第 4 次课 用户输入和错误管理

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

如何在程序中指定输入数据？

之前程序中数据的输入：在程序中设定，只能进行特定数据的计算

- 变量的值在程序中设定 `v0, t, g = 5, 0.6, 9.8`
- 修改变量值需修改程序 `y = v0*t - 1/2*g*t**2`

程序中数据输入的其它方式

- 用户交互的输入
- 读取命令行参数
- 从文件读取数据
- 图形化用户界面

Notes

提要

Notes

用户交互的数据输入

修改变量值需修改程序

```
C = 21
F = 9/5*C + 32
print(F)
```

```
>>>
69.80000000000001
```

改进：程序问用户“C = ?”，然后读取用户输入，并赋值给 C

```
C = input('C = ? ')      # C 是字符串
C = float(C)             # 转为浮点数
F = 9/5*C + 32
print(F)
```

```
>>>
C = ? 21
69.80000000000001
```


Notes

例子：输出前 n 个偶数

```
n = int(input('n = ? '))

for i in range(2, 2*n+1, 2):
    print(i)
print('-'*15)

# or:
print(list(range(2, 2*n+1, 2)))
print('-'*15)

# or:
for i in range(1, n+1):
    print(2*i)
```

```
>>>
n = ? 4
2
4
6
8
---------------
[2, 4, 6, 8]
---------------
2
4
6
8
>>>
```


Notes

提要

Notes

从命令行读取数据

在 Unix/Linux 系统运行程序常避免询问用户，而从命令行获取参数

- 例如用命令 `ls -s -t` 列出当前文件夹中文件，就带了两个命令行参数
- `-s`：让 ls 列出文件名和文件大小
- `-t`：根据方件的最后修改时期排序

Linux Terminal

```
nbu@ubuntu:~# ls -s -t
   2048  2023 年清华大学录取通知.pdf
   1024  2023 年清华大学退学申请.doc
   3929  2024 年麻省理工录取通知.pdf
   6666  2025 年宁波大学报考指南.pdf
nbu@ubuntu:~#
nbu@ubuntu:~# python myprog.py arg1 arg2 arg3
```


Notes

例子：温度转换 - 单个参数

c2f_cml.py

```
import sys
# 命令行参数存放在 sys.argv 列表中，sys.argv[0] = 'c2f_cml.py'
C = float(sys.argv[1])
F = 9/5*C + 32
print(F)
```

Windows Command Prompt

```
C:\Users\nbu> python c2f_cml.py 21
69.80000000000001
C:\Users\nbu>
```

Linux Terminal

```
nbu@ubuntu:~# python c2f_cml.py 21
69.80000000000001
nbu@ubuntu:~#
```


Notes

例子：竖直上抛 - 多个参数

根据输入 v_0 和 t 求小球位置 y，其中 $g = 9.8\,\mathrm{m/s^2}$

$$y = v_0 t + \frac{1}{2} g t^2$$

location_sys.py

```
import sys
v0, t = float(sys.argv[1]), float(sys.argv[2])
y = v0*t + 1/2*9.8*t**2
print('v0 = %.2f m/s, t = %.2f s, y = %.2f m' % (v0, t, y))
```

Linux Terminal

```
nbu@ubuntu:~# python location_sys.py 1 3
v0 = 1.00 m/s, t = 3.00 s, y = 47.10 m
nbu@ubuntu:~#
nbu@ubuntu:~# python location.py --t 3 --v0 1   # 如何指定参数?
```

Notes

可选命令行参数

location_arg.py

```
import argparse
parser = argparse.ArgumentParser() # 创建参数解析器

# 定义命令行选项：参数选项、参数名、类型、默认值和帮助字符串
parser.add_argument('--v0', '--initial_velocity',
        type=float, default=0.0, help='initial velocity')
parser.add_argument('--t',  '--time',
        type=float, default=1.0, help='time')

# 读入命令行先项并进行解析
args = parser.parse_args()
v0, t= args.v0, args.t

y = v0*t + 1/2*9.8*t**2
print('v0 = %.2f m/s, t = %.2f s, y = %.2f m' % (v0, t, y))
```

