## 1、简明Python教程

## 2、安装包

<https://www.python.org/downloads/windows/>

IDE：

IDLE

## 3、输出

print()

print(‘\*’, end=’’) #在同一行输出

注释

#

## 4、输入

input()

## 5、变量

四种基本类型：

字符串---表示一串字符，需要用’’或””引起来

整数

浮点数---就是小数

bool

变量的特性：

变量是能变的。所以在一次赋值操作之后，还可以继续给它赋予新的值，而且可以是不同类型的值

## 6、bool变量

True、False

关系运算符：

>、<、>=、<=、==、!=

逻辑运算符：

not、and、or

## 7、if的语法：

if 条件**:**

选择执行的语句

## 8、while的语法：

while 条件**:**

循环执行的语句

else:

条件语句为False时

## 9、引入模块的方法：

from 模块名 import 方法名

eg: from random import randint, choice

randint()用法：

randint(5, 10) – 产生一个5到10之间（包括5和10）的随机整数

choice()用法：

从一个list中随机挑选一个元素：

myStrList = ['left','mid','right']

print(choice(myStrList))

## 10、变量的命名规则：

第一个字符必须是字母或者下划线

剩下的部分可以下字母、数字、下划线

变量名称对大小写敏感

## 12、for循环的语法：

for iIndex in range(a, b)：

循环语句

else:

XXXX

从a循环到b-1。range产生一组整数序列

本质：

对一个序列中的元素进行递归

## 13、字符串

单引号（’）和双引号（”）：效果一样

三个引号（’’’或”””）：可以方便地使用单引号和双引号，并且可以直接换行

转义字符：\’、\”、[\\、\n](\\\\、\\n)

## 14、字符串格式化

两个字符串相加：+

字符变量与字符串相加：+

星号 (\*) 表示复制当前字符串，紧跟的数字为复制的次数

字符和数字不能直接相加。解决方案：

用str()把数字转换成字符串

用%对字符串进行格式化

**%d**—格式化整数

num = 18

print('My age is %d' % num)

print('My age is %d' % 18)

**%f**—格式化小数

**%.2f**—保留2位小数

**%s**—格式化字符串

strWeek = 'Friday'

print('Today is %s' % strWeek)

print('Today is %s' % 'Friday')

注意：有引号的表示一段字符，没有引号的就是一个变量，这个变量可能是字符，也可能是数字。但是一定要和%所表示的格式相一致。

## 16、字符串格式化2

格式化多个值的语法：

print("%s's score is %d" % ('Mike'**,** 86))

('Mike', 86)这种用()表示的一组数据在python中被称为元组（tuple）

## 17、类型转换

int(x) #把x转换成整数

float(x) #把x转换成浮点数

str(x) #把x转换成字符串

bool(x) #把x转换成bool值

## 18、bool类型转换：

在python中，以下数值会被认为是False：

为0的数字，包括0，0.0

空字符串，包括’’，””

表示空值的None

空集合，包括()，[]，{}

print(bool(-123)) #True print(bool("abc")) #True

print(bool("false")) #True print(bool("False")) #True

print(bool(0)) #False print(bool('')) #False

## 19、函数

自定义函数，关键字def，格式如下：

def sayHello()**:**

print('Hello')

def IsEqual(num1, num2):

if num1 > num2:

print('too big')

return False …

## 20、if、elif、else

示例：

def IsEqual(num1, num2):

if num1 > num2:

print('too big')

return False

elif num1 < num2:

print('too small')

return False

else**:**

print('bingo')

return True

## 25、list

打印：

print(list(range(1,7)))

遍历：

使用for ... in ...遍历：

myList = [23, 3.5, 'HaHa', False]

for item in myList:

print(item)

访问

下标从0开始，不能访问不存在的元素：

print('第1个元素是%d' % myList[0])

修改

修改某一个元素，直接给那个元素赋值就可以了：

myList[0] = 365

删除

使用del函数：

del(myList[0])

增加

使用append函数：

myList.append(1024)

## 27、list切片

list的索引操作

下标从0开始：

myStrList[0]

负数下标：

myStrList[-1]表示最后一个元素

myStrList[-3]表示倒数第3个元素

list的切片操作

切片操作符是在[]中提供一对可选数字，用**:**分割。

冒号前的数表示切片的开始位置

冒号后的数表示切片的结束位置

下标从0开始，同样也可以使用负数

注意：开始位置包含在切片中，结束位置不包含

如果不指定第一个数，切片从列表第一个元素开始

如果不指定第二个数，切片一直到最后一个元素结束

如果两个数都不指定，则返回整个列表的一个拷贝

## 28、字符串的分割

split()会把字符串按照其中的空格进行分割，分割后的每一段都是一个新的字符串，最终返回这些字符串组成一个list。

除了空格外，split()也会按照换行符\n，制表符\t等进行分割。默认按照空格进行分割。

注意：即使分割符后面没有其它字符，也会有一个字串被分割出来。

eg1：

str = 'Hello**,**world**,**haha**,**'

strList = str.split(',') # ['Hello', 'world', 'haha', '']

eg2：

str = 'Hello**,**world**,**haha'

strList = str.split(',') # ['Hello', 'world', 'haha']

