

## 第三章 流程控制

### 3.1 条件语句

### 3.2 条件流程控制

### 3.3 循环流程控制

### 3.4 实验

### 3.5 小结

### 习题





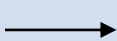
流程控制是指在程序运行时，对指令运行顺序的控制。

通常，程序流程结构分为三种：顺序结构、分支结构和循环结构。顺序结构是程序中最常见的流程结构，按照程序中语句的先后顺序，自上而下依次执行，称为顺序结构；分支结构则根据if条件的真假（True或者False）来决定要执行的代码；循环结构则是重复执行相同的代码，直到整个循环完成或者使用break强制跳出循环。

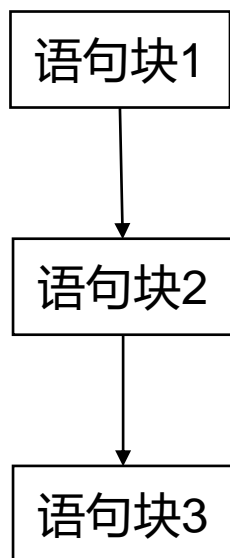
Python语言中，一般来说，我们使用if语句实现分支结构，用for和while语句实现循环结构。

流程图，是使用图形来表示流程控制的一种方法，是一种传统的算法表示方法，用特定的图形符号和文字对流程和算法加以说明，叫做算法的图，也称为流程图。俗话说千言万语不如一张图。

流程图有它自己的规范，按照这样的规范所画出的流程图，便于技术人员之间的交流，也是软件项目开发所必备的基本组成部分，因此画流程图也应是开发者的基本功。

符号	说明
	圆角矩形用来表示“开始”与“结束”。
	矩形用来表示要执行的动作或算法。
	菱形用来表示问题判断。
	平行四边形用来表示输入输出。
	箭头用来代表工作流方向。

**顺序结构**是程序中最常见的流程结构，按照程序中语句的先后顺序，自上而下依次执行，称为顺序结构；



```
1 a = 2
2 b = 3
3 c = a + b
4 print(a, "+", b, "=", c)
5 print("%d + %d = %d" % (a, b, c))
6 print("{0:d} + {1:d} = {2:d}".format(a, b, c))
```

Run: demo\_03\_01 ×

"C:\Program Files\Python38\python.exe" C:/Users/ruanz.  
2 + 3 = 5  
2 + 3 = 5  
2 + 3 = 5

Process finished with exit code 0

条件语句是用来判断给定的条件是否满足，并根据判断的结果（True或False）决定是否执行或如何执行后续流.....程的语句，它使代码的执行顺序有了更多选择，以实现更多的功能。

一般来说，条件表达式是由条件运算符和相应的数据所构成的，在Python中，**所有合法的表达式都可以作为条件表达式**。条件表达式的值只要不是**False、0、空值（None）、空列表、空集合、空元组、空字符串**等，其它均为True。

## 第三章 流程控制

3.1 条件语句

3.2 条件流程控制

3.3 循环流程控制

3.4 实验

3.5 小结

习题

if语句是由if发起的一个条件语句，在满足此条件后执行相应内容，Python的语句基本结构如下。

**if** 表达式1:

语句块1

**elif** 表达式2:

语句块2

.....

**else**:

语句块n

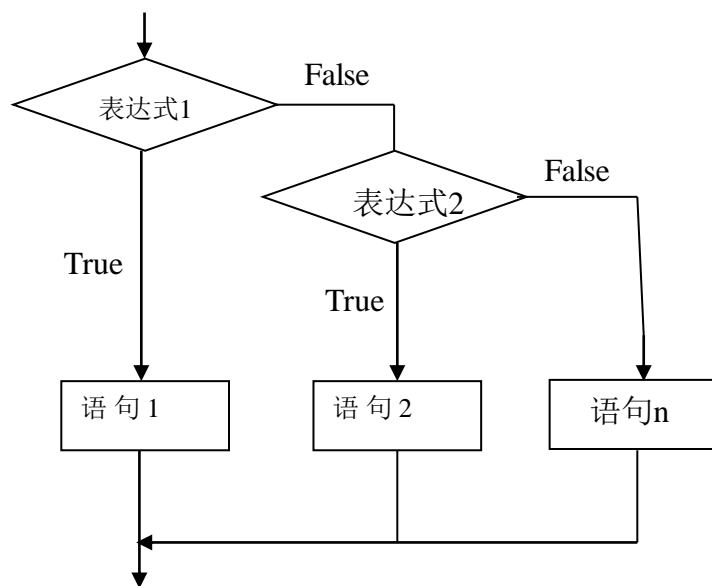


图3.1 分支选择结构

这里的elif, 为else if 的缩写, 同时需要注意的是:

- 1、else、elif为if语句的子语句块, 不能独立使用。
- 2、每个条件后面要使用冒号 “:”, 表示满足条件后需要执行的语句块, 后面几种其它形式的选择结构和循环结构中是冒号也是必须要有的。
- 3、使用缩进来划分语句块, 相同缩进数的语句组成一个语句块。
- 4、在Python中**没有switch...case语句**。



