Preview

2018年8月7日 19:29

1. 常用操作符

a. 算术操作符（返回数或者文本）

操作符	名称	用途	实例	备注
+	加法	返回左右相加	3+2返回5	
-	减法	返回左减去右	3-2返回1	
*	乘法	返回左右相乘	3*2返回6	
**	指数	返回左取右次方数	3**2返回9	
/	除法	返回左除以右	3.0/2返回1.5	Python 3后，/ 只用于浮点数除法，返回浮点数
//	地板除（取商）	返回右对左取商	3.0//2返回1.0	Python 3后，// 只用于整数除法，返回向下取整的整数
%	取模（取余）	返回右对左取模	5/3返回2；	取模即返回左数被右数整除后的余数

b. 赋值操作符

i. 简单赋值符：= ，将右值赋值给左

ii. 符合赋值符：+=，-=，*= 等等， 将左右先进行算术运算符计算后的返回的值，赋值给左

c. 比较操作符（返回布尔值）

操作符	用途	实例
==	如果左右的值相等则返回True，否则返回False	3==2返回False
!=	如果左右的值不等则返回True，否则返回False	3!=2返回True
<>	同上	3<>2返回True

>	如果左大于右则返回True，否则返回False	3>2返回True
<	如果左小于右则返回True，否则返回False	3<2返回False
>=	如果左大于或等于右则返回True，否则返回False	3>=2返回True
<=	如果左大于或等于右则返回True，否则返回False	3<=2返回False

d. 逻辑操作符（返回布尔值）

操作符	名称	用途	实例
and	逻辑与	只有左右均为True时返回True，否则返回False	True and True = True True and False = False False and False = False
or	逻辑或	只有左右至少一个为True时返回True，否则返回False	True or True = True True or False = True False or False = False
not	逻辑非	返回逻辑状态相反的布尔值	not True = False not False = True

e. 成员操作符

操作符	用途
in	如果左是在右中则返回True，否则返回False
not in	如果左不在右中则返回True，否则返回False

f. 操作符优先级

优先级			操作符		
最高	**				
	*	/	%	//	

	+	-			
	<=	<	>	>	=
	==	!=	<>		
	=	.=			
	in	not in			
最低	not	or	and		

2. 数据结构

a. 变量与赋值

Python 的变量是不可变对象，如果变量的值发生改变，将新创建一个新的对象申请一个新的内存地址，用于储存新的值，并将该变量指向新的内存地址（获得新值），而原有值的内存地址作废，等待回收。不可变对象的设计对程序的执行效率带来一定的影响

b. 数据类型

float：单精度浮点数，使用32位（4字节）来储存一个数字

double：双精度浮点数，使用64位（8字节）来储存一个数字

i. Python 三大基本数据类型：整数，浮点数，布尔值
（Python的浮点数是双精度浮点数，即double）

ii. type(x) 可以返回数据 x 的数据类型

iii. 整数与整数的运算结果仍然是整数，但只要其中一个为浮点数，运算结果即为浮点数

iv. 整数运算的结果是精确的，浮点数运算结果不一定精确。

3. 流程控制

a. If 语句

i. 布尔表达式：默认返回False 的值：False，None，0， “” ， （），[]，{} （其余单值默认返回True）

ii. 条件分支：if, elif, else

b. While 循环：while 后面的布尔表达式为True时，执行，直至布尔表达式为假

i. break 语句：跳出最内层的循环 （函数里用来跳过）
ii. continue 语句：跳到最内层循环的首行

c. for 循环
i. range() 函数：生成列表，左包含右不包含
ii. len()函数：返回列表的长度
! iii. 使用 for I in range(len())可以实现遍历列表中所有元素的同时，可以实现修改元素（通过L[i]来索引访问），例子

```
L = [1, 2, 3]
for I in range (len (L))
    L [i] += 1
print L  # 输出的L是[2,3,4]
```

! d. else 语句：当for 遍历整个列表，或者 while 的循环条件为False 时，如果没有break 终止循环，则可以通过 else 来执行终止循环，例子：for 循环结束时通过 else 过渡

```
# 简单搜索质数
for n in range (2, 10):
    for x in range (2, n):
        if n % x == 0:
        print n, "equals", x, "*", n/x
        break
    else:
        print n, "是一个质数"
```

4. 数据结构

a. 三种数据结构之一——标量（Scaler）：如整数，浮点数
b. 三种数据结构之二——序列（Sequence）：

i. 列表（List）：一个任意类型的对象的位置相关的有序集合，大小可变

1) 列表是有序的，通过索引（下标）来访问列表中的元素
2) 索引：从左索引，从0开始。从右索引，以负数表示，从-1开始
3) 切片：L[0:2]→ 表示从左起的索引0，到左起的索引1（右不包含），切出新列表；
   切片符：的左右如果不输入，则表示从左（或者右）起第一个索引开始（结束）
4) 列表方法：

list.append(x)	添加一个元素到列表的末尾
list.extend(L)	将列表 L 拼接到列表 list 的末尾
list.insert(i, x)	在索引 i 元素的前面插入一个元素x
list.remove(x)	删除列表第一个值为x的元素
list.pop([i])	删除列表中指定索引i位置的元素，并返回，如不输入索引参数，默认删除返回最后一个元素
list.index(x)	返回列表第一个值为x的元素的索引
list.count(x)	返回列表中值为x的元素的个数
list.sort()	排序列表中的元素
list.reverse()	反转列表中的元素