## 29、连接list

join把list中的所有字符串连接成一个字符串。

注意：join是字符串的方法

strList = ['Hello', 'world', 'haha', '']

'#'.join(strList) #: Hello#world#haha#

注意：list只能是字符串的list时才可以

strList = ['Test',5,False,'String']

'$'.join(strList)

# TypeError: sequence item 1: expected str instance, int found

## 30、字符串的索引和切片

遍历：

使用for ... in ... 遍历：

strWord = 'Hello'

for szChar in strWord:

print(szChar)

索引访问：

使用[]加索引的方式，访问字符串的某个字符

strWord[0]

strWord[-1]

注意：字符串是常量，不能通过索引访问去更改其中的字符。

切片：

通过两个参数，截取一段子串

strWord[2:3] #：l

strWord[:-2] #：Hel

strWord[:] #：Hello

连接字符：

join也可以对字符串使用，作用就是用连接符把字符串中的每个字符重新连接成一个新字符串

'$'.join(strWord) #：H$e$l$l$o

## 35、break、continue

break

彻底跳出循环

continue

略过本次循环的余下内容，直接进入下一次循环

注意：无论是break还是continue，其改变的仅仅是当前所处的最内层循环的运行，如果外层还有循环，并不会因此跳出或略过

## 36、异常处理

语法：

try:

可能出现异常的语句

except:

当异常出现时如何处理

finally:

无论异常是否发生都必须执行的语句

示例：

try:

f = open('xx.txt')

print('open sucess')

#except:

# print('file not exists.')

except Exception as err: #把异常对象放到except组中，将异常错误信息准确输出

print(str(err))

finally:

#如果文件打开，才去关闭。locals()的作用是作用域内的局部变量集合

if 'f' in locals():

f.close()

## 37、字典

语法：

d = {key1:value1, key2:value2}

注意：

键必须是唯一的

键只能是简单对象，比如字符串、整数、浮点数、bool值

list不能作为键，但是可以作为值

字典中的键值对没有顺序，故无法用索引访问字典中的某一项，而是要用键来访问

遍历：

通过for ... in遍历：

dInfo = {'name':'张三', 'sex':'男'}

for key in dInfo:

print(dInfo[key])

修改：

如果要改变某一项的值，直接给这一项赋值：

dInfo['name'] = '李四'

增加：

增加一项字典项的方法是，给一个新键赋值：

dInfo['age'] = 18

删除：

删除一项字典项的方法是del：

del(dInfo['sex'])

新建空字典：

d = {}

## 38、模块

引入某个模块，就可以使用其中的函数和变量：

import random

eg：

random.randint(1, 10)

random.choice([1, 3, 5])

注意：函数前面要加上“random.”，这样python才知道是要调用random中的方法

想知道random中有哪些函数和变量，可以使用dir()方法：

dir(random)

如果只用到模块中某一个函数或变量，可以使用from ... import指明：

from math import pi

为了便于理解和避免冲突，可以给引入的方法换个名字：

from math import pi as math\_pi

print(math\_pi)

## 31、读文件

方法：

open方法，默认以只读模式打开文件，打开文件的时候指明编码

eg：

f = open('new 1.txt', 'r', encoding='utf-8') #打开文件，并没有得到其中的内容

data = f.read() #把文件内所有内容读进一个字符串中

print(data)

f.close() #关闭文件，释放资源

其它读取文件内容的方法：

readline() #读取一行内容

readlines() #把内容按行读取至一个list中

## 32、写文件

open函数的打开模式：

‘w’—写模式，原来文件中的内容会被新写入的内容覆盖掉，如果文件不存在，会自动创建

‘r’—读模式，默认模式，文件必须存在，否则引发异常

‘a’—追加模式，新写入的内容不会覆盖之前的内容，而是添加到文件中

eg：

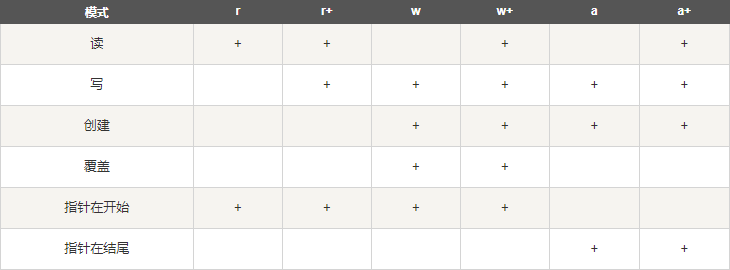
data = """It is a Test.

