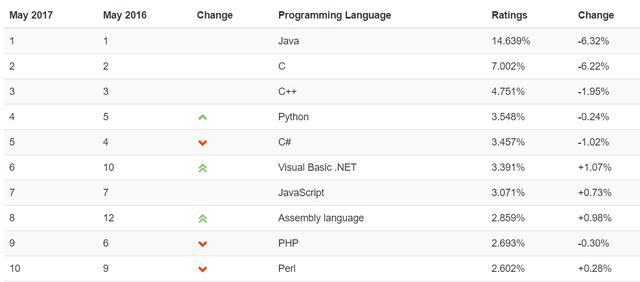
# python概述

## Python语言的发展历史

自从20世纪90年代初Python语言诞生至今，它已被逐渐广泛应用于专业数据采集处理、数据科学计算分析以及自动化测试运维领域。



## Python 是一种 面向对象的解释型计算机程序设计语言

Python 是纯粹的自由软件， 源代码和解释器CPython遵循GPL (GNU General Public License)协议。

Python 语法简洁清晰，特色之一是强制用空白符(white space)作为语句缩进 (标准四个空格) 。

c语言：

public void main(){

代码段;

}

Python语言:

def main():

代码;

## python的应用领域

### 数据采集与处理

使用 Requets/Urllib/Re 模块库实现典型的网络爬虫程序，采集各种结构化和非结构化数据，通过动态代理Proxy自动轮询，突破网站403反爬虫拦截机制。同时使用 Scrapy 或 BS4 企业级爬虫框架快速完成网络深维度自动探索采集。

### 数据计算与分析领域

在数据处理方面使用强大的 NumPy / SciPy / Pandas 模块库实现数据规整化操作标准流程：采集->加载->清洗->转换->重塑；充分利用强大的 Pandas模块库实现聚合与分组算法、时间序列算法等核心的数据分析计算；最终使用 Matpolitlib模块库进行可视化数据呈现。

### 人工智能与机器学习领域

使用 Scikit-Learn 模块库实现机器学习，使用AIML人工智能标记语言。掌握 Theano/Keras 模块库搭建各种深度学习模型，如自编码、循环神经网络、递归神经网络等。了解Google公司的 TensorFlow人工智能系统。

### 自动化测试 领域

使用 Selenium2 模块库实现典型的网络模拟点击和虚拟操作，编写测试脚本完成对网站及应用的自动化测试，并进行测试日志存储记录和跟踪。结合相关平台和测试工具形成一整套的自动化测试标准流程和规范。

### 系统集成运维 领域

Fabric 模块库是基于Python实现的SSH命令行工具，简化了SSH的应用程序部署及系统管理任务，它提供了系统基础的操作组件，可以实现本地或远程shell命令，包括：命令执行、文件上传、下载及完整执行日志输出等功能。使用 Re 模块库对 Log 日志进行分析和处理。

### Web互联网 领域

使用 Socket 模块库实现服务器及客户端编程，实现TCP/UDP的协议下的数据通信操作。利用Django 框架快速实现网站开发，了解web服务器端框架 Flask/Tornado 在实际开发中的应用。