5) 列表用作栈（栈：后进先出）：在list中使用 append() 进行压入，使用 pop() 进行弹出
6) ★ 列表用作队列（队列：先进先出）：使用 collections.deque (popleft和popright)

ii. 字符串（String）：单个字符的序列，但是具有不可变性

1) 创建字符串：“” 或 ‘’
2) 跨行：'''...'''
3) 转义：\ 一般会作为转义符，\n 转义为换行符，\t 转义为制表符，如不想转义，则在代码前加 r
4) 占位符：%s %d 等
5) 字符串方法：

str.find(sub, [start, [end]]) str.find('to')	用于搜寻的方法， 返回在字符串中找到的子字符串 sub 的最低索引，使得sub 包含在切片 s[start:end] 中，如果未找到sub， 则返回-1 (简单来说就是找子字符串，如果有的话，返回其第一个字符的索引)
str.split(sep, [maxsplit]) str.split()	返回字符串中的单词列表， sep 是分隔符号， maxsplit 是最大拆分次数（因此列表中最多有 maxsplit+1个元素），如果没有指定，则无限分隔
str.join(iterator)	链接字符串数组。将字符串，元祖，列表中的元素以指定的字符链接成新的字符串
str.strip([chars])	返回字符串删除掉chars 之后的一个副本，如果chars留空，则是删除空格
str.lower()	大写变小写
str.upper()	小写变大写
★ str.isalnum	★如果字符串中至少有一个字符，并且都是数字或者字母，则返回True，否则返回False
str.count(sub, [start, [end]])	返回在[start, end] 范围内的子字符串sub 非重叠出现的次数（AA相较于AB就是重叠出现）
str.replace(old,new, [count])	返回字符串的一个副本，其中old子串都被new替换，count参数是指只有前面count个替换

iii. 元组（Tuple）：
   1) 元组与列表的区别：元组具有不可变性
   2) 元组的方法：与列表类似，但由于不变形，append() 和 pop() 方法 没有
iv. Unicode字符串： print u'\u4f60\u597d'， （ u' 字符串 ' ）
v. 字节数组
vi. 缓冲区
vii. xrange对象

c. 三种数据结构之三——映射（Mapping）：字典（Dictionary）
   i. 字典内部实现是基于二叉树，数据没有严格的顺序（字典是无序的）
   ii. 字典的方法：
      1) 创建：直接用 {} 赋值，其中用：隔开键和值，用逗号隔开不同的键值
      2) 返回值：D[key]
      3) 插入键值：D[key] = value， 如果字典里没有 这个键，则会创建这个键值，如果有，则

是修改值

4) 删除键值：del D[key]

iii. 字典的遍历：由于字典无序，所以遍历的方法是无穷的。经典的方法：先获取字典的所有键，并储存在列表中，由此，根据列表的序列（偏序关系），打印所有键值

d. 独立于三种数据结构之外的数据结构：集合（Set）

i. 集合既不是序列也不是映射，也不是标量

ii. 集合是唯一（add进相同元素的时候不会报错但不会多出一个一样的元素），不可变，无序的

iii. 集合的运算：

1) 创建：{a,b,c} 或者使用 set() 函数

2) 集合的差：x - y 返回包含在 x 且不包含在 y 中元素的集合

3) 集合的并：x | y 返回并集

4) 集合的交：x&y 返回交集

5) 集合的异或：x ^ y 返回只被x 包含或者只被y包含的元素的集合

6) 真子集：x > y 如果x 真包含 y ， 则返回True 否则返回Flase

iv. 集合的方法：

set.add(x)	往集合中插入元素x
set1.update(set2)	把set2中的元素添加到set1中
set.remove(x)	删除集合中的元素x
set1.union(set2)	并赋：set1 = set1 | set2
set1.intersection(set2)	交赋：set1 = set1 & set2
set1.difference(set2)	差赋：set1 = set1 - set2
set1.issuperset(set2)	判断 set1 >= set2 （子集）

5. 文件读写

a. 改变环境变量：引入 os 模块

i. 查看当前工作目录：os.getwd()

ii. 改变当前工作目录：os.chdir(路径)

