
          * * * * * * *
           * * * * * *
            * * * * *
             * * * *
              * * *
               * *
                *
```

课堂练习

练习：输入一个正整数，输出相应长度的斐波那契数列，分别用for和while循环实现

```
输入斐波那契数列的数字个数：8
0 1 1 2 3 5 8 13
```

```
n = int(input("输入斐波那契数列的数字个数："))
a,b = 0,1
for i in range(n):
    print(a, end =' ')
    a,b = b, a+b
```

```
n = int(input("输入斐波那契数列的数字个数："))
a,b = 0,1
while n > 0:
    print(a, end =' ')
    a,b = b, a+b
    n-= 1
```

嵌套循环结构案例：打印九九乘法表

```
for i in range(1,10):
    for j in range(1,i+1):
        print(f"{i}*{j}={i*j:<6}",end='')
    print()
```

```
1*1=1
2*1=2   2*2=4
3*1=3   3*2=6   3*3=9
4*1=4   4*2=8   4*3=12  4*4=16
5*1=5   5*2=10  5*3=15  5*4=20  5*5=25
6*1=6   6*2=12  6*3=18  6*4=24  6*5=30  6*6=36
7*1=7   7*2=14  7*3=21  7*4=28  7*5=35  7*6=42  7*7=49
8*1=8   8*2=16  8*3=24  8*4=32  8*5=40  8*6=48  8*7=56  8*8=64
9*1=9   9*2=18  9*3=27  9*4=36  9*5=45  9*6=54  9*7=63  9*8=72  9*9=81
```

for循环和while循环

- 基本的while循环
- 基本的for循环
- 循环的辅助控制

循环的辅助控制：break语句

break语句位于循环体内，用于提前结束整个循环

```
for i in range(5):
    print("i = ", i, end = ',          ')
    print("j = ",end = '')
    for j in range(5):
        if j == 3:
            break
        print(j, end= ', ')
    print()
    print("jEnd = ", j)
```

```
i =  0,  j = 0,  1,  2,
jEnd =  3
i =  1,  j = 0,  1,  2,
jEnd =  3
i =  2,  j = 0,  1,  2,
jEnd =  3
i =  3,  j = 0,  1,  2,
jEnd =  3
i =  4,  j = 0,  1,  2,
jEnd =  3
```

循环的辅助控制：continue语句

continue语句位于循环体内，用于提前终止本次循环

```
for i in range(5):
    print("i = ", i, end = ',          ')
    print("j = ",end = '')
    for j in range(5):
        if j == 3:
            continue
        print(j, end= ', ')
    print()
    print("jEnd = ", j)
```

```
i =  0, j = 0, 1, 2, 4,
jEnd =  4
i =  1, j = 0, 1, 2, 4,
jEnd =  4
i =  2, j = 0, 1, 2, 4,
jEnd =  4
i =  3, j = 0, 1, 2, 4,
jEnd =  4
i =  4, j = 0, 1, 2, 4,
jEnd =  4
```

循环的辅助控制：else扩展

```
while <条件表达式>:
    <循环体>
else:
    <else语句块>
```

```
for <变量> in <序列>:
    <循环体>
else:
    <else语句块>
```

循环的辅助控制： else扩展

```
for j in range(10):
    if j == 5:
        break
    print(j,end=' ')
else:
    print("\n循环正常结束！")
```

```
0 1 2 3 4
```

```
for j in range(10):
    if j == 5:
        continue
    print(j,end=' ')
else:
    print("\n循环正常结束！")
```

```
0 1 2 3 4 6 7 8 9
循环正常结束！
```

循环的辅助控制：组合示例

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

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

```
97
```

```
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
```

循环控制：综合示例

例：输入一个正整数，对其进行因数分解。若输入为质数，则输出“这是一个质数！”；否则输出形如：6=2*3, 30=2*3*5。

1.将输入整数存入循环变量m

2. 执行循环：

2.1 寻找m的最小因子i

2.2 找到i: i存入因数列表，m=m//i，以新的m执行2

2.3未找到i, 寻找结束，退出循环

3. 根据运行结果输出

循环控制：综合示例

```
n = int(input("输入一个数：  "))
m,s = n,[]
while m > 1:
    for i in range(2,int(m**0.5)+1):
        if m%i == 0:
            m = m//i
            s.append(i)
            break
    else:
        break
if m == n:
    print("这是一个质数！  ")
else:
    print(f"{n}=",end='')
    for i in s:
        print(f"{i}*",end='')
    print(m)
```

m:因数分解的中间数；s:存储因数
while循环:执行因数分解直至无法分解
for循环:找到当前中间数m的最小因数
若i为m的因数，m分解出i，i存入因数列表，退出for循环

else表示for循环正常退出，即该次循环未找到因数，寻找过程结束，退出while循环

也可以用len(s)==0判断

输出因数分解式

循环控制：综合示例

```
n = int(input("输入一个数：  "))
m,s = n,[]
while m > 1:
    for i in range(2,int(m**0.5)+1):
        if m%i == 0:
            m = m//i
            s.append(i)
            break
    else:
        break
if m == n:
    print("这是一个质数！  ")
else:
    print(f"{n}=",end='')
    for i in s:
        print(f"{i}*",end='')
    print(m)
```

```
输入一个数：  23
这是一个质数！

输入一个数：  256
256=2*2*2*2*2*2*2*2

输入一个数：  289
289=17*17

输入一个数：  28356724
28356724=2*2*11*103*6257
```

程序控制结构：总结

- 几乎所有合法的Python表达式都可以作为选择结构和循环结构中的条件表达式
- 数字0、0.0、0j、布尔值False、空的组合数据类型、空字符串、空值None以及任意与这些值等价的值或表达式作为条件表达式时，均被认为条件表达式不成立，否则认为条件表达式成立
- 选择结构和循环结构往往会互相嵌套使用来实现复杂的业务逻辑
- 关键字elif表示的意思是else if

程序控制结构：总结

- 应优先考虑使用for循环，特别是对序列元素进行遍历的场合
- break语句用于提前结束其所在的循环，continue语句用于提前结束本次循环并进入下一次循环
- for循环和while循环都可以带有else子句，如果循环因条件表达式不满足而自然结束时，执行else子句中的代码；否则不执行