## Python语言特点

### 简单易学

适合没有任何编程语言基础的人稍微看一下资料，就可以写出功能强大的程序。

### 开发效率高

很难像Java那样开发出完整的大型综合性网站或平台，但其起到画龙点睛的作用。同时也是一门典型的“胶水语言”，整合其他各种编程语言。

### 典型的工具语言

它是一门解释型编程语言，编译完毕后可直接运行，发现Bug后立即修改，剩下大量的编译时间。

### 强大丰富的模块库

高度代码重用性，编写各种工具模块引用的系统工程中，丰富的模块强大到恐怖的地步，几乎无处不在适用于各种领域。

### 优秀的跨平台

几乎所有的Python程序，都可以不加修改地运行在不同的操作系统平台。

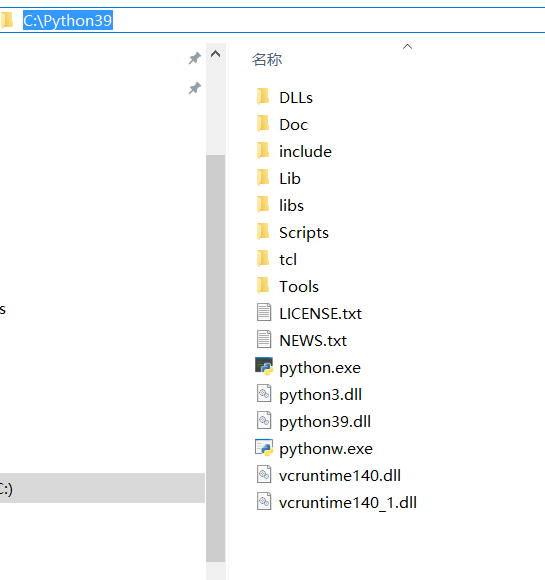
# 二．Python在Windows运行环境的搭建

## 2.1.可以从官方网站 https://www.python.org/ 下载对应版本的Python安装包

### 2.1.1.直接从qq群下载Python39.rar解压缩，并将解压后的文件夹复制到指定路径

例如：复制到如下路径

C:\Python39

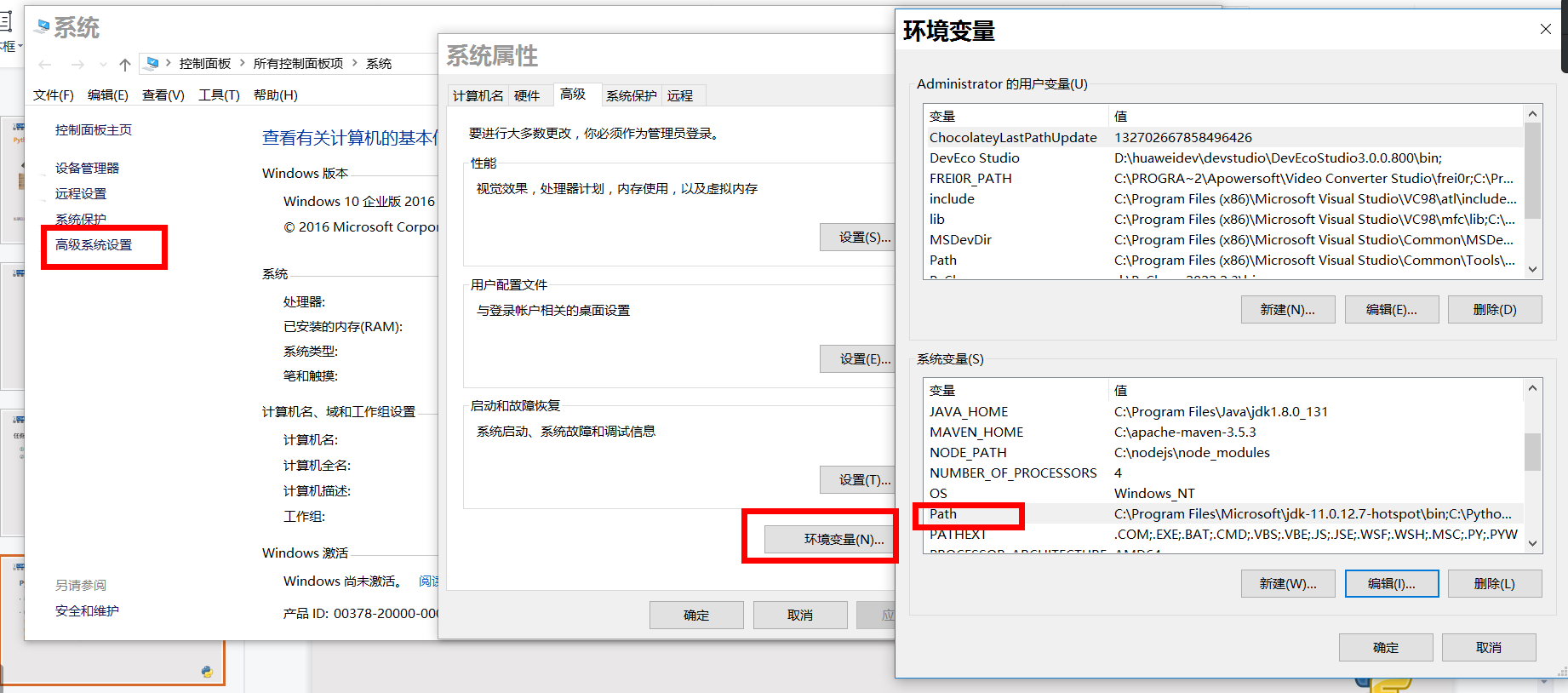


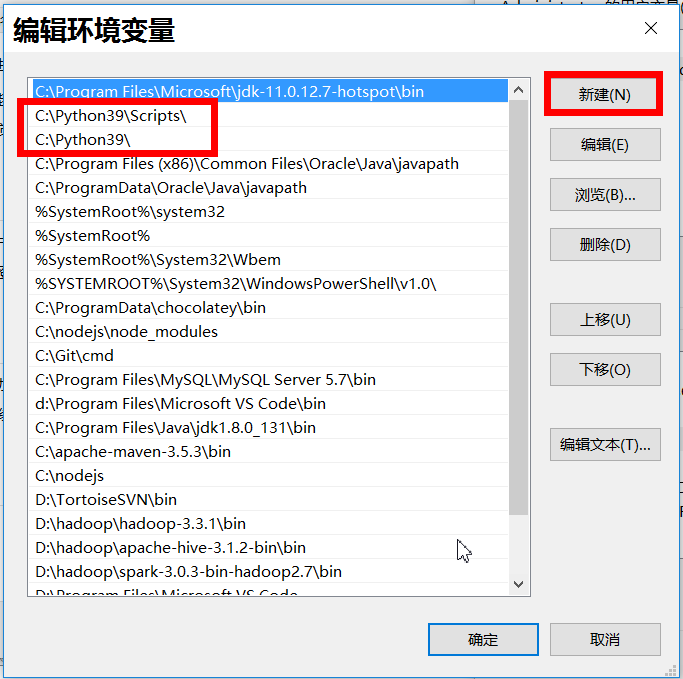
### 2.1.2.配置windows的环境变量

PYTHONHOME=python的安装路径

PYTHONPATH=python的安装路径

Path=...;python的安装路径;python的安装路径\Scripts

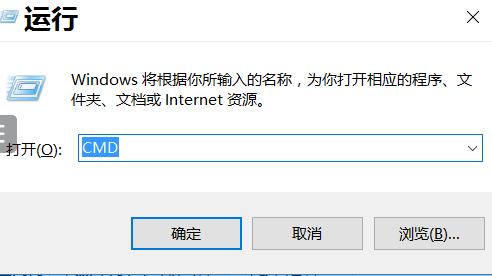


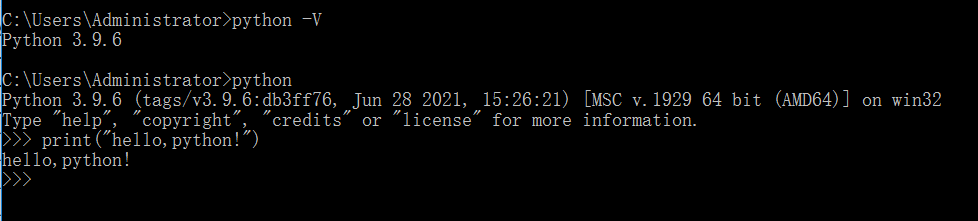


### 2.1.3.验证环境

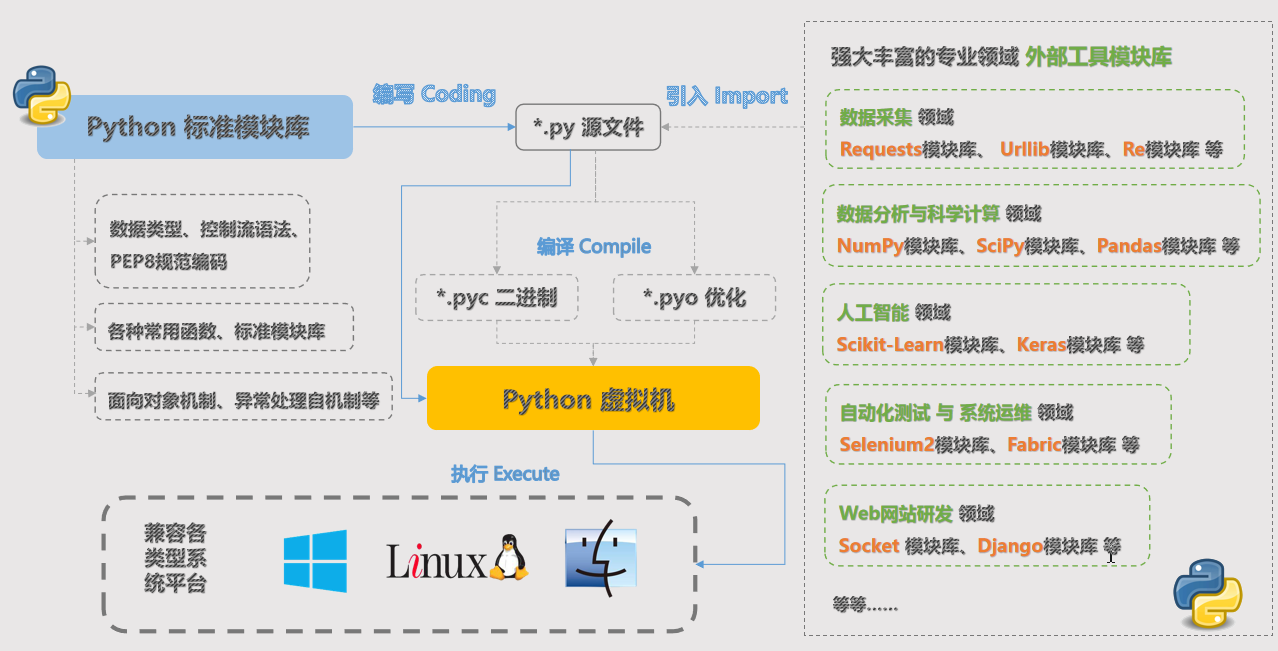
打开Windows下的命令行工具

使用 win+R 快捷键启动 运行，输入 cmd 启动 命令行工具





## 2.2.python工作原理



### 2.2.1.python加载源文件到python虚拟机

python的原文件以.py为结尾，在源文件中根据自己的业务需求引入不同组件库实现业务功能。

### 2.2.2.python虚拟机将源代码中的代码转译为操作系统能够识别的指令集

### 2.2.3.操作系统执行指令并返回结果给调用者或者输出结果给用户呈现

## 2.3.python源文件的运行方式

### 2.3.1.直接运行.py源文件

Python 语言是典型的脚本语言，通过解析器直接运行\*.py文件。所有Python脚本程序的后缀名都是以 \*.py 结尾。

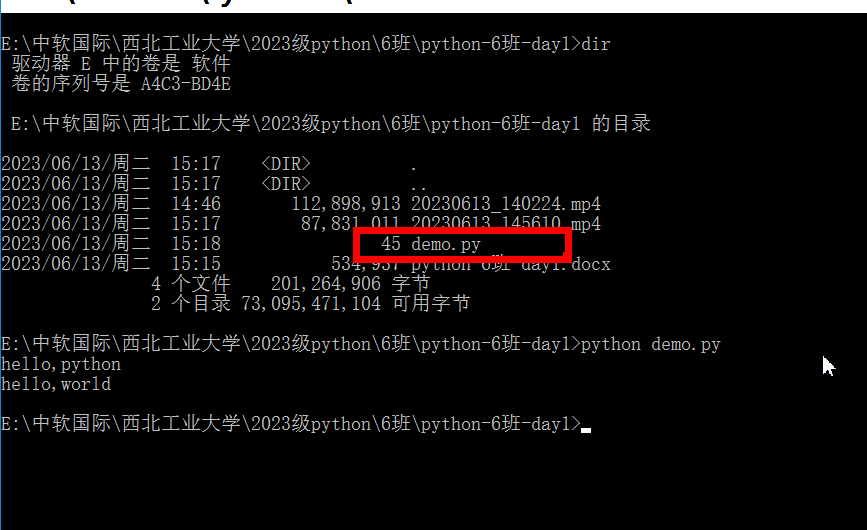
python 原文件名.py

编写python源文件：demo.py

源文件内容如下：

print("hello,python");

print("hello,world");



### 2.3.2.编译执行

#### 1.先将源文件编译为.pyc二进制字节码文件

Python -m py\_compile 源文件.py

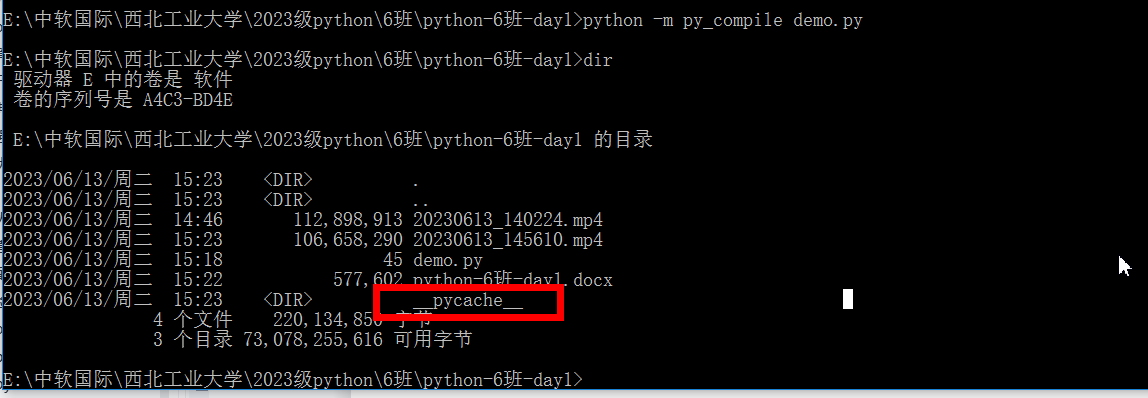
#### 使用python解释器执行pyc文件

Python 文件名.pyc

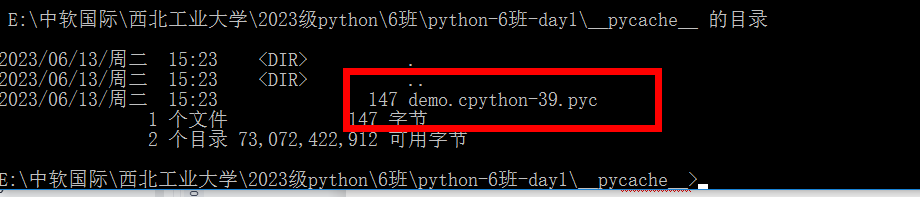
示例:

编译demo.py文件

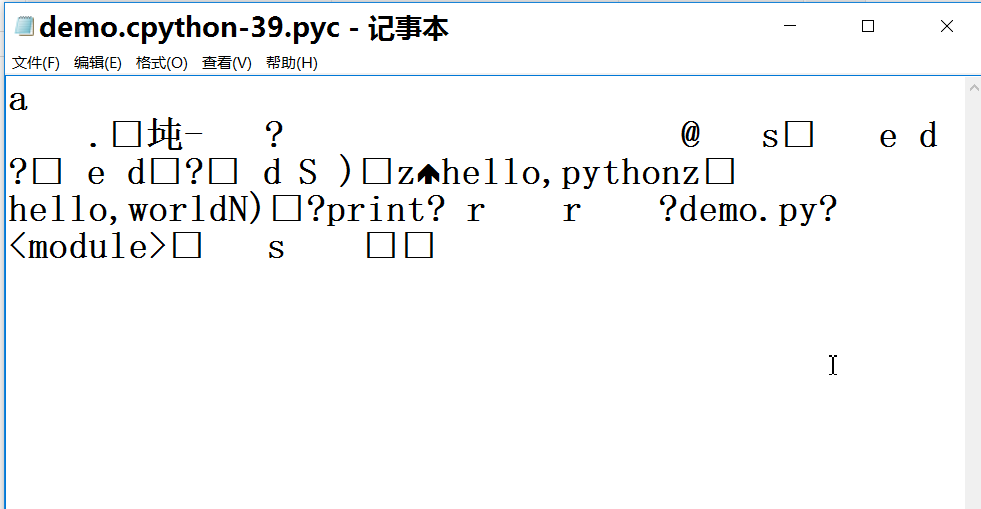
python -m py\_compile demo.py



进入到 \_\_pycache\_\_



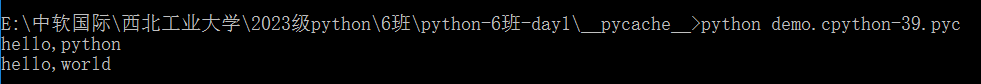
打开pyc文件结果如下：



源码不可见，可以隐藏源代码。

执行pyc文件

python demo.cpython-39.pyc



pyc是一种二进制文件，是由py文件经过解释器编译后，在磁盘上生成的文件形式，是一种byte code，py文件变成pyc文件后，加载的速度有所提高，而且pyc是一种跨平台的字节码，是由python的解析器来执行的。pyc的内容是跟python的版本相关的，不同版本编译后的pyc文件是不同的。

