**Python学习笔记**

LouisozZ

**基础知识**

**-----------------------------------------------------------**

**输出为**print(‘str1’,’str2’,’str3’)会依次打印每个字符串，遇到逗号“,”会输出一个空格。

**输入为**input(‘输入前打印该引号内的信息’)。

Name = input(‘What\’s your name?’) # \’ 是对 ‘ 进行转译

Show:

What’s your name? 光标

以#开头的语句是注释，注释是给人看的，可以是任意内容，解释器会忽略掉注释。其他每一行都是一个语句，当语句以冒号:结尾时，缩进的语句视为代码块，缩进标准为4个空格。

**Python的数据类型：**

整数、浮点数、字符串、布尔值、空值（None）（注意大小写）、常量

Python是动态语言，变量本身类型不固定

e.g.对于a = ‘abc’ 这个语句

Python解释器干了两件事情：

1.在内存中创建了一个'abc'的字符串；

2.在内存中创建了一个名为a的变量，并把它指向'abc'。

当对另一个变量b赋值时，b=a 然后改变a的值，b的值是不会改变的，因为a与‘abc’的关系只是指向的关系，后面改变a的值只是修改了指向关系而已

Python提供了ord()函数获取字符的整数表示，chr()函数把编码转换为对应的字符

>>> ord('A')

65

>>> ord('中')

20013

>>> chr(66)

'B'

>>> chr(25991)

'文'

Python对bytes类型的数据用带b前缀的单引号或双引号表示 x = b'ABC'

以Unicode表示的str通过encode()方法可以编码为指定的bytes

>>> 'ABC'.encode('ascii')

b'ABC'

>>> '中文'.encode('utf-8')

b'\xe4\xb8\xad\xe6\x96\x87'

反过来，如果我们从网络或磁盘上读取了字节流，那么读到的数据就是bytes。要把bytes变为str，就需要用decode()方法

>>> b'ABC'.decode('ascii')

'ABC'

>>> b'\xe4\xb8\xad\xe6\x96\x87'.decode('utf-8')

'中文'

由于Python源代码也是一个文本文件，所以，当你的源代码中包含中文的时候，在保存源代码时，就需要务必指定保存为UTF-8编码。当Python解释器读取源代码时，为了让它按UTF-8编码读取，我们通常在文件开头写上这两行：

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

第一行注释是为了告诉Linux/OS X系统，这是一个Python可执行程序，Windows系统会忽略这个注释；

第二行注释是为了告诉Python解释器，按照UTF-8编码读取源代码，否则，你在源代码中写的中文输出可能会有乱码。

**格式化输出**：其中，格式化整数和浮点数还可以指定是否补0和整数与小数的位数，%%表示一个%

>>> 'Hello, %s' % 'world'

'Hello, world'

>>> 'Hi, %s, you have $%d.' % ('Michael', 1000000)

'Hi, Michael, you have $1000000.'

>>> '%2d-%02d' % (3, 1)

' 3-01'

>>> '%.2f' % 3.1415926

'3.14'

**列表 & 元组**

列表是可变的，而元组是静态不可变的，所指的不可变是对于元组元素的不可变。如下元组t[2][0]和t[2][1]是可变的，因为列表p才是元组元素，而p是列表，列表元素是可变的。

列表格式：>>> p = ['asp', 'php']

>>> L = ['Apple', 123, True, p]

元组格式：>>> t = ('a', 'b',p)

列表和元组都是可以通过下标索引的，跟C语言一样，但是索引的区别在于索引下标可以是负数，用来表示倒序查找，其中 -1 表示倒数第一个， -2 表示倒数第二个

>>>t[-2]

>>>’b’