Notes

可选命令行参数

Linux Terminal

```
nbu@ubuntu:~# python location_arg.py --v0 1 --t 3
v0 = 1.00 m/s, t = 3.00 s, y = 47.10 m
nbu@ubuntu:~#
nbu@ubuntu:~# python location_arg.py --v0 1
v0 = 1.00 m/s, t = 1.00 s, y = 5.90 m
nbu@ubuntu:~#
nbu@ubuntu:~# python location_arg.py -h
usage: location_arg.py [-h] [--v0 V0] [--t T]

options:
  -h, --help            show this help message and exit
  --v0 V0, --initial_velocity V0
                        initial velocity
  --t T, --time T       time
nbu@ubuntu:~#
```


Notes

课堂练习

1 求差数列求和　　文件名：sn_input.py, sn_sys.py & sn_arg.py

根据首项 a_1 和公差 d，求等差数列前 n 项和

$$S_n = n a_1 + \frac{n(n-1)}{2} d$$

用三种方式，分别计算 $a_1 = 2$、$d = 3$ 等差数列前 $n = 10$ 项和

- 用户交互的输入：sn_input.py 使用 input 函数询问用户获取参数
- 读取命令行参数：sn_sys.py 使用 sys 库，从命令行直接获取参数
- 可选命令行参数：sn_arg.py 使用 argparse 库，可选命令行参数

Notes

提要

Notes

函数 eval：将字符串转化为表达式计算

```
>>> s = '1 + 2'
>>> r = eval(s)
>>> r
3
>>> type(r)
<class 'int'>
```

```
>>> r = eval('[1, 6] + [1, 2]')
>>> r
[1, 6, 1, 2]
>>> type(r)
<class 'list'>
```

注意：eval 的参数必须是可计算的字符串

```
>>> eval( 'NBU' )    # 相当于 r = NBU
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<string>", line 1, in <module>
NameError: name 'NBU' is not defined

>>> eval('"NBU"')    # 相当于 r = "NBU"
'NBU'
```

Notes

例子：加运算

add_input.py

```
a = eval(input('请给一个输入：'))
b = eval(input('再给一个输入：'))
c = a + b
print('%s + %s = %s (%s)' % (type(a), type(b), type(c), c))
```

Linux Terminal

```
nbu@ubuntu:~# python add_input.py
请给一个输入：12.34
再给一个输入：56.78
<class 'float'> + <class 'float'> = <class 'float'> (69.12)
nbu@ubuntu:~# python add_input.py
请给一个输入：[1, 2]
再给一个输入：[0, 3]
<class 'list'> + <class 'list'> = <class 'list'> ([1, 2, 0, 3])
nbu@ubuntu:~#
```

Notes

例子：加运算 - 输入不当会导致错误

Linux Terminal

```
nbu@ubuntu:~# python add_input.py
请给一个输入：(1, 2)
再给一个输入：[3, 4]
Traceback (most recent call last):
  File "add_input.py", line 3, in <module>
    c = a + b
TypeError: can only concatenate tuple (not "list") to tuple
nbu@ubuntu:~# python add_input.py
请给一个输入：4
再给一个输入：'Hello, NBU!'
Traceback (most recent call last):
  File "add_input.py", line 3, in <module>
    c = a + b
TypeError: unsupported operand type(s) for +: 'int' and 'str'
nbu@ubuntu:~#
```

Notes

函数 exec：执行任意 Python 代码

函数 eval 能计算表达式，但不能执行语句

```
>>> eval('r = 1 + 1')
 File "<string>", line 1
   r = 1 + 1
     ^
SyntaxError: invalid syntax
```

```
>>> eval('1 + 1')
2
```

函数 exec 能执行任意 Python 语句

```
>>> exec('r = 1 + 1')
>>> r
2
>>> exec('''
... a = [1, 2]
... c = a*2''')
>>> c
[1, 2, 1, 2]
```

```
>>> somecode ='''
... def f(t):
...     return t**2
... '''

>>> exec(somecode)
>>> f(2)
4
```

Notes

函数 exec 可在程序运行时创建函数

exec_formula.py

```
formula = input('请输入一个关于 x 的表达式: ')
exec("""
def f(x):
    return %s""" % formula)

while True:
    x = eval(input('请输入 x (退出请输入 None): '))
    if x is None: break
    print('f(%g) = %g' % (x, f(x)))
```

```
>>>
请输入一个关于 x 的表达式: x**2
请输入 x (x 为 None 时退出): 2
f(2) = 4
请输入 x (退出请输入 None): None
```

