Python 语言程序设计:Python 语法基础元素

李宽
likuan@dgut.edu.cn

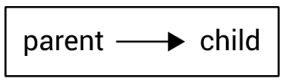
东莞理工学院

2019.9

目录

目录

程序的格式框架：缩进




```
def my_other_function():
    print('Hello, there!')
    answer = input('Do you like chocolate?')
    if answer == 'no':
        print("I can't believe it!")
    else:
        print('Me, too!')
```

```
def bad_indentation(num):
    if num < 10:
        a = 10
      b = 5
```

This code causes Python to output the following:

```
  File "<stdin>", line 4
    b = 5
        ^
```

- 严格明确：缩进是语法的一部分，缩进不正确程序会运行错误
- 所属关系：表达代码间包含和层次关系的唯一手段
- 长度一致：同一程序内缩进要保持长度一致[1]

[1] 一般用 4 个空格或者 1 个 TAB 来表示

程序的格式框架：注释

```
from yapf.yapflib.yapf_api import FormatCode
def fu( a) :
    """ This is a
    multiline comment
    """
      #this function prints
        #and increments
    print(  "a\'",a)
    print(  'b"',a)
    a=a+1 #indentation "error"
    return a
def fn(wx , z):
  ''' comment 1
   comment 2 '''
  print("a text\n with a value {}".format(z))
  s =np.abs(wx)
  return z*s
```

- 单行注释：以 # 开头，其后内容为注释内容
- 多行注释：以三个单引号（或三个双引号）开头和结尾

变量 1/2

变量：程序中用于保存和表示数据的占位符号

```
1  #TempConv.py
2
3  val = input("请输入带温度表示符号的温度值（例如：32C）：")
4
5  if val[-1] in ['C','c']:
6      f = 1.8 * float(val[0:-1]) + 32
7      print("转换后的温度为：%.2fF"%f)
8  elif val[-1] in ['F','f']:
9      c = (float(val[0:-1]) - 32) / 1.8
10     print("转换后的温度为：%.2fC"%c)
11 else:
12     print("输入有误")
```

变量 2/2

命名规则：大小写字母、数字、下划线和汉字等字符及组合
如：TempStr，Python_great，我在学 Python

注意事项：

- 大小写敏感，首字符不能是数字，不与关键字[2]相同
- 赋值符号：使用等号 (=) 向变量赋值或修改值,= 被称为赋值符号

示例:

- GOOD: spam, _speed, eggs, spam123
- BAD: 123spam, #sign, var.123
- DIFFERENT: spam SPAM Spam SPam

[2]又称保留字

关键字

保留字

and	elif	import	raise	global
as	else	in	return	nonlocal
assert	except	is	try	True
break	finally	lambda	while	False
class	for	not	with	None
continue	from	or	yield	
def	if	pass	del	

(26/33)

关键字/保留字是被编程语言内部定义并保留使用的标识符:

- Python 语言目前有 33 个保留字（关键字）：if,elif,else,in
- 保留字是编程语言的基本单词，大小写敏感，if 是保留字，If 是变量

变量与关键字

```
1  #TempConv.py
2
3  val = input("请输入带温度表示符号的温度值（例如：32C）：")
4
5  if val[-1] in ['C','c']:
6      f = 1.8 * float(val[0:-1]) + 32
7      print("转换后的温度为：%.2fF"%f)
8  elif val[-1] in ['F','f']:
9      c = (float(val[0:-1]) - 32) / 1.8
10     print("转换后的温度为：%.2fC"%c)
11 else:
12     print("输入有误")
```

绿色：变量，红色：关键字

数据类型

整数, 浮点数, 字符串, 列表

```
1  #TempConv.py
2
3      val = input('请输入带温度表示符号的温度值')
4
5  if val[-1] in ['C','c']:
6      f = 1.8 * float(val[0:-1]) + 32
7      print("转换后的温度为: %.2fF"%f)
8  elif val[-1] in ['F','f']:
9      c = (float(val[0:-1]) - 32) / 1.8
10     print("转换后的温度为: %.2fC"%c)
11 else:
12     print("输入有误")
```