**单向分支选择结构**是最简单的一种形式，不包含elif和else，当表达式值为True时，执行语句块，否则该语句块不执行，继续执行后面的代码。其语法如下。

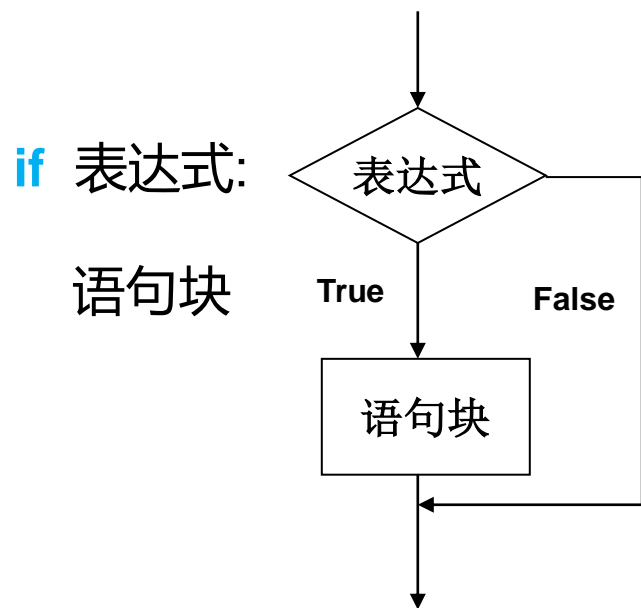


图3.2 单分支选择结构

```
1  # 判断变量a的值是否为True，是则执行a=0，否则直接输出a的值。  
2  a = 1  
3  if a:  # 等价于a>0或a!=0  
4      a = 0  
5  print(a)  # 如果if条件的值为False则输出结果为1
```

Run: demo\_03\_02\_01 x

"C:\Program Files\Python38\python.exe" C:/Users/ruanz  
0

Process finished with exit code 0

双分支语句是由if和else两部分组成，当表达式的值为True时，执行语句块1否则执行语句块2。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

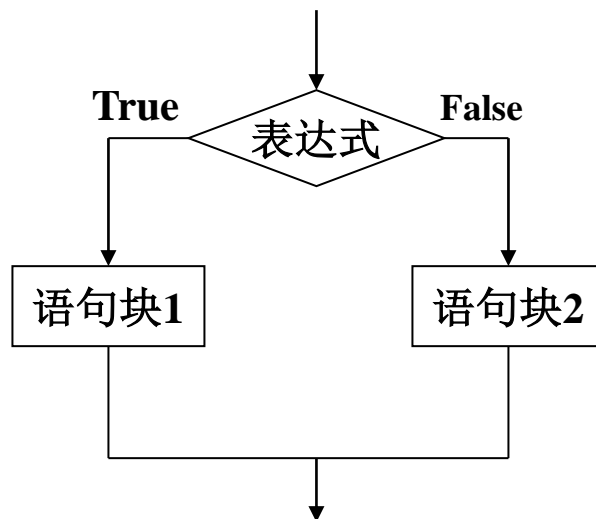


图3.3双分支选择结构

例如判断条件表达式的值是否为True，是则执行语句块1，否则执行else部分，最后再输出a的值。


```
1  # 判断变量a的值是否小于0，是则if部分，否则执行else部分。
2  a = int(input("输入一个整数a="))
3  if a < 0:
4      print("a<0")
5      a -= 1
6  else:
7      print("a≥0")
8      a += 1
9  print(a)
```








Run: demo\_03\_02\_02 ×

▶ ↑ "C:\Program Files\Python38\python.exe" C:/Users/ruanz  
■ ↓ 输入一个整数a=5  
⏏ ↺ a≥0  
⏏ ↻ 6  
🚀 ⏏ Process finished with exit code 0

又如判断三个正整数是否能构成三角形。

```
1  # 输入三个正整数, 判断它们是否能构成三角形。
2  a = int(input("输入a="))
3  b = int(input("输入b="))
4  c = int(input("输入c="))
5  if (a + b > c) and (a + c > b) and (b + c > a):
6      print("a={0:d},b={1:d},c={2:d}能构成三角形".format(a, b, c))
7  else:|
8      print("a={0:d},b={1:d},c={2:d}不能构成三角形".format(a, b, c))
    else
```

Run:  demo\_03\_02\_03 x

  "C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmPro  
 输入a=3  
 输入b=4  
 输入c=5  
 a=3,b=4,c=5能构成三角形  
 Process finished with exit code 0

多分支选择结构由if、一个或多个elif和一个else子块组成，else子块可省略。一个if语句可以包含多个elif语句，但结尾最多只能有一个else。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