Notes

函数 StringFunction：将字符串表达式转化为函数

需要安装 scitools3 库：pip install scitools3

```
>>> from scitools.StringFunction import StringFunction
>>> formula = 'exp(x)*sin(x)'
>>> f = StringFunction(formula)
>>> f(0)
0.0
>>> print(str(f))
exp(x)*sin(x)
```

StringFunction 也可以指定参数，以 $g(t) = Ae^{-at}\sin(\omega x)$ 为例

```
>>> g = StringFunction('A*exp(-a*t)*sin(omega*x)',\
...     independent_variable='t', A=1, a=0.1, omega=3, x=5)
>>> g(1.2)
0.5767535751840072
>>> g.set_parameters(A=2, x=10); g(1.2)
-1.7526108790616068
```

Notes

例子：数值微分

数值微分 $f'(x) \approx \dfrac{f(x+h) - f(x-h)}{2h}$（$h$ 足够小）　diff.py

```
from scitools.StringFunction import StringFunction
import sys, math

def numerical_derivative(f, x, h=1E-5):
    return (f(x+h) - f(x-h))/(2*h)

f = StringFunction(sys.argv[1], independent_variable='x')
x = float(sys.argv[2])
print('数值微分:', numerical_derivative(f, x))
```

Linux Terminal（若在 Winodows 中，表达式无需加引号）

```
nbu@ubuntu:~# python diff.py 'exp(x)*sin(x)' 3.4
数值微分: -36.626296916386636
nbu@ubuntu:~#
```

Notes

例子：扩展到符号计算

diff_sym.py

```
import sympy as sym; from diff import * # 载入 diff.py
x_value = x          # 存一下 x 的值，x 将被用作符号变量
x = sym.symbols('x')
formula = sym.sympify(str(f))            # 将 f 转换为 表达式
dfdx = sym.diff(formula, x)              # 符号微分
dfdx_value = dfdx.subs(x, x_value)       # 用 x_value 替换 x
print('精确微分:', dfdx_value)
print('微分表达式:', dfdx)
```

Linux Terminal（若在 Winodows 中，表达式无需加引号）

```
nbu@ubuntu:~# python diff_sym.py 'exp(x)*sin(x)' 3.4
数值微分: -36.626296916386636
精确微分: -36.626296954476
微分表达式: exp(x)*sin(x) + exp(x)*cos(x)
nbu@ubuntu:~#
```

Notes

课堂练习

2 中点法求积分　　文件名：integrate_exec.py & integrate_sfun.py

中点法的思想是将曲线 $f(x)$ 下的区域划分成 n 个等宽矩形，矩形高度为中点的 f 值。计算所有矩形面积并累加，就得到了中点法的公式：

$$\int_a^b f(x)\,\mathrm{d}x \approx h\sum_{i=0}^{n-1} f\left(a+ih+\frac{1}{2}h\right), \quad h=\frac{b-a}{n}$$

根据以上公式，编写程序计算指定函数在指定区间内的数值积分

- 从命令行读取构成 $f(x)$ 的公式、以及 a、b 和 n
- integrate_exec.py：基于函数 exec 实现
- integrate_sfun.py：基于函数 StringFunction 实现
- 对以下情况计算数值积分，并与理论值对比

$$f(x)=x^2, \quad a=0,\ b=3,\ n=100$$

Notes

提要

Notes

摄氏度转华氏度

c2f_gui.py

```
from tkinter import *
def compute():
    C = float(C_entry.get())
    F = 9/5*C + 32
    F_lbout.configure(text='%g' % F)

root = Tk()
C_entry = Entry(root, width=4);  C_entry.pack(side='left')
C_label = Label(root, text='C'); C_label.pack(side='left')

compute = Button(root, text=' 等于', command=compute)
compute.pack(side='left', padx=4)

F_lbout = Label(root, width=4);  F_lbout.pack(side='left')
F_label = Label(root, text='F'); F_label.pack(side='left')
```

tk

21 C 等于 69.8 F

Notes

提要

Notes

一行一行地从文件读取数据

一般的程序结构

```
infile = open('data.txt', 'r')     # 打开文件, 'r' 表示只读
for line in infile:
    # 对每行数据操作
infile.close()                     # 关闭文件
```

计算文件 data.txt 中所有数的均值 mean.py

```
infile = open('data.txt', 'r')
mean, num = 0, 0
for line in infile:
    mean = mean + float(line)
    num = num + 1
infile.close()
mean = mean/num
print(' 均值为: %.2f' % mean)
```

data.txt

```
21.8
18.1
19
23
26
```

IDLE Shell

```
均值为: 21.58
```


Notes

其它读文件的方式

将文件的所有行读入到一个字符串的列表中

```
infile = open('data.txt', 'r')
lines = infile.readlines()
for line in lines:
    # 对每行数据操作
infile.close()
```