数据类型: 数字类型

范例代码中出现了两种数字类型[3]：

- 整数：数学中的整数, 举例：12 或-34
- 浮点数：数学中的实数，带有小数部分 1.8 或-1.0

[3]更多相关内容会在稍后课程中展开，此处仅围绕示例代码铺开

数据类型: 字符串 1/2

```
1  #TempConv.py
2
3  val = input('请输入带温度表示符号的温度值（例如：32C）：')
4
5  if val[-1] in ['C','c']:
6      f = 1.8 * float(val[0:-1]) + 32
7      print('转换后的温度为：%.2fF'%f)
8  elif val[-1] in ['F','f']:
9      c = (float(val[0:-1]) - 32) / 1.8
10     print('转换后的温度为：%.2fC'%c)
11 else:
12     print('输入有误')
```

字符串类型：由 0 个或多个字符组成的有序字符序列
字符串由一对单引号或一对双引号表示

数据类型: 字符串 2/2

字符串是字符的有序序列，可以对其中的字符进行索引

字符串的使用;

- 索引：返回字符串中单个字符 < 字符串 >[M]
- 切片：返回字符串中一段字符子串 < 字符串 >[M:N], 半闭半开

val[0:-1] vs val[0:]

数据类型: 列表 1/2

```
#TempConv.py

val = input("请输入带温度表示符号的温度值（例如：32C）：")

if val[-1] in [C,c]:
    f = 1.8 * float(val[0:-1]) + 32
    print("转换后的温度为：%.2fF"%f)
elif val[-1] in [F,f]:
    c = (float(val[0:-1]) - 32) / 1.8
    print("转换后的温度为：%.2fC"%c)
else:
    print("输入有误")
```

数据类型: 列表 2/2

列表类型是由 0 个或多个数据组成的有序序列

- 列表用中括号表示，采用逗号分隔元素
- 使用保留字in判断一个元素是否在列表中

语句与函数: 赋值语句 1/2

```
#TempConv.py

val = input(请输入带温度表示符号的温度值（例如：32C）：)

if val[-1] in ['C','c']:
    f = 1.8 * float(val[0:-1]) + 32
    print("转换后的温度为：%.2fF"%f)
elif val[-1] in ['F','f']:
    c = (float(val[0:-1]) - 32) / 1.8
    print("转换后的温度为：%.2fC"%c)
else:
    print("输入有误")
```

语句与函数: 赋值语句 2/2

赋值语句：由赋值符号构成的一行代码, 用来给变量赋予新的数据值

```
C = (eval(TempStr[0:-1]) - 32)/1.8 # 右侧运算结果赋给变量 C
```

```
#input() 返回一个字符串, TempStr 也是一个字符串
TempStr = input("请输入带有符号的温度值：")
```

重要: 赋值语句右侧的数据类型同时作用于变量

语句与函数: 分支语句

```
1  #TempConv.py
2  val = input("请输入带温度表示符号的温度值（例如：32C）：")
3
4  if val[-1] in [C,c]:
5    f = 1.8 * float(val[0:-1]) + 32
6    print("转换后的温度为：%.2fF"%f)
7  elif val[-1] in [F,f]:
8    c = (float(val[0:-1]) - 32) / 1.8
9    print("转换后的温度为：%.2fC"%c)
10 else:
11   print("输入有误")
```

分支语句是由判断条件决定程序运行方向的语句

- 使用保留字 if elif else 构成条件判断的分支结构
- 所在行最后的冒号是语法的一部分，冒号以及后续的缩进用来表示后续语句与条件的所属关系

语句与函数: 函数

```
1  #TempConv.py
2  val = input("请输入带温度表示符号的温度值（例如：32C）：")
3
4  if val[-1] in ['C','c']:
5      f = 1.8 * float(val[0:-1]) + 32
6      print("转换后的温度为：%.2fF"%f)
7  elif val[-1] in ['F','f']:
8      c = (float(val[0:-1]) - 32) / 1.8
9      print("转换后的温度为：%.2fC"%c)
10 else:
11     print("输入有误")
```