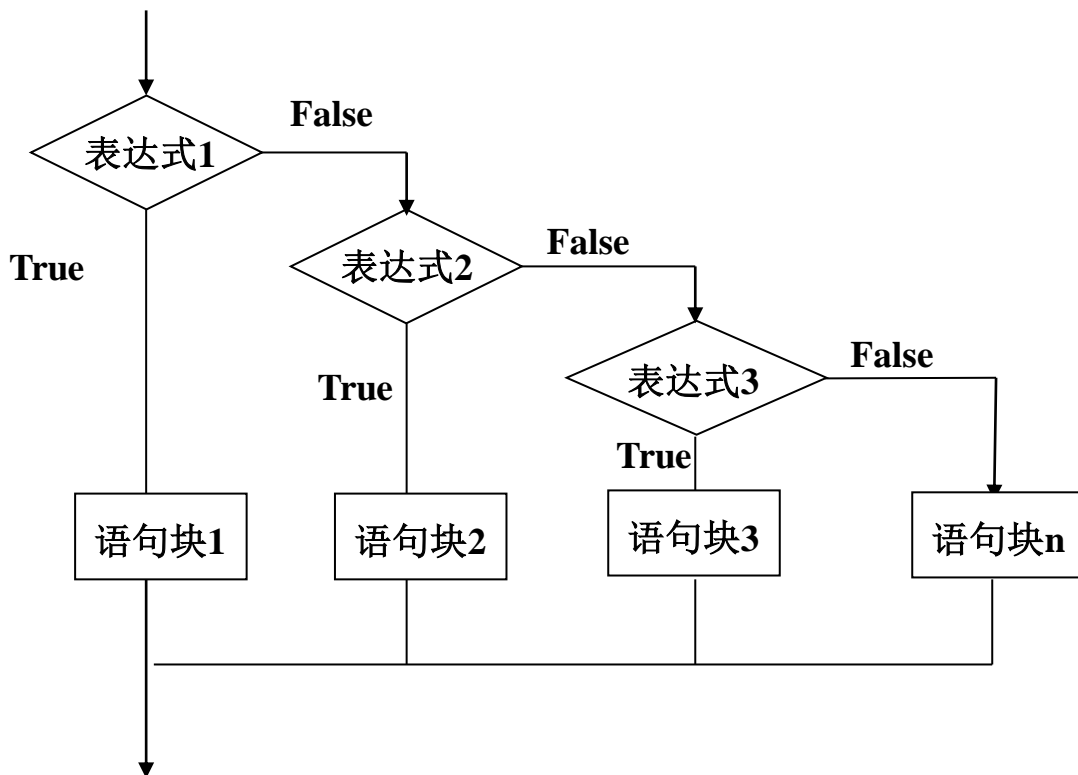


图3.4多分支选择结构

例如，根据你身上带的钱，来决定你今天中午能吃什么。

```
1  # 根据你身上带的钱，来决定你今天中午能吃什么。
2  money = float(input(" 请输入你带的钱： "))
3  if (money >= 1) and (money <= 5):  # 判断money是否在到5之间
4      print("你可以吃包子")
5  elif (money > 5) and (money <= 10):  # 判断money是否在6到10之间
6      print("你可以吃面条")
7  elif money < 0:  # 如果money小于0，就说明你没有钱，否则就说明你的钱大于10元
8      print("你的钱不够")
9  else:
10     print("你可以吃大餐")
```

Run: demo\_03\_02\_04 ×

▶ ↑ "C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmProjec  
■ ↓ 请输入你带的钱： 8  
— ↕ 你可以吃面条  
≡ ⌂  
— ↩  
▲

Process finished with exit code 0

选择结构可以进行嵌套来表达更复杂的逻辑关系。使用选择结构嵌套时，一定要控制好不同级别的代码块的缩进，否则就不能被Python正确理解和执行。在if 语句嵌套中，if、if...else、if...elif...else它们可以进行一次或多次相互嵌套，例如结构如下。

