Python

一种解释型的、面向对象的、带有动态语义的高级程序设计语言。

Python doc:

<https://docs.python.org>

<https://docs.python.org/3.5/library/functions.html>

产生：

由Guido van Rossum在20世纪90年代初创造的。

用途：

系统管理工作（比如它是很多Linux发行版的重要组成部分）。

Industrial Light&Magic使用其制作影片特效

Yahoo!使用它管理讨论组

Google用它实现Web爬虫和搜索引擎中的很多组件。

游戏和生物信息等领域。

扩展名：

.py

.pyw

被设计来运行开发完成的纯图形界面程序 的。

Python内建对象类型（数字、字符串、列表、元组和字典）

Python.exe与pythonw.exe

运算符：

+

-

\*

/

//：整除

%：取余

\*\*：乘方，-3\*\*2=-(3\*\*2)，(-3)\*\*2，函数pow(2.3)=8

普通整数不能大于2147483647（也不能小于-2147483647），如果真的需要大数，需使用长整型，结尾有个L。

让脚本像普通程序一样运行：

脚本首行加：

#!/usr/bin/env python

使其具有可执行性：

chmod a+x hello.py

abs(number)

cmath.sqrt(number)

float(object)

help()

input(prompt)

int(object)

long(object)

math.ceil(number)

math.floor(number)

math.sqrt(number)

pow(x,y[,z]) ：返回x的y次幂（所得结果对z取模）

raw\_input(prompt)

repr(object)

round(number[,ndigits])

str(object)

第二章 列表和元组

在python中，最基本的数据结构是序列（sequence）。序列中的每个元素被分配一个序号——即元素的位置，也称为索引。第一个索引是0，第二个是1，以此类推。也可以从最后一个元素开始计数，最后一个是-1， 倒数第二个为-2。

Python有6种内建序列：列表、元组、字符串、Unicode字符串、buffer对象、xrange对象。

列表和元组的主要区别在于，列表可以修改，元组则不能。

Python中的数据结构：容器（container）：容器基本上包含其他对象的任意对象。序列（列表和元组）和映射（字典）是两类主要的容器，序列中的每个元素都有自己的编号，而映射中的每个元素则有一个名字（也称为键）。既不是序列也不是映射的容器类型的一个例子就是集合。

通用序列操作：索引（indexing）、分片（slicing）、加（adding）、乘（multiplying）以及检查某个元素是否属于序列的成员（成员资格）。除此之外，Python还有计算序列长度、找出最大元素和最小元素的内建函数。

迭代（iteration），对序列进行迭代的意思是：依次对序列中的每个元素重复执行某些操作。

字符串字面值能够直接使用索引，而不需要一个变量引用它们。‘Hello’[1]

分片：分片操作的实现需要提供两个索引作为边界，第1个索引的元素是包含在分片内的，而第2个则不包含在分片内。

序列相加+

两种相同类型的序列才能进行连接操作。

乘法：用数字 x乘以一个序列会生成新的序列，而在新的序列中，原来的序列将被重复x次。

None、空列表和初始化：

空列表表示方法：[]，初始化长度为10的列表：sequence = [None]\*10

成员资格：

为了检查一个值是否在序列中，可以使用in运算符。

Len, max, min

List：根据字符串创建列表。

List(‘Hello’)

基本的列表操作

1. 改变列表：元素赋值 x[1]=2
2. 删除元素：del

列表方法：

对象.方法（参数）

Append

Count

Extend

Index

Insert

Pop：pop方法是唯一一个既能修改列表又返回元素值（除了None）的列表方法。

后进先出可以用append, pop, 先进先出：insert/append, pop(0)，或deque。

Remove

Reverse

Sort：sort, reverse搭配用排序再反转。

元组：不可变序列

元组与列表一样，唯一不同的是元组不能修改。

元组没有像列表一样的方法。

Tuple函数

与list函数基本上一样：以一个序列作为参数并把它转换为元组。如果参数就是元组，那原样返回。

元组意义何在：

元组可以在映射（和集合的成员）中当作键使用——而列表则不行。

元组作为很多内建函数和方法的返回值存在。

本章函数

Len(seq)

List(seq)

Max(args)

Min(args)