你猜一下"""

f = open('text.txt', 'w')

f.write(data)

f.close()

文件读写的模式 

## 42、函数的默认参数：

当你没有提供参数时，这个参数使用默认值；如果你提供了，就用你给的：

eg：

def Hello(name = 'world'):

print('Hello ' + name)

Hello()

Hello('Python')

## 44、查天气

使用urllib.request下载：

eg：

import urllib.request

web = urllib.request.urlopen('http://www.baidu.com')

content = web.read()

如果编码有问题：

str = '\xe5\xae\x9d\xe9\xb8\xa1\xe5\xb8\x82'

print(str.decode('utf-8'))

json格式解析：

使用json模块提供的loads方法，将json数据转换成字典

eg：

data = json.loads(content)

weather = data['weatherinfo']['weather']

获取地区编码：

说明：

天气网的城市代码按层级放在很多xml为后缀的文件中。而这些所谓的xml文件不符合xml的格式规范，导致在浏览器中无法显示。

抓取省份列表：

http://m.weather.com.cn/data3/city.xml

抓取城市列表（用省份代码）：

http://m.weather.com.cn/data3/city%s.xml

抓取地区列表（用城市代码）：

http://m.weather.com.cn/data3/city%s.xml

保存成文件。

## 48、面向对象

类：

eg：

class MyClass:

name = 'Sam'

def sayHi(self):

print(('Hello, %s') % self.name)

调用：

mc = MyClass()

mc.sayHi()

注意：类方法和我们之前定义的函数区别在于，第一个参数必须是self。而在调用类方法的时候，通过“对象.方法名()”格式进行调用，而不需要额外提供self这个参数的值。self在类方法中的值，就是你调用的这个对象本身。

继承：

eg：

class Vehicle:

def \_\_init\_\_(self, speed): #构造函数

self.speed = speed #属性

def drive(self, distance): #方法

print('需要行驶%.2f小时' % (float(distance)/self.speed))

class Bike(Vehicle):

pass # pass：表示一个空的代码块

class Car(Vehicle):

def \_\_init\_\_(self, speed, fuel):

Vehicle.\_\_init\_\_(self, speed) #调用父类的构造函数

self.fuel = fuel

def drive(self, distance):

Vehicle.drive(self, distance) #调用父类的方法

print('需要耗油%.2f升' % (float(distance) \* self.fuel))

调用：

b = Bike(15)

b.drive(100)

c = Car(80, 0.012)

c.drive(100)

说明：

1、\_\_init\_\_是python的内置方法，函数名前后是两个英文下划线

2、class定义后面的括号里表示这个类继承于哪个类

3、Car类中，重新定义\_\_init\_\_和drive函数，这样会覆盖掉它继承自Vehicle的同名函数。我们依然可以通过”Vehicle.函数名”来调用它的超类方法，以此来获得它作为Vehicle所具有的功能。注意，因为是通过类名调用方法，而不是像之前一样通过对象来调用，所以这里必须提供self的参数值。

## 51、and-or

eg1：

a = 'heaven'

b = 'hell'

c = True and a or b #'heaven'

d = False and a or b #'hell'

eg2：

a = ''

b = 'hell'

c = True and a or b #'hell'

技巧：and-or真正的技巧在于，确保a的值不会为假。最常用的方式是使a成为[a]、b成为[b]，然后使用返回值列表的第一个元素：

eg3：

a = ''

b = 'hell'

c = (True and [a] or [b])[0] #'' #类似C++中的三元表达式

## 52、元组（tuple）

元组也是一种序列，与list类似，只是元组中的元素在创建之后不能被修改。同样有索引、切片、遍历等操作。

用法：

在print语句中：

print('%s is %d years old' % ('Mike', 23))

作为函数返回值：

def get\_pos(n):

return (n/2, n\*2)

得到这个函数的返回值有两种形式：

一种是根据返回值元组中的元素的个数提供变量：

x, y = get\_pos(50)

一种是用一个变量记录返回的元组：

pos = get\_pos(50) #pos[0]、pos[1]

## 53、正则表达式（区分大小写）

在Python的string前面加’r’，是为了告诉编译器这个string是原生字符串，不要转义。

最简单的正则表达式，没有特殊的符号，只有基本的字母或数字。它满足的匹配规则就是完全匹配。

import re

text = r'Hi, I am Shirley Hilton. I am his wife.'

m = re.findall(r'hi', text) #两个’hi’，分别来自’Shirley’和’his’

if m:

print(m)

else:

print('not match')