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

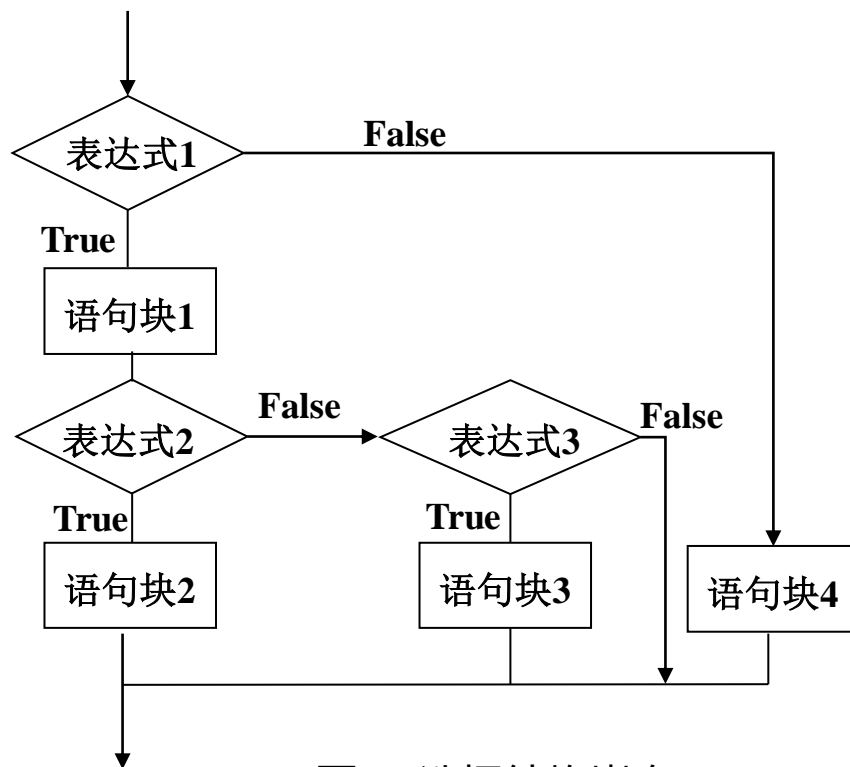


图3.5选择结构嵌套

例如请输入一个正整数，判断它是否能同时被2和3整除。

```
1  # 请输入一个正整数，判断它是否能同时被2和3整除。
2  a = int(input("请输入一个正整数："))
3  if a % 2 == 0: # 判断一个数是否能被2整除
4      if a % 3 == 0: # 判断一个数是否能被3整除
5          print(a, "能同时被2和3整除")
6      else:
7          print("此数能够被2整除，但是不能被3整除！")
8  else:
9      print("此数不能被2整除!")
```

Run: demo\_03\_02\_05 ×

▶ "C:\Program Files\Python38\python.exe" C:/Users/ruanz.  
请输入一个正整数：30  
30 能同时被2和3整除

Process finished with exit code 0



## 第三章 流程控制

3.1 条件语句

3.2 条件流程控制

3.3 循环流程控制

3.4 实验

3.5 小结

习题

循环，是我们生活中常见的，比如每天都要吃饭、上课、睡觉等，这就是典型的循环。循环结构是指在程序中需要反复执行某个功能而设置的一种程序结构。

Python提供for和while两种循环语句。

- **for语句**，用来遍历序列对象内的元素，通常用在已知的循环次数；
- **while语句**，提供了编写通用循环的方法。

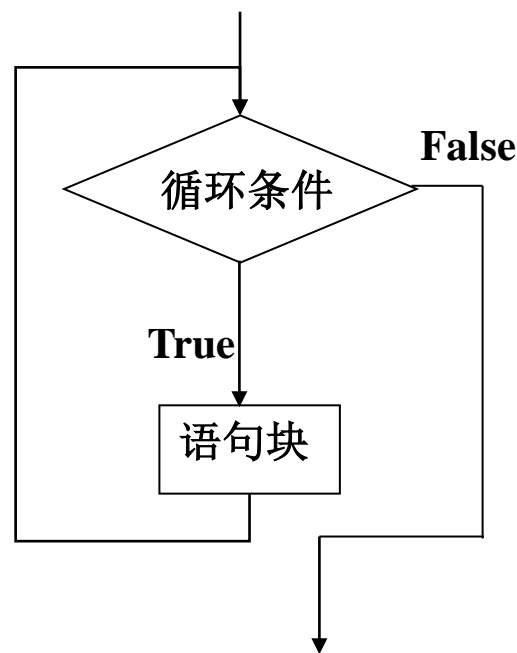


图3.6 循环流程图

for循环的语法结构跟前面讲的if...else有点类似，记的时候不要记混淆了。for执行时，依次将可迭代对象中的值赋给变量，变量每赋值一次，则执行一次循环体。循环执行结束时，如果有else部分，则执行对应的语句块。else只有在循环正常结束时执行。如果使用break跳出循环，则不会执行else部分，且根据实际编程需求，else部分可以省略。其结构如下。

**for** 变量 **in** 序列或迭代对象:

循环体（语句块1）

**else:**

语句块2

for和else后面冒号不能丢，循环体、语句块缩进严格对齐。

例如：求1~100的累加和，range()函数是生成1到100的整数。

```
1 s = 0
2 for x in range(1, 101): # 循环从1到100, 当101时就退出循环
3     s += x # 累加
4 print(s)
```

Run: demo\_03\_03\_01 ×

▶ ↑ "C:\Program Files\Python38\python.exe" C:/Users/ruanz  
■ ↓ 5050  
☐ ↺ Process finished with exit code 0

例如，删除列表对象[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]中的所有偶数。

```
1  # 删除列表对象中所有偶数。
2  x = list(range(10)) # 创建列表对象
3  for i in x: # i从x的第k=0个元素开始，
4      # 下一次循环则取下一个 (k=k+1) 元素(注意x是变化的，但k总是每次循环后加1)
5      print("x =", x, ", i =", i)
6      x.remove(i) # 从列表x中移除第一个值为i的元素
7  else:
8      print("delete over") # for循环正常执行完成后，执行else部分
9  print("x =", x)
```