因为py文件是可以直接看到源码的，如果你是开发商业软件的话，不可能把源码也泄漏出去。所以就需要编译为pyc后，再发布出去。

## 2.4.python虚拟机的运行原理

PyCodeObject：pyc字节码在Python虚拟机中对应的是PyCodeObject对象，虚拟机先把字节码封装成一个PyCodeObject对象后再一条条执行字节码指令。

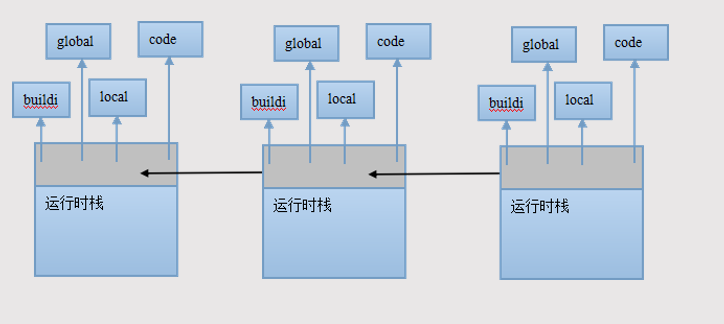
虚拟机中的PyCodeObject对象什么时候创建？

PyCodeObject对象的创建时机是模块加载的时候，即import。

1、执行 python test.py 会对test.py进行编译成字节码并解释执行，但不会生成test.pyc  
2、如果test.py中加载了其他模块，如import urllib2，那么python会对urllib2.py进行编译成字节码，生成urllib2.pyc，然后对字节码解释执行。  
3、如果想生成test.pyc，我们可以使用python内置模块py\_compile来编译。  
也可以执行命令 python -m test.py 这样，就生成了test.pyc  
4、加载模块时，如果同时存在.py和.pyc，python会使用.pyc运行，如果.pyc的编译时间早于.py的时间，则重新编译.py，并更新.pyc文件。

PyFrameObject

当发生函数调用时，创建新的栈帧，对应Python的实现就是PyFrameObject对象。



## 2.5. pyo 是优化编译后的程序

从Python3开始没有pyo为后缀名的文件，取而代之是生成对应的pyc文件

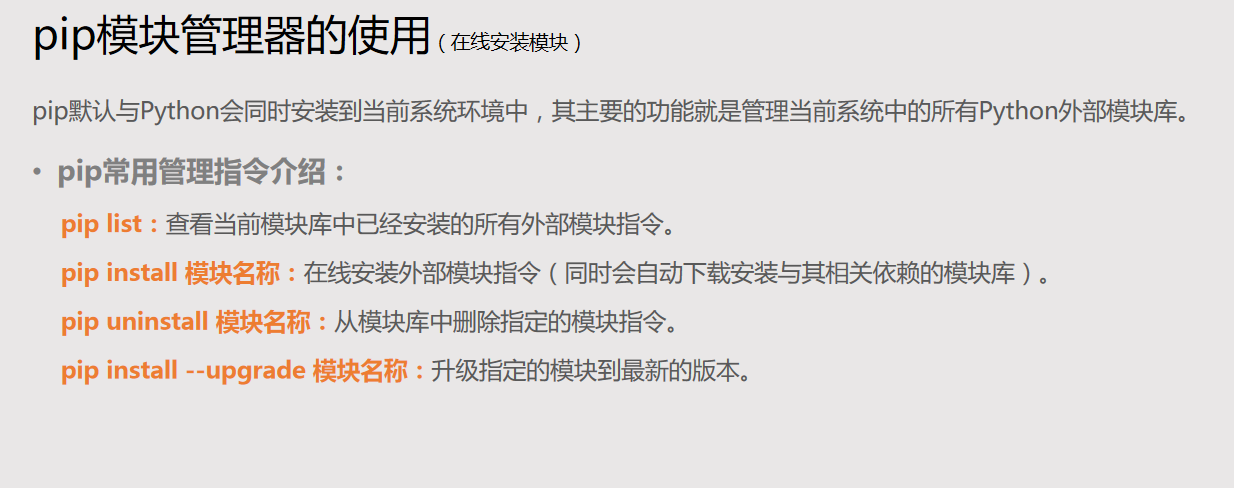
## 2.6.pyd 是 python 的动态链接库文件，在这里我们先不做过多的介绍。

## 2.7.py、pyc、pyo 和 pyd 之间的区别

py是源文件，pyc是源文件编译后的字节码文件，pyo是源文件优化编译后的字节码文件，pyd是其他语言写的Python库

Python并非完全是解释性语言，它是有编译的，先把源码py文件编译成pyc或者pyo，然后由python的虚拟机执行，相对于py文件来说，编译成pyc和pyo本质上和py没有太大区别，只是对于这个模块的加载速度提高了，并没有提高代码的执行速度。

## 2.8.python插件库管理工具pip（easyinstall）

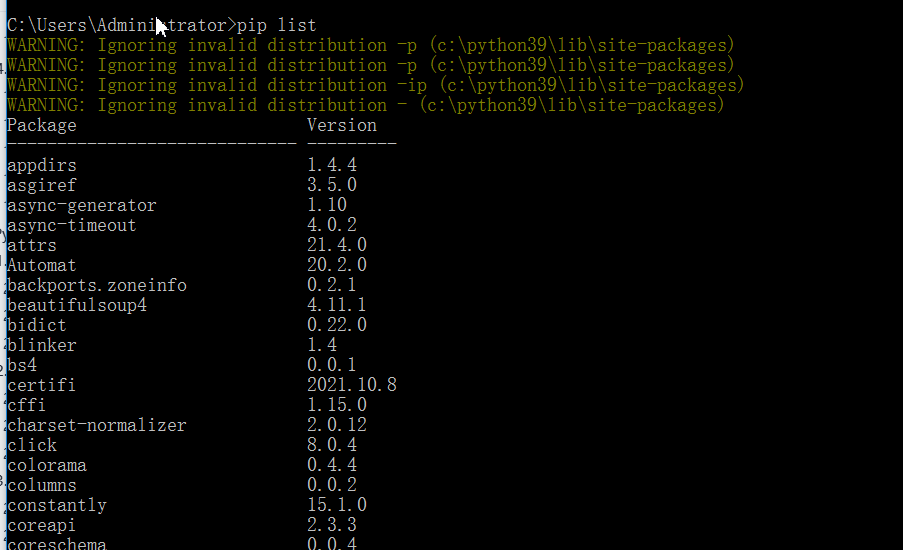


pip模块下载安装的磁盘位置

Python安装路径\Lib\site-packages 文件夹中

### 2.8.1.使用pip工具查看已经安装组件库

pip list



python -m pip list

### 2.8.2.使用pip安装第三方组件库

pip install 组件库名称

例如，安装mysql的组件库pymysql

pip install pymysql

因为 pip 的服务器一般安装在国外，基于国内糟糕的网络环境，使得 pip 安装 Python 第三方模块将是一个 很痛苦 的过程。

使用国内的清华镜像站点下载资源库

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple 包名称

例如：从清华镜像站点安装pandas组件库

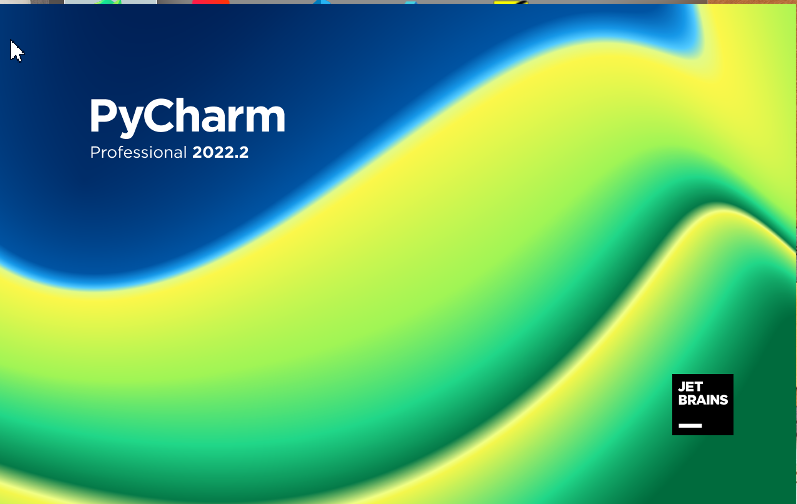
pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pandas

