# Python数据科学基础入门

# Python基础第一部分

# 1.0了解Python

## 1.1课程介绍 & 为什么Python如此受欢迎？

（1）课程介绍

Tips:Python基础+数据科学基础(Numpy\Pandas\Matplotlib\Scipy)

（2） Python为什么如此受欢迎？

Python语言介绍：

* 面向对象，解释型计算机程序设计语言
* 1989年Guido van Rossum发明，1991年公开版本发行
* Python3.0在2008年发布，为了不带来过多的累赘，没有考虑向下兼容。虽然Python3已经发布多年，目前版本3.6.3，但Python2使用热度不减。
* 对于一般学习来讲，常见的Python2和Python3的区别之一是**输出语句**。
* 解析器：语言只是符号、语法、语义定义及使用规则的集合。使用这些规则编写的程序（就是Python程序）并不能被计算机直接执行。解析执行Python源程序的程序叫做Python解析器（Interpreter），而由解析器解析执行的过程就是Python的实现。Python解析器有几种：

--CPython 官方提供的解析器就是C语言实现的，所以称之为CPython，也是最常用的Python实现，课程中使用的就是CPython作为解析器。

--JPython 使用Java语言实现的Python解析器，将Python代码编程成Java字节码执行。

--IronPython 是运行在微软Net平台上的Python解析器，直接把Python代码编译成Net字节码。

--PyPy 使用Python语言实现的Python解析器。

* 广泛应用于科学计算，众多开源科学计算软件包提供Python接口

(e.g.:计算机视觉库OpenCV，三维可视化库VTK，医学图像处理库ITK)

* 应用：科学计算、自然语言处理、图形图像处理、脚本开发、Web应用
* 当今最受欢迎的语言之一，2004年后使用率线性增长 TIOBE排名
* 美国大学计算机编程入门课程最流行的语言(前10名计算机系中有8名使用Python）
* **设计哲学**：优雅，明确，简单，可读性强
* 优点：功能强大，开发效率高，应用广泛，易上手，语法简洁
* 用途：

网页开发

可视化（GUI）界面开发

网络

系统编程

数据分析

机器学习

网络爬虫

科学计算

* 哪些著名网站是用Python实现的？

成千上万，例如：

雅虎地图

谷歌中的很多组成部分

Youtube

豆瓣网

代码举例：

===C++===

#include<iostream>

using namespace std;

int main()

{

cout<<"Hello World";

return 0;

}

===Java===

public class Main{

public static void main(String[] args)

{

System.out.println("Hello World");

}

}

===Python===

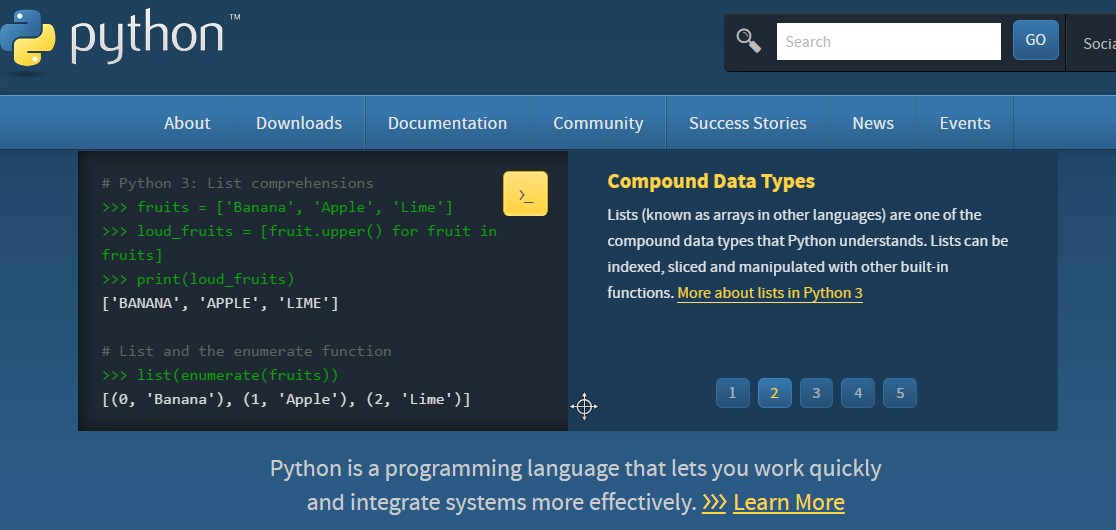
#!/usr/bin/python

print "Hello World";

## 1.2安装Python和配置环境

### 1.2.1基础版本Python安装

* 1. 下载和安装Python



* 1. 1.2 Python包管理

<https://pypi.org/project/matplotlib-helpers/#files>

PYPI：https://pypi.python.org/pypi

PyPI(Python Package Index)是python官方的第三方库的仓库，所有人都可以下载第三方库或上传自己开发的库到PyPI。PyPI推荐使用pip包管理器来下载第三方库。

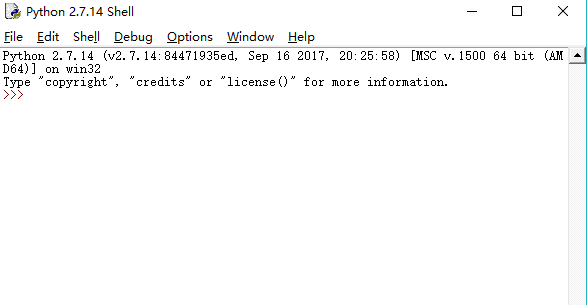
PIP可正常工作在Windows、Mac OS、Unix/Linux等操作系统上，但是需要至少2.6+和3.2+的CPython或PyPy的支持。python 2.7.9 和3.4以后的版本已经内置累pip程序，所以不需要安装。

pip install 包名 安装包

pip uninstall 包名 卸载包

pip list 查看所有的安装的python的第三方包

### 1.2.2 IDLE介绍



演示Python2和Python3的输出语句print差别

## 1.3 Anaconda数据科学环境安装

Anaconda（水蟒）：是一个科学计算软件发行版，集成了大量常用扩展包的环境，包含了 conda、Python 等 180 多个科学计算包及其依赖项，并且支持所有操作系统平台。下载地址：https://www.continuum.io/downloads

安装包：pip install xxx,conda install xxx

卸载包：pip uninstall xxx,conda uninstall xxx

升级包：pip install upgrade xxx,conda update xxx

IDE

**Jupyter Notebook：**

命令：jupyter notebook

1.Anaconda自带，无需单独安装

2.实时查看运行过程

3.基本的web编辑器（本地）

4..ipynb 文件分享

5.可交互式

6.记录历史运行结果

修改jupyter显示的文件路径：

通过jupyter notebook --generate-config命令创建配置文件，之后在进入用户文件夹下面查看.jupyter隐藏文件夹，修改其中文件jupyter\_notebook\_config.py的202行为计算机本地存在的路径。