Run: demo\_03\_03\_02 x

```
"C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmPro:
x = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] , i = 0
x = [1, 2, 3, 4, 5, 6, 7, 8, 9] , i = 2
x = [1, 3, 4, 5, 6, 7, 8, 9] , i = 4
x = [1, 3, 5, 6, 7, 8, 9] , i = 6
x = [1, 3, 5, 7, 8, 9] , i = 8
delete over
x = [1, 3, 5, 7, 9]

Process finished with exit code 0
```

**for循环嵌套**是指在for循环里有一个或多个for语句，循环里面再嵌套一重循环的叫双重循环，嵌套两层以上的叫多重循环。

例如使用两个for循环打印出九九乘法表，使用for循环和range()函数，变量i控制外层循环，变量j是控制内层循环的次数。

```
1  # 打印九九乘法表
2  for i in range(1, 10): # 外循环循环9次
3      for j in range(1, i): # 内循环控制每行输出的个数
4          print("{0:d}x{1:d}={2:d},|".format(j, i, i * j), end='')
5      else: # 一行中的最后一项
6          j = i
7          print("{0:d}x{1:d}={2:d} ".format(j, i, i * j))
```

for i in range(1, 10) > for j in range(1, i)


Run: demo\_03\_03\_03 x






```
"C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmProjects,
1x1=1
1x2=2, 2x2=4
1x3=3, 2x3=6, 3x3=9
1x4=4, 2x4=8, 3x4=12, 4x4=16
1x5=5, 2x5=10, 3x5=15, 4x5=20, 5x5=25
1x6=6, 2x6=12, 3x6=18, 4x6=24, 5x6=30, 6x6=36
1x7=7, 2x7=14, 3x7=21, 4x7=28, 5x7=35, 6x7=42, 7x7=49
1x8=8, 2x8=16, 3x8=24, 4x8=32, 5x8=40, 6x8=48, 7x8=56, 8x8=64
1x9=9, 2x9=18, 3x9=27, 4x9=36, 5x9=45, 6x9=54, 7x9=63, 8x9=72, 9x9=81

Process finished with exit code 0
```

例如，求 $1! + 2! + 3! + 4! + \dots + 10!$  的和

```
1  # 求1! +2! +3! +4! +...+N! 的和
2  N = int(input("请输入N: "))
3  s = 0
4  for n in range(1, N + 1):  # 外循环完成各阶乘的累加
5      m = 1  # 阶乘初始化为1
6      for i in range(1, n + 1):  # 内循环完成n! 的计算
7          m *= i  # m为n的阶乘
8      s += m  # 累加
9  print("1! + 2! +...+ ", N, "! = ", s, sep='')
```

Run:  demo\_03\_03\_04 x

  "C:\Program Files\Python38\python.exe" C:/Users/ruanz  
 请输入N: 10  
 1! + 2! +...+ 10! = 4037913  
 Process finished with exit code 0

当不知道循环次数，但知道循环条件时，一般使用while语句。与for循环类似，else部分可以省略。

注意：在Python中没有do...while语句。其语法结构如下。

**while** 循环条件:

    循环体（语句块1）

**else:**

    语句块2



例如求50以内所有5的倍数的和。

```
1  # 求50以内所有5的倍数的和
2  s = 0
3  i = 1
4  while i <= 50: # 从1循环到50
5      if i % 5 == 0: # 判断变量i是否能被5整除
6          s += i
7          print(i, end=' ')
8          i += 1 # 循环控制变量
9  else: # 循环正常结束, 就执行else部分
10     print(" \nover")
11     print(s)
```

Run: demo\_03\_03\_05 x

5 10 15 20 25 30 35 40 45 50  
over  
275  
Process finished with exit code 0

例如使用双重while循环打印出一个倒三角形图案。

```
1 # 打印出一个倒三角形的图案
2 i = 0
3 while i < 5: # 外循环, 循环行
4     j = 5 # 每循环一次变量j初始化为5
5     while j > i: # 内循环从5减到变量i
6         print('◆', end=' ')
7         j -= 1 # 控制内循环
8     print() # 换行
9     i += 1 # 控制外循环
```

Python还可以更简单

```
1 # 打印出一个倒三角形的图案
2 i = 0
3 while i < 5: # 循环行
4     print('◆ ' * (5 - i)) # 输出一行
5     i += 1 # 改变循环变量
```