# python的集成开发工具

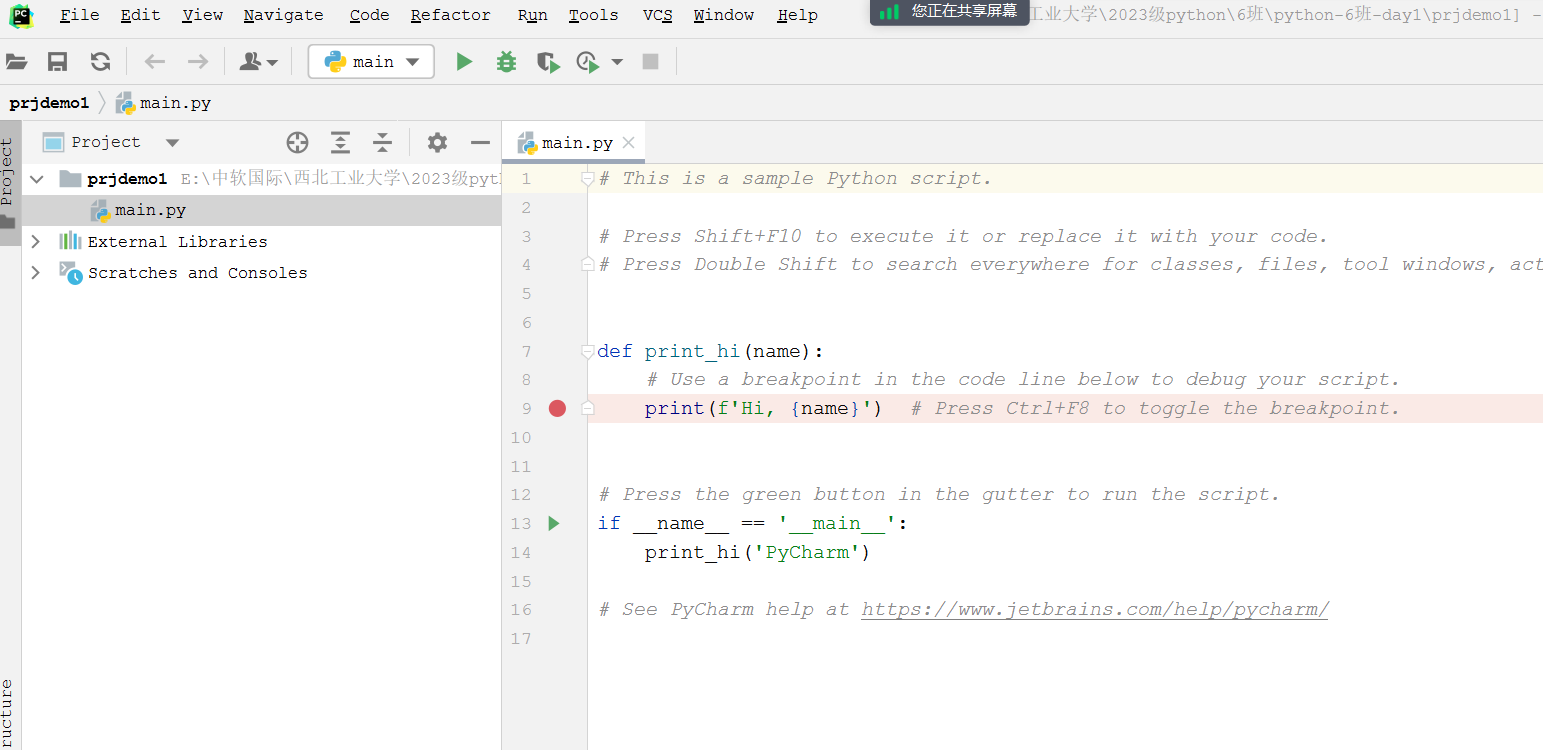
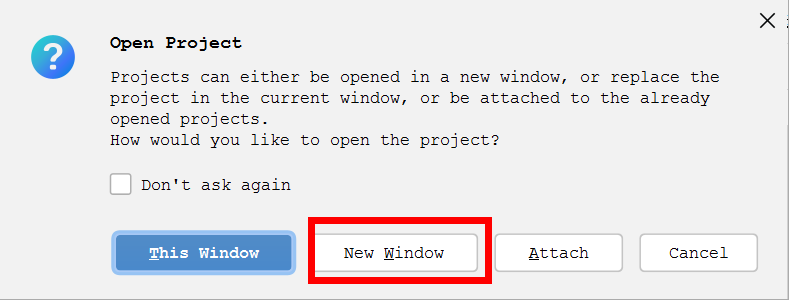
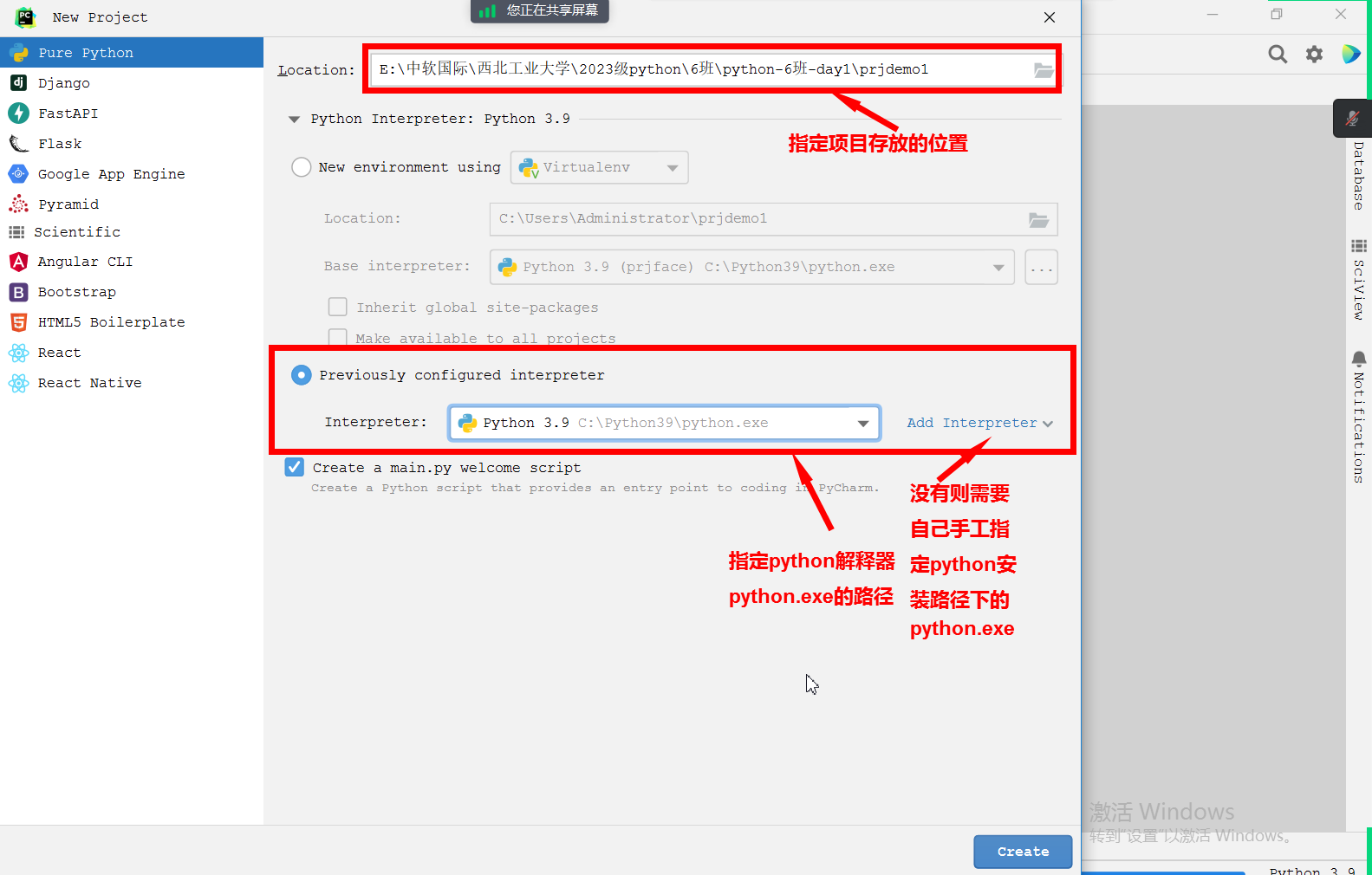
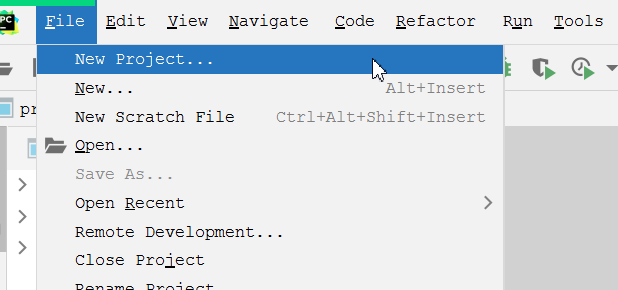
## 3.1.pycharm开发工具

是由jetbrains.com/pycharm工具，分为社区和专业两个版本

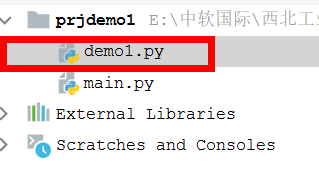
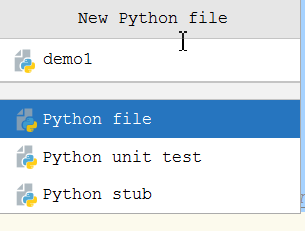
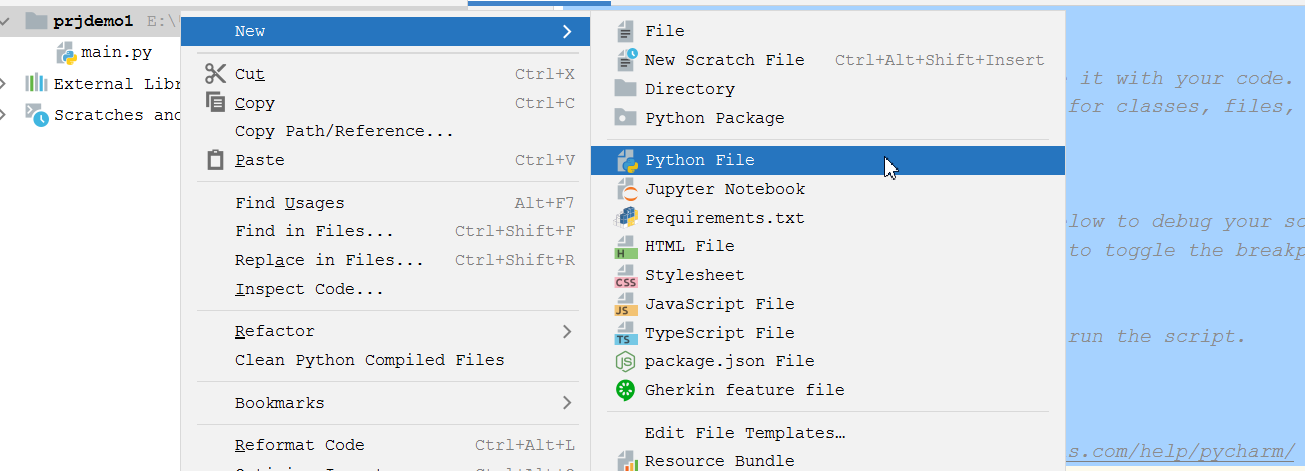
或者下载群里面的pycharmpro19\_jb51.rar文件，解压缩后，直接运行bin文件夹中的pycharm64.exe