如果只想找到’hi’这个单词，而不把包含它的单词也算在内，就可以使用’\bhi\b’。\b在正则表达式中表示单词的开头或结尾，空格、标点、换行都算是单词的分割。而’\b’自身又不会匹配任何字符，它代表的只是一个位置。所以单词前后的空格标点之类不会出现在结果里。

m = re.findall(r'\bhi\b', text) #not match

m = re.findall(r'\bhi', text) #一个’hi’，来自his

[]在正则表达式中表示满足括号中任一字符。比如’[hi]’，它就不是匹配’hi’了，而是匹配’h’或’i’。

m = re.findall(r'[Hh]i', text) #四个’hi’，分别来自’Hi’、’Shirley’、’Hilton’和’his’

**‘.’**在正则表达式中表示除换行符以外的任意字符。

m = re.findall(r'i.', text) # ['i,', 'ir', 'il', 'is', 'if']

m = re.findall(r'.', text) # ['H', 'i', ',', ' ', 'I', ' ', 'a', 'm', ' ', 'S', 'h', 'i', 'r', 'l', 'e', 'y', ' ', 'H', 'i', 'l', 't', 'o', 'n', '.', ' ', 'I', ' ', 'a', 'm', ' ', 'h', 'i', 's', ' ', 'w', 'i', 'f', 'e', '.']

‘\S’表示 不是空白符的任意字符，注意是大写字符S

m = re.findall(r'\S', text) # ['H', 'i', ',', 'I', 'a', 'm', 'S', 'h', 'i', 'r', 'l', 'e', 'y', 'H', 'i', 'l', 't', 'o', 'n', '.', 'I', 'a', 'm', 'h', 'i', 's', 'w', 'i', 'f', 'e', '.']

‘\*’表示数量，表示前面的字符可以重复任意多次（包括0次），只要满足这样的条件，都会被表达式匹配上。该匹配为贪婪匹配，会匹配尽可能长的结果。

m = re.findall(r'I.\*e', text) # ['I am Shirley Hilton. I am his wife']

‘?’的匹配方式为懒惰匹配，匹配到最短的就停止。需要用’.\*?’

m = re.findall(r'I.\*?e', text) # ['I am Shirle', 'I am his wife']

练习：

从下面的一段文本中，匹配出所有s开头，e结尾的单词

site sea sue sweet see case sse ssee loses

答案：

m=re.findall(r'\bs\S\*e\b', text) # ['site', 'sue', 'see', 'sse', 'ssee']

## 57、匹配手机号

匹配手机号，就是找出一串连续的数字。更进一步，是11位，以1开头的数字。

步骤：

[]表示其中任意一个字符，所以要匹配数字，可以用[0123456789]

由于它们是连续的字符，可以简化为[0-9]，类似的还有[a-zA-Z]

还有另外一种表示数字的方法：\d

要表示任意长度的数字，就可以用[0-9]\*或\d\*

注意：\*表示的任意长度包括0，也就是没有数字的空字符也会被匹配出来。一个与\*类似的符号+，表示的则是1个或者更长

所以要匹配所有的数字串，应当用[0-9]+或\d+

如果要限定长度，就用{}代替+，大括号里写上你想要的长度。比如11位的数字：\d{11}

再把第一位限定为1，就在前面加上1，后面去掉一位：1\d{10}

## 58、正则表达式的元字符

\w—匹配字母、数字、下划线或汉字

\s—匹配任意的空白符

^--匹配字符串的开始

$--匹配字符串的结束

注意：大写其实就是小写的反义

\W—匹配任意不是字母、数字、下划线或汉字的字符

\S—匹配任意不是空白符的字符

\D—匹配任意非数字的字符

\B—匹配任意不是单词开头或结束的位置

[a]的反义是[^a]，表示除a以外的任意字符

[^abcd]就是除abcd以外的任意字符

？--重复0次或一次

{n,}—重复n次或更多次

{n,m}—重复n到m次

eg:

^\w{4,12}$--表示一段4到12位的字符，包括字母、数字、下划线或汉字，可以用来作为用户注册时检测用户名的规则

\d{15,18}表示15到18位的数字，可以用来检测身份证号

^1\d\*x?表示以1开头的一串数字，数字结尾有字母x，也可以没有。有的话就带上x。

如果确实要匹配.或者\*字符本身，而不是要它们所代表的元字符，那么就需要用\.或\\*。

比如”\d+\.\d+”可以匹配出123.456这样的结果。

## 59、匹配多种格式的手机号

写一个正则表达式，能匹配出多种格式的电话号码，包括

(021)88776543

010-55667890

02584453362

0571 66345673

一个可以匹配出所有结果的表达式是：\(?0\d{2,3}[\) -]?\d{7,8}。理论上会匹配到错误的数据。因为()是成对出现的。

一个是合并多条件的匹配：\(0\d{2,3}\)\d{7,8} **|** 0\d{2,3}[ -]?\d{7,8}

注意，使用”|”时，要特别注意不现条件之前的顺序。匹配时，会按照从左往右的顺序，一旦匹配成功就停止验证后面的规则。

## 60、随机数

random.randint(a,b)