对于列表，有如下**方法：**

列表名.append(列表元素) ----末尾追加

列表名.insert(下标，列表元素) ----在指定位置插入

列表名.pop() ----删除列表末尾元素

列表名.pop(下标) ----删除指定位置元素

列表名.sort() ----对列表进行排序

修改列表元素值直接赋值

**循环**

1. for 变量名 in 元组or列表 **:**

执行代码

循环就是把元组或者列表中的每个元素代入变量，然后执行缩进块的语句

1. while判断条件 **:**

执行代码

**字典**

字典格式：

>>>字典名 = {‘KEY1’:VALUE1,’KEY2’:VALUE2,’KEY3’:VALUE3}

创建字典的时候，必须键入key和value，而且一个key对应一个value，当多次对key放入value，前面的值会被后面的值冲掉。

对字典的操作：

>>> 'KEY' in 字典名 ----判断字典中是否有该KEY

>>>字典名.get(‘KEY’,返回值) ----默认如果KEY不存在，则返回None,且交互命令行不显示结果。

Set和字典的区别在于有没有值，set只有key而没有value，要创建一个set，需要提供一个list作为输入集合，set可以看成数学意义上的无序和无重复元素的集合，因此，两个set可以做数学意义上的交集、并集等操作

**可变与不可变对象**

对可变对象进行操作，可变对象的内容是会改变的。而对于不可变对象，对他进行的变化操作，实际上是新建了一个新的对象，原对象没有发生任何改变。

**函数**

**函数定义格式：**

def【关键字def】 func\_name【函数名】(parament\_1,parament\_2,…【参数列表】):【冒号】

statements【缩进表示函数内部语句】

return

如果函数在交互环境中定义，那么定义过程中会出现…提示，定义结束后需要按两次回车退出函数定义状态。如果函数已经在.py文件中定义好了，那么使用文件中的函数需要在交互环境中用语句 from filename\_without\_py import func\_name

**空函数**

def nop():

pass

pass代表什么都不做，有两个作用，一个是定义空函数，一个是先定义接口，使程序能跑起来，后面再来补充函数具体内容。

**返回值**

函数可以返回多个返回值，当有多个返回值的时候其实返回的是一个元组，可以用一个元组来接收，也可以用对应个数的变量接收，但是要注意顺序

**参数**

对于默认参数（即在定义函数时候的参数列表中给参数赋值，那么该参数就有了默认值，该参数变成了默认参数），位置应该在参数列表的后面（右边），如果有多个默认参数，但是只需要修改中间某一个（非第一个默认参数），则调用函数的时候需要写出该参数名且赋值。

**可变参数（参数个数可变）**

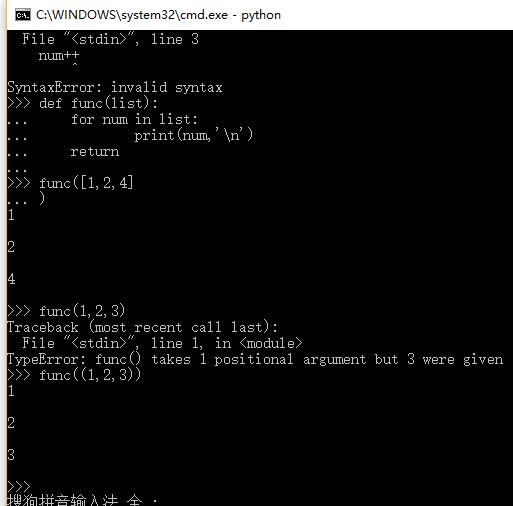
1. 将参数做成一个列表或者元组传递进来

def calc(numbers):

sum = 0

for n in numbers:

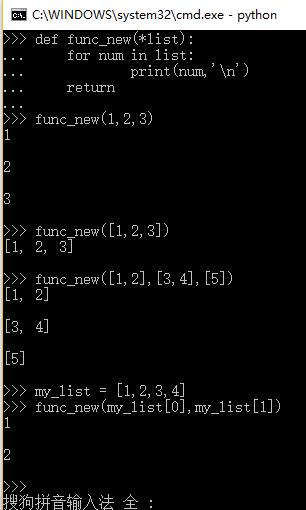
note：这种情况下只能是列表或者元组作为参数



1. 将参数改为可变参数，即在参数前面加一个 ‘ \* ‘

这样在调用函数的时候参数的传入就方便了，func(1,2,3)。但是需要注意的是这时候传入进去的参数一定要正确，不然会出现歧义，与函数设计的初衷出现偏差。

对于事先已经定义了一个元组或者列表，希望传入的是其中的元素的话，可以在调用的时候，对形参加一个 ‘ \* ‘，这样传入的就是里面的元素了。



**关键字参数**

关键字参数会在函数内部自动组装为一个dict。关键字参数允许你传入0个或任意个含参数名的参数，这些关键字参数在函数内部自动组装为一个dict

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 复习 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\* 字典：关键字+值