Run: demo\_03\_03\_06 ×

"C:\Program Files\Python38\python.exe

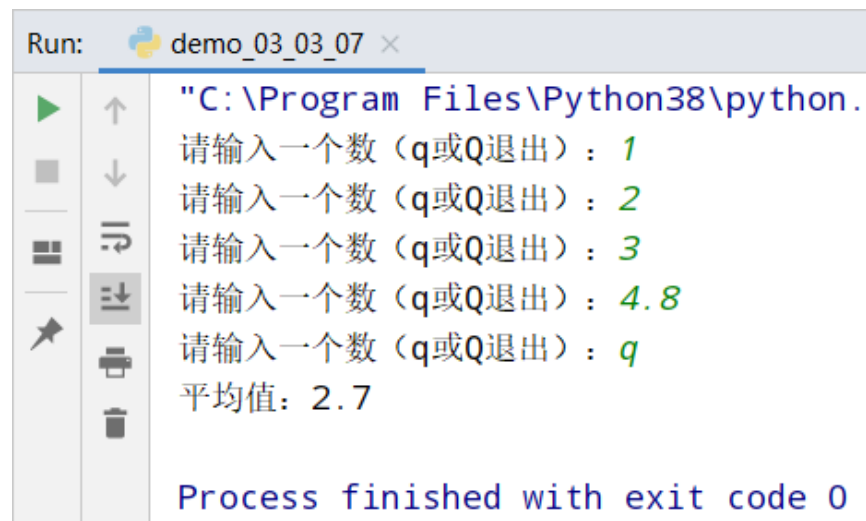
```
◆◆◆◆◆
◆◆◆◆
◆◆◆
◆◆
◆
```

Process finished with exit code 0

- **break语句**：跳出循环或叫终止循环，执行循环后面的语句。
- **continue语句**：结束本次循环（循环体中continue后面的语句不执行），进入下一次循环。

例如，从键盘输入若干个数，计算其平均值。循环条件为True，当输入为q或Q时强制跳出循环。

```
1  # 从键盘输入若干个数，计算其平均值
2  n, s = (0, 0) # 初始化两个变量
3  while True: # 无限循环
4      x = input("请输入一个数 (q或Q退出): ")
5      if x.lower() == 'q':
6          break
7      s += float(x) # 累加
8      n += 1 # 计数器
9  if n > 0:
10     print("平均值: " + str(s / n))
```








Run: demo\_03\_03\_07 x

```
"C:\Program Files\Python38\python.  
请输入一个数 (q或Q退出): 1  
请输入一个数 (q或Q退出): 2  
请输入一个数 (q或Q退出): 3  
请输入一个数 (q或Q退出): 4.8  
请输入一个数 (q或Q退出): q  
平均值: 2.7  
  
Process finished with exit code 0
```

例如，把50~80的不能被3整除的数输出。

```
1  # 把50~80的不能被3整除的数输出
2  for i in range(50, 80): # 从50循环到79
3      if i % 3 == 0: # 如果能被3整除就不输出
4          continue
5      print(i, end=' ')
```

Run:  demo\_03\_03\_08 x

  "C:\Program Files\Python38\python.exe" C:/Users/ruanz/Pychari  
  50 52 53 55 56 58 59 61 62 64 65 67 68 70 71 73 74 76 77 79  
Process finished with exit code 0

在循环体中可以包含另一个循环或分支语句，在分支语句中也可以包含另一个分支或循环。

例如，将大小写字母相互转换，注意代码缩进。

```
1  # 将小写字母转换成大写字母，大写字母转换成小写字母
2  x = input("请输入字母。")
3  for i in range(len(x)):
4      if (x[i] >= 'a') and (x[i] <= 'z'): # 判断是否为小写字母
5          print(chr(ord(x[i]) - 32), end="") # 首先将字符转换成ASCII码进行计算，
6                                              # 然后再将ASCII码转换成字符
7      elif (x[i] >= 'A') and (x[i] <= 'Z'): # 判断是否为大写字母
8          print(chr(ord(x[i]) + 32), end="")
9      elif x[i] == ' ': # 如果遇到空格，原样输出
10         print(end=" ")
11 else:
12     print(" \nover!")
```

Run: demo\_03\_03\_09 x

```
"C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmProjects/new
请输入字母。Hello World!
hELLO wORLD
over!
```

Process finished with exit code 0

## 第三章 流程控制

3.1 条件语句

3.2 条件流程控制

3.3 循环流程控制

3.4 实验

3.5 小结

习题

1、从键盘输入三个同学的成绩，然后找出最高分。

```
1      # 输入三个同学的成绩数，找出最高分
2      st1 = float(input("请输入第一位同学的成绩："))
3      st2 = float(input("请输入第二位同学的成绩："))
4      st3 = float(input("请输入第三位同学的成绩："))
5      Max = st1  # 假设第一个为最高分
6      if Max < st2:  # 如果第一个数小于第二个数，最大的数就变成第二个
7          Max = st2
8      if Max < st3:  # 把前面两个最大的数和第三个比
9          Max = st3
10     print(Max)
```

Run: demo03\_04\_01 x

D:\Python\Python36\python.exe F:/python/new/demo03\_04\_01.py  
请输入第一位同学的成绩：75  
请输入第二位同学的成绩：90  
请输入第三位同学的成绩：88  
90.0