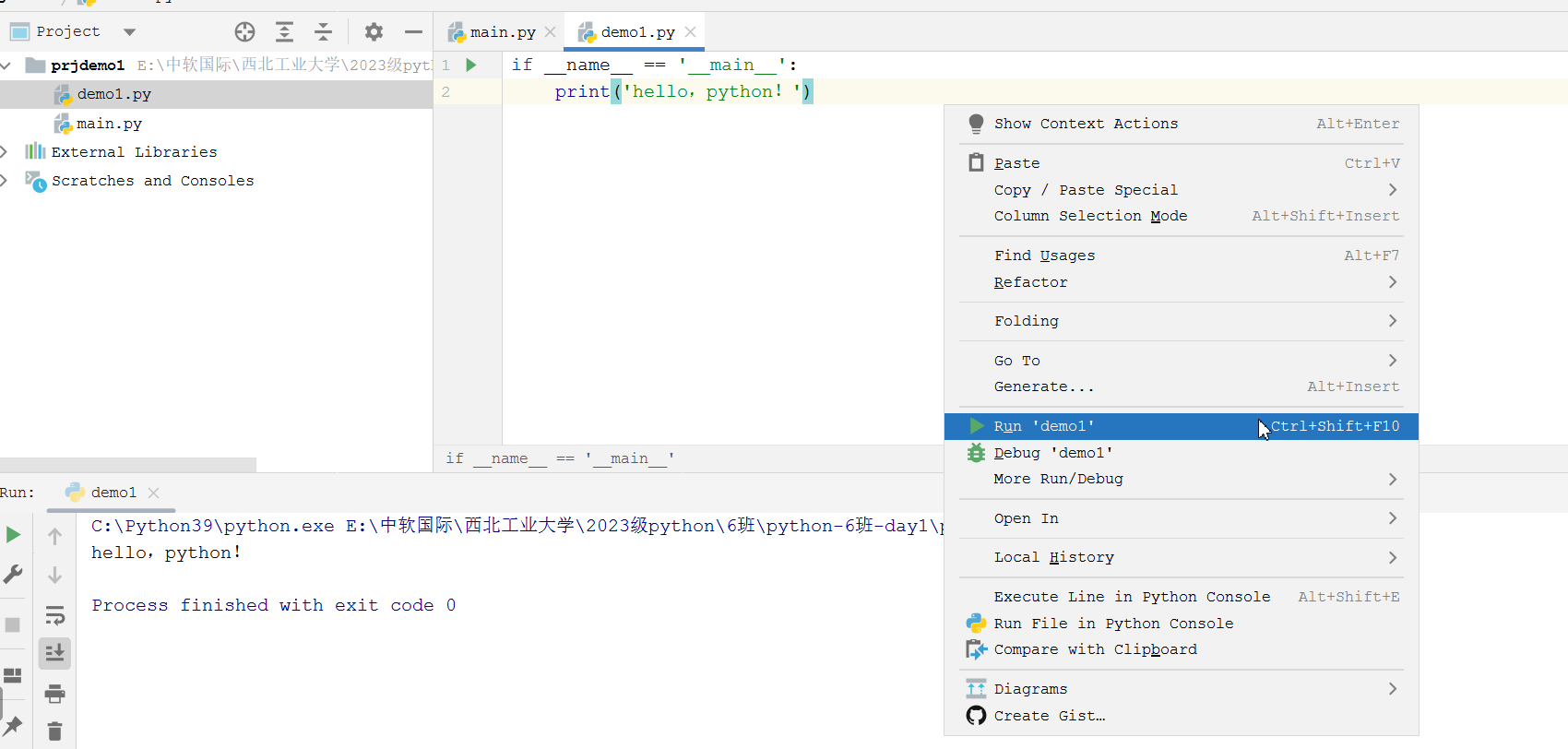
### 3.1.1.在pycharm中新建项目



### 3.1.2.新建python的源文件

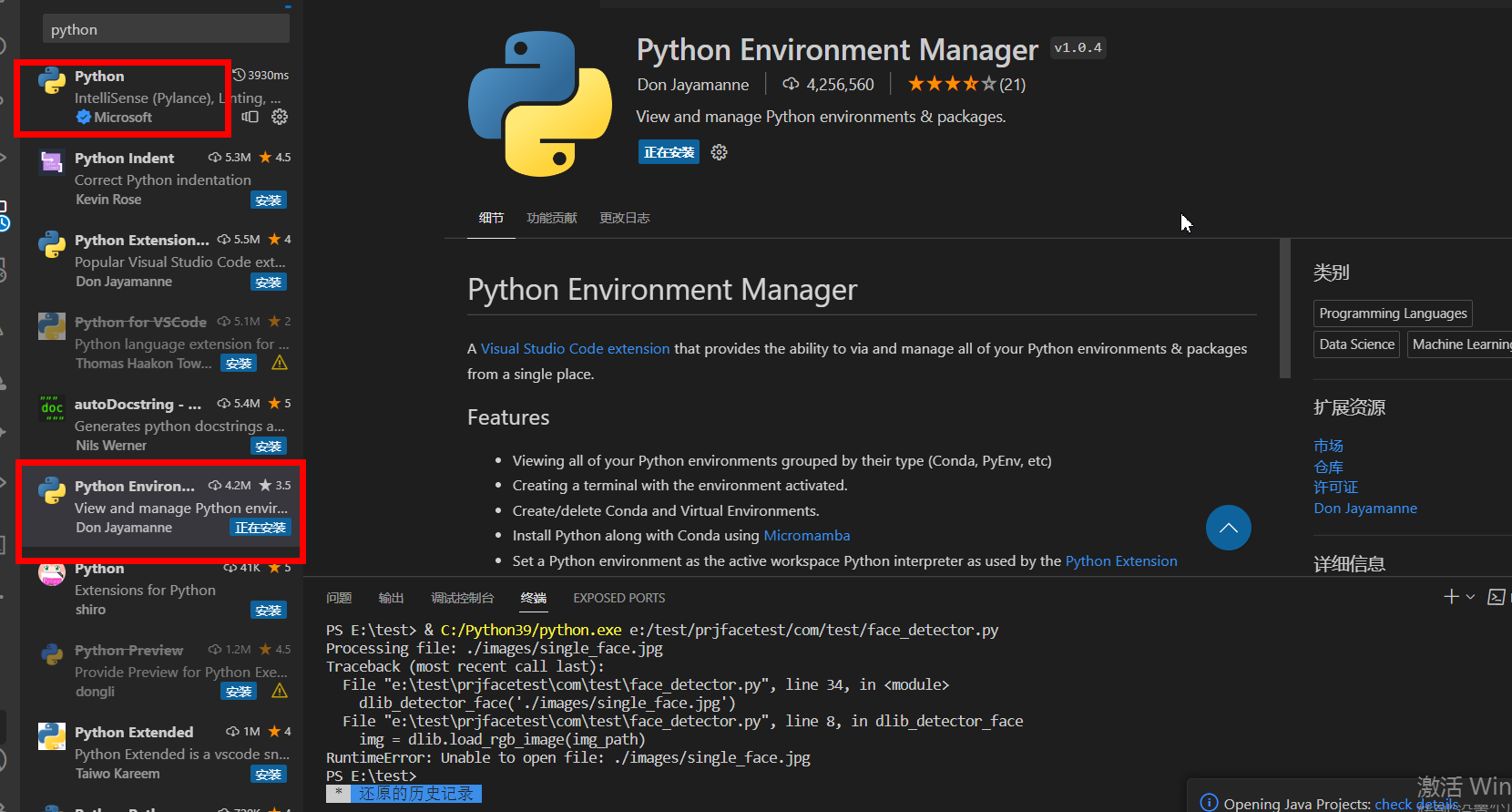
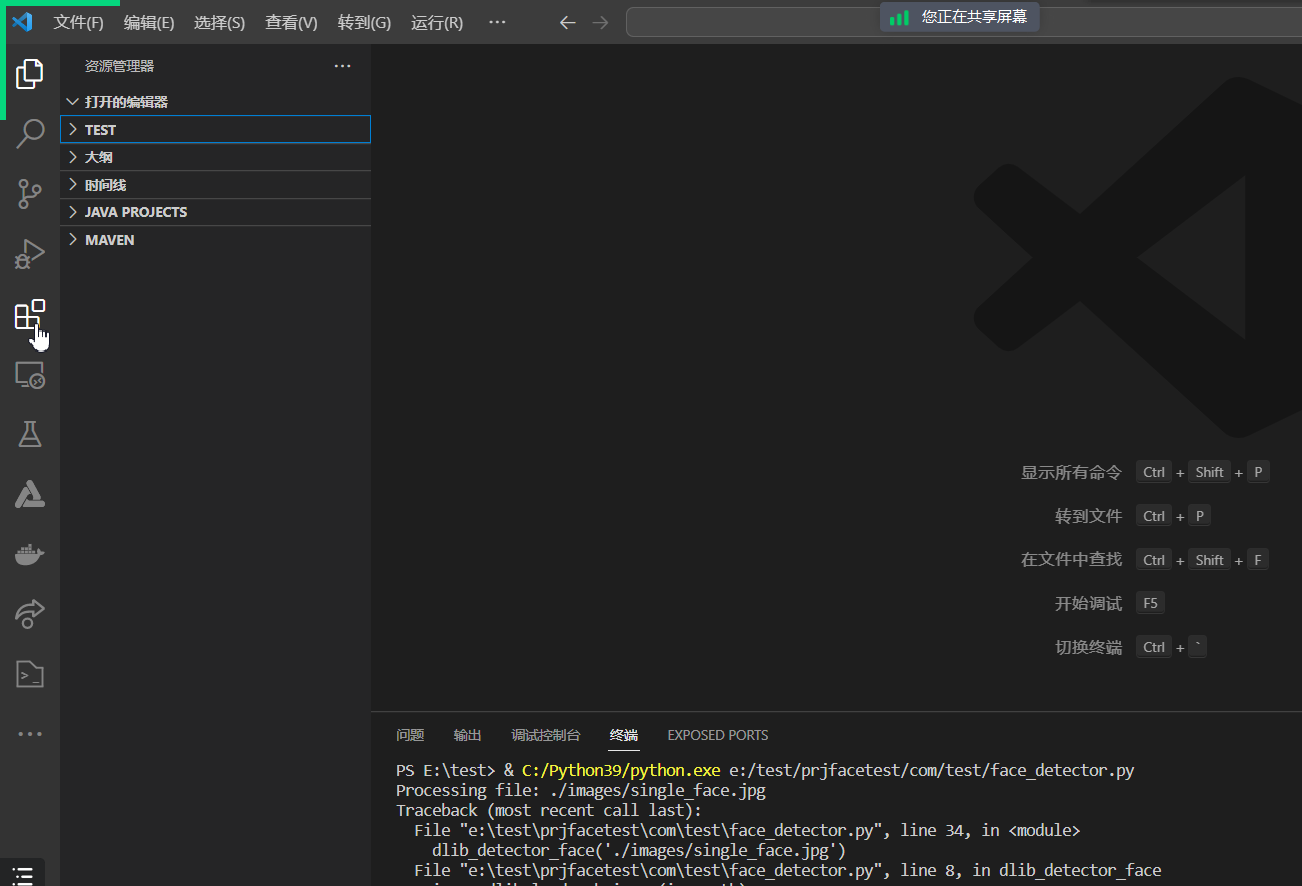


编写完成代码后，可以直接在编辑区点击右键运行，在下方观察运行的结果



大家打开的时候需要点击eval--->continue

## 3.2.使用vscoed安装python开发环境



练习：

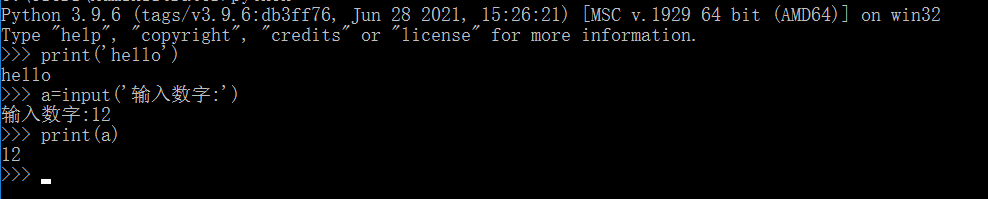
1. 配置python的开发环境，并进行验证
2. 使用pycharm建立项目并编写第一个python程序，内容为,hello python!
3. 在vscode中建立python运行环境并编写第一个python程序，输出结果为,hello vscode!