函数：根据输入参数产生不同输出的功能过程

- 类似数学中的函数，y=f(x)
- 函数采用 < 函数名 >(< 参数 >) 方式使用

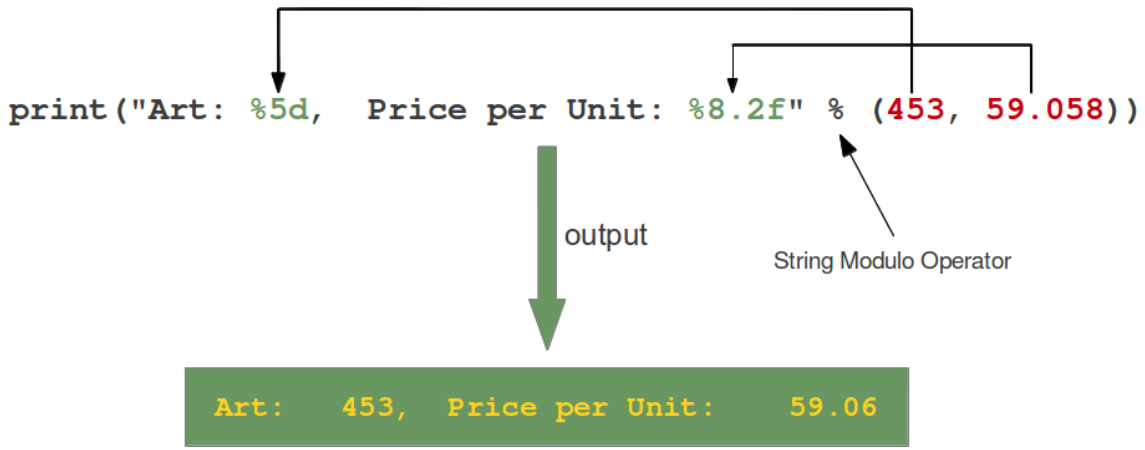
输入输出 1/2

input()：从控制台获得用户输入的函数
input() 函数的使用格式: 变量 =input(提示信息字符串)
用户输入的信息以字符串类型保存在 < 变量 > 中

```
1 #TempStr 保存用户输入的信息
2 TempStr = input("请输入带有符号的温度值:")
```

输入输出 2/2

print()：以字符形式向控制台输出结果的函数



print() 函数的基本使用格式：print(< 拟输出字符串或字符串变量 >)
体会如下代码的异同：

```
1 print("output is %d" % 123)
2 print("output is %5d" % 123)
3 print("output is %15d" % 123)
```

代码重温

```
#TempConv.py

val = input("请输入带温度表示符号的温度值（例如：32C）：")

if val[-1] in ['C','c']:
    f = 1.8 * float(val[0:-1]) + 32
    print("转换后的温度为：%.2fF"%f)
elif val[-1] in ['F','f']:
    c = (float(val[0:-1]) - 32) / 1.8
    print("转换后的温度为：%.2fC"%c)
else:
    print("输入有误")
```

目录

整数类型

与数学中整数的概念一致
可正可负, 没有范围限制[4]

整数类型有 4 种进制表示形式

- 十进制：1010, 99, -217
- 二进制，以 0b 或 0B 开头：0b010, -0B101
- 八进制，以 0o 或 0O 开头：0o123, -0O456
- 十六进制，以 0x 或 0X 开头：0x9a, -0X89

[4]仅受内存的限制

浮点数 1/2

与数学中实数的概念一致

- 浮点数取值范围 (约 -10^{308} 到 10^{308})
- 小数精度 (10^{-16}

常规计算可忽略如此限制
浮点数间运算存在不确定尾数，不是 bug, 举例:

```
1 >>> 0.1 + 0.2
2 0.30000000000000004
```