使用 with 语句

```
with open('data.txt', 'r') as infile:
    for line in infile:
        # 对每行数据操作
```

将整个文件的内容读入到一个字符串中

```
text = infile.read()
s = text.split()
```


Notes

其它读文件的方式

```
>>> infile = open('data.txt', 'r')
>>> lines = infile.readlines()
>>> lines
['21.8\n', '18.1\n', '19\n', '23\n', '26\n']
>>> infile = open('data.txt', 'r')
>>> infile.readline()
'21.8\n'
>>> for line in infile: print(line.strip(), end = ' ')
...
18.1 19 23 26
>>> infile = open('data.txt', 'r')
>>> filestr = infile.read()
>>> filestr
'21.8\n18.1\n19\n23\n26\n'
>>> filestr.split('\n')
['21.8', '18.1', '19', '23', '26', '']
```

Notes

读取包含更多数据信息的文件

读取罗马 1782-1970 每月和年平均降雨 rainfall.py

```
infile = open('rainfall.txt', 'r')
months, values = [], []
for line in infile:
    words = line.split()  # 将分字符串切分成词
    if words[0] != 'Year':
        months.append(words[0])
        values.append(float(words[1]))
    else:
        avg = float(words[1])
print('每月平均降雨量如下 (mm): ')
for month, value in zip(months, values):
    print('%4s %5.1f' % (month, value))
print('每年平均降雨为 (mm): ', avg)
```

rainfall.txt

```
Jan 81.2
Feb 63.2
Mar 70.3
Apr 55.7
May 53.0
Jun 36.4
Jul 17.5
Aug 27.5
Sep 60.9
Oct 117.7
Nov 111.0
Dec 97.9
Year 792.9
```


Notes

将数据写入文件

写文件的一般流程

```
outfile = open(filename, 'w')
for data in somelist:
    outfile.write(sometext + '\n')
outfile.close()
```

文件打开方式

- 'w': 文件不存在时先创建再打开，文件存在时先清空再打开
- 'a': 可以打开文件后追加写入文件内容

Notes

例子：将二维表的内容写入文件

write2table.py

```
data = [[ 0.75,  0.29, -0.29, -0.75,  0.29,  0.11, -0.11],
        [-0.29, -0.11,  0.11,  0.29, -0.75, -0.29,  0.29]]

outfile = open('table.txt', 'w')

for row in data:
    for column in row:
        outfile.write('%8.2f' % column)
    outfile.write('\n')
outfile.close()
```

table.txt

```
    0.75    0.29   -0.29   -0.75    0.29    0.11   -0.11
   -0.29   -0.11    0.11    0.29   -0.75   -0.29    0.29
```


Notes

课堂练习

3 文本数据读写 文件名：beltway.py

从 beltway.txt 中读入事件发生时间和经纬度：

- 将经度，纬度以浮点数分别存入 longs 和 lats 列表
- 将日期转换为自 2002 年 10 月 1 日起的天数，以整数存入 days 列表
- 将 longs 和 lats 以 '%8.4f' 的格式写入文件 xy.txt

beltway.txt: mm/dd/yyyy,lat,long

```
Beltway sniper data.
10/02/2002,39.05987314,-77.04737177
10/03/2002,38.98390032,-77.02674218
10/04/2002,38.29423783,-77.51456929
10/07/2002,38.95778241,-76.74620882
10/09/2002,38.79769595,-77.51791951
10/14/2002,38.86910131,-77.14935544
```

xy.txt

```
     lat      long
 39.0599 -77.0474
 38.9839 -77.0267
 38.2942 -77.5146
 38.9578 -76.7462
 38.7977 -77.5179
 38.8691 -77.1494
```


Notes

什么是文件？

文件是字符的序列

- 对于文本文件，每个字符占 1 个字节（8 位，即 8 个 0 或者 1），所以最多可能有 $2^8 = 256$ 种不同字符
- 中文字符需要多个字节存储（和编码有关）
- Python 读取文件的三种方式：read、readline 和 readlines

dat.txt

```
L 1
L 2
L 3
```

```
file = read('dat.txt')
```

```
>>> file.read()
'L 1\nL 2\nL 3\n'
```

```
>>> file.readline()
'L 1\n'
```

```
>>> file.readlines()
['L 1\n', 'L 2\n', 'L 3\n']
```