/\* 格式：dict = {‘key1’:value1 , ‘key2’:vlaue2 , …}

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

def person(name, age, \*\*kw):

其中 \*\*kw 就是一个关键字参数，他的参数类型就是一个字典，可以通过先定义一个字典然后传入，也可以直接在调用函数的时候写 关键字=值（e.g. city = ‘chengdu’）

person('Adam', 45, gender='M', job='Engineer')

extra = {'city': 'Beijing', 'job': 'Engineer'}

person('Jack', 24, city=extra['city'], job=extra['job'])

person('Jack', 24, \*\*extra)

**命名关键字**

命名关键字的作用是只接收字典中被函数命名的关键字参数，命名关键字参数需要一个特殊分隔符‘ \* ‘，’ \* ‘后面的参数被视为命名关键字参数。命名关键字的传入必须要有参数名（和位置参数不同！），如果函数定义中已经有了一个可变参数，后面跟着的命名关键字参数就不再需要一个特殊分隔符\*了

def person(name, age, \*, city, job):

person('Jack', 24, city='Beijing', job='Engineer')

def person(name, age, \*args, city, job)

**要注意定义可变参数和关键字参数的语法**

\*args是可变参数，args接收的是一个tuple；

\*\*kw是关键字参数，kw接收的是一个dict。

以及调用函数时如何传入可变参数和关键字参数的语法：

可变参数既可以直接传入：func(1, 2, 3)，又可以先组装list或tuple，再通过\*args传入：func(\*(1, 2, 3))；

关键字参数既可以直接传入：func(a=1, b=2)，又可以先组装dict，再通过\*\*kw传入：func(\*\*{'a': 1, 'b': 2})。

**NOTE：**

我们在编写程序时，如果可以设计一个不变对象，那就尽量设计成不变对象。因为不变对象一旦创建，他的内部数据就不能修改，这样就减少了由于修改数据导致的错误。同时在多任务环境下同时读取对象不需要加锁，且不会出现问题。

**尾递归**

尾递归是指，在函数返回的时候，调用自身本身，并且，return语句不能包含表达式。这样，编译器或者解释器就可以把尾递归做优化，使递归本身无论调用多少次，都只占用一个栈帧，不会出现栈溢出的情况

**切片**

类似MATLAB，列表|元组|字符串 [startnum:endnum:等差值]

**Web开发（TDD）**

**-----------------------------------------------------------**

首先安装Django(python 的 web 框架)，用python的包管理工具下载

下载安装命令：

pip install Django

然后安装浏览器自动化工具，selenium

下载安装命令：

pip install --upgrade selenium

由于selenium3比较之前版本有了很大改动，其中之一就是把webdriver从浏览器中分离出来了，所以需要重新安装，以他来驱动浏览器。下载解压缩之后需要将他放到python的环境变量所在位置，至于为什么我好没有搞清楚，按理来说应该放在哪里都行，添加环境变量就好了，但是实际表明不行。

<https://segmentfault.com/a/1190000007249396>

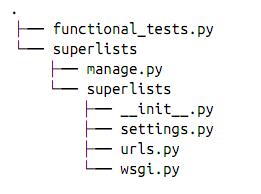
<https://selenium-python.readthedocs.io/api.html> （web driver API文档）

**开启一个服务器：**

首先用web框架（Django）新建一个项目，命令:

django-admin startproject name\_of\_your\_project

但是该命令要能运行需要添加django-admin.exe所在位置到系统环境变量

该命令执行完成之后会建立一个文件夹，文件夹的名字就是你所命名的工程的名字，里面的文件结构如下

这里的工程名字是superlists，该文件夹下面还有一个同名的文件夹，这是为了保存应用于整个项目的文件，其中settings.py作用就是存储网站的全局配置信息。manage.py的作用是运行开发服务器。

运行服务器的命令：

python manage.py runserver