IPython：

命令：ipython

1.Anaconda自带，无需单独安装

2.Python的交互式命令行 Shell

3.可交互式

4.记录历史运行结果

5.及时验证想法

Spyder：

命令：spyder

1.Anaconda自带，无需单独安装

2.完全免费，适合熟悉Matlab的用户

3.功能强大，使用简单的图形界面开发环境

## 1.4conda管理数据科学环境

conda和pip类似均为安装、卸载或管理Python第三方包

Conda install 包名 pip install 包名

Conda uninstall 包名 pip uninstall 包名

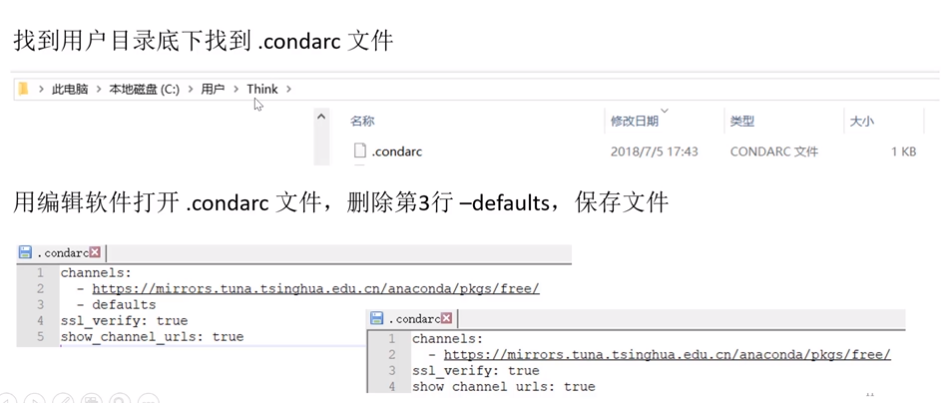
Conda install -U 包名 pip install -U 包名

## 1.4 Anaconda设置为国内下载镜像

conda config --add channels <https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/>

conda config --set show\_channel\_urls yes





## 1.4 安装pycharm的IDE工具

需要自行安装：https://www.jetbrains.com/pycharm/download

PyCharm，JetBrains的精品，全平台支持，不多解释了。

Pycharm新建模板默认添加作者时间等信息

1.打开pycharm，选择File-Settings

2.选择Editor--Color&Style--File and Templates--Python-Script

3.填加

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

# @Time : ${DATE} ${TIME}

# @Author : Z

# @Email : S

# @File : ${NAME}.py

注意：创建python文件而不是文件.py，否则不能识别

## 1.3 Python版本差异

python版本

Python 2 or Python 3

Python 2.x 是早期版本，Python 3.x是当前版本

Python 2.7 (2.x的最终版)于2010年发布后很少有大的更新

Python 2.x 比 Python3.x 拥有更多的工具库

大多数Linux系统默认安装的仍是 Python 2.x

版本选择取决于要解决的问题

建议选择 Python 2.x 的情况：

部署环境不可控，Python版本不能自行选择

某些工具库还没有提供支持 Python 3.x。

如果选择使用 Python 3.x，需要确定要用的工具库支持新版本。

课程中会综合python2和3版本使用，大家要熟知两者的部分区别

注：python的结构：module模块--->package模块--->project项目(eclipse)

python file ----- package -------project(Pycharm)

（1）脚本语言的第一行#!/usr/bin/python，只对Linux/Unix用户适用，用来指定本脚本用什么interperter来执行。

有这句的，加上执行权限后，可以直接用./执行，不然会出错，因为找不到python解释器。

#!/usr/bin/python是告诉操作系统执行这个脚本的时候，调用/usr/bin下的python解释器；

#!/usr/bin/env python这种用法是为了防止操作系统用户没有将python装在默认的/usr/bin路径里。当系统看到这一行的时候，首先会到env设置里查找python的安装路径，再调用对应路径下的解释器程序完成操作。

#!/usr/bin/python相当于写死了python路径;

#!/usr/bin/env python会去环境设置寻找python目录，可以增强代码的可移植性，推荐这种写法

（2）# -\*- coding:utf-8 -\*-

#coding=utf-8

如果要在python2的py文件里面写中文，则必须要添加一行声明文件编码的注释，否则python2会默认使用ASCII编码。

（3）#查看编码方式

import sys

print(sys.getdefaultencoding())

Python3字符编码

#编码方式的解读

Python3 最重要的一项改进之一就是解决了 Python2 中字符串与字符编码遗留下来的这个大坑。

#Python2 字符串设计上的一些缺陷：

1.使用 ASCII 码作为默认编码方式，对中文处理很不友好。

2.把字符串的牵强地分为 unicode 和 str 两种类型，误导开发者

#Python3 改进

当然这并不算 Bug，只要处理的时候多留心也可以避免这些坑。但在 Python3 两个问题都很好的解决了。

首先，Python3 把系统默认编码设置为 UTF-8

其次，文本字符和二进制数据区分得更清晰，分别用 str 和 bytes 表示。文本字符全部用 str 类型表示，str 能表示 Unicode 字符集中所有字符，而二进制字节数据用一种全新的数据类型，用 bytes 来表示

然后，Python3 中，在字符引号前加‘b'，明确表示这是一个 bytes 类型的对象，实际上它就是一组二进制字节序列组成的数据

最后，encode 负责字符（unicode）到字节（byte）的编码转换。默认使用 UTF-8 编码准换。

注意区分：Unicode和UTF-8的区别（utf-8是为传送unicode字符的一种再编码的方式）

#（python3下面可以演示）

>>> s = "Python之禅"

>>> s.encode()

b'Python\xe4\xb9\x8b\xe7\xa6\x85'

>>> s.encode("gbk")

b'Python\xd6\xae\xec\xf8'

decode 负责字节（byte）到字符（unicode）的解码转换，通用使用 UTF-8 编码格式进行转换。

>>> b'Python\xe4\xb9\x8b\xe7\xa6\x85'.decode()

'Python之禅'

>>> b'Python\xd6\xae\xec\xf8'.decode("gbk")

'Python之禅'

# 2.0Python数据类型

## 2.1 Package以及数据类型1

1. 自带package和外部package

1.1 自带package举例： os; os.getcwd();os.chdir(path)

2. 外部package以及管理系统介绍: easy\_install, pip (comes with Python 3.6)

3. 环境变量中配置easy\_install, pip

4. 使用easy\_install, pip安装package举例

>>> import requests

>>> r = requests.get('https://www.baidu.com/')

>>> r.text

>>> r.url

>>> r.encoding

## 2.2import和from...import区别

语法层面：

（1）import 模块名

使用方式：模块名.函数名(参数)

模块名.变量名

（2）from 模块名 import 函数名或变量名（多个函数名用逗号分割）

#-\*-coding:utf8-\*-

#导入模块的案例

#import和from....import区别

Demo1:

import math

math.sin(0.5)

from math import sin #只导入模型中的指定对象

print sin(3)

from math import sin as f

print f(3)

Demo2:

import numpy as np

a=np.array((1,2,3,4))

print a

## 2.3 数据类型2: Numeric & String

1. Python数据类型

1.1 总体：numerics, sequences, mappings, classes, instances, and exceptions

1.2 Numeric Types: int (包含boolean), float, complex

1.3 int: unlimited length; float: 实现用double in C, 可查看 sys.float\_info; complex: real(实部) & imaginary(虚部）,用z.real 和 z.imag来取两部分

1.4 具体运算以及法则参见：https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex

1.5 例子

id() #id函数可以获得对象的id标识

type() #type获取当前数据的数据类型

a=122

b=122

id(122)

id(a)

id(b)

大家认为id得到的值一致吗？

2.案例

import sys

a = 5

b = 3

e = complex(a, b)

f = complex(float(a), float(b))

print ("a is type" , type(a))

print ("b is type" , type(b))

print ("e is type" , type(e))

print ("f is type" , type(f))

print ("e's real part is: " , e.real)

print ("e's imaginary part is: " , e.imag)

+ - \*乘法 /除法 //整除 \*\*幂 %取模

print(a + b)

print(d / c)

print (b / a)

print (b // a)

print (e)

print (e + f)