不确定尾数一般发生在 10^{-16} 精度左右
round()可以解决浮点数计算精度的问题

浮点数 2/2

四舍五入取整

```
round(number[, ndigits])
```

对 number 四舍五入，ndigits 是小数截取位数（默认为 0 位）

强调: 整数相等判断 vs 浮点数相等判断

除 round 外其他取整方式：

- 向下取整。用内建的 int()
- 向上取整。用到 math 模块中的 ceil() 方法

浮点数的科学计数法表示: 使用字母 e 或 E 作为幂的符号，以 10 为基数
格式：<a>e<b> 表示 $a*10^b$：

```
4.3e-3      # 0.0043
9.6E5       # 960000.0
```

复数

与数学中复数的概念一致

```
1 z = 12e3 + 45.678e+9j
2 z_re = z.real          # 12000.0, 实部
3 z_im = z.imag          # 45678000000.0, 虚部
```

数值运算操作符

数值运算操作符

操作符及使用	描述
x + y	加，x与y之和
x – y	减，x与y之差
x * y	乘，x与y之积
x / y	除，x与y之商　10/3结果是3.3333333333333335
x // y	整数除，x与y之整数商　10//3结果是3
+ x	x本身
- y	x的负值
x % y	余数，模运算　10%3结果是1
x ** y	幂运算，x的y次幂，x^y
	当y是小数时，开方运算　10**0.5结果是 $\sqrt{10}$

类型间可进行混合运算，生成结果为”最宽”类型间可进行混合运算

复数 > 浮点数 > 整数

数值运算函数

函数及使用	描述
abs(x)	绝对值，x的绝对值 abs(-10.01) 结果为 10.01
divmod(x,y)	商余，(x//y, x%y)，同时输出商和余数 divmod(10, 3) 结果为 (3, 1)
pow(x, y[, z])	幂余，(x**y)%z，[..]表示参数z可省略 pow(3, pow(3, 99), 10000) 结果为 4587
round(x[, d])	四舍五入，d是保留小数位数，默认值为0 round(-10.123, 2) 结果为 -10.12
max(x_1,x_2, ... ,x_n)	最大值，返回x_1,x_2, ... ,x_n中的最大值，n不限 max(1, 9, 5, 4 3) 结果为 9
min(x_1,x_2, ... ,x_n)	最小值，返回x_1,x_2, ... ,x_n中的最小值，n不限 min(1, 9, 5, 4 3) 结果为 1
int(x)	将x变成整数，舍弃小数部分 int(123.45) 结果为123； int("123") 结果为123
float(x)	将x变成浮点数，增加小数部分 float(12) 结果为12.0； float("1.23") 结果为1.23
complex(x)	将x变成复数，增加虚数部分 complex(4) 结果为 4 + 0j

逻辑运算操作符

逻辑操作

操作符	数学符号	描述
<	<	小于
<=	≤	小于等于
>=	≥	大于等于
>	>	大于
==	=	等于
!=	≠	不等于
x and y	两个条件x和y的逻辑**与**	
x or y	两个条件x和y的逻辑**或**	
not x	条件x的逻辑**非**	

位运算操作符 1/2

仅限于整形：

X	Y	X&Y	X|Y	X^Y	~(X)
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

位运算操作符 2/2

```
>>> x = 240
>>> y = 1
>>> x|y
241
```

x =240 11110000
y =1 00000001

```
11110000
00000001
--------
11110001
```

```
>>> x&y
0
```

```
11110000
00000001
--------
00000000
```

```
>>> x^y
241
>>>
```

```
11110000
00000001
--------
11110001
```

```
>>> x<<2
960
```

```
11110000
--------
1111000000
```

```
>>> x>>2
60
```

```
11110000
--------
00111100
```