第 2 次课 循环和列表

Python 科学计算

周吕文

宁波大学，机械工程与力学学院

2024 年 9 月 1 日

Notes

提要

Notes

如何制作摄氏、华氏温度对照表格

摄氏温度 C 与华氏温度 F 的转换关系

$$F = \frac{9}{5}C + 32 \qquad \Longleftrightarrow \qquad C = \frac{5}{9}(F - 32)$$

如何用程序输出这样一张表？

C	F
-20	-4.0
-15	5.0
-10	14.0
- 5	23.0
0	32.0
5	41.0

C	F
10	50.0
15	59.0
20	68.0
25	77.0
30	86.0
35	95.0

Notes

简单的做法

根据转换关系，我们知道如何输出表中的任意一行

```
>>> C = -20
>>> F = 9/5*C + 32
>>> print(C,F)
-20 -4.0
```

重复以上操作，即可得到摄氏、华氏温度对照表格 c2f_repeat.py

```
C = -20;  F = 9.0/5*C + 32;  print(C, F)
C = -15;  F = 9.0/5*C + 32;  print(C, F)
                  ...
C =  40;  F = 9.0/5*C + 32;  print(C, F)
```

- 缺点：代码冗长乏味，可读性差，易出错，不易维护
- 改进：计算机非常擅长使用“循环”执行重复性任务

Notes

while 循环结构

结构

```
while <布尔条件>:
    <语句 1>
    <语句 2>
      ...

<循环结构外语句>
```

解释

- while 循环重复执行语句，直到<布尔条件>为假（False）
- 循环内的所有语句都必须缩进
- 当遇到未缩进的语句时，循环结构结束

Notes

使用 while 循环制作摄氏、华氏温度对照表

c2f_while.py

```
print('--------------')
dC = 5
C = -20
while C <= 5:
    F = 9/5*C + 32
    print('%5d %5.1f' % (C, F))
    C = C + dC
print('--------------')
```

IDLE Shell

```
>>>
----------------
  -20  -4.0
  -15   5.0
  -10  14.0
   -5  23.0
    0  32.0
    5  41.0
----------------
>>>
```

注意

- 循环体中的语句必须缩进（四个空格）
- while 所在行的行尾冒号“:”不要遗漏

Notes

布尔表达式

布尔表达：计算结果为布尔数据类型（True 或 False）的表达式

例如：$C = 40,\quad C \neq 40,\quad C \geq 40,\quad C < 40$

```
>>> C = 41
>>> C == 40
False
>>> C != 40
True
>>> C >= 40
True
>>> C < 40
False
>>> C == 41
True
```


Notes

while 循环与布尔表达式

while 循环中可用 and/or 连接多个布尔表达式

```
while <布尔表达式 1> and <布尔表达式 2>:
    ...
while <布尔表达式 1> or  <布尔表达式 2>:
    ...
```

```
>>> x = 0; y = 1.2
>>> x >= 0 and y < 1
False
>>> x >= 0 or y < 1
True
>>> x > 0 or not y > 1
False
>>> not (x > 0 or y > 0)
False
```


Notes

while 循环示例：累加求和

正弦函数泰勒级数展开近似

$$\sin x \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

loop_sum.py

```
import math
x = 1.2; N = 25; k = 1; s = x; sign = 1.0

while k < N:
    sign = -sign
    k = k + 2
    term = sign*x**k/math.factorial(k)
    s = s + term

print('sin(%g) = %g (%d 项近似值)' % (x, s, N))
```


Notes

课堂练习

1 级数 $\frac{1}{k}$ 求和 文件名：sum_1_k.py

使用循环计算数学累加和

$$s = \sum_{k=1}^{100} \frac{1}{k}$$

2 饭桌游戏报数 文件名：except7.py

100 以内循环报数，不能报包含 7 和 7 的倍数的数，编程输出这些数。

Notes

提要

Notes

列表：一个有序且可更改的集合

为什么要引入列表？

- 我们并不总是像前面的简单做法一样，一个变量一个变量的处理。
- 更自然的方式是将一系列元素组成有序集合，类似数学中的向量。
- **简单做法：**一个变量对应一个值

$$C_1 = -20,\ C_2 = -15,\ C_3 = -10,\ \cdots,\ C_{12} = 35,\ C_{13} = 40$$

如果变量非常多，这将程序将变得冗长乏味。

- **列表做法：**可将摄氏温度定义为

$$C = [-20, -15, -10, \cdots, 35, 40]$$

这样我们只需要 C 这一个变量就可以保存所有的值了。

Notes

列表的基本操作：初始化和索引

初始化：方括号 + 逗号分割

```
>>> L = [-91, 'a string', 7.2, 0]
>>> print(L)
[-91, 'a string', 7.2, 0]
```