Reversed(seq)

Sorted(seq)

Tuple(seq)

字符串

字符串是不可变的。

字符串格式化用字百分号%来实现。在%左侧放置一个字符串（格式化字符串），而右侧则放置希望格式化的值。可以使用一个值，如一个字符串或者数字。

格式化字符串的%s部分称为转换说明符（conversion specifier），它们标记了需要插入转换值 的位置。S表示值会被格式化的字符串——如果不是字符串，则会用str将其转换为字符串。

如果要在格式化字符串里面包括百分号，那必须使用%%。

如果要格式化实数（浮点数），可以使用f说明符类型，同时提供所需要的精度：一个句点再加上希望保留的小数位数。

Format = “Pi with three decimals: %.3f”

From math import pi

Print format % pi

格式化操作符的右操作数可以是任何东西。如果右操作数是元组的话，则其中的每一个元素都会被单独格式化，每个值 都需要一个对应的转换说明符。

基本的转换说明符

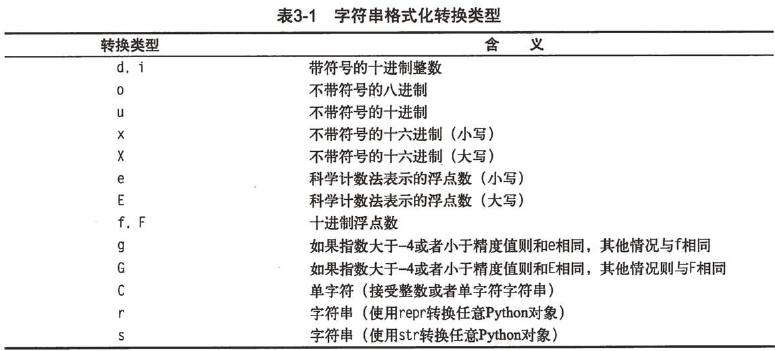
%字符：标记转换说明符的开始

转换标志（可选）：-表示左对齐；+表示在转换值之前要加上正负号；“”（空白字符）表示正数之前保留空格；0表示转换值 若位数不够则用0填充。

最小字段宽度（可选）：转换后的字符串至少应该具有该值指定的宽度。如果是\*， 则宽度会从值 元组中读出。

点（.）后跟精度值（可选）：如果转换的是实数，精度值就表示出现在小数点后的位数。如果转换的是字符串，那么该数字就表示最大字段宽度。如果是\*，那么精度将会从元组中读出。

转换类型如下：



字段宽度和精度

首先是字段宽度，然后是精度，通过（.）分隔，如果只给出精度，必须包含点号。

有用的字符串常量：

String.digits: 包含数字0~9的字符串。

String.letters: 包含所有字母（大写或小写）的字符串。3.0已被移除，由string.ascii\_letters替代。

String.lowercase: 包含所有小写字母的字符串。

String.printable: 包含所有可打印字符的字符串。

String.punctuation: 包含所有标点的字符串。

String.uppercase: 包含所有大写字母的字符串。

Find在一个较长的字符串中查找子字符串。返回子串所在位置的最左端索引。如果没有找到则返回-1。

Rfind, index, rindex, count, startwith, endswith

Join: split的逆方法，用来在队列中添加元素。需要添加的队列元素必须是字符串。

Lower：返回字符串的小写字母版。

Translate, islower, capitalize, swapcase, title, istitle, upper, isupper.

Title：将字符串转换为标题——也就是所有单词首字母大写，但它使用的单词划分方法可能会得到并不自然的结果：

Capwords

Replace

Split：如果不提供任何分隔符，程序会把所有空格作为分隔符。

Join, rsplit, splitlines

Strip：返回去除两侧（不包括内部）空格的字符串。它和lower方法一起使用的话就可以很方便的对比输入的和存储的值。也可以指定需要去除的字符，将其列为参数即可。