# Python语法规范与数据类型

## 4.1.常见的编程模式

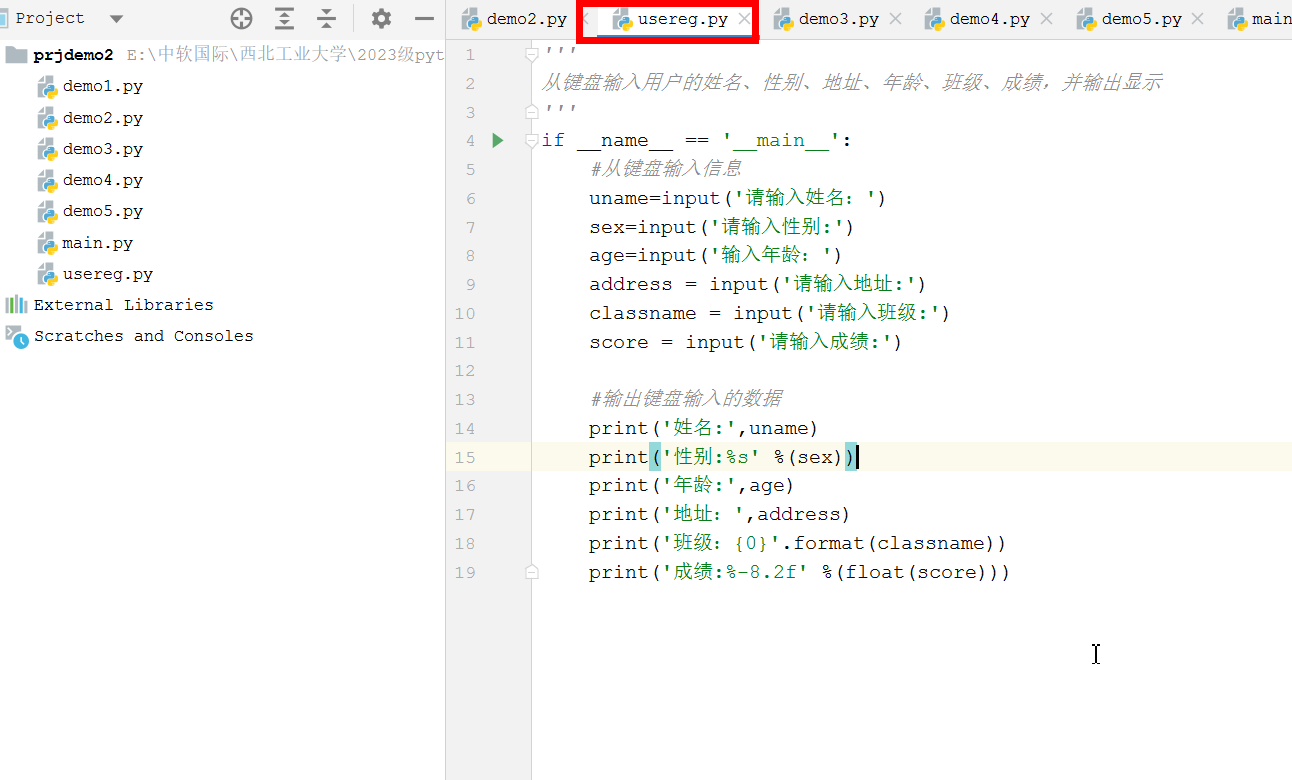
### 4.1.1.交互式编程

在命令行输入程序代码执行的方式，称为交互式编程。



### 4.1.2.脚本式编程

将程序代码写在文件中，统一由解释器执行的方式，称为脚本式编程。



## 4.2.字符编码

### 4.2.1.ascii编码

美国国家编码标准。这个编码表被称为ASCII编码，比如大写字母A的编码是65，小写字母z的编码是122。



最早只有127个字符被编码到计算机里

### 4.2.2.Unicode字符集

Python3之所以能够很好地解决中文乱码问题，在于其将所有的字符串都是用unicode进行字符编码。

Unicode把所有的语言统一到一套编码里，这样就不会有乱码了。

Unicode也在不断的发展，但最常用的是用两个字节表示一个字符（如果遇到非常生僻的字符，就需要4个字节）。现在我们见到的大多数操作系统和大多数编程语言都支持unicode。

ASCII编码是1个字节，而Unicode编码通常是2个字节

### 4.2.3.UTF-8编码

可变长的unicode字符集，UTF-8编码把一个Unicode字符根据不同的数字大小编码成1-6个字节，常用的英文字母被编码成1个字节，汉字通常是3个字节，只有很生僻的字符才会被编码成4-6个字节。

如果你要传输的文本包含大量英文字符，用UTF-8编码就能节省空间。

UTF-8编码有一个额外的好处，就是ASCII编码实际上可以被看成是UTF-8编码的一部分，所以，大量只支持ASCII编码的历史遗留软件可以在UTF-8编码下继续工作。

## 4.3.在python源文件中使用utf-8编码

在源文件的开始位置输入

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

则，解释器会以utf-8的编码执行代码，中文则不会出现乱码

示例如下：

|  |
| --- |
| ***#-\*- coding:utf-8 -\*-***if \_\_name\_\_ == '\_\_main\_\_':  print('hello,大蟒蛇python!') |

## 4.4.python标识符和保留字

### 4.4.1.标识符

用于标识变量、函数、代码段的名称。类似于给容器定义一个名称，方便进行使用。

标识符（Identifier）是指用来标识某个实体的一个符号。在不同的应用环境下有不同的含义。

在编程语言中，标识符是用户编程时使用的名字，对于变量、常量、函数、语句块 定义的名字，统称之为标识符。

### 4.4.2.标识符的命名规则

在编程语言中，标识符就是程序员自己规定的具有特定含义的词，比如类名称，属性名称，变量名等。

在 Python 里，标识符有字母、数字、下划线组成，但**不能以数字开头**。

Python 中的标识符是区分大小写的。

以下划线开头的标识符是有特殊意义的。以单下划线开头 \_foo 的代表不能直接访问的类属性，需通过类提供的接口进行访问，不能用 from xxx import \* 导入；

以双下划线开头的 \_ \_foo 代表类的私有成员；

以双下划线开头和结尾的 \_ \_foo\_ \_ 代表 Python 里特殊方法专用的标识，如\_ \_init\_ \_() 代表类的构造函数。

### 4.4.3.python中的保留字

python语言已经使用的标识名称，在系统内部用于特定的功能和含义，不能够在使用它去定义变量、函数或者语句块的名称。这样的标识名称称为保留字。

例如：

try=20 try-----错误，因为try在python作为异常处理的关键字使用了。

保留字包括关键字和未使用的保留字。

关键字则指在语言中有特定含义（如 for / if / pass / ……），成为语法中一部分的那些字。



## 4.5.语法格式规范

### 4.5.1.语法规范

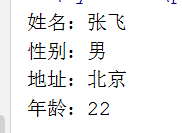
学习 Python 与其他语言最大的区别就是，Python 的代码块不使用大括号 { } 来控制区分 函数、逻辑判断和代码块 等语句块的作用域范围和控制区域。python 最具特色的就是用缩进来写模块。

缩进的空白数量是可变的（一般是1个Tab制表位），但是所有代码块语句必须包含相同的缩进空白数量，具有行缩进一致的相邻代码被认定为是1个块结构，这个必须严格执行。如下所示：



示例如下：

|  |
| --- |
| class Student:  def show(self):  print('姓名：张飞')  print('性别：男')  print('地址：北京')  print('年龄：22')  pass if \_\_name\_\_ == '\_\_main\_\_':  Student().show()  pass |



### 4.5.2.分行编写代码

有的时候一行代码太长，不便于我们书写清晰的代码结构，还可能造成代码阅读起来很不方便。因此，我们可以使用斜杠（ \）将一行的语句分为多行显示，如下所示：

|  |
| --- |
| n1=10 n2=20 n3=30 n4=n1+\  n2+\  n3 |

语句中包含 [ ], { } 或 ( ) 括号就不需要使用多行连接符，因为它们属于序列数据类型

### 4.5.3.python中的注释

注释主要用于帮助开发人员理解代码的含义。注释部分不参与程序的执行

#### 单行注释#

对单行语句进行注释，表示当前语句的含义。

|  |
| --- |
| *#-\*- coding:utf-8 -\*-* from demo2 import Student *#从demo2.py文件中引入Student类* if \_\_name\_\_ == '\_\_main\_\_':    print('hello,大蟒蛇python!')  st=Student()*#创建Student类的对象* st.show()*#使用Student对象st调用show函数* n1=10  n2=20  n3=30  n4=n1+\  n2+\  n3   print(n4) |