+= -= \*= /= %= \*\*= //=

a+=b

a-=b

a\*=b

a//=b

a/=b

a%=b

print a

& 按位与 | 按位或 ^按位异或 <<左移 >> 右移

a,b=5,3

#0101 0011

print a&b

print a|b

print a^b

print a<<2 #左移乘2，这里乘4

print a>>2 #右移除2，这里除以4

print (sys.float\_info)

## 2.4 数据类型3：字符串、变量

1. 字符串：

一串字符

显示或者打印出来文字信息

导出

编码：# -\*- coding: utf-8 -\*-

单引号，双引号，三引号

不可变（immutable)

Format字符串

【Demo】 age = 3

name = "Tom"

print("{0} was {1} years old".format(name, age))

print("%s was %d years old"%(name, age))

联合：+: print(name + " was " + str(age) + " years old")

换行符: print("What's your name? \nTom")

2. 字面常量（literal constant):

可以直接以字面的意义使用它们：

如：6，2.24，3.45e-3, "This is a string"

常量：不会被改变

3. 变量：

储存信息

属于identifier

identifier命名规则：

第一个字符必须是字母或者下划线

其余字符可以是字母，数字，或者下划线

区分大小写

如：合法：i, name\_3\_4, big\_bang

不合法：2people, this is tom, my-name, >123b\_c2

4. 注释： #

5. 缩进(Indentation)

#参考代码：

print("hello world") #单引号

print('hello world') #双引号

print('''this is 1 line #三引号

this is 2 line

this is 3 line!''')

## 2.5输入/输出

语法1：<变量>=input([<提示>])

提示 的信息是在屏幕上，提示用户输入什么数据，可以没有

python3中从键盘输入的均为string字符串类型，如果需要整形或浮点型需要强转

代码1：

a=int(input("请输入整数~"))

b=float(input("请输入浮点数"))

print(a,b)

语法2：print(<输出项列表>，sep=分隔符，end=结束符)

代码2:

print("Sun","Mon","Tue","Fri",sep=',',end=';')

注：（1）\*python2中有raw\_input和input

raw\_input获取到的输入永远都是str类型的

input获取到的输入会自动判断其类型

\*\*\*python3将raw\_input和input进行了整合，只有input

（2）Python2.x中，input()函数让我们明确我们输入的是数字格式还是字符格式，就是我们 自己要知道我们想要的是什么，数字格式直接输入，字符格式必须加上单引号或者双引号，以确定我们输入的是字符串。

>>> a = input("Please input your name: ")

Please input your name: apple

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

File "<string>", line 1, in <module>

NameError: name 'apple' is not defined

从结果可以看出，提示我们输入的名字没有定义，报错了，说明不能以字符形式直接没有引号的情况下进行输入；

我们验证当我们以引号的方式输入字符串的时候，这个时候没有提示我们错误，由此可以看出，是没有问题的

>>> a = input("Please input your name: ")

Please input your name: "apple"

>>> a

'apple'

## 2.6随机数生成的几种方式-掌握Random方式

#-\*-coding:utf8-\*-

import numpy as np

(1)使用random

import random

x=random.random() #获取[0,1)内的随机小数

print(x)

#了解shuffle  
l1 = [1, 2, 3, 4] #有序  
random.shuffle(l1)  
print l1

n=random.randint(1,100) #获取[1,100]区间上的随机整数

print "n:",n

print (random.randrange(90,100))

(2)使用numpy

#1.给定上下限范围内选取整数

x=np.random.randint(0,10,size=(1,10)) #size指定1行10列,size关键字可以省

y=np.random.randint(0,10,7) #0-10之间的7个数

#2.产生[a,b)中均匀分布的样本值

z=np.random.uniform(-1, 5, size = (3, 4)) # 'size='可省略，（0,1）之间的均匀分布

#3.rand产生均匀分布的样本值，3行4列

t = np.random.rand(3, 4) #产生0-1之间的均匀分布的样本值

#4.产生二项分布的样本值

n, p = 10, .5 # number of trials, probability of each trial试验次数，每次试验的概率

s = np.random.binomial(n, p, 1000) #产生二项分布的样本值

#5.产生高斯分布的样本值

k=np.random.normal(0,0.1,10) #参数顺序：1.均值 2.标准差 3.生成多少样本

#6.产生卡方分布的样本值

s = np.random.chisquare(2,size=(2,3)) #2为自由度

# 3.0Python数据结构

## 3.1List基础数据结构

1. print中的编码：

编码：# -\*- coding: utf-8 -\*-

2. print中的换行

print("What's your name? \nTom")

3. List

创建

访问

更新

删除

脚本操作符

函数方法

Code:

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

#创建一个列表

number\_list = [1, 3, 5, 7, 9]

string\_list = ["abc", "bbc", "python"]

mixed\_list = ['python', 'java', 3, 12]

#访问列表中的值

second\_num = number\_list[1]

third\_string = string\_list[2]

fourth\_mix = mixed\_list[3]

print("second\_num: {0} third\_string: {1} fourth\_mix: {2}".format(second\_num, third\_string, fourth\_mix))

#更新列表

print("number\_list before: " + str(number\_list))

number\_list[1] = 30

print("number\_list after: " + str(number\_list))

#删除列表元素

print("mixed\_list before delete: " + str(mixed\_list))

del mixed\_list[2]

print("mixed\_list after delete: " + str(mixed\_list))

#Python脚本语言

print(len([1,2,3])) #长度

print([1,2,3] + [4,5,6]) #组合

print(['Hello'] \* 4) #重复

print(3 in [1,2,3]) #某元素是否在列表中

#列表的截取

abcd\_list =['a', 'b', 'c', 'd']

print(abcd\_list[1])

print(abcd\_list[-2])

print(abcd\_list[1:])

#补充

#6.7列表创建

print [1,'a','four'] #创建列表

print [] #创建空列表

print list((1,2,3,3,4)) #将元组转化为列表

print list(range(1,10,2)) #将range对象转化为列表

print list('hello') #将字符串转化为列表

print list({3,5,7}) #将集合转化为列表

print list({'a':3,'b':9,'c':78}) #将字典的”键“转化为列表

print list({'a':3,'b':9,'c':78}.items()) #将字典的”键：值“转化为列表

#这里面注意一般将tuple()set()dict()list()这样的函数称之为工厂函数，因为这些函数可以生成新的数据类型

#6.8切片

aList9=[3,4,5,6,7,9,11,13,15,17]

aList9[::] #返回所有元素

aList9[::-1] #所有元素的逆序

aList9[::2]#隔一个元素获取一个元素，获取偶数位置的元素

aList9[1::2]#隔一个元素获取一个元素，获取奇数位置的元素

aList9[0:100]#切片位置大于列表长度，从源列表尾部截断

aList9[100:] #切片的开始位置大于列表长度，返回空列表

# aList9[100]#不允许越界访问，抛出IndexError异常

## 3.2part2-zip函数和enumerate函数

#6.5zip函数:用于将多个列表中元素重新组合为元组并返回包含这些元组的zip对象

#enumerate()函数返回包含若干下标和值的迭代对象

x=list(range(10))

import random

random.shuffle(x)

print x

print list(zip(x,[1]\*11)) #zip函数也可以用于一个序列或迭代对象

print list(zip(range(1,4)))