Lstrip, rstrip

Translate：可以替换字符串中的某些部分，只处理单个字符，可以同时进行多个替换。

第四章 字典 当索引不好用时

Mapping：通过名字引用值的数据结构，字典是python中唯一内建的映射类型。

字典中的值没有特殊的顺序，但是都存储在一个特定的键（key）里。键可以是数字、字符串甚至是元组。

某些情况下，字典比列表更加适用，比如：

表征游戏棋盘的状态，每个键都是由坐标值组成的元组；

存储文件修改次数，用文名作为键；

数字电话/地址簿

创建和使用字典：

每个键和值之间用冒号（：），项之间用逗号（，），整个由大括号括起来（{}）

注意：字典中的键是唯一的（其他类型的映射也是如此），而值并不唯一。

Phonebook = {‘Alice’: ‘1234’, ‘Beth’: ‘9102’, ‘Cecil’:’3258’}

基本字典操作：

Len(d)：返回d项（键-值对）的数量。

d[k]：返回关联到键k上的值。

d[k]=v：将值v关联到键k上

del d[k]：删除键为k的项

k in d：检查d中是否有含有键为k的项。查找的是键，而不是值。表达式v in l （l为列表）则用来查找值，而不是索引。

字典的特点：

键可以为任何不可变类型。

字典方法：

Clear，类似于list.sort，无返回值 （或者说返回None）。

Copy，返回一个具有相同键-值对的新字典（这个方法实现的是浅复制（shallow copy），因为值本身就是相同的，而不是副本）。

当在副本中替换值的时候，原始字典不受影响，但，如果修改了某个值（原地修改，而不是替换），原始的字典也会改变，因为同样的值也存储在原字典中。

避免此类问题可以使用深复制（deep copy），复制其包含所有的值，可以使用cipy模块的deepcopy函数来完成操作。

Fromkeys，使用给定的键建立新的字典，每个键默认对应的值为None。

Get，更宽松的访问字典项的方法。即使试图访问字典中不存在的键时不会异常。

注意，get方法带来的灵活性如何使得程序在用户输入我们并未准备的值时也能做出合理反应。

has\_key，可以检查字典中是否含有给出的键。表达式d.has\_key(k)相当于k in d。python 3.0中不包括这个函数。

Items和iteritems，items将所有的字典项以列表方式返回，这些列表项中的每一项都来自于（键，值）。但是项在返回时并没有特殊的顺序。

Iteritems方法的作用大致相同，但是会返回一个迭代器对象而不是列表：

Keys and iterkeys，keys方法将字典中的键以列表形式返回，而iterkeys则返回针对键的迭代器。

Pop方法用来获得对应于给定键的值，然后将这个键-值对从字典中移除。

Popitem类似list.pop，后者会弹出列表的最后一个元素。不同的是，popitem弹出随机的项。若想一个接一个地移除并处理项，这个方法就非常有效了（因为不用首先获取键的列表）。

Setdefault方法类似get方法，就是能够获得与给定键相关联的值，除此之外，setdefault还能在字典中不含有给定键的情况下设定相应的键值。当键不存在的时候，setdefault返回默认值并且相应地更新字典。如果键存在，就返回与其对应的值，但不改变字典。默认值是可选的，这点和get一样。如果不设定，会默认使用None。

Update， 可以利用一个字典项更新另外一个字典，提供的字典中的项会被添加到旧的字典中，若有相同的键则会进行覆盖。

Values and itervalues，values方法以列表的形式返回字典中的值（itervalues返回值的迭代器）。与返回键的列表不同的是，返回值的列表中可以包含复复的元素。

小结：

映射：映射可以使用任何不可变对象标识元素。最常用的类型是字符串和元组。Python唯一内建的映射类型是字典。

利用字典格式化字符串

字典的方法：与调用列表及字符串类似。

第5章 条件、循环和其他语句

Print, import，使用logging模块记日志比print语句更合适。

从模块导入函数的时候，可以使用

Import somemodule

From somemodule import somefunction

From somemodule import somefunction, anotherfuncion, yetanotherfunction

From somemodule import \*只有确定想要从给定的模块导入所有功能时，才使用这个版本。

如果两个模块都有open函数，可以使用

Module1.open(…)

Module2.open(…)

另一种方法，在语句末尾增加一个as子句，在该子句后给出名字，或为整个模块提供别名：

Import math as foobar

Foobar.sqrt(4)

也可以为函数提供别名：

From math import sqrt as foobar

Foobar(4)