#### 多行注释

主要用于描述语句块或者函数的功能以及作用，作者、日期等进行描述

多行注释 (三个单引号) ’’’ 或 （三个双引号） ”””

|  |
| --- |
| ***''' 声明了一个Student类 该类主要用于描述学生的属性和业务 date:2023-06-15 author:zhhlk '''***class Student:  ***'''  show函数用于显示学生的自我介绍信息  在屏幕输出相关的信息  '''*** def show(self):  print('姓名：张飞')  print('性别：男')  print('地址：北京')  print('年龄：22')  pass if \_\_name\_\_ == '\_\_main\_\_':  Student().show()  pass |

## 4.6.等待用户输入

### 4.6.1.使用input函数实现人机交互

等待用户输入，实际上是将当前运行的程序线程挂起，暂停程序的运行。等待用户交互操作之后，在按回车或输入特定字符之后，恢复程序挂起的线程，继续执行，同时处理输入的数据。

### 4.6.2.语法格式

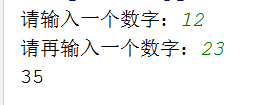
变量=input(“提示信息：”)

从键盘输入数据并将数据赋值给变量。

**注意：从键盘输入的数字都是字符串类型的值**

**示例如下：**

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#声明变量接收从简哦按输入的数据* **n1=input('请输入一个数字：')**  **n2 = input('请再输入一个数字：')**   num=int(n1)+int(n2)  print(num) |



## 4.7.print输出函数的格式

print( ) 打印输出函数是在开发中用得很多的函数，代表输出并换行。其语法结构也有很多……

### 4.7.1.格式化输出规范 1：

print(字符常量 + 字符变量) 说明：+ 加号仅用于连接两个字符串类型

print(字符常量 ，任意数据类型) 说明：, 逗号用于连接任意数据类型

print(输出的数据 , end=‘ ’) 说明：输出不换行，与下一个输出在同行显示，并使用end指定的字符连接

### 4.7.2.格式化输出规范 2：

print(格式占位符号 % 变量名称)

print(多个格式占位符号 %(变量1, 变量2, …. , 变量N))

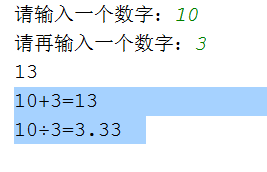
常用格式占位符号 ：%s：输出字符串类型； %d：整数类型；%f：浮点数类型（小数） 等等.

%-10s：占位10个字符，左对齐，多与的占位使用空格填充

%-8.2f：占位8个字符，左对齐，.2代表小数点保留两位

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#声明变量接收从简哦按输入的数据* n1=input('请输入一个数字：')  n2 = input('请再输入一个数字：')   num=int(n1)+int(n2)  print(num)   *#使用%s格式化输出* print("%s+%s=%d" %(n1,n2,int(num)))   *#输出浮点数，并保留2位小数点* print("%s÷%s=%-8.2f" % (n1, n2, float(int(n1)/int(n2)))) |



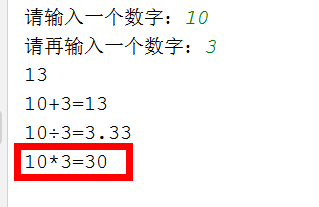
### 4.7.3.格式化输出规范 3：

print(‘{0} + {1} = {2}’.format(num1, num2, num1 + num2))

说明：{ } 方式为占位的另一种表现，但后续需要通过使用 format函数绑定变量

变量或数据的个数要与占位符的个数保持一致。

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#声明变量接收从简哦按输入的数据* n1=input('请输入一个数字：')  n2 = input('请再输入一个数字：')   num=int(n1)+int(n2)  print(num)   *#使用%s格式化输出* print("%s+%s=%d" %(n1,n2,int(num)))   *#输出浮点数，并保留2位小数点* print("%s÷%s=%-8.2f" % (n1, n2, float(int(n1)/int(n2))))   *#使用{n}格式化占位输出* **print('{0}\*{1}={2}'.format(n1,n2,int(n1)\*int(n2)))** |



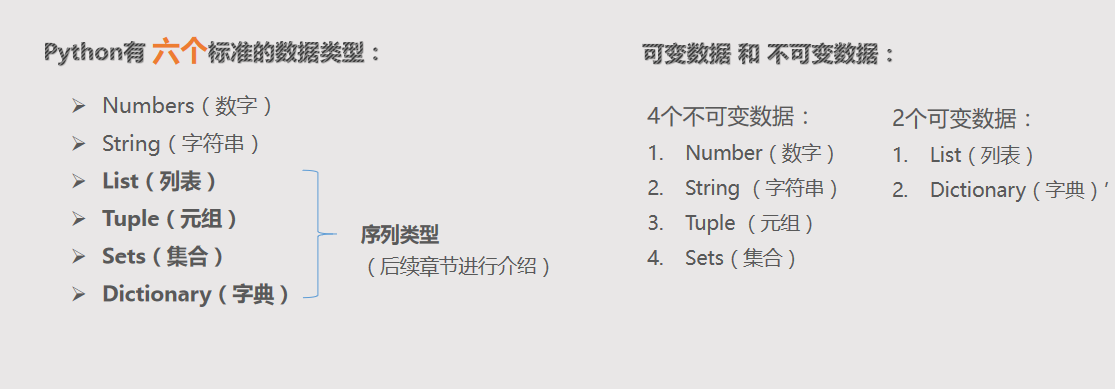
### 4.7.4.模拟用户个人信息注册，需要输入用户个人信息 姓名、性别、年龄、血型、身高、电话 信息，并输出显示。

|  |
| --- |
| *''' 模拟用户个人信息注册，需要输入用户个人信息 姓名、性别、年龄、血型、身高、电话 信息，并输出显示。  '''* if \_\_name\_\_ == '\_\_main\_\_':  uname=input('输入姓名：')  sex = input('输入性别：')  age = input('输入年龄：')  xue = input('输入血型：')  height = input('输入身高：')  tel = input('输入电话：')   print('='\*5,'个人注册信息','='\*5)  print('='\*20)  print('姓名:',uname)  print('性别:%s' %sex)  print('年龄:%d'%(int(age)))  print('血型:{0}'.format(xue))  print('身高:%-8.2f' %(float(height)))  print('电话:', tel)   print('=' \* 20)    pass |



## 4.8.python中的数据类型

### 4.8.1.标准数据类型



### 4.8.2.Python 数字类型

数字数据类型用于存储数值。

他们是不可改变的数据类型，这意味着改变数字数据类型会分配一个新的对象。

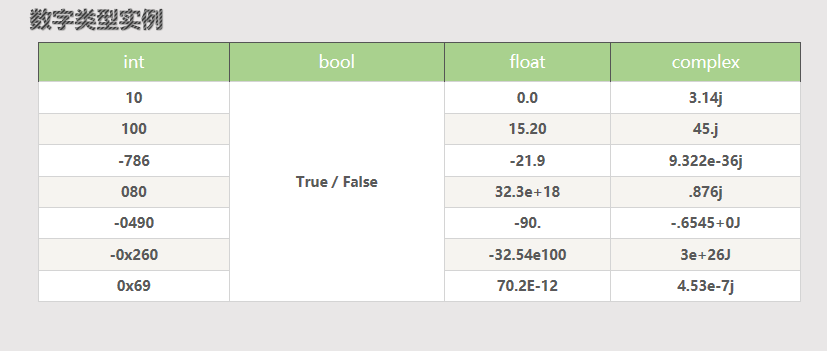
Python支持 四种 不同的数字类型：

int（有符号整型，通常被称为是整型或长整数，是正或负整数，不带小数点）

float（浮点型，由整数部分与小数部分组成，浮点型也可以使用科学计数法表示（2.5e2 = 2.5 x 102 = 250））

bool（布尔类型，由 True 或 False，即 真 1 或 假 0 python2.7中。常用于逻辑判断使用）

complex（复数，由实数部分和虚数部分构成，可以用a + bj,或者complex(a,b)表示， 复数的实部a和虚部b都是浮点型）



## 4.9.python字符串类型

使用单引号或者双引号括起来的字符都称为字符串。

S1=”abcd”

S1=’abcd’

### 4.9.1.字符串的访问方式

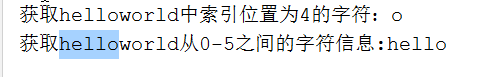
Python的字串列表有 2种 取值顺序:

从左到右下标索引默认0开始的，最大范围是(字符串长度-1)

从右到左下标索引默认-1开始的，最大范围是字符串开头

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  ss1="helloworld"  print('获取%s中索引位置为4的字符：%s' %(ss1,ss1[4]))   *#获取从0-5之间的字符信息:* s2=ss1[0:5]  print('获取%s从0-5之间的字符信息:%s' %(ss1,s2))  pass |



### 4.9.2.字符串转义



例如：

s3='hello\'world\''  
print(s3)

输出的结果：

hello'world'

### 4.10.字符串中常用的函数

#### 1.字符串运算符 in 和 not in

我们经常可以判断字符串中是否包含某个指定的字符或子字符串，这个时候就需要使用 in 或 not in 来进行操作。

语法格式：

字符串2 in 字符串1 返回值为bool类型，true表示存在，false表示不存在

字符串2是否存在于字符串1中

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  s='hello,python'   if 'python' in s:  print('python在%s中！' %(s))  else:  print('python不在%s中！' %(s))   pass |



#### 2.count函数

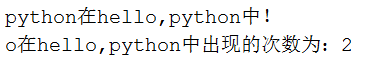
语法：

字符串1.count(字符串2)

字符串2在字符串1中出现的次数，如果不存在返回-1

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  s='hello,python'   if 'python' in s:  print('python在%s中！' %(s))  else:  print('python不在%s中！' %(s))   *#统计o在hellopython中出现的次数* n=s.count('o')  print('o在%s中出现的次数为：%s' %(s,n))   pass |



#### find函数

str. find(str, beg=0, end=len(string))

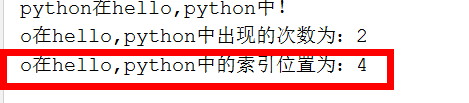
检测字符串中是否包含子字符串 str ，如果指定 beg（开始） 和 end（结束） 范围，则检查是否包含在指定范围内，如果包含子字符串返回开始的索引值，否则返回-1。

字符串1.find(字符串2)

在字符串1中查找字符串2的索引位置

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  s='hello,python'   if 'python' in s:  print('python在%s中！' %(s))  else:  print('python不在%s中！' %(s))   *#统计o在hellopython中出现的次数* n=s.count('o')  print('o在%s中出现的次数为：%s' %(s,n))    ***#在s中查找o第一次出现的索引位置* oindex=s.find('o')  print('o在%s中的索引位置为：%s' %(s,oindex))**   pass |