向python输入路径时，需要使用斜杠 / 而不能使用 \ ，以避免自动转义

b. txt 文件读取

i. 读写模式： file1 = open(filename, mode = 'r') # 返回文件句柄

'r'	以读方式打开文件，仅可读取文件信息
'w'	以写方式开始文件，仅可以写入信息。（如文件已存在，则覆盖，如不存在，则创建）
'a'	以追加模式打开文件（即续写），自动移到文件末尾，仅可从末尾开始写入，如文件不存在，则创建
'r+'	以读写方式打开文件，可读可写
'w+'	清除文件内容，然后以读写方式打开文件，如文件不存在，则创建
'a+'	以读写+追加方式打开文件，自动移到文件末尾从末尾开始读写，如文件不存在，则创建
'b'	以二进制模式打开文件，而不是文本模式

ii. 文件句柄方法

f.close()	关闭文件，如果用open() 打开过文件，必须关闭，否则会一直占用系统资源（占用可打开句柄数）
f.flush()	刷新输出内存
f.read([count])	独处文件，如果有count参数，则读出count个字节
f.readline([count])	读出一行信息，如果有count参数，则读出count行
f.readlins	读出所有行，即读出整个文件所有内容
f.tell	获取当前文件指针位置
f.seek(p, [where])	把指针移动到指定位置，p是位置偏移量，如不输入即不偏移；where 是定位，0 即从文件开头，1 是从当前指针位置， 2是从文件结尾。 例如f.seek(2, 1) 表示将指针从当前位置移到当前位置往后2个字节处
f.write(string)	把string字符串写入文件

f.writelines(list)	把list中的字符串一行一行地写入文件

iii. 注意事项：

1) 如果文件有多行，使用readline读取某一行时，会把换行符 \n 也读取出来，而最后一行就不会有。如果想去掉换行符，可使用 strip() 方法，把空白符（包括换行符）删除

2) readlines 返回的是字符串列表

c. csv 文件读取

i. excel 文件无法直接读取到python 中，可以讲excel 数据保存为csv格式，即使用逗号分隔符的数据表，就可以使用python读取了

ii. csv 文件的读取和txt 文件一样

d. 文件输出

i. 一种更便捷的文件输出方法： print>>f 把原本print 函数 输出到shell 的内容改为输出到文件 f 例如，将二维数组[['1','2'],['3','4']]输出到文件：

★ for line in data:

★ print>>f, str(line[0]) + ',' + str(line[1]) + '\n'

6. 函数基础

a. 创建函数

i. def 语句： def fun_name (para1, para2, ...)

ii. return 语句： 当程序执行到return语句的时候，会将值返回并结束函数，如果return后面还有语句，将不会被执行

iii. lambda 语句：用于创建匿名函数（即无具体名称）： lambda para1, para2, ... : 返回的值

1) 例如： g = lambda x : x+1 创建一个匿名的累进函数，返回到g (注意返回给g的是一个函数)

2) 注意事项：

a) lambda 定义时单行函数，如果需要多行实现，必须用def

b) lambda 参数列表可以包含多个

c) lambda 语句有且只有一个返回值，如需要多个返回值，必须用def

d) lambda 语句中的返回表达式不能含有命令（如print），而且只限一条表达式

b. 参数类型

i. 位置参数，关键字参数：python 默认的参数类型，

1) 位置参数即根据定义时的位置，来确定输入的多个参数的具体指向。因此，参数之间有顺序关系

如 def fun1(a, b, c) 引用时， g = fun1(1, 2, 3) 各自按位置对应传递参数值

2) 关键字参数： 引用函数时声明需要传递的参数是哪个，因此，参数顺序没有关系

如 def fun1(a, b, c) 引用是， g = fun1(a=1, c=3, b=2)

3) 两者可以混合使用

4) 默认参数： 可以在定义函数的时候就把参数的默认值定义好

如 def fun2(a=0, b=0, c=0)

但是，不允许带默认值的参数定义在无默认值的参数之前

ii. 可变数量的位置参数，可变数量的关键字参数：

1) 可变数量的位置参数是在参数前面加*，而关键字参数是加**

2) 可变的意思是传入参数的数量是任意的，可以是0个，也可以是1个，也可以是多个

3) 各类参数的定义位置关系：

★ def fun(关键字参数，位置参数，可变位置参数，可变关键字参数)

```
# 如  def fun4 (a=0, b, *c, **d)
                print a, b, c, d
        fun4 (1, 2, 3, 4, 5, A=1, B=2)  返回 1 2 (3, 4, 5) {'A'=1, 'B'=2}
```

iii. 可变对象与不可变对象

1) 包括：

可变对象	字典，列表
不可变对象	数值类型，字符串，元组

2) Python 函数的参数都是对对象的引用，如果引用不可变对象时尝试修改对象，程序会在函数中生成新的对象，但引用的不可变对象不会被修改

iv. 作用域

1) Python 在创建、改变或者查找变量的时候，都是在命名空间中进行， 即在特定作用域下

进行，因此有局部变量（Local Variable）和全局变量（Global Variable）的关系

2) 当局部变量和全局变量发生重名时，在作用域内，全局变量会被局部变量屏蔽，正常来说无法访问，换言之，在特定的作用域内，默认访问其作用域内的局部变量。
3) 如果想访问全局变量，可以使用 globals 函数