生成一个a到b之间的随机整数，包括a和b。且a、b都必须是整数，且b>=a。

random.random()

生成0到1之间的随机浮点数，包括0但不包括1，即[0.0,1.0)。

random.uniform(a,b)

生成a、b之间的随机浮点数，a、b无需是整数，也不用考虑大小。

random.choice(sqe)

从序列中随机选取一个元素。seq需要是一个序列，比如list、元组、字符串。

eg：

random.choice([1, 2, 3, 5, 8, 13]) #list

random.choice('hello') #字符串

random.choice(['hello', 'world']) #字符串组成的list

random.choice((1, 2, 3)) #元组

random.randrange(start, stop, step)

生成一个从start到stop（不包括stop），间隔为step的一个随机数。start、stop、step都要为整数，且start<stop。

eg：

random.randrange(1, 9, 2) #就是从[1, 3, 5, 7]中随机选取一个。

start和step都可以不提供参数，默认是从0开始，间隔为1。但如果需要指定step，则必须指定start。

eg：

random.randrange(4) #[0, 1, 2, 3]

random.randrange(1, 4) #[1, 2, 3]

random.randrange(start, stop, step)其实在效果上等同于

random.choice(range(start, stop, step))

random.sample(population, k)

从population序列中，随机获取k个元素，生成一个新序列。sample不改变原来序列。

random.shuffle(x)

把序列x中的元素顺序打乱。shuffle直接改变原有的序列。

random.seed(x)

指定随机数种子。在python中，默认用系统时间作为seed

## 61、time模块

方法：

time.time()

返回当前时间的时间戳（1970纪元后经过的浮点秒数）

time.sleep(secs)

让程序暂停secs秒。在抓取网页的时候，适当让程序sleep一下，可以减少短时间内的请求，提高请求的成功率。

## 62、序列化：pickle和cPickle（c语言编写的pickle，比pickle快1000倍）

序列化：将对象状态转换为可保持或传输的格式的过程

eg：

import pickle

data = ['SaveMe!', 123.456, True]

f = open('test.data', 'wb+')

pickle.dump(data, f)

f.close()

注意：

若f = open('test.data', 'w')，则会提示错误：

TypeError: write() argument must be str, not bytes

产生问题的原因是因为pickle存储方式默认是二进制方式

反序列化：从序列化的格式中解析对象状态的过程

eg：

f = open('test.data', 'rb+')

data = pickle.load(f)

f.close()

保存多个对象的两种方法：

一是把这些对象全部放在一个序列中，再对这个序列进行存储

a = 123

b = 'Hello'

data = (a, b)

pickle.dump(data, f)

一是依次保存和提取

pickle.dump(a, f)

pickle.dump(b, f)

## 66、列表解析（列表综合）

通过一个已有的列表生成一个新的列表。

eg：

假设有一个由数字组成的 list，现在需要把其中的偶数项取出来，组成一个新的 list：

list\_1 = [1, 2, 3, 5, 8, 13, 22]

list\_2 = [i for i in list\_1 if i % 2 == 0] # [2, 8, 22]

进一步的，在构建新列表时，还可以对于取出的元素做操作。比如，对于原列表中的偶数项，取出后要除以2，则可以通过 [i / 2 for i in list\_1 if i % 2 == 0] 来实现。输出为 [1, 4, 11]

## 67、函数的参数传递

第1类：def func(arg1, arg2):

调用：

根据调用时提供参数的位置进行匹配，要求实参与行参的数量相等，默认按位置匹配参数。调用时，少参数或者多参数都会引起错误

func(3, 7)

在调用时，也可以根据形参的名称指定实参。但同样，必须提供所有的参数：

func(arg2=3, arg1=7)

第2类：def func(arg1=1, arg2=2, arg3=3):

调用：

提供的参数会按顺序先匹配前面位置的参数，后面未匹配到的参数使用默认值。也可以指定其中的部分参数，或者混合起来用

注意：没有指定参数名的参数必须在所有指定参数名的参数前面，且参数不能重复

第3类：def func(\*args) #可以接受任意数量的参数

调用：

在变量前加上星号前缀（\*），调用时的参数会存储在一个 tuple（元组）对象中，赋值给形参。在函数内部，需要对参数进行处理时，只要对这个 tuple 类型的形参（这里是 args）进行操作就可以了。因此，函数在定义时并不需要指明参数个数，就可以处理任意参数个数的情况。由于tuple 是有序的，所以 args 中元素的顺序受到赋值时的影响。

def calcSum(\*args):

sum = 0

for i in args:

sum += i

print(sum)

calcSum(1,2)

calcSum(1,2,3)

