第二章：Python起步

目录

[2.1 程序输出：print 语句 1](#_Toc350418121)

[2.2 程序输入： raw\_input()内建函数 2](#_Toc350418122)

[2.3 注释 2](#_Toc350418123)

[2.4 运算符 3](#_Toc350418124)

[2.5 变量和赋值 3](#_Toc350418125)

[2.6 数字 3](#_Toc350418126)

[2.7 字符串 4](#_Toc350418127)

[2.8 列表和元组 5](#_Toc350418128)

[2.9 字典 6](#_Toc350418129)

[2.10 代码块及缩进对齐 6](#_Toc350418130)

[2.11 if 语句 7](#_Toc350418131)

[2.12 while 循环 7](#_Toc350418132)

[2.13 for 循环和 range()内建函数 7](#_Toc350418133)

[2.13 列表解析 9](#_Toc350418134)

[2.15 文件和内建函数 open() 、file() 9](#_Toc350418135)

[2.16 错误和异常 9](#_Toc350418136)

[2.17 函数 10](#_Toc350418137)

[2.18 类 10](#_Toc350418138)

[2.19 模块 11](#_Toc350418139)

[2.19 实用的函数 12](#_Toc350418140)

[2.20关于进制 13](#_Toc350418141)

[2.21第一二章的课后题 13](#_Toc350418142)

## 2.1 程序输出：print 语句

1、print（通配符支持，重定向支持）

print支持类c的printf格式化输出: print “%s is number %d!” % (“python”, 1)

print的输入内容后面加逗号, 就会使其输入不换行。

把输出重定向到日志文件:

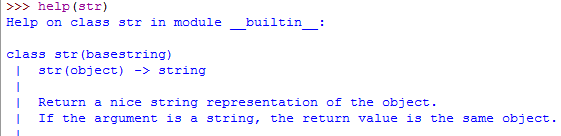
logfile = open(“c:/1.log”, “a”); //打开文件c:/1.log使用a模式..即add, 添加.

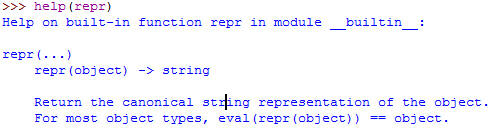
print >> logfile, “Fatal error: invalid input!”; >>为重定向..将print的结果重定向到logfile, 输出内容是”Fatal error: invalid input!”…

logfile.close(); //关闭文件流…

2、\_在解释器中表示最后一个表达式的值.

3、在交互式解释器中，你可以用 print 语句显示变量的字符串表示，或者仅使用变量名查看该变量的原始值。在仅用变量名时，输出的字符串是被用单引号括起来了的。即它显示的是该对象的字符串表示，而不仅仅是字符串本身。其实，print语句调用str()函数显示对象，而交互式解释器则调用repr()函数来显示对象。





canonical 标准的

## 2.2 程序输入： raw\_input()内建函数

程序输入: raw\_input(“提示字符串”): user = raw\_input(“请输入您的姓名”);

int(数值)…..将数值字符串转换成整数值…

## 2.3 注释

Python 也使用 # 符号标示注释

## 2.4 运算符

1、+ - \* / %是和其他语言相同的加减乘及取模运算.取余运算

1. / 在浮点取模中得到的结果是完整的浮点数
2. // 在浮点取模中得到的结果是经过舍去运算的结果.
3. \*\* 是乘方
4. >>和<<的移位运算也支持. 但不支持java中的>>> 和<<< 移位.
5. < <= > >= ++ != <> 等比较运算符

7、and or not 等逻辑运算符

## 2.5 变量和赋值

1、变量名：以字母或下划线开头，其它的字符可以是数字，字母，或下划线。Python变量名是大小写敏感的。

2、Python 是动态类型语言， 也就是说不需要预先声明变量的类型。 变量的类型和值在赋值那一刻被初始化。Python 也支持增量赋值，也就是运算符和等号合并在一起， 看下面的例子:n \*= 10

3、Python 不支持 C 语言中的自增 1 和自减 1 运算符， 这是因为 + 和 － 也是单目运算符，Python 会将 --n 解释为-(-n) 从而得到 n , 同样 ++n 的结果也是 n.