索引：从 0 开始到 len(L) - 1

```
>>> mylist = [4, 6, -3.5]
>>> mylist[0]
4
>>> mylist[1]
6
>>> mylist[2]
-3.5
>>> len(mylist)
3
```


Notes

列表的基本操作：append, extend, insert, delete

```
>>> C = [-10, -5, 0, 5, 10, 15, 20, 25, 30]
>>> C.append(35)      # 列表尾部追加单个元素
>>> C
[-10, -5, 0, 5, 10, 15, 20, 25, 30, 35]
>>> C = C + [40, 45] # 合并（扩展）列表
>>> C
[-10, -5, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45]
>>> C.insert(0, -15) # 在索引为 0 的元素前插入元素 -15
>>> C
[-15, -10, -5, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45]
>>> del(C[2])        # 删除第 2 个元素（注意：从 0 开始计数）
>>> C
[-15, -10, 0, 5, 10, 15, 20, 25, 30, 35, 40, 45]
>>> len(C)           # 列表长度
12
```


Notes

列表的基本操作：查找，负索引

```
>>> C = [-10, -5, 0, 5, 10, 15, 20, 25, 30]
>>> C.index(10)      # 第一个值为 10 的元素索引
4
>>> 10 in C          # 10 这个元素是否在列表 C 中
True
>>> C[-1]            # 列表最后一个元素
30
>>> C[-2]            # 列表中倒数第二个元素
25
>>> somelist = ['book.tex', 'book.log', 'book.pdf']
>>> texfile, logfile, pdf = somelist
>>> texfile
'book.tex'
>>> pdf
'book.pdf'
```


Notes

列表的一些方法和函数

a = []、a = [1, "nbu"]	初始化空、非空列表
a.append(818)	增加元素到列表尾部
a.pop(i)	取出 a 位置为 i 的元素，i 缺省时默认为 -1
a + [1, 2]	合并两个列表
a.insert(k, e)	在 a 的位置 k 处插入元素 e
a[3]、a[-1]	通过索引访问 a 中元素
a[1:3]	获取子列表，包含序号为 1 和 2 的元素
del a[3]	从列表 a 中删除 a[3]
a.index(4)	获取元素 4 的序号
a.remove(4)	删除 a 中的第一个 4
4 in a	测试 4 是否在 a 中
a.count(4)	计算 a 中 4 的个数
a.sort()、a.reverse()	对 a 中元素排序、将 a 中元素反转
len(a)	计算 a 中元素个数
min(a)、max(a)	计算 a 中最小、最大元素
sorted(a)	对 a 中元素排序，返回新列表
sum(a)	求 a 中元素和
isinstance(a, list)	检查 a 是否是列表，等价于 type(a) is list

Notes

课堂练习

3 列表操作 文件名：list_operate.py

对列表 a = [1, 2, 9, 5, 8, 9] 和 b = [3, 4, 6, 9, 4] 进行以下操作：

- 将列表 a 和 b 合并成一个列表 c
- 统计列表 c 中 9 的个数
- 对列表 c 中元素进行排序，生成列表 d
- 求列表 c 中序号为 3 的元素在列表 d 中的位置 k
- 取出 d 中序号为 k-1 的元素
- 求列表 d 最后 5 个元素的和

Notes

提要

Notes

for 循环结构

结构

```
for <循环变量> in <序列>:
    <语句 1>
    <语句 2>
      ...

<循环结构外语句>
```

解释

- for 循环重复执行语句，直到整个<序列>被完全遍历
- 循环内的所有语句都必须缩进
- 当遇到未缩进的语句时，循环结构结束

Notes

使用 for 循环遍历列表元素

c2f_list_for.py

```
print('---------------')
degrees = [-20, -15, -10, -5, 0, 5]

for C in degrees:
    F = 9/5*C + 32
    print('%5d %5.1f' % (C, F))

print('---------------')
```

IDLE Shell

```
>>>
---------------
  -20  -4.0
  -15   5.0
  -10  14.0
   -5  23.0
    0  32.0
    5  41.0
---------------
>>>
```