第4类：func(\*\*kargs)

说明：上次说的 func(\*args) 方式是把参数作为 tuple 传入函数内部。而 func(\*\*kargs) 则是把参数以键值对字典的形式传入。

eg：

def printAll(\*\*kargs):

for k in kargs:

print k, ':', kargs[k]

printAll(a=1, b=2, c=3)

调用：

字典是无序的，所以在输出的时候，并不一定按照提供参数的顺序。同样在调用时，参数的顺序无所谓，只要对应合适的形参名就可以了。于是，采用这种参数传递的方法，可以不受参数数量、位置的限制。

混合用法：

定义：

def func(x, y=5, \*a, \*\*b):

print x, y, a, b

调用：

func(1) 1 5 () {}

func(1,2) 1 2 () {}

func(1,2,3) 1 2 (3,) {}

func(1,2,3,4) 1 2 (3, 4) {}

func(x=1) 1 5 () {}

func(x=1,y=1) 1 1 () {}

func(x=1,y=1,a=1) 1 1 () {'a': 1}

func(x=1,y=1,a=1,b=1) 1 1 () {'a': 1, 'b': 1}

func(1,y=1) 1 1 () {}

func(1,2,3,4,a=1) 1 2 (3, 4) {'a': 1}

func(1,2,3,4,k=1,t=2,o=3) 1 2 (3, 4) {'k': 1, 't': 2, 'o': 3}

注意：

在混合使用时，首先要注意函数的写法，必须遵守：

带有默认值的形参(arg=)须在无默认值的形参(arg)之后；

元组参数(\*args)须在带有默认值的形参(arg=)之后；

字典参数(\*\*kargs)须在元组参数(\*args)之后。

可以省略某种类型的参数，但仍需保证此顺序规则。

调用时也需要遵守：

指定参数名称的参数要在无指定参数名称的参数之后；

不可以重复传递，即按顺序提供某参数之后，又指定名称传递。

而在函数被调用时，参数的传递过程为：

1.按顺序把无指定参数的实参赋值给形参；

2.把指定参数名称(arg=v)的实参赋值给对应的形参；

3.将多余的无指定参数的实参打包成一个 tuple 传递给元组参数(\*args)；

4.将多余的指定参数名的实参打包成一个 dict 传递给字典参数(\*\*kargs)。

## 70、lambda表达式

简单实例：

实现三数相加：

sum = lambda a,b,c:a+b+c

print(sum(1,2,3))

说明：定义 lambda 表达式时，参数列表周围没有括号，返回值前没有 return 关键字，也没有函数名称

复杂实例：

把 lambda 表达式用在 def 函数定义中：

def fn(x):

return lambda y:x+y

a = fn(2)

print(a(3))

解释：

这里，fn 函数的返回值是一个 lambda 表达式，也就等于是一个函数对象。当以参数2来调用 fn 时，得到的结果就是：

lambda y: 2 + y

a = fn(2) 就相当于：

a = lambda y: 2 + y

所以 a(3) 的结果就是5。

## 71、变量的作用域

作用域是从变量被定义的位置开始。

在函数内部去改变一些变量的值，并且这些变量在函数外部同样被使用到。怎么办？

方法1：

用 return 把改变后的变量值作为函数返回值传递出来，赋值给对应的变量。

eg：

def func(x):

x = 2

return x

x = 50

x = func(x) #x=2

方法2：

使用“全局变量”。在 Python 的函数定义中，可以给变量名前加上 global 关键字，这样其作用域就不再局限在函数块中，而是全局的作用域

当内部作用域想修改外部作用域的变量时，就要用到global和nonlocal关键字了

eg：修改全局变量的值

x = 50

def func():

global x #需要使用global关键字声明

x = 2

func() #x=2

eg：修改嵌套作用域中的变量，则需要用nonlocal关键字

def outer():

num = 10

def inner():

nonlocal num

num = 100

inner()

print(num) #100

outer()

## 72、map函数

问题1：假设有一个数列，如何把其中每一个元素都翻倍？

三种方法：

lst\_1 = [1,2,3,4,5,6]

#lst\_2 = [i\*2 for i in lst\_1]

#def double\_func(x):

# return x\*2

#lst\_2 = map(double\_func, lst\_1)

lst\_2 = map(lambda x:x\*2, lst\_1)

print(list(lst\_2))

map 是 Python 自带的内置函数，它的作用是把一个函数应用在一个（或多个）序列上，把列表中的每一项作为函数输入进行计算，再把计算的结果以列表的形式返回。

map 的第一个参数是一个函数，之后的参数是序列，可以是 list、tuple

问题2：假设有两个数列，如何求和？

map 中的函数可以对多个序列进行操作。

方法：

lst\_1 = [1,2,3,4,5,6]

lst\_2 = [1,3,5,7,9,11]

lst\_3 = map(lambda x, y: x + y, lst\_1, lst\_2)

print(list(lst\_3)) #[2, 5, 8, 11, 14, 17]

lst\_2 = [1,3,5,7]

print(list(lst\_3)) #[2, 5, 8, 11]