## 2、输入三个同学的成绩，然后大到小排列。

```
1 # 输入三个同学的成绩，然后大到小排列
2 st1 = float(input("请输入第一位同学的成绩："))
3 st2 = float(input("请输入第二位同学的成绩："))
4 st3 = float(input("请输入第三位同学的成绩："))
5 if st1 < st2: # 第一个和第二个进行比较
6     tmp = st1
7     st1 = st2
8     st2 = tmp # 交换两个数的值
9 if st1 < st3: # 第一个和第三个进行比较
10     tmp = st1
11     st1 = st3
12     st3 = tmp
13 if st2 < st3: # 第二个和第三个进行比较
14     tmp = st2
15     st2 = st3
16     st3 = tmp
17 print(st1, st2, st3)
```

Run: demo03\_04\_02 x

D:\Python\Python36\python.exe F:/python/new/demo03\_04\_02.py

请输入第一位同学的成绩：78

请输入第二位同学的成绩：66

请输入第三位同学的成绩：80

80.0 78.0 66.0

Process finished with exit code 0



1、求出1000以内的所有完数，如 $6=1+2+3$ 除了它自身以外的因子之和等于它本身叫完数。

```
1  # 求出1000以内的所有完数，如6=1+2+3除了它自身以外的因子之和等于它本身的叫完数。
2  for i in range(1, 1000):  # 外循环从1到999
3      Sum = 0
4      for j in range(1, i):  # 内循环判断这个数是不是完数
5          if i % j == 0:  # 求出它的所有因数
6              Sum += j  # 把求出来的因数相加
7      if Sum == i:  # 判断它的因子之和是否等于它本身
8          print(i, end=' ')
```

Run: demo03\_04\_03 x

D:\Python\Python36\python.exe F:/python/new/demo03\_04\_03.py

6 28 496

Process finished with exit code 0

2、用循环语句求 $1+22+333+4444+55555$ 的和。

```
1 # 用循环语句求1+22+333+4444+55555的和。
2 Sum = 1 # 直接把1放到总和里面
3 for i in range(2, 6): # 外层循环运行4次
4     x = i # 控制最高位的数
5     for j in range(1, i+1):
6         x = x * 10 + i # 如22=2*10+2, 加的变量i就是个位的数
7     Sum += x
8 print("1+22+333+...+55555的和为:%d" % Sum)
```

Run: demo03\_04\_04 x



D:\Python\Python36\python.exe F:/python/new/demo03\_04\_04.py



1+22+333+...+55555的和为:603555



Process finished with exit code 0

## 1、求出2000~2100的所有闰年。

- 公历闰年规律为：四年一闰、百年不闰、400年再闰。
- 普通年能被4整除且不能被100整除的为闰年，如2020年是闰年，2019年不是如年。
- 世纪年能被400整除的是闰年，如2000年是闰年，1900年不是闰年。

闰年条件：能被4且不能被100整除，或者能被400整除的是闰年。

```
1 # 求出2000~2100的所有闰年
2 # 闰年条件：能被4且不能被100整除，或者能被400整除的是闰年
3 k = 0
4 for year in range(2000, 2101): # 循环从2000到2100
5     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0): # 闰年条件
6         print(year, end=' ')
7         k += 1 # 输出计数
8         if k == 10: # 每输出10个年份，就换行
9             k = 0
10            print() # 换行
```

Run: demo\_03\_04\_01 x

```
"C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmProjects/new/c
2000 2004 2008 2012 2016 2020 2024 2028 2032 2036
2040 2044 2048 2052 2056 2060 2064 2068 2072 2076
2080 2084 2088 2092 2096
Process finished with exit code 0
```

2、输入两个正整数，并求出它们的**最大公约数**( Greatest Common Divisor, GCD) 和**最小公倍数**(Least Common Multiple, LCM) 。

- **辗转相除法**, 又名**欧几里德算法**(Euclidean algorithm), 是GCD的一种方法。方法是：用第一个数m除以第二个数n, 得余数r, 再用出现的余数r (第1个余数) 去除除数, 再用出现的余数 (第2个数) 去除第一余数, 如此反复, 直到最后余数是0为止。那么最后的除数就是这两个数的GCD。

```
1  # 输入两个正整数，并求出它们的最大公约数和最小公倍数
2  a = int(input("输入第一个正整数: "))
3  b = int(input("输入第二个正整数: "))
4  m, n = (a, b) # 另存这两个数
5  r = m % n # 求余数
6  while r: # 当r!=0
7      m, n = (n, r) # 更新变量
8      r = m % n # 求余数
9  gcd = n # GCD
10 lcm = a*b / gcd # LCM
11 print("%d与%d的最大公约数为%d,最小公倍数为%d" % (a, b, gcd, lcm))
```