注意

- 循环体中的语句必须缩进（四个空格）
- for 所在行的行尾冒号“:”不要遗漏

Notes

可以用 while 循环实现 for 循环

```
for element in somelist:
    <process element>
```

⇑

```
index = 0
while index < len(somelist):
    element = somelist[index]
    <process element>
    index += 1
```

注意

- 所有 for 循环都可以按照以上方式转为 while 循环
- 反之，不是所有 while 循环都可以用 for 循环实现

Notes

使用 while 循环遍历列表元素

c2f_list_while.py

```
Cdegrees = [-20, -15, -10, -5,  0, 5]
index = 0

print('%5s %5s'%('C', 'F'))
while index < len(Cdegrees):
    C = Cdegrees[index]
    F = 9/5*C + 32
    print('%5d %5.1f' % (C, F))
    index += 1
```

IDLE Shell

```
>>>
    C     F
  -20  -4.0
  -15   5.0
  -10  14.0
   -5  23.0
    0  32.0
    5  41.0
>>>
```


Notes

将表格存储为列表

c2f_for_append.py

```
Cdegrees = [-20, -15, -10, -5,  0, 5]
Fdegrees = [] # 初始化为空列表

for C in Cdegrees:
    F = 9/5*C + 32
    Fdegrees.append(F)

print(Fdegrees)
```

```
>>>
[-4.0, 5.0, 14.0, 23.0, 32.0, 41.0]
```


Notes

for 循环使用 range

for 循环通常需要在一个列表上遍历迭代

```
for element in somelist:
    ...
```

可用 range 函数生成列表索引用于 for 循环的遍历

```
for i in range(0, len(somelist), 1):
    element = somelist[i]
    ...
```

range(start=0, stop, inc=1): 生成可循环的整数等差序列

```
>>> range(3) # = range(0,3,1)
range(0, 3)
>>> list(range(3))
[0, 1, 2]
```

```
>>> range(2, 8, 3)
range(2, 8, 3)
>>> list(range(2, 8, 3))
[2, 5]
```


Notes

如何改变列表中的元素

v 为什么没有改变

```
>>> v = [-1, 1, 10]
>>> v
[-1, 1, 10]

>>> for e in v:
...     e = e + 2
...
>>> v
[-1, 1, 10]
```

修改列表元素需直接对其赋值

```
>>> v = [-1, 1, 10]
>>> v[1] = 4
>>> v
[-1, 4, 10]
>>> for i in range(len(v)):
...     v[i] = v[i] + 2
...
>>> v
[1, 6, 12]
```

当修改列表元素值时，实际上是使元素指向（引用）新的地址

```
>>> v[1] = 2
>>> id(v[1]) # 获取内存地址
130966451306768
```

```
>>> v[1] = 3
>>> id(v[1])
130966451306800
```


Notes

如何复制列表中的元素

列表赋值给变量赋的是地址

```
>>> v = [-1, 1, 10]
>>> id(v)
130966423376192
>>> u = v
>>> u
[-1, 1, 10]
>>> u.append(2)
>>> u
[-1, 1, 10, 2]
>>> v
[-1, 1, 10, 2]
>>> id(u) # 与 v 地址相同
130966423376192
>>> id(v) # 改值后地址没变
130966423376192
```

使用 copy() 复制列表

```
>>> v = [-1, 1, 10]
>>> id(v)
130966423377856
>>> u = v.copy()
>>> u
[-1, 1, 10]
>>> u.append(2)
>>> u
[-1, 1, 10, 2]
>>> v
[-1, 1, 10]
>>> id(u) # 与 v 地址不同
130966423381120
>>> id(v)
130966423377856
```

Notes

使用列表推导式快速创建列表

使用循环创建 2 个列表 c2f_for_append2.py

```
>>> n = 5
>>> Cdegrees = []; Fdegrees = []
>>> for i in range(n):
...     Cdegrees.append(-20 + i*5)
...     Fdegrees.append(9/5*Cdegrees[i] + 32)
...
>>> print(Cdegrees, Fdegrees)
[-20, -15, -10, -5, 0] [-4.0, 5.0, 14.0, 23.0, 32.0]
```

更简洁的方法是使用列表推导式 c2f_for_expression.py

```
>>> # somelist = [expression for element in otherlist]
>>> Cdegrees = [-5 + i*5 for i in range(n)]
>>> Fdegrees = [9/5*C + 32 for C in Cdegrees]
>>> print(Cdegrees, Fdegrees)
[-5, 0, 5, 10, 15] [23.0, 32.0, 41.0, 50.0, 59.0]
```