print list(zip(['a','b','c'],[1,2])) #如果两个列表不等长，以短的为主

print enumerate(x) #枚举列表对象，返回enumerate对象

print list(enumerate(x)) #enumreate对象可迭代

## 3.3part3\_List函数操作

# 列表操作包含以下函数:

# 1、cmp(list1, list2)：比较两个列表的元素

# 2、len(list)：列表元素个数

# 3、max(list)：返回列表元素最大值

# 4、min(list)：返回列表元素最小值

# 5、list(seq)：将元组转换为列表

# 列表操作包含以下方法:

# 1、list.append(obj)：在列表末尾添加新的对象

# 2、list.count(obj)：统计某个元素在列表中出现的次数

# 3、list.extend(seq)：在列表末尾一次性追加另一个序列中的多个值（用新列表扩展原来的列表）

# 4、list.index(obj)：从列表中找出某个值第一个匹配项的索引位置

# 5、list.insert(index, obj)：将对象插入列表

# 6、list.pop(obj=list[-1])：移除列表中的一个元素（默认最后一个元素），并且返回该元素的值

# 7、list.remove(obj)：移除列表中某个值的第一个匹配项

# 8、list.reverse()：反向列表中元素

# 9、list.sort([func])：对原列表进行排序

a=list(range(1,10,1))

b=list(range(2,11,2))

print cmp(a,b)

print max(a)

print a.index(2)

c=a.insert(0,"app") #c无效

print a #对a本地操作

a.extend(b)

print a

2. 元组（tuple)

创建

访问

删除

脚本操作符

函数方法

## 3.4part2列表list与元祖tuple的比较

#创建只有一个元素的tuple，需要用逗号结尾消除歧义

a\_tuple = (2,)

#tuple中的list

mixed\_tuple = (1, 2, ['a', 'b'])

print("mixed\_tuple: " + str(mixed\_tuple))

mixed\_tuple[2][0] = 'c'

mixed\_tuple[2][1] = 'd'

print("mixed\_tuple: " + str(mixed\_tuple))

Tuple 是不可变 list。 一旦创建了一个 tuple 就不能以任何方式改变它。

Tuple 与 list 的相同之处

\*定义 tuple 与定义 list 的方式相同, 除了整个元素集是用小括号包围的而不是方括号。

\*Tuple 的元素与 list 一样按定义的次序进行排序。

\*Tuples 的索引与 list 一样从 0 开始, 所以一个非空 tuple 的第一个元素总是 t[0]。

负数索引与 list 一样从 tuple 的尾部开始计数。

\*与 list 一样分片 (slice) 也可以使用。注意当分割一个 list 时, 会得到一个新的 list ；当分割一个 tuple 时, 会得到一个新的 tuple。

Tuple 不存在的方法

\*您不能向 tuple 增加元素。Tuple 没有 append 或 extend 方法。

如：

list\_2=[1,2,3,4]

list\_2.append('a')

print list\_2

list\_2.extend([1,2,3,4])

print list\_2

\*您不能从 tuple 删除元素。Tuple 没有 remove 或 pop 方法。

\*然而, 您可以使用 in 来查看一个元素是否存在于 tuple 中。

用 Tuple 的好处

\*Tuple 比 list 操作速度快。如果您定义了一个值的常量集，并且唯一要用它做的是不断地遍历它，请使用 tuple 代替 list。

\*如果对不需要修改的数据进行 “写保护”，可以使代码更安全。使用 tuple 而不是 list 如同拥有一个隐含的 assert 语句，说明这一数据是常量。如果必须要改变这些值，则需要执行 tuple 到 list 的转换。

Tuple 与 list 的转换

Tuple 可以转换成 list，反之亦然。内置的 tuple 函数接收一个 list，并返回一个有着相同元素的 tuple。而 list 函数接收一个 tuple 返回一个 list。从效果上看，tuple 冻结一个 list，而 list 解冻一个 tuple。

Tuple 的其他应用

一次赋多值====序列解包

>>> v = ('a', 'b', 'e')

>>> (x, y, z) = v

解释：v 是一个三元素的 tuple, 并且 (x, y, z) 是一个三变量的 tuple。将一个 tuple 赋值给另一个 tuple, 会按顺序将 v 的每个值赋值给每个变量。

## 3.5Dictionary字典数据结构

键(key)，对应值(value)

结构介绍

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

#创建一个词典

phone\_book = {'Tom': 123, "Jerry": 456, 'Kim': 789}

mixed\_dict = {"Tom": 'boy', 11: 23.5}

#访问词典里的值

print("Tom's number is " + str(phone\_book['Tom']))

print('Tom is a ' + mixed\_dict['Tom'])

#修改词典

phone\_book['Tom'] = 999

phone\_book['Heath'] = 888

print("phone\_book: " + str(phone\_book))

phone\_book.update({'Ling':159, 'Lili':247})

print("updated phone\_book: " + str(phone\_book))

#删除词典元素以及词典本身

del phone\_book['Tom']

print("phone\_book after deleting Tom: " + str(phone\_book))

#清空词典

phone\_book.clear()

print("after clear: " + str(phone\_book))

#删除词典

del phone\_book

# print("after del: " + str(phone\_book))

#不允许同一个键出现两次

rep\_test = {'Name': 'aa', 'age':5, 'Name': 'bb'}

print("rep\_test: " + str(rep\_test))

#键必须不可变，所以可以用数字，字符串或者元组充当，列表不行

list\_dict = {['Name']: 'John', 'Age':13}

list\_dict = {('Name'): 'John', 'Age':13}

# 六、字典内置函数&方法

# Python字典包含了以下内置函数：

# 1、cmp(dict1, dict2)：比较两个字典元素。

# 2、len(dict)：计算字典元素个数，即键的总数。

# 3、str(dict)：输出字典可打印的字符串表示。

# 4、type(variable)：返回输入的变量类型，如果变量是字典就返回字典类型。

# Python字典包含了以下内置方法：

# 1、radiansdict.clear()：删除字典内所有元素

# 2、radiansdict.copy()：返回一个字典的浅复制

# 3、radiansdict.fromkeys()：创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值

如：print "fromkeys",dict\_2.fromkeys(dict\_2,10)

# 4、radiansdict.get(key, default=None)：返回指定键的值，如果值不在字典中返回default值

# 5、radiansdict.has\_key(key)：如果键在字典dict里返回true，否则返回false

# 6、radiansdict.items()：以列表返回可遍历的(键, 值) 元组数组

# 7、radiansdict.keys()：以列表返回一个字典所有的键

# 8、radiansdict.setdefault(key, default=None)：和get()类似, 但如果键不已经存在于字典中，将会添加键并将值设为default

# 9、radiansdict.update(dict2)：把字典dict2的键/值对更新到dict里

# 10、radiansdict.values()：以列表返回字典中的所有值

#备注：

#Python的最重要的数据类型-字典dictionnary

#创建字典

dic1={"a":1,"b":2,"c":'apple'}

print dic1

#创建字典的方式2

dict2=dict(zip(range(3),["a","b","c"]))

print dict2

print type(range(3)) #list

#创建字典的方式3

dict3=dict(name="wang",age="23")

print dict3

#字典的遍历

for elem in dic1:

print elem

print dic1[elem] #切记容易出错

for elem in dic1.items():

print elem

for key,elem in dic1.items():

print key,elem

## 3.6集合补充

#-\*-coding:utf8-\*-

#集合中只能包含数字、字符串、元祖等不可变类型的数据，而不能包含列表、字典、集合等可变类型数据

#有一个方法：利用内置的hash()函数计算对象的哈希值，凡是无法计算哈希值的对象都不能作为集合的元素，也不能作为字典对象的“键”