注意：

map 中的函数会从对应的列表中依次取出元素，作为参数使用，同样将结果以列表的形式返回。所以要注意的是，函数的参数个数要与 map 中提供的序列组数相同，即函数有几个参数，就得有几组数据

## 73、reduce函数

map 可以看作是把一个序列根据某种规则，映射到另一个序列。reduce 做的事情就是把一个序列根据某种规则，归纳为一个输出。

eg：

求1累加到100的和：

from functools import reduce

def add(x,y):

return x+y

print(reduce(add, range(101))) #5050

reduce((lambda x, y: x + y), xrange(1, 101), 3) #5053

reduce(function, iterable[, initializer])

第一个参数是作用在序列上的方法，

第二个参数是被作用的序列，这与 map 一致。

第三个参数是可选参数，是初始值。

function 需要是一个接收2个参数，并有返回值的函数。它会从序列 iterable 里从左到右依次取出元素，进行计算。每次计算的结果，会作为下次计算的第一个参数。

提供初始值 initializer 时，它会作为第一次计算的第一个参数。否则，就先计算序列中的前两个值

所以，在对于一个序列进行某种统计操作的时候，比如求和，或者诸如统计序列中元素的出现个数等，可以选择使用 reduce 来实现。

eg：统计序列中元素的出现个数

lst = [1,1,2,3,2,3,3,5,6,7,7,6,5,5,5]

def stat(dic, k):

if dic.get(k) is None:

dic[k] = 1

else:

dic[k] += 1

return dic #注意返回值

print(reduce(stat, lst, {}))

## 74、多线程

\_thread，重命名已废弃的thread（）

start\_new\_thread(function, args[, kwargs])

function 是开发者定义的线程函数，

args 是传递给线程函数的参数，必须是tuple类型，

kwargs 是可选参数。

调用 start\_new\_thread 之后，会创建一个新的线程，来执行 function 函数。而代码原本的主线程将继续往下执行，不再等待 function 的返回。通常情况，线程在 function 执行完毕后结束。

eg：

import urllib.request, \_thread, time

def getContent(i):

d = 1764798 + i

url = ('https://api.douban.com/v2/movie/subject/%d' % d)

content = urllib.request.urlopen(url).read()

print('%d\n%s\n\n\n' % (d, content))

data.append(content)

data = []

for i in range(3):

\_thread.start\_new\_thread(getContent, (i,)) #元组中只包含一个元素时，需要在元素后面添加逗号，否则括号会被当作运算符使用

time.sleep(70000)

threading推荐使用

## 75、Set（集合）

集合（set）是一个无序不重复元素的序列。基本功能是进行成员关系测试和删除重复元素。

可以使用大括号 { } 或者 set() 函数创建集合，注意：创建一个空集合必须用 set() 而不是 { }，因为 { } 是用来创建一个空字典。

创建格式：

parame = {value01,value02,...}

或者

set(value)

eg：

#!/usr/bin/python3

student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}

print(student) # 输出集合，重复的元素被自动去掉

# {'Mary', 'Jim', 'Rose', 'Jack', 'Tom'}

# 成员测试

if('Rose' in student) :

print('Rose 在集合中') Rose 在集合中

else :

print('Rose 不在集合中')

# set可以进行集合运算

a = set('abracadabra')

b = set('alacazam')

print(a) #{'b', 'a', 'c', 'r', 'd'}

print(a - b) # a和b的差集 #{'b', 'd', 'r'}

print(a | b) # a和b的并集 #{'l', 'r', 'a', 'c', 'z', 'm', 'b', 'd'}

print(a & b) # a和b的交集 #{'a', 'c'}

print(a ^ b) # a和b中不同时存在的元素 #{'l', 'r', 'z', 'm', 'b', 'd'}

## 76、小知识

1、/除：21 / 10 输出结果 2.1

//取整除-返回商的整数部分。9//2 输出结果 4 , 9.0//2.0 输出结果 4.0

2、print ("1 - c 的值为：", 3)

3、[id()](http://www.runoob.com/python/python-func-id.html" \t "_blank) 函数用于获取对象内存地址

4、\_\_name\_\_属性：

一个模块被另一个程序第一次引入时，其主程序将运行。如果我们想在模块被引入时，模块中的某一程序块不执行，我们可以用\_\_name\_\_属性来使该程序块仅在该模块自身运行时执行。

if \_\_name\_\_ == '\_\_main\_\_':

print('程序自身在运行')

else:

print('我来自另一模块')