Notes

使用 enumerate 同时获得列表索引和元素

使用索引获得列表元素

```
for i in range(len(Cdegrees)):
    print(i, Cdegrees[i])
```

使用 enumerate 同时获得列表索引和元素

```
for i, C in enumerate(Cdegrees):
    print(i, C)
```

使用 enumerate 同时操作 3 个列表

```
L1 = [3, 6, 1]
L2 = [1, 1, 0]
L3 = [9, 3, 2]
for i, a in enumerate(L1):
    print(i, a, L2[i], L3[i])
```

```
>>>
0 3 1 9
1 6 1 3
2 1 0 2
```


Notes

使用 zip 同时操作多个列表

使用索引同时操作多个列表

```
for i in range(len(Cdegrees)):
    print(Cdegrees[i], Fdegrees[i])
```

使用 zip 同时操作多个列表

```
for C, F in zip(Cdegrees, Fdegrees):
    print(C, F)
```

使用 zip 同时操作 3 个列表

```
L1 = [3, 6, 1]
L2 = [1, 1, 0]
L3 = [9, 3, 2]
for a, b, c in zip(L1, L2, L3):
    print(a,b,c)
```

```
>>>
3 1 9
6 1 3
1 0 2
```


Notes

课堂练习

4 级数 k^2 求和 文件名：sum_k2.py

分别使用 while 和 for 计算以下数学累加和

$$s = \sum_{k=1}^{10} k^2$$

5 理想气体体积 文件名：ideal_gas.py

理想气体状态方程表明，气体压强 p、体积 V 和温度 T 存在以下关系

$$pV = nRT, \quad R = 8.3145\,\mathrm{J/(K \cdot mol)}$$

现有 $n = 40$ mol 气体，恒定温度 $T = 250$ K，求不同压强下的气体体积：

- 在 100 kPa - 200 kPa 之间取 $n = 51$ 个间隔均匀的值构成压强列表 p。
- 根据以上公式计算列表 p 中的压强对应的体积，并存在 V 列表中。
- 使用 for 循环遍历这两个列表，并打印出表格。

Notes

提要

Notes

嵌套列表：列表的列表

- 列表可以包含任何对象，甚至是另一个列表
- 可以将两个或多个列表放在一起形成一个新列表

```
>>> s = [['N', 'B', 'U'], [315211, 818], ['Y', 'E', 'S']]
>>> s[0]
['N', 'B', 'U']
>>> s[1]
[315211, 818]
>>> s[0][0]
'N'
>>> s[2][1]
'E'
```


Notes

嵌套列表的两种方式

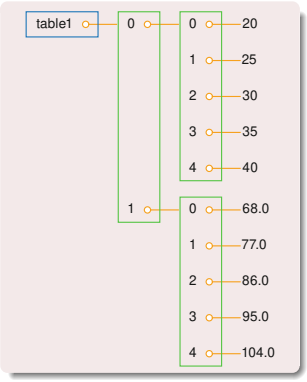
c2f_nested_list.py

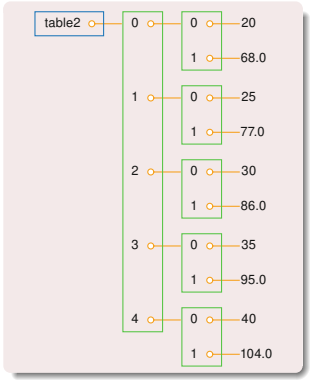
```
Cdegrees = list(range(20, 45, 5))
Fdegrees = [9/5*C + 32 for C in Cdegrees]

# 按列嵌套，table1 中包含两列
table1 = [Cdegrees, Fdegrees]
print(table1)

# 按行嵌套，table2 中每一行为序偶对 [C, F]
table2 = [[C, F] for C, F in zip(Cdegrees, Fdegrees)]
print(table2)
```

```
>>>
[[20, 25, 30, 35, 40], [68.0, 77.0, 86.0, 95.0, 104.0]]
[[20, 68.0], [25, 77.0], [30, 86.0], [35, 95.0], [40, 104.0]]
```

Notes

嵌套列表的两种方式





Notes

提取列表子表（切片）

```
>>> A = [2, 3.5, 8, 10]
>>> A[2:]
[8, 10]
>>> A[1:3]
[3.5, 8]
>>> A[:3]
[2, 3.5, 8]
>>> A[1:-1]
[3.5, 8]
>>> A[:]
[2, 3.5, 8, 10]
```