#字典的定义

a={3,5} #字典定义方法1

type(a)

a\_set=set(range(8,14))#将range对象转化为集合

b\_set=set([0,1,2,3,0,1,2,3,7,8])#转化时自动去掉重复元素

c\_set=set((1,2,3))

d\_set=set({1:'a',2:'b'}.items())

f\_set=set() #空集合

del b\_set #删除集合

#集合的操作和运算

#集合元素的增加、更新和删除

s={1,2,3}

s.add(3)

print s

s.update({3,4}) #更新当前元素，自动忽略重复元素

print s

s.discard(5) #删除元素，不存在则忽略该操作

# s.remove(5) #删除元素，不存在则抛出异常

#集合的运算

（在mspaint中绘图解释）

d\_set=set([8,9,10,11,12,13])

f\_set={0,1,2,3,7,8}

d\_set | f\_set #并集

d\_set.union(f\_set) #并集

d\_set & f\_set #交集

d\_set.intersection(f\_set) #交集

d\_set.difference(f\_set) #差集

d\_set-f\_set #差集

d\_set.symmetric\_difference(f\_set) #对称差集

d\_set.issubset(f\_set) #测试是否为子集

x1={1,2,3}

y1={1,2,5}

z1={1,2,3,4}

print x1<y1 #< > >=这些符号作用于集合的时候用于表示集合之间的包含关系

print x1<z1

## 3.7列表推导(掌握)

（回顾）

(1)回顾列表的创建和切片的知识

#列表创建

print [1,'a','four'] #创建列表

print [] #创建空列表

print list((1,2,3,3,4)) #将元组转化为列表

print list(range(1,10,2)) #将range对象转化为列表

print list('hello') #将字符串转化为列表

print list({3,5,7}) #将集合转化为列表

print list({'a':3,'b':9,'c':78}) #将字典的”键“转化为列表

print list({'a':3,'b':9,'c':78}.items()) #将字典的”键：值“转化为列表

#这里面注意一般将tuple()set()dict()这样的函数称之为工厂函数，因为这些函数可以生成新的数据类型

#切片

aList9=[3,4,5,6,7,9,11,13,15,17]

aList9[::] #返回所有元素

aList9[::-1] #所有元素的逆序

aList9[::2]#隔一个元素获取一个元素，获取偶数位置的元素

aList9[1::2]#隔一个元素获取一个元素，获取奇数位置的元素

aList9[0:100]#切片位置大于列表长度，从源列表尾部截断

aList9[100:] #切片的开始位置大于列表长度，返回空列表

# aList9[100]#不允许越界访问，抛出IndexError异常

#列表的增删改查

aList10=[3,5,7]

aList10[len(aList10):]=[9] #在尾部增加元素

aList10[:3]=[1,2,3] #替换列表中的元素

aList10[:3]=[] #删除列表元素

del aList10[:3] #删除列表元素

print aList10

bList=[3,5,7]

aList10==bList #两个列表的值是相等的

aList10 is bList #两个列表是同一对象

id(aList10)==id(bList) #两个列表的内存地址相等

bList=aList[::] #切片，浅复制

bList.append(9) #使用append增加元素

cList=[1,1,1,3,1,5,1,7,1,9]

del cList[::2]

print cList #[1, 3, 5, 7, 9]

(2)列表推导式

#-\*-coding:utf8-\*-

#列表推导式语法

#[表达式 for 变量 in 序列或迭代对象]

#快速生成满足特定需求的列表

#列表推导在逻辑上相当于一个循环，只是形式更加简洁

#Demo1

aList=[x\*x for x in range(10)]

print aList

#等同于

aList1=[]

for x in range(10):

aList1.append(x\*x)

print aList1

#等同于

aList2=list(map(lambda x:x\*x,range(10)))

print aList2

#Demo2 练习：

freshfruit=['banana','loganberry','passion fruit']

aList3=[]

for item in freshfruit:

aList3.append(item.strip())

print aList3

#等同于

aList4=list(map(lambda x:x.strip(),freshfruit))

print aList4

#等同于

aList5=list(map(str.strip,freshfruit))

print aList5

（熟悉内容）

（加强）

（3）列表推导的其他应用

#1.使用列表推到式实现嵌套列表的平铺（两个for循环）

vec = [[1,2,3],[4,5,6],[7,8,9]]

new1=[num for elem in vec for num in elem]

print new1

#等同于

result=[]

for elem1 in vec: #[1,2,3],[4,5,6],[7,8,9]

for num in elem1:1,2,3,4,5,6,7,8,9

result.append(num)

print result

#2.过滤不符合条件的元素

aList5=[-1,-4,6,7.5,-2.3,9,-11]

print [i for i in aList5 if i>0]#所有大于零的元素

#3.列表推导中使用多个循环实现多序列元素任意的组合，并过滤元素

aList6=[(x,y)for x in [1,2,3] for y in [3,1,4] if x!=y]

print aList6

#等同于

result1=[]

for x in [1,2,3]:

for y in [3,1,4]:

if x!=y:

result1.append((x,y))

print result1

#4.实现列表推到式实现矩阵转置

matrix=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]

#先取行在取出列

aList7=[ [row[i] for row in matrix] for i in range(4)]

print aList7

#或者使用zip函数

aList8=list(map(list,zip(\*matrix)))

print aList8

#可以print zip(\*matrix) 吗？------因为zip生成的是元祖而不是list，因此需要利用list函数作用将类型改变为list类型。

#zip函数：返回元组列表，其中每个元组包含第i个元素来自每个参数序列。 返回的列表被截断长度与最短参数序列的长度相同。

#5.使用列表推导生成100以内的所有素数

import numpy as np

# [p for p in range(2,100) if 0 not in [p%d for d in range(2,int(np.sqrt(p))+1)]] #+1的意思是range不包括末尾值

（非重点掌握内容）

## 3.8.补充函数编程内容：

#6.1map函数将一个函数依次作用或映射到序列或迭代器对象的每个元素上，并返回一个map对象作为结果

案例1:

list(map(str,range(5))) #转化为字符串

案例2:

def add1(v):

return v+5

print list(map(add1,range(10)))#将单参数函数映射到1个序列上

#案例3：

def add(x,y):

return x+y

list(map(add,range(5),range(5,10)))#将双参数函数映射到两个序列上

#等同于

print list(map(lambda x,y:x+y,range(5),range(5,10)))

#等同于

print [add(x,y) for x,y in zip(range(5),range(5,10))]

#6.2reduce函数

#标准库functions中的函数reduce()可以将一个接受2个参数以累积的方式

# 从左到右一次作用到一个序列或迭代器对象的所有元素上

#在Python2中reduce是内置函数，不需要导入任何模块即可使用,py3中需要导入包

from functools import reduce

print reduce(add,range(10)) #add是上面定义的函数

seq=[1,2,3,4,5,6,7,8,9]

#将lambda函数应用在seq序列上

print reduce(lambda x,y:x+y,seq)

#6.3filter函数将一个单参数函数作用到一个序列上，返回该序列中

# 使得该函数返回值为True的那些元素组成的filter对象

#如果指定函数为None，则返回序列中等价于True的元素

#下面代码在python3中使用

# seq1=['foo','x41','?1','\*\*\*']

# def func(x):

# return x.isalnum()

# filter(func,seq) #返回filter对象

# print list(filter(func,seq)) #将filter转化为列表

# #使用列表推到式实现相同的功能

# print [x for x in seq if x.isalnum()]

# #使用lambda表达式实现相同功能

# print list(filter(lambda x:x.isalnum(),seq))

# print list(filter(None,[1,2,3,0,0,4,0,5]))

#6.4 列表的运算

print [1,2,3]\*3

print [1,2,3]+[4,5,6]