输入第一个正整数: 12

输入第二个正整数: 40

12与40的最大公约数为4,最小公倍数为120

## 3、输出100以内的所有质数(素数, Prime Number)。

```
1  # 输出100以内的所有质数(素数)
2  k = 0
3  for x in range(2, 100): # 从2~99循环
4      isPrimeNumber = True # 每次初始化为True, 默认是质数
5      for n in range(2, x): # 内层循环判断是否为质数
6          if x % n == 0: # 判断x是否能被n整除
7              isPrimeNumber = False
8              break # 跳出内层循环
9      if isPrimeNumber: # 等价于isPrimeNumber==True
10         print(x, end=' ')
11         k += 1 # 输出计数
12         if k == 10: # 每输出10项, 就换行
13             k = 0 # 重置计数器
14             print() # 换行
```

试一试:  
X改为 `int(math.sqrt(x))+1`

Run: demo\_03\_04\_03 ×

```
"C:\Program Files\Python38\python.exe" C:/Users/ruanz
2 3 5 7 11 13 17 19 23 29
31 37 41 43 47 53 59 61 67 71
73 79 83 89 97
Process finished with exit code 0
```

## 3、求100以内最大的10个质数的和。

```
1  # 求100以内最大的10个质数的和
2  k = s = 0  # k-质数计数器,s-累加和
3  x = 99
4  while x >= 2:
5      isPrimeNumber = True  # 每次初始化为True, 默认是质数
6      for n in range(2, x):  # 内层循环判断是否为质数
7          if x % n == 0:  # 判断x是否能被n整除
8              isPrimeNumber = False
9              break  # 跳出内层循环
10     if isPrimeNumber:  # 等价于isPrimeNumber==True
11         print(x, end=' ')
12         s += x  # 累加和
13         k += 1  # 输出计数
14         if k == 10:  # 如果质数个数等于10, 则退出外循环
15             break
16     x -= 1  # 改变外循环变量
17 print("\n100以内最大的10个质数的和为"+str(s))
```

97 89 83 79 73 71 67 61 59 53  
100以内最大的10个质数的和为732

4、求1-10的所有偶数的和。

```
1  # 求1-10的所有偶数的和。
2  s = 0  # 累加和
3  for i in range(1, 11):  # 从1循环到10
4      if i % 2 != 0:  # 如果是偶数就不输出，也不做计算
5          continue
6      print(i, end=' ')
7      s += i
8
9  print("\n10以内的偶数和为: %d" % s)
```

Run: demo\_03\_04\_05 x

"C:\Program Files\Python38\python.exe" C:/Users/s  
2 4 6 8 10  
10以内的偶数和为: 30

Process finished with exit code 0

5、将10-20不能被2或3整除的数输出。

```
1  # 将10-20不能被2或3整除的数输出。
2  for i in range(10, 21): # 从10循环到20
3      if i % 2 == 0 or i % 3 == 0: # 判断是否能否被2或3整除，是就不输出
4          continue
5      print(i, end=' ')
```

Run: demo\_03\_04\_06 x

"C:\Program Files\Python38\python.exe" C:/Users/ruanz/PycharmPro:  
11 13 17 19  
Process finished with exit code 0



## 第三章 流程控制

3.1 条件语句

3.2 条件流程控制

3.3 循环流程控制

3.4 实验

3.5 小结

习题

本章讲解了Python的流程控制：if分支、for循环和while循环。if、for和while语法上很简单，但通过组合或嵌套，可以实现各种简单到复杂的程序逻辑结构。

为了保证程序流程控制的灵活性，Python提供了continue和break两个语句来控制循环语句。continue语句用来结束本次循环，提前进入下一次循环。break语句用于强制退出循环，不执行循环体中剩余的循环次数。

## 第三章 流程控制

3.1 条件语句

3.2 条件流程控制

3.3 循环流程控制

3.4 实验

3.5 小结

习题

# 习题：

1. \_\_\_\_\_语句是else和if的组合。
2. \_\_\_\_\_、\_\_\_\_\_不能单独和if分支配合使用。
3. 每个流程结构语句后面必须要有\_\_\_\_\_。
4. Python中的流程控制语句有\_\_\_\_\_、\_\_\_\_\_和\_\_\_\_\_。
5. 当循环\_\_\_\_\_结束时才会执行else部分。
6. 自然常数e，有时也称为欧拉数（Euler number），是一个无限不循环小数，其值约为2.718281828。利用公式 $e = 1/1! + 1/2! + \dots + 1/n!$ ，近似计算e的值。
7. 寻找100~999中的所有可能的水仙花数。设三位数ABC，若 $ABC = A^3 + B^3 + C^3$ ，则称ABC为水仙花数。例如371， $3^3 + 7^3 + 1^3 = 27 + 343 + 1 = 371$ ，故371为水仙花数。
8. 我国古代数学家张丘建在《算经》一书中曾提出过著名的“百钱买百鸡”问题，该问题叙述如下：“鸡翁一，值钱五；鸡母一，值钱三；鸡雏三，值钱一；百钱买百鸡，则翁、母、雏各几何？”试编程求解之。

感谢聆听