注意：子表（切片）是原列表的副本

```
>>> B = A[:]
>>> B[1] = 4.5
>>> B
[2, 4.5, 8, 10]
>>> A
[2, 3.5, 8, 10]
>>> C = A[:]; C is A
False
>>> C == A
True
>>> D = A;    D is A
True
```


Notes

遍历嵌套列表

iterate_c2f_nested_list.py

```
Cs = list(range(20, 45, 5))
Fs = [9/5*C + 32 for C in Cs]
table = [[C, F] for C, F in zip(Cs, Fs)]

for C, F in table:      # 遍历全部
    print('%5.0f %5.1f' % (C, F))

print('\n--------------\n')

for C, F in table[1:3]: # 遍历部分
    print('%5.0f %5.1f' % (C, F))
```

```
>>>
   20  68.0
   25  77.0
   30  86.0
   35  95.0
   40 104.0

--------------

   25  77.0
   30  86.0
```


Notes

遍历嵌套列表

通过索引遍历多重嵌套列表

```
for i1 in range(len(somelist)):
    for i2 in range(len(somelist[i1])):
        for i3 in range(len(somelist[i1][i2])):
            for i4 in range(len(somelist[i1][i2][i3])):
                value = somelist[i1][i2][i3][i4]
                # work with value
```

通过子表（切片）索引多重嵌套列表

```
for sublist1 in somelist:
    for sublist2 in sublist1:
        for sublist3 in sublist2:
            for sublist4 in sublist3:
                value = sublist4
                # work with value
```


Notes

遍历嵌套列表

iterate_nested_list.py

```
L = [[9, 7], [1, 5, 6]]
for i in range(len(L)):
    for j in range(len(L[i])):
        print(L[i][j], end=' ')
```

```
>>>
9 7 1 5 6
```

```
for row in L:
    for column in row:
        print(column, end=' ')
```

```
>>>
9 7 1 5 6
```


Notes

课堂练习

6 访问列表　　文件名：nbu_list.py

对列表 `q = [['N', 'B', 'U'], [315211, 818]]` 进行以下操作：

- 通过索引取出字母 B 和列表 `[315211, 818]`
- 使用循环访问列表中的所有元素并逐一打印

7 嵌套列表　　文件名：ball_list.py

根据竖直上抛公式

$$y(t) = v_0 t - \frac{1}{2} g t^2$$

其中 $v_0 = 5$ m/s，$g = 9.8$ m/s^2，完成以下任务：

- 在区间 $[0, 2v_0/g]$ 中取 $n = 101$ 个间隔均匀的值构成时间列表 t。
- 根据以上公式计算列表 t 中的时间对应的位置，并存在 y 列表中。
- 使用两种方式（按列/行嵌套）将 t 和 y 嵌套成一个列表。

Notes

提要

Notes

元组（Tuple）：不能修改的列表

元组的定义

```
>>> t = (2, 4, 'abc')
>>> t
(2, 4, 'abc')
```

```
>>> t = 2, 4, 'abc'
>>> t
(2, 4, 'abc')
```

元组不能修改（改/加/删）

```
>>> t[1] = -1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> t.append(0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
```


Notes

元组能干什么？

```
>>> t
(2, 4, 'abc')
>>> 2 in t  # 成员检测
True
```

```
>>> t[1]     # 索引
4
>>> t[1:]    # 切片
(4, 'abc')
```

```
>>> t = t + (-1.0, -2.0)
>>> t
(2, 4, 'abc', -1.0, -2.0)
>>> t = t*2
>>> t
(2, 4, 'abc', -1.0, -2.0, 2, 4, 'abc', -1.0, -2.0)
```

为什么要有元组

- 元组不能被修改，可以防止数据的意外修改
- 元组比列表快
- 元组可以作为字典（dictionary）的关键字（后面介绍），而列表不行

Notes

课堂练习

8 修改元组　　文件名：nbu_tuple.py

元组虽然不能修改，但可以通过函数 `list` 将其转为列表，再进行修改，修改完后再用 `tuple` 函数转为元组。据此，请对元组

```
q = ('N', 'B', 'U', 315211, 818, 'Y', 'E', 'S')
```

做如下操作：

- “删除”元组 q 中的 'E'
- 并在 'Y' 和 'S' 之间插入 'Y' 和 'D'。
- 用 ('N', 'B', 'U')、(315211, 818) 和 ('Y', 'Y', 'D', 'S') 构造一个嵌套元组 p

Notes

The End!

Notes