对于open函数，可以像下面这样使用：

From module1 import open as open1

From module2 import open as open2

链式赋值：（chained assignment）

X = y = somefunction()

增量赋值：（augmented assignment）

X += 1

语句块：

Python一个tab字符位置为8个空格。

Python中，冒号（：）用来标识语句块的开始，块中的每一个语句都是缩进的。当回退到和已经闭合的块一样的缩进量时，就表示当前块已经结束了。

以下值在作为布尔表达式的时候，会被解释器看作假（false）：

False None 0 “” () [] {}

其他的一切都被解释为真，包括特殊值True。

If condition:result

Elif condition:result

Else:result

比较运算符

X == y

X < y

X > y

X >= y

X <= y

X != y

X is y x和y是同一个对象

X is not y x和y是不同的对象

X in y x是y容器的成员

X not in y x不是y容器的成员

And, or, not

循环遍历

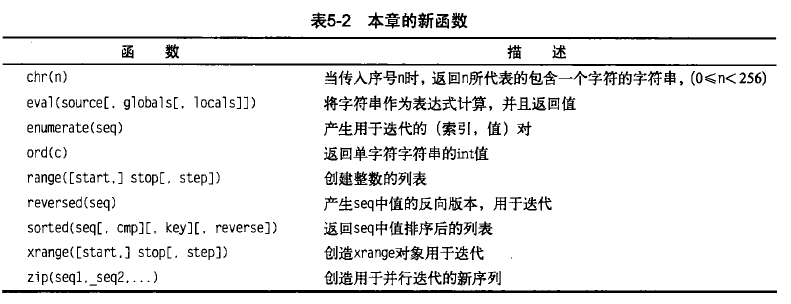
For key in keys: print …

While x <= 100:

print(x)

x += 1

第五章新函数



第六章 抽象

第七章 更加抽象

创建对象、多态、封装、方法、特性、超类以及继承的概念。

7.1 对象的魔力

面向对象程序设计中的对象基本上可以看做数据（特性）以及由一系列以存取、操作这些数据的方法所组成的集合。使用对象替代全局变量和函数的原因有：

多态Polymorphism：意味着可以对不同类的对象使用同样的操作。

封装Encapsulation：对外部世界隐藏对象的工作细节。

继承Inheritance：以普通的类为基础建立专门的类对象。

7.1.1 多态

1、多态和方法

2、多态的多种形式

任何不知道对象到底是什么类型，但是又要对对象“做点什么”的时候，都会用到多态。很多内建运算符和函数都有多态的性质。

唯一能够毁掉多态的就是使用函数显示地检查类型，比如type、isinstance以及issubclass函数等。如果可能的话，应该尽量避免使用这些毁掉多态的方式。真正重要的是如何让对象按照你所希望的方式工作，不管它是否是正确的类型（或者类）。

多态是Python的核心，也被称为鸭子类型（duck typing）。

7.1.2 封装

7.1.3 继承

7.2 类和类型

7.2.1 类到底是什么

所有的对象都属于某一个类，称为类的实例（instance）。

当一个对象所属的类是另外一个对象所属类的子集时，前者就被称为后者的子类（subclass）。

7.2.2 创建自己的类

7.2.3 特性、函数和方法

Self参数事实上正是方法和函数的区别。

一些关于面向对象设计的思考

* 将属于一类的对象放在一起。如果一个函数操纵一个全局变量，那么两者最好都在类内作为特性和方法出现。
* 不要让对象过于亲密。方法应该只关心自己实例的特性。让其他实例管理自己的状态。
* 要小心继承，尤其是多重继承。继承机制有时很有用，但也会在某些情况下让事情变得过于复杂。多继承难以正确使用，更难以调试。
* 简单就好。让你的方法小巧。一般来说，多数方法都应能在30秒内被读完（以及理解），尽量将代码行数控制在一页或者一屏之内。
* 当考虑需要什么类以及类要有什么方法时，应该尝试下面的方法：
  + 写下问题的描述（程序要做什么？），把所有名词、动词和形容词加下划线。
  + 对于所有名词，用作可能的类。
  + 对于所有的动词，用作可能的方法。
  + 对于所有形容词，用作可能的特性。
  + 把所有方法和特性分配到类。

第八章 异常