5、预定义的清理行为

关键词 with 语句就可以保证诸如文件之类的对象在使用完之后一定会正确的执行他的清理方法。就算在处理过程中出问题了，文件总是会关闭。

with open(r'C:\Users\Suo\Desktop\new 2.txt') as f:

for line in f:

print(line)

6、类的私有属性与私有方法：

私有属性：

\_\_private\_attrs：两个下划线开头，声明该属性为私有，不能在类地外部被使用或直接访问。在类内部的方法中使用时 self.\_\_private\_attrs。

私有方法：

\_\_private\_method：两个下划线开头，声明该方法为私有方法，只能在类的内部调用 ，不能在类地外部调用。self.\_\_private\_methods。

7、'tea for too'.replace('too', 'two') # 'tea for two'

## 77、迭代器与生成器

迭代器：

迭代器对象从集合的第一个元素开始访问，直到所有的元素被访问完结束迭代器只能往前不能后退。字符串、列表、元组对象都可用于创建迭代器。

eg：

两个基本方法：iter()和next()：

li = [1, 2, 3, 4]

it = iter(li)

print(next(it)) #1

print(next(it)) #2

迭代器对象可以使用常for语句进行遍历：

for x in it:

print(x)

也可以使用next()：

while True:

try:

print(next(it))

except StopIteration:

sys.exit()

生成器：

使用了yield的函数被称为生成器。生成器是一个返回迭代器的函数，只能用于迭代操作。

在调用生成运行的过程中，每次遇到yield时，函数会暂停并保存当前所有的运行信息，返回yield的值。并在下一次执行next()方法时从当前位置继续运行。

eg：使用yield实现斐波那契数列：

def fibonacci(n):

a, b, counter = 0,1,0

while True:

if counter>n:

print(counter)

return

yield a

a, b = b, a+b

counter += 1

print('%d %d %d' % (a,b,counter))

f = fibonacci(10)

while True:

try:

print(next(f))

except StopIteration:

sys.exit()

## 78、OS文件/目录方法

os.listdir(strDir)

返回指定的文件夹包含的文件或文件夹的名字的列表。这个列表以字母顺序。 它不包括 '.' 和'..' 即使它在文件夹中

os.path.join(strDirPath, strPath)

获取组合路径

os.path.isdir(path)

是否是目录

os.path.splitext(r'D:\img\1.png')

('D:\\img\\1', '.png')

eg：搜索指定目录的所有文件

方法1：深度优先搜索（用栈）：

import os,queue

data = []

def DepthFirstSearch(strDirPath):

fileList = os.listdir(strDirPath)

for strPath in fileList:

pathTmp = os.path.join(strDirPath, strPath) #获取组合路径

if os.path.isdir(pathTmp):

DepthFirstSearch(pathTmp)

else:

data.append(pathTmp)

方法2：广度优先搜索（用队列）

data = []

subDirList = queue.Queue()

def PutList2Queue(strDir, listTmp):

for tmp in listTmp:

strTmp = os.path.join(strDir, tmp)

subDirList.put(strTmp)

def BreadthFirstSearch(strDirPath):

PutList2Queue(strDirPath, os.listdir(strDirPath))

while not subDirList.empty():

pathTmp = subDirList.get()

if os.path.isdir(pathTmp):

PutList2Queue(pathTmp, os.listdir(pathTmp))

else:

data.append(pathTmp)



## 79、标准库概览

操作系统接口

import os

print(os.getcwd()) # 返回当前的工作目录

os.chdir(r'C:\Users\Suo\Desktop') # 修改当前的工作目录

os.system('mkdir today') # 执行系统命令 mkdir

日常的文件和目录管理任务

import shutil

shutil.copyfile('city.py', 'citycode.txt')

shutil.move('test.data', r'C:\Users\Suo\Desktop\a.data')

文件通配符：glob模块提供了一个函数用于从目录通配符搜索中生成文件列表

import glob

print(glob.glob('\*.txt')) # ['citycode.txt', 'guessgame.txt', 'text.txt']

命令行参数：

import sys

print(sys.argv)

日期和时间

from datetime import date

print(date.today())

数据压缩

以下模块直接支持通用的数据打包和压缩格式：zlib，gzip，bz2，zipfile，以及 tarfile：

import zlib

s = b'witch which has which witches wrist watch' #len(s)=42

t = zlib.compress(s) #len(t) = 37

t = zlib.decompress(t)

print(zlib.crc32(s))

## 80、安装Python插件

在命令行(以管理员身份运行)中输入：python -m pip install 模块名

eg：

读excel模块：

python -m pip install xlrd

下载地址：https://pypi.python.org/pypi/xlrd

写excel模块：

python -m pip install xlwt

下载地址：https://pypi.python.org/pypi/xlwt

修改excel模块：

python -m pip install xlutils

下载地址：https://pypi.python.org/pypi/xlutils