## 2.6 数字

1、int(有符号整数)、long(长整数)、bool(布尔值)、float(浮点值)、complex(复数)

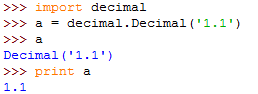


布尔值由常量 True 和 False 来表示， 如果将布尔值放到一个数值上下文环境中（比方将 True 与一个数字相加）， True 会被当成整数值 1， 而 False则会被当成整数值 0。

2、python2.3开始, 如果结果从int溢出, 会自动转型为long，int和long以后可能被转化成一种类型。

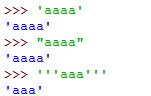
3、其实还有第六种数字类型， decimal， 用于十进制浮点数。不过它并不是内建类型， 你

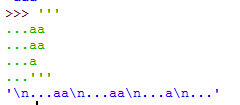
必须先导入 decimal 模块才可以使用这种数值类型。 由于需求日渐强烈， Python 2.4 增加了这种类型。举例来说， 由于在二进制表示中有一个无限循环片段，数字 1.1 无法用二进制浮点数精确表示。因此， 数字 1.1 实际上会被表示成：1.1000000000000001. 使用print decimal.Decimal(‘1.1’);则可以得到精确的1.1。



## 2.7 字符串

Python 支持使用成对的单引号或双引号，三引号（三个连续的单引号或者双引号）可以用来包含特殊字符。使用索引运算符( [ ] )和切片运算符( [ : ] )可以得到子字符串。字符串有其特有的索引规则：第一个字符的索引是 0， 最后一个字符的索引是 －1。加号( + )用于字符串连接运算，星号( \* )则用于字符串重复。





你会发现他把换行符给你检测出来了。三引号(三个连续单引号或双引号)可以用来包含特殊字符。

1、切片操作：

单个元素的切片，直接使用一个索引就可以了。





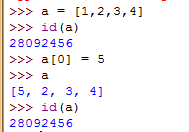
切片操作还支持负数：负数就是从后往前数。



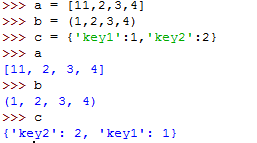
我们也可以省略前边或者后边，第一个即是默认是你的开始，第二个默认就是你的长度（从1开始数）。

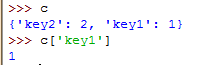


## 2.8 列表和元组

1、可以将列表和元组当成普通的“数组”，它能保存任意数量任意类型的 Python 对象。和数组一样，通过从 0 开始的数字索引访问元素，但是列表和元组可以存储不同类型的对象。 

列表元素用中括号包裹, 元素的个数及元素的值可以改变.元组元素用小括号包裹, 不可以更改, 尽管他们的内容可以, 元组可以看成是只读的列表. 可以使用切片运算得到子集。这一点与字符串的使用方法一样。



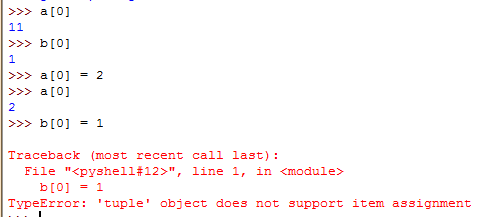


2、注意：

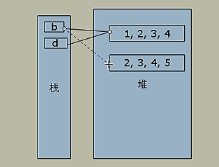
\* 4 得到的结果是一个新的 list | tuple | map, 是原数据的4份。

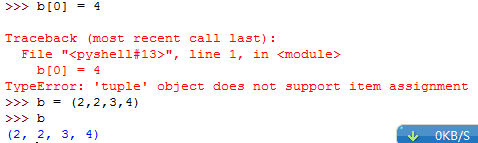
元组类型不支持为里面的元素指定新的值，所以说元组是不可变对象。但是列表是可变对象。

在学习python的过程中，希望大家建立起来可变对象和不可变对象。理解这个东西，可能对你写程序快不快没有影响，但是对你提高自己的能力和提高程序的安全性很有用的。



对不可变的理解：





即单个内容不可改，整体可改。

## 2.9 字典

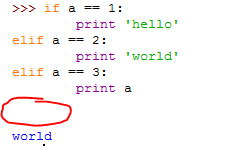
相当于其他语言中的map, 使用{key: value}的方式表示. 取值的方式和其他语言的map一致. 也可以直接使用map[key] = value的方式为其赋值.几乎所有类型的 Python 对象都可以用作键，不过一般还是以数字或者字符串最为常用。字典元素用大括号({ })包裹。