#以下为内置函数对列表的操作

#6.5zip函数:用于将多个列表中元素重新组合为元组并返回包含这些元组的zip对象

#enumerate()函数返回包含若干下标和值的迭代对象

x=list(range(10))

import random

random.shuffle(x)

print x

print max(x),min(x),sum(x),len(x)

print list(zip(x,[1]\*11)) #zip函数也可以用于一个序列或迭代对象

print list(zip(range(1,4)))

print list(zip(['a','b','c'],[1,2])) #如果两个列表不等长，以短的为主

print enumerate(x) #枚举列表对象，返回enumerate对象

print list(enumerate(x)) #enumreate对象可迭代

#6.6sort()和reverse()

print "原来的：",x

g=lambda item:len(str(item))

print g(x) #30?

print str(x) #[0, 3, 2, 8, 1, 5, 4, 7, 6, 9]

print len(str(x)) #30? len指的是列表元素的个数

x.sort(key=lambda item:len(str(item)),reverse=True) #按指定规则排序

print "排序的：",x

x.reverse()

print "逆序的",x

x.sort(key=str) #按转化为字符串后的大小排序

print "按转化为字符串的大小排序",x

## 3.9元素和生成器推导（了解）

（回顾和复习）

#-\*-coding:utf8-\*-

#元祖的创建

#元祖一旦创建，里面的元素是不可以更改的，无法为元祖增加或删除元素

#1.元祖没有提供append(),extend()和insert()方法

#2.也没有remove()和pop()方法，也不支持del()删除元祖元素

#但是del可以删除整个元祖

x=(1,2,3)

type(x)

x=(3) #和x=3一样，这个不是元祖，有歧义

x=(3,) #如果元祖中只有一个元素，必须在后面多写一个逗号

#空元祖的表示

x=()

x=tuple()

#将其他迭代对象转化为元祖

tuple(range(5))

（重点）

#生成器推导---

#和列表推导的差别在于使用的是圆括号而不是列表推导的方括号

#生成器推导得到是一个生成器对象而不是列表或元祖，可以将生成器推导转化为元祖或列表

g=((i+2)\*\*2 for i in range(10))

print g #表示生成器返回的是生成器对象

print tuple(g) #将生成器对象转化为数组

print list(g) #生成器对象已遍历结束，没有元素了

g=((i+2)\*\*2 for i in range(10))

print g.next() #Python3中使用g.\_\_next\_\_()

print next(g)

#总结：生成器对象具有惰性求值的特点，只在需要时返回元素，比列表推导有更高的效率

（非重点内容）

#扩展：

包含yield语句的函数可以用来创建可迭代的生成器对象

# def 函数名(参数)：

# ...

# yield 变量

#普通函数使用return返回一个数据，生成器使用yield产生一系列数据

#当f函数被调用的时候，会输出yield后面跟的变量的值

#斐波那契数列1,1,2,3,5,8,13,21

def f():

a,b=1,1 #为多个元素赋值

while True:

yield a #暂停执行，需要时在产生一个新元素

a,b=b,a+b #序列解包，继续生成新元素

a=f() #创建生成器对象

for i in range(10): #数列的前10个元素

print a.next()

#常规的例子

def counter(start=0):

while True:

yield start

start+=1

b=counter()

print 'first:',b.next()

print 'second:',b.next()

## 3.10字典推导式（了解）

（主要内容）

#字典推导式

f\_dict={i:str(i) for i in range(1,5)}

print f\_dict

x=['A','B','C','D']

y=['a','b','c','d']

e\_dict={i:j for i,j in zip(x,y)}

print e\_dict

#有序字典（python默认的字典是无序的）

import collections

x=collections.OrderedDict()

x['a']=3

x['b']=5

x['c']=8

print x

（回顾及加强内容）

#-\*-coding:utf8-\*-

#字典：包含若干“键值对”元素的无序可变序列

#键和值用冒号分割，不同元素用逗号分割，所有元素放在{}中

#键可以是Python中任意不可变的类型（整数、实数、复数、字符串、元祖 ）

#键不允许重复

#字典的创建

#创建方式1：

a\_dict={1:'apple',2:'bnalna',3:'pear'}

print a\_dict

#创建方式2：

b\_dict=dict(zip([1,2,3],['apple','bnalana','pear']))

print b\_dict

#创建方式3：

c\_dict=dict(name='ni',age='20')

print c\_dict

#创建方式4:创建给定的内容为键，值为空的字典

d\_dict=dict.fromkeys(['name','age','sex'])

print d\_dict

dict\_3['name']="zhao" #更新值的内容

#空字典

x=dict()

y={}

#查看字典类型

type(x)

#修改元素值

c\_dict['age']=30

print c\_dict

#添加新元素

c\_dict['address']='SDIBT'

print c\_dict

#返回所有元素

c\_dict.items()

#返回所有的键

c\_dict.keys()

#返回所有的值

c\_dict.values()

#字典的update方法将另一个字典的键：值一次性添加到当前字典对象

c\_dict.update({'a':97,'age':39})#修改age的值，同时添加键值a：97

print c\_dict

#删除字典元素

del c\_dict['age']

#使用if判断键是否存在

if 'age' in c\_dict:

print c\_dict['Age']

else:

print 'NoExists'

#可以使用异常结构

try:

print c\_dict['address']

except:

print 'NoExists'

#Python提供了强大的get函数可以实现上述的功能

print c\_dict.get('age')

print c\_dict.get('Address','Not Exists.')

print c\_dict

#删除整个字典

# del c\_dict

#字典的遍历

for item in c\_dict: #默认遍历的是字典的键

print item

for item in c\_dict.items():#明确遍历字典的所有元素

print item

（非重点内容）

（加强）

#字典的排序

g\_dict={'name': 'Dong', 'age': 30}

from operator import itemgetter

print sorted(g\_dict.items(),key=itemgetter(1))#按字典的“值”排序

print sorted(g\_dict.items(),key=itemgetter(0))#按字典的“键”排序

print sorted(g\_dict.items(),key=lambda item:item[0])#按字典的“键”排序

#Demo

h\_dict=[{'name': 'Dong', 'age': 30},{'name': 'zhang', 'age': 31},

{'name': 'Li', 'age': 32},{'name': 'Dong', 'age': 33}]

print h\_dict

#按照key指定排序依据，先按姓名升序，姓名相同按照年龄降序

print sorted(h\_dict,key=lambda x:(x['name'],-x['age']))

# Python基础第二部分

# 4.0Python函数

## 4.1part1\_函数

函数：程序中可重复使用的程序段

给一段程程序起一个名字，用这个名字来执行一段程序，反复使用 （调用函数）

用关键字 ‘def' 来定义，identifier(参数)

* identifier
* 参数list
* return statement
* 局部变量 vs 全局变量

Code:

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

#没有参数和返回的函数

# def say\_hi():

# print(" hi!")

#

# say\_hi()

# say\_hi()

#

#

#有参数，无返回值

# def print\_sum\_two(a, b):

# c = a + b

# print(c)

#

# print\_sum\_two(3, 6)

# def hello\_some(str):

# print("hello " + str + "!")

#

# hello\_some("China")

# hello\_some("Python")

#有参数，有返回值

# def repeat\_str(str, times):

# repeated\_strs = str \* times

# return repeated\_strs

#

#

# repeated\_strings = repeat\_str("Happy Birthday!", 4)

# print(repeated\_strings)

#全局变量与局部 变量

# x = 60

#

# def foo(x):

# print("x is: " + str(x))

# x = 3

# print("change local x to " + str(x))

#

# foo(x)

# print('x is still', str(x))

x = 60

def foo():

global x

print("x is: " + str(x))

x = 3

print("change local x to " + str(x))

foo()

print('value of x is' , str(x))

## 4.2part2\_函数

* 默认参数
* 关键字参数
* VarArgs参数

Code:

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

# 1.默认参数，且默认值参数一定在最右端

def repeat\_str(s, times = 1):

repeated\_strs = s \* times

return repeated\_strs

repeated\_strings = repeat\_str("Happy Birthday!")

print(repeated\_strings)

repeated\_strings\_2 = repeat\_str("Happy Birthday!" , 4)

print(repeated\_strings\_2)

#不能在有默认参数后面跟随没有默认参数

#f(a, b =2)合法

#f(a = 2, b)非法

#2.关键字参数: 指的是调用时候的参数传递形式，可以按参数名字指定值

def func(a, b = 4, c = 8):

print('a is', a, 'and b is', b, 'and c is', c)

func(13, 17)

func(125, c = 24)

func(c = 40, a = 80)

#3.VarArgs参数（重要）

#\*参数和\*\*参数，前者用来接收任意多个实参并将其放在一个元祖中

#\*\*用来接收字典类型的参数

def print\_paras(fpara, \*nums, \*\*words):

print("fpara: " + str(fpara))

print("nums: " + str(nums))

print("words: " + str(words))

print\_paras("hello", 1, 3, 5, 7, word = "python", anohter\_word = "java")

## 4.3函数补充内容

3.5(补充)函数中需要注意的事项

#-\*-coding:utf8-\*-

#函数中需要注意的事项

#1.默认值参数，且默认值参数一定在最右端

def say(mess,times=1):

print (mess+' ')\*times

print say.func\_defaults #查看函数所有默认值参数的当前值

say('hello')

say('hello',3)

#2.关键参数：指的是调用时候的参数传递形式，可以按参数名字指定值

def demo(a,b,c=5):

print a,b,c

print demo(3,7)

print demo(a=7,b=3,c=6)

print demo(a=8,b=9,c=0)

#3.可变长参数

#\*参数和\*\*参数，前者用来接收任意多个实参并将其放在一个元祖中

#\*\*用来接收字典类型的参数

def demo1(\*p):

print p

demo1(1,2,3)

demo1(1,2,3,34)

def demo2(\*\*p):

for item in p.items():

print item

demo2(x=1,y=2,z=3) #key和value形式

#4.传递参数时的序列解包

def demo3(a,b,c):

print a+b+c

seq=[1,2,3]

demo3(\*seq)

tup=(1,2,3)

demo3(\*tup)

dict={1:'a',2:'b',3:'c'}

#字典对象作为实参时默认使用字典的键，如果需要将字

# 典中键值元素作为参数则需要使用items()方法明确说明，

# 如果单独取出“值”需要调用字典的values()方法说明

demo3(\*dict)

demo3(\*dict.items())

set={1,2,3}

demo(\*set)

demo3(\*dict.values())

## 4.4 lambda表达式

语法；

函数名=lambda 参数：表达式

代码：

#1

g=lambda x,y:x+y

print(g(4,5))

#2.带默认参数的lambda表达式

g1=lambda x,y=0,z=0:x+y+z

print(g(5))

#3.直接使用lambda表达式

(lambda x,y=0,z=0:x+y+z)(4,5,6)

使用建议：

当函数非常简单，只有一个表达式的时候，可以考虑lambda表达式，否则建议用普通函数

（重点知识）

#5.lambda表达式--声明匿名函数（没有函数名字的临时使用的小函数）

#lambda表达式只可以使用一个表达式，不允许包含其他复杂语句，但是在表达式中可以调用其他函数

#并支持默认值参数和关键参数，该表达式的计算结果相当于函数的返回值

f=lambda x,y,z:x+y+z

print f(1,2,3)

g=lambda x,y=2,z=3:x+y+z #含有默认参数

print g(1)

print g(2,4,5)

print g(2,y=4,z=5) #调用时使用关键参数

L=[(lambda x:x\*\*2),(lambda x:x\*\*3),(lambda x:x\*\*4)]

print L[0](2),L[1](2),L[2](2)

L=[1,2,3,4,5]

print map((lambda x:x+10),L) #没有名字的lambda表达式作为函数参数

print map((lambda x,y,z:x\*y\*z),[1,2,3],[1,2,3],[1,2,3])

#lambda使用自定义函数（平方）

def demo4(n):

return n\*n

demo4(5)

a\_list=[1,2,3,4,5]

map(lambda x:demo4(x),a\_list)

**#非重点掌握**

data=list(range(20))

print data

import random

random.shuffle(data)

print data

data.sort(key=lambda x:x)

print data

data.sort(key=lambda x:len(str(x))) #使用lambda表达式指定排序规则

print data

data.sort(key=lambda x:len(str(x)),reverse=True)

print data

# 5.0Python程序设计(一)

## 5.1控制流1\_if\_for语句

1. if 语句

if condition:

do something

elif other\_condition:

do something

2. for 语句

Code:

#猜数字游戏，每次猜一个，告诉用户猜的数值低还是高

# #if statement example

#

# number = 59

# guess = int(input('Enter an integer : '))

#

# if guess == number:

# # New block starts here

# print('Bingo! you guessed it right.')

# print('(but you do not win any prizes!)')

# # New block ends here

# elif guess < number:

# # Another block

# print('No, the number is higher than that')

# # You can do whatever you want in a block ...

# else:

# print('No, the number is a lower than that')

# # you must have guessed > number to reach here

#

# print('Done')

# # This last statement is always executed,

# # after the if statement is executed.

#the for loop example

#1.for的基本语法

# for i in range(1, 10): #在python中range是关键字，从a到b的值列表，不包括b

# print(i)

# else: #for循环执行完毕之后使用else

# print('The for loop is over')

#

# 2.for循环用于list

# a\_list = [1, 3, 5, 7, 9]

# for i in a\_list:

# print(i)

# 3.for循环用于元祖

# a\_tuple = (1, 3, 5, 7, 9)

# for i in a\_tuple:

# print(i)

# 4.for循环用于字典

# a\_dict = {'Tom':'111', 'Jerry':'222', 'Cathy':'333'}

# for ele in a\_dict:

# print(ele) #键

# print(a\_dict[ele]) #值

#

# for key, elem in a\_dict.items(): #items遍历的是键和值

# print(key, elem)

## 5.2控制流2\_while\_range语句

1. while语句

2. range语句

Code：

#2.1while example

# number = 59

# guess\_flag = False

#

#

# while guess\_flag == False:

# guess = int(input('Enter an integer : '))

# if guess == number:

# # New block starts here

# guess\_flag = True

#

# # New block ends here

# elif guess < number:

# # Another block

# print('No, the number is higher than that, keep guessing')

# # You can do whatever you want in a block ...

# else:

# print('No, the number is a lower than that, keep guessing')

# # you must have guessed > number to reach here

#

# print('Bingo! you guessed it right.')

# print('(but you do not win any prizes!)')

# print('Done')

#2.2For example 适合于固定循环次数的循环中

（仅需更改一行即可实现固定次数循环）

number = 59

num\_chances = 3

print("you have only 3 chances to guess")

for i in range(1, num\_chances + 1):

print("chance " + str(i))

guess = int(input('Enter an integer : '))

if guess == number:

# New block starts here

print('Bingo! you guessed it right.')

print('(but you do not win any prizes!)')

break

# New block ends here

elif guess < number:

# Another block

print('No, the number is higher than that, keep guessing, you have ' + str(num\_chances - i) + ' chances left')

# You can do whatever you want in a block ...

else:

print('No, the number is lower than that, keep guessing, you have ' + str(num\_chances - i) + ' chances left')

# you must have guessed > number to reach here

print('Done')

## 5.3控制流2\_break\_continue语句

4.3\_控制流3break\_continue\_\_pass

1. break

2. continue

3. pass

Code:

#break & continue example

# number = 59

#

# while True:

# guess = int(input('Enter an integer : '))

# if guess == number:

# # New block starts here

# break

#

# # New block ends here

# if guess < number:

# # Another block

# print('No, the number is higher than that, keep guessing')

# continue

# # You can do whatever you want in a block ...

# else:

# print('No, the number is a lower than that, keep guessing')

# continue

# # you must have guessed > number to reach here

#

# print('Bingo! you guessed it right.')

# print('(but you do not win any prizes!)')

# print('Done')

#continue and pass difference

# a\_list = [0, 1, 2]

#

# print("using continue:")

# for i in a\_list:

# if not i:

# continue

# print(i)

#

# print("using pass:")

# for i in a\_list:

# if not i:

# pass

# print(i)

# 6.0Python程序设计(二)

## 6.1 输入输出方式介绍(Output Format)

接受用户的输入： input()

输入格式：str(), format

Code:

str\_1 = input("Enter a string: ")

str\_2 = input("Enter another string: ")

print("str\_1 is: " + str\_1 + ". str\_2 is :" + str\_2)

print("str\_1 is {} + str\_2 is {}".format(str\_1, str\_2))

## 6.2\_读写文件File\_IO

1. 写出文件

2. 读入文件

Code:

#文件写入磁盘

some\_sentences = '''\

I love learning python

because python is fun

and also easy to use

'''

#Open for 'w'irting

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

#Write text to File

f.write(some\_sentences)

f.close()

#文件从磁盘读取

#If not specifying mode, 'r'ead mode is default

f = open('sentences.txt')#默认是r读模式

while True:

line = f.readline()

#Zero length means End Of File

if len(line) == 0:

break

print(line)

# close the File

f.close

## 6.3\_错误与异常处理Error\_\_Exceptions

Python有两种错误类型：

1. 语法错误(Syntax Errors)

2. 异常（Exceptions)

首先，try语句下的（try和except之间的代码）被执行

如果没有出现异常，except语句将被忽略

如果try语句之间出现了异常，try之下异常之后的代码被忽略，直接跳跃到except语句

如果异常出现，但并不属于except中定义的异常类型，程序将执行外围一层的try语句，如果异常没有被处理，将产生unhandled exception的错误

处理异常（Handling Exceptions)

Exception doc: https://docs.python.org/3.4/library/exceptions.html

Code:

#Example of Syntax errors

# while True print("Hello World!")

#Examples of exceptions

# print(8/0)

# print(hello \* 4)

# num = 6

# print("Hello World " + num )

#Handling exceptions

# while True:

# try:

# x = int(input("Please enter a number"))

# break

# except ValueError:

# print("Not valid input, try again...")

# 7.0Python面向对象程序设计基础

## 7.1\_面向对象编程Object-Oriented

1. 面向对象编程

Python支持面向对象编程

类(class)：现实世界中一些事物的封装 （如：学生）

类：属性 （如：名字，成绩）

类对象

实例对象

引用：通过引用对类的属性和方法进行操作

实例化：创建一个类的具体实例对象 （如：学生张三）

## 7.2面向对象实现加减乘除简单四则运算(补充)

#-\*-coding:utf8-\*-

#python的面向对象

class ArithmetricOperation:

def \_\_init\_\_(self,xnew,ynew):

self.x=xnew

self.y=ynew

def add(self):

return self.x+self.y

def sub(self):

return self.x-self.y

def mul(self):

return self.x\*self.y

def div(self):

return self.x/self.y

# obj.x=float(input("X:"))

# obj.y=float(input("Y:"))

obj=ArithmetricOperation(float(input("X:")),float(input("Y:")))

print "x+y=",obj.add()

print "x-y=",obj.sub()

print "x\*y=",obj.mul()

print "x/y=",obj.div()

#需求：将上述面向对象程序单独写在一个py文件中，在另外一个类中使用：

**from 文件名 import 类名 的方式输出对应的结果信息**

# 8.0Python图形界面GUI设计

## 8.1图形界面GUI

1. GUI： Graphical User Interface

2. tkinter: GUI library for Python

3. GUI Example

该代码在Python3.6中演示并讲解

Code:

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

from tkinter import \*

import tkinter.simpledialog as dl

import tkinter.messagebox as mb

#tkinter GUI Input Output Example

#设置GUI

root = Tk() #创建应用程序窗口

w = Label(root, text = "Label Title")#创建标签

w.pack() #画布框架

#欢迎消息

mb.showinfo("Welcome", "Welcome Message")#第一个参数title，第二个参数message

guess = dl.askinteger("Number", "Enter a number")

#第一个参数caotion标题，第二个参数是prompt提示

output = 'This is output message'

mb.showinfo("Output: ", output)

## 8.2猜数字小游戏

8.2\_猜数字游戏

1. GUI from tkinter

2. 逻辑层

Code:

# #设置GUI

# root = Tk()

# w = Label(root, text = "Guess Number Game")

# w.pack()

#

# #欢迎消息

# mb.showinfo("Welcome", "Welcome to Guess Number Game")

#

#

# #处理信息

# number = 59

#

# while True:

# #让用户输入信息

# guess = dl.askinteger("Number", "What's your guess?")

#

# if guess == number:

# # New block starts here

# output = 'Bingo! you guessed it right, but you do not win any prizes!'

# mb.showinfo("Hint: ", output)

# break

# # New block ends here

# elif guess < number:

# output = 'No, the number is a higer than that'

# mb.showinfo("Hint: ", output)

# else:

# output = 'No, the number is a lower than that'

# mb.showinfo("Hint: ", output)

#

# print('Done')

#Tips：

原始问题：下面的代码执行后为什么x的值是[2, 2]呢？

>>> x = [3, 5, 7]

>>> x = x[1:] = [2]

>>> x

[2, 2]

进一步，我们修改x列表的初始内容，会发现不管x的初始值是什么，执行x = x[1:] = [2]之后的x的值都是[2, 2]：

>>> x = [1, 2, 3, 3, 4, 5, 6]

>>> x = x[1:] = [2]

>>> x

[2, 2]

ans：

实际上这里有个小坑，这个问题的根源在于x = [1:] = [2]相当于x = [2]和x[1:] = [2]这两条语句，也就是说先创建列表x的值为[2]，然后使用切片为其追加一个元素2，然后得到[2, 2]。

# 9.0Python补充问题

## 9.1print换行打印问题？

在python中使用print方法时，默认是换行打印的，如果我们不想换行打印呢？

这样的，在python3.x之前，可以通过在print语句之后加逗号解决

for i in range(1, 5):

print i,

在python3.x之后，可以在print()之中加end=""来解决，可以自定义结尾字符

如：

for j in range(1, 5):

print(j, end=" ")