Notes

提要

Notes

输入中的错误处理

c2f_cml.py

```
import sys
C = float(sys.argv[1])
F = 9/5*C + 32; print(F)
```

Linux Terminal

```
nbu@ubuntu:~# python c2f_cml.py
Traceback (most recent call last):
  File "c2f_cml.py", line 2, in <module>
    C = float(sys.argv[1])
IndexError: list index out of range
nbu@ubuntu:~#
```

用户运行程序时忘了提供命令行参数

- sys.argv 是一个列表，只包含一个元素 sys.argv[0]，也就是程序名
- sys.argv[1] 访问了列表中不存在的元素，导致了 IndexError。

Notes

程序需要处理输入错误

c2f_cml_if.py

```
import sys
if len(sys.argv) < 2:
    print('输入参数失败')
    sys.exit(1)  # 终止程序
C = float(sys.argv[1])
F = 9/5*C + 32
print('%gC = %.1fF' % (C, F))
```

Linux Terminal

```
nbu@ubuntu:~# python c2f_cml_if.py
输入参数失败
nbu@ubuntu:~# python c2f_cml_if.py 21
21C = 69.8F
nbu@ubuntu:~#
```


Notes

更常用的处理方式：使用 try-except 异常处理机制

c2f_cml_try.py

```
import sys
try:                                    # 如果 try 块中出现错误，程
    C = float(sys.argv[1])              # 序会抛出一个异常，并马上
except:                                 # 转入到 except 块中执行
    print('输入参数失败')
    sys.exit(1)  # 终止程序
F = 9/5*C + 32
print('%gC = %.1fF' % (C, F))
```

Linux Terminal

```
nbu@ubuntu:~# python c2f_cml_try.py
输入参数失败
nbu@ubuntu:~# python c2f_cml_try.py 21
21C = 69.8F
nbu@ubuntu:~#
```

Notes

明确异常类型是良好的编程风格

c2f_cml_try2.py

```
import sys
try:
    C = float(sys.argv[1])
except IndexError:     # 超出 sys.argv 索引 -> IndexError 异常
    print('没有命令行参数 C! '); sys.exit(1)
except ValueError:     # float 转换失败 -> ValueError 异常
    print('参数 C 必需是数字! '); sys.exit(1)
F = 9/5*C + 32
print('%gC = %.1fF' % (C, F))
```

Linux Terminal

```
nbu@ubuntu:~# python c2f_cml_try2.py
没有命令行参数 C!
nbu@ubuntu:~# python c2f_cml_try2.py twenty-one
参数 C 必需是数字!
nbu@ubuntu:~#
```

Notes

抛出包含自定义信息的异常

可通过 raise 抛出包含特定信息的异常，下面的例子展示了两种情况：

- 捕获异常后，抛出一个带有改进错误信息的新异常
- 因为输入数据错误而抛出异常

c2f_cml_try3.py（接下页）

```
import sys
def read_C():
    try:
        C = float(sys.argv[1])
    except IndexError:
        raise IndexError ('需要提供参数 C')
    except ValueError:
        raise ValueError ('"%s"不是一个数字' % sys.argv[1])
    if C < -273.15:
        raise ValueError ('C = %g 不是一个合理的温度' % C)
    return C
```

Notes

抛出包含自定义信息的异常

c2f_cml_try3.py（接上页）

```
try:
    C = read_C()
except (IndexError, ValueError) as e:
    print(e); sys.exit(1)

F = 9.0*C/5 + 32
print('%gC = %.1fF' % (C, F))
```

Linux Terminal

```
nbu@ubuntu:~# python c2f_cml_try3.py
需要提供参数 C
nbu@ubuntu:~# python c2f_cml_try3.py twenty-one
"twenty-one"不是一个数字
nbu@ubuntu:~# python c2f_cml_try3.py -288
C = -288 不是一个合理的温度
nbu@ubuntu:~#
```