## 2.10 代码块及缩进对齐

代码块通过缩进对齐表达代码逻辑而不是使用大括号，因为没有了额外的字符，程序的可

读性更高。

## 2.11 if 语句



1、if expression:

if\_suite

2、if expression:

if\_suite

else:

else\_suite

3、if expression1:

if\_suite

elif expression2:

elif\_suite

else:

else\_suite

4、在本书写作之时， 正在进行一个关于是否需要增加 switch/case 语句的讨论， 不过目前并没有什么实质性的进展。

## 2.12 while 循环

while expression:

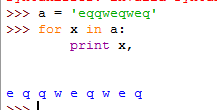
while\_suite

## 2.13 for 循环和 range()内建函数

Python 中的 for 循环与传统的 for 循环（计数器循环）不太一样， 它更象 shell 脚本里

的 foreach 迭代。Python 中的 for 接受可迭代对象（例如序列或迭代器）作为其参数，每次迭代其中一个元素。

1、for item in list|tuple|map:



for item in list|tuple|map:

print item

1. range()

Python 提供了一个 range()内建函数来生成这种列表。它正好能满足我们的需要，接受一个数值范围，生成一个列表。

Range的帮助文档的解释：

range([start,] stop[, step]) -> list of integers

range(i, j) returns [i, i+1, i+2, ..., j-1]; start (!) defaults to 0.

When step is given, it specifies the increment (or decrement).

For example, range(4) returns [0, 1, 2, 3].

当有step的时候，必须有start



例如：

>>> for eachNum in range(3):

... print eachNum

...

0

1

2

range()函数经常和 len()函数一起用于字符串索引。 在这里我们要显示每一个元素及其

索引值：

>>> foo = 'abc'

>>> for i in range(len(foo)):

... print foo[i], '(%d)' % i

...

a (0)

b (1)

c (2)

1. enumerate()

因为以上的这些循环有一个约束，你要么循环索引，要么循环元素。这导致了 enumerate()

函数的推出（Python2.3 新增）。 它同时做到了这两点，即循环索引又循环元素。你可以通过enumerate()遍历list, 同时得到索引和值。

例如：

>>> for i, ch in enumerate(foo):

... print ch, '(%d)' % i

...

a (0)

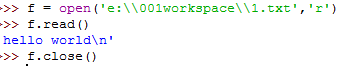
b (1)

c (2)

## 2.13 列表解析

## 2.15 文件和内建函数 open() 、file()

handle = open(file\_name, access\_mode = “r”), 只读方式打开文件, 得到的句柄是handle..该方法如果没有提供access\_mode, 默认是r



## 2.16 错误和异常

要给你的代码添加错误检测及异常处理， 只要将它们封装在 try-except 语句当中。 try之后的代码组， 就是你打算管理的代码。 except 之后的代码组， 则是你处理错误的代码。另外，程序员也可以通过使用 raise 语句故意引发一个异常。

例如：

try:

filename = raw\_input('Enter file name: ')

fobj = open(filename, 'r')

for eachLine in fobj:

print eachLine, fobj.close()

except IOError, e:

print 'file open error:', e

## 2.17 函数

Python 中的函数使用小括号( () )调用。函数在调用之前必须先定义。如果函数中没有 return 语句， 就会自动返回 None 对象。 Python 是通过引用调用的。这意味着函数内对参数的改变会影响到原始对象。不过事实上只有可变对象会受此影响，对不可变对象来说，它的行为类似按值调用。

def function\_name([arguments]):

“optional document string”

function\_suite

* 1. python的函数调用中参数是引用传递
  2. 可以在定义函数的时候, 在参数列表中通过=设置参数的默认值.

## 2.18 类

1、定义:

class class\_name:

static\_variable\_name = value

def \_\_init\_\_(self, [arguments]):

//operation

//self in here is the reference for this class instance

def general\_method\_name(self, [arguments]):

//operation

//self is the class instance

//if you want to use class variable, please use like self.\_\_class\_\_.\_\_name\_\_

2、实例化: instance = class\_name([arguments, …]);

当一个类实例被创建时， \_\_init\_\_() 方法会自动执行， 在类实例创建完毕后执行， 类似构建函数。\_\_init\_\_() 可以被当成构建函数，不过不象其它语言中的构建函数，它并不创建实例--它仅仅是你的对象创建后执行的第一个方法。它的目的是执行一些该对象的必要的初

始化工作。通过创建自己的 \_\_init\_\_() 方法，你可以覆盖默认的 \_\_init\_\_()方法（默认的

方法什么也不做），从而能够修饰刚刚创建的对象。在这个例子里， 我们初始化了一个名为 name 的类实例属性（或者说成员）。这个变量仅在类实例中存在， 它并不是实际类本身的一部分。 \_\_init\_\_()需要一个默认的参数， 前一节中曾经介绍过。毫无疑问，你也注意到每个方法都有的一个参数，什么是 self ? 它是类实例自身的引用。其他语言通常使用一个名为 this 的标识符。

## 2.19 模块

模块:带.py后缀名的文件名…一个模块创建之后, 可以使用import导入这个模块使用.

模块是一种组织形式， 它将彼此有关系的 Python 代码组织到一个个独立文件当中。

模块可以包含可执行代码， 函数和类或者这些东西的组合。

1、一个模块创建之后， 你可以从另一个模块中使用 import 语句导入这个模块来使用

如何导入模块

import module\_name

如何访问一个模块函数或访问一个模块变量，一旦导入完成， 一个模块的属性（函数和变量）可以通过熟悉的 .句点属性标识法访问。

module.function()

module.variable

2、sys模块概览

sys.stdout.write(‘Hello World!\n’) //使用sys模块的标准输出

sys.platform //返回系统的标记

sys.version //返回系统的版本

## 2.19 实用的函数

表 2.1 对新 Python 程序员有用的内建函数

dir([obj]) 显示对象的属性, 如果没有提供参数, 显示全局变量的名字

dir([object]) 如果没有参数获得当前脚本scope内定义的对象, 如果有参数, 返回该对象内部定义的对象, 如果该对象有一个\_\_dir\_\_方法, 该方法将被调用, 并且必须返回属性的列表…这就允许通过自定义\_\_getattr\_\_()或\_\_getattribute\_\_()方法的方式实现dir的自定义显示属性列表….如果没有指定参数, 是根据该对象的\_\_dict\_\_内存字典的最佳聚合信息显示的..

help([obj]) 显示对象的文档, 如果没有参数, 进入交互式帮助

int(obj) 将一个对象转换为整数

len(obj) 返回对象的长度

open(file\_name, mode) 以mode(r|w|a…)方式打开一个文件

range([[start, ]stop[, step]]) 返回一个整数列表…结束值是stop-1, step默认是1

raw\_input(str) 提示str等待用户输入

str(obj) 将一个对象转换为字符串

type(obj) 返回对象的类型…返回值本身是一个type对象

type([object]) 参数为空显示<type ‘type’>, 参数不为空显示该对象的类型

type(name, bases, dict) 通过名称, 基类, 内存字典动态创建一个类型

sum(iterable[, start=0]) 可以对纯数值的list|tuple|map进行求和操作..

object\_\_name.\_\_doc\_\_ 查看该对象的文档字符串

\_\_doc\_\_ 对象的文档字符串, 该文档字符串在定义对象时写在对象语句块中第一句, 使用单纯的字符串的方式表示

sys.exit() 退出python解释器

append(Object) 给list添加一个元素，这跟数组是不一样的，他可以接受不同类型的元素。

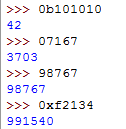
os.linesep 返回的是系统换行符…不同的系统换行符是不同的, 使用linesep可以提高代码跨平台性，首先要引入os



string\_variable\_name.strip([chars]) 脱离, 滤去字符串中的某些字符, 如果没有参数返回原字符串

## 2.20关于进制

* 1. 二进制: 0b101010
  2. 八进制: 07167
  3. 十进制: 98767
  4. 十六进制: 0xf2134



## 2.21第一二章的课后题

具体参python核心编程175页