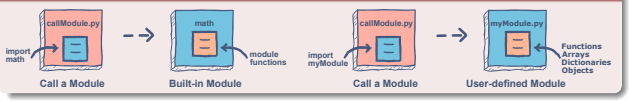
Notes

提要

Notes

为什么要制作模块

模块是一些有用的函数、数据、类等的集合



模块的作用和好处

- 提高可维护性：代码独立分块，易于定位，修改和更新不影响整体。
- 提高复用性：模块化代码可跨项目复用，提升效率。
- 提升开发效率：团队可并行开发，减少冲突和合并问题。
- 降低测试难度：模块独立测试，简化整体测试过程。

Notes

例子：利率计算模块-计算公式

利率相关公式：初始金额 A_0，最终金额 A，年利率 $p\%$，天数 n

$$A = A_0\left(1+\frac{p}{360\times 100}\right)^{n} \tag{1}$$

$$A_0 = A\left(1+\frac{p}{360\times 100}\right)^{-n} \tag{2}$$

$$n = \frac{\ln(A/A_0)}{\ln\left(1+\frac{p}{360\times 100}\right)} \tag{3}$$

$$p = 360\times 100\left[\left(\frac{A}{A_0}\right)^{1/n}-1\right] \tag{4}$$

Notes

例子：利率计算模块-创建

将函数实现并存为 interest.py，这就是一个名为 “insterest” 的模块了

```
from math import log as ln

def present_amount(A0, p, n):
    return A0*(1 + p/(360.0*100))**n

def initial_amount(A, p, n):
    return A*(1 + p/(360.0*100))**(-n)

def days(A0, A, p):
    return ln(A/A0)/ln(1 + p/(360.0*100))

def annual_rate(A0, A, n):
    return 360*100*((A/A0)**(1.0/n) - 1)
```


Notes

例子：利率计算模块-使用

如果模块文件和程序在一个目录，可直接使用 import 载入

```
from interest import days
n = days(A0=1, A=2, p=5)
print('%d 天后金额翻倍' % n)
```

```
>>>
4991 天后金额翻倍
```

可将模块文件集中放于某目录，然后将该目录加入 python 路径列表

```
>>> import interest
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'interest'
>>> import sys
>>> sys.path.append('/usr/lib')
>>> import interest
>>> interest.days(A0=1, A=2, p=5)
4991.006265599126
```

Notes

例子：利率计算模块-增加测试

interest.py（续）

```
def test_all_functions():
    A = 2.2133983053266699; A0 = 2.0; p = 5; n = 730
    A_cmpt = present_amount(A0, p, n)
    A0_cmpt = initial_amount(A, p, n)
    n_cmpt = days(A0, A, p)
    p_cmpt = annual_rate(A0, A, n)
    def feq(a, b, tolerance=1E-12):
        return abs(a - b) < tolerance
    success = feq(A_cmpt, A) and feq(A0_cmpt, A0) and \
              feq(p_cmpt, p) and feq( n_cmpt,  n)
    if success: print("测试通过！")
    assert success, "没试不通过！"

if __name__ == '__main__': # 当模块被直接运行时为真，否则为假
    test_all_functions()
```

Notes

例子：利率计算模块-增加测试

模拟程序入口的一般格式

```
if __name__ == '__main__':
    <语句块>
```

当模块被直接运行时，文件名__name__ == '__main__'

```
nbu@ubuntu:~# python interest.py
测试通过!
nbu@ubuntu:~#
```

模块被 import 进其它程序时，文件名__name__ != '__main__'

```
>>> from interest import days
>>> days(A0=1, A=2, p=5)
4991.006265599126
```


Notes

课堂练习

4 制作温度转换模块　　文件名：convert_temp.py

摄氏（C）、华氏（F）、开氏（K）温度转换关系如下：

$$F = \frac{9}{5}C + 32, \quad K = C + 273.15$$

根据以上公式及其衍生公式，制作温度转换模块 convert_temp：

1. 编写六个温度转换函数，用于在摄氏、开氏和华氏度之间进行转换：C2F、F2C、C2K、K2C、F2K 和 K2F。
2. 在 IDLE Shell 中导入这个模块并进行一些温度转换。
3. 将 2 中部分插入三引号中，放在模块开头，作为文档字符串。
4. 编写 test_conversion 函数验证模块：若命令行第一个参数为 verify，则调用该测试函数。提示：C2F(F2C(f)) 的结果应为 f。
5. 为该模块添加一个用户界面，使用户可以将一个温度值和相应温标作为命令行参数，然后输出其它两个温标下的温度值。例如，命令行输入 21.3 C，输出结果为 70.34 F 294.45 K。

Notes

The End!

Notes

Notes

Notes

Notes