#### 函数名称：replace

str. replace(old, new, [ , max])

把字符串中的 old（旧字符串） 替换成 new(新字符串)，如果指定第三个参数max，则替换不超过 max 次。

参数说明：

old -- 将被替换的子字符串。

new -- 新字符串，用于替换old子字符串。

max -- 可选字符串, 替换不超过 max 次。

函数返回值：

返回字符串中的 old（旧字符串） 替换成 new(新字符串)后生成的新字符串，如果指定第三个参数max，则替换不超过 max 次。

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  s1='hellx,pythxn'  *#将s1中的x替换为o* s2=s1.replace('x','o')  print('%s替换后的字符串为：%s' %(s1,s2))  pass |



#### 函数名称：split

str. split(split [, num=string.count(str)])

通过指定分隔符对字符串进行切片，如果参数num 有指定值，则仅分隔 num 个子字符串

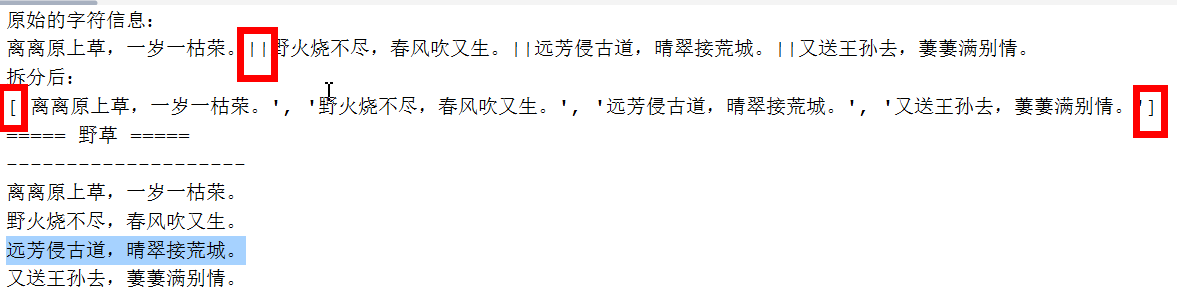
str -- 分隔符，默认为所有的空字符，包括空格、换行(\n)、制表符(\t)等。

num -- 分割次数。

返回值为分割后的字符串列表。

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  ts='离离原上草，一岁一枯荣。||野火烧不尽，春风吹又生。||远芳侵古道，晴翠接荒城。||又送王孙去，萋萋满别情。'  print('原始的字符信息：')  print(ts)   *#根据||将以上的文本差分为列表集合（数组）* tss=ts.split("||")*#根据||拆分字符串* print("拆分后：")  print(tss)  print('='\*5,'野草','='\*5)  print('-'\*20)  for ts1 in tss:*#从列表中获取元素的值并传递给ts1变量，在循环内部输出显示* print(ts1)  print('-' \* 20)  pass |



字符串处理函数的汇总分类

字符串大小写转换

lower() / upper() /swapcase() / title()

字符串搜索、替换

find() /count() / replace / strip() / lstrip() /rstrip()

字符串分割、组合

split() / join()

字符串编码、解码

decode()/encode()

字符串测试

isalpha() / isdigit() / isspace() / islower() / isupper() / istitle()

### 4.11.作业

1. 使用input实现学生注册信息，信息包括姓名、性别、地址、生日、电话、邮箱
2. 使用print的%s和{}形式输出注册的信息
3. 使用字符串函数实现以下功能
4. 从键盘输入email邮箱，并判断email邮箱格式是否正确(@和.必须出现)
5. 从下面文字中查找字符串’新加坡’出现的次数

“研究人员称，他们对中国、越南、韩国和新加坡等亚太地区9个国家的45个城市进行了调查，通过抽样等手段测算了这些城市房价和租金的中位数。研究提到，新加坡的房价在2022年上涨了8%。

分析认为，当地房价上涨的主要因素有海外移民的大量涌入、大量年轻人不再与父母合住等。另外，建筑材料供应链受到冲击导致新增住房减少，也是房价上涨的重要推手。除了房价外，研究显示新加坡的租房价格也是亚太地区最贵的。

1. 将下面的字符串按照指定的分隔符‘￥’进行拆分并逐行显示

strval=’寒雨连江夜入吴，平明送客楚山孤。￥洛阳亲友如相问，一片冰心在玉壶。’

1. 将下面文档中的先生字符替换为‘老师’

有个人要请一位先生教育自己的孩子。有一天，一个人来应聘，主人说：“我们家贫穷，可能有很多对先生失礼的地方，您看怎么样啊？”这位先生说：“不用这么客气，我本来就没什么计较的。”主人说：“吃蔬菜，可以吗？”答：“可以。”主人说：“家里没有奴仆，凡是打扫庭院，开门关门，有劳先生做，可以吗？”答：“行。”主人说：“有时家里人，妇女孩子想买零星杂物，委屈先生去跑一趟，可以吗？”答：“可以 。”主人说：“如果是这样，就太好了！”之后，先生也说：“我也有一句话，希望主人不要惊讶。”主人问他什么话？先生说：“我自愧小时候没有好好学习！”主人说：“何必说这样谦虚的话。”先生说：“不敢欺骗你，我其实一字不识呀！”

# 五.序列类型

## 5.1.序列对象的定义

“序列”是程序设计中经常用到的数据存储方式。在其他程序设计语言中，“序列”通常被称为”数组“，用于存储相关数据项的数据结构。几乎每一种程序设计语言都提供了“序列”数据结构，如C和Basic中的一维、多维数组等。

## 5.2.序列与数组的区别：

数组是提供了能够存放同一数据类型且连续的内存空间。有长度限制。

序列虽然是连续的存储空间，但可以存放不同数据类型，也可以理解为更加 “高级的数组”。

## 5.3.Python中常用的序列对象

列表 List （可变数据类型）

元组 Tuple（不可变数据类型）

集合 Sets（可变数据类型）

字典 Dictionary（可变数据类型）

字符串 String（不可变）

range( ) --访问序列元素的函数

## 5.4.列表类型list(有序集合)

### 5.4.1.用于存放一组数据的容器，并且指定每个数据元素的存放位置

放置在列表集合中的数据称为元素，每个元素都可以使用下标索引位置获取器元素的值。

### 5.4.2.列表集合的定义语法

列表对象=[元素1，……,元素n]

### 5.4.3.管理列表集合的常用方法

#### 1.获取列表元素的值

变量=列表对象[索引位置]

#### 2.更改列表元素的值

列表对象[索引位置]=值

#### 3.向列表集合中添加数据

列表对象.append(元素值)

#### 从列表对象中删除元素

列表对象.remove(元素值)

#### 获取列表集合的长度(元素个数)

size=len(列表对象)

#### 6.统计指定元素在列表集合中的个数count

n=列表对象.count(元素值)

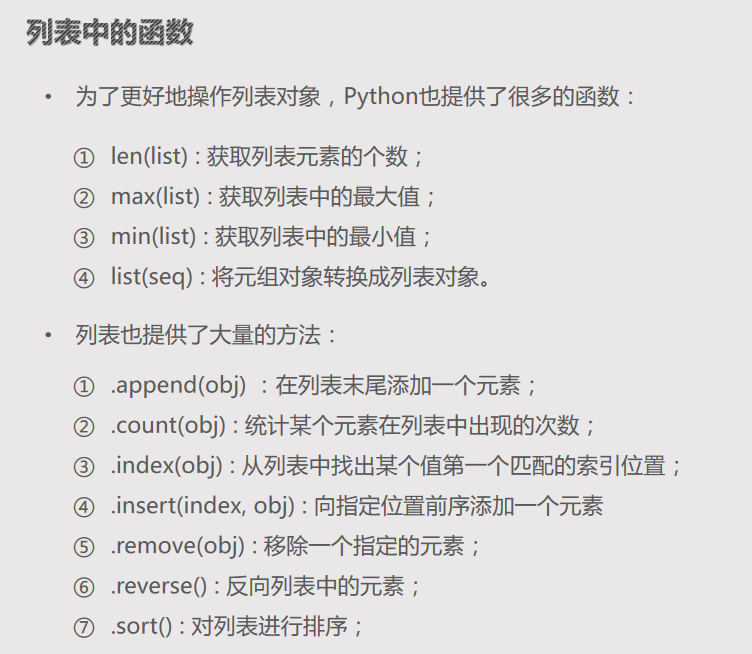
n表示查找到的元素值个数

没有找到返回值为0

#### 7.查找元素在当前集合中的索引位置index

索引位置=列表对象.index(元素值)

如果在列表中没有找到元素值，则抛出异常消息



### 5.4.4.列表集合的示例

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#定义列表集合对象* slist=['庄周','蔡文姬','大乔','牛魔','混子瑶']  size=len(slist)  print('元素的个数为：',size)   *#获取索引位置为3的元素* hs=slist[3]  print('索引位置为3的元素值为：',hs)   *#获取大乔所在的索引位置* dindex=slist.index('大乔')  print('大乔所在的索引位置为：',dindex)  print('修改前的值为:',slist[dindex])   *#修改大乔为刘禅* slist[dindex]='刘禅'*#根据索引位置修改元素的值* print('修改后的值为:',slist[dindex])   print('获取列表集合中的所有元素值：')  for i,sname in enumerate(slist):*#使用for循环获取列表集合的索引位置和元素值* print(i,'.',sname)*#i表示索引位置，sname表示slist集合元素的值* print('='\*40)   *#向列表集合中添加新的元素值* slist.append('太乙真人')  slist.append('百里守约')  slist.append('百里玄策')   print('获取添加以后的列表集合中的所有元素值：')  for i, sname in enumerate(slist): *# 使用for循环获取列表集合的索引位置和元素值* print(i, '.', sname) *# i表示索引位置，sname表示slist集合元素的值* print('=' \* 40)   *#从键盘输入一个名称，判断是否在列表中存在* seacherName=input('请输入要查找的英雄名称:')   n=slist.count(seacherName)  if n>0:  print('%s在列表中存在！' %(seacherName))  else:  print('不存在于列表中！')   *#从列表中删除元素* slist.remove('百里守约')  print('获取删除以后的列表集合中的所有元素值：')  for i, sname in enumerate(slist): *# 使用for循环获取列表集合的索引位置和元素值* print(i, '.', sname) *# i表示索引位置，sname表示slist集合元素的值* print('=' \* 40)  pass |

运行结果如下：

|  |
| --- |
| 元素的个数为： 5  索引位置为3的元素值为： 牛魔  大乔所在的索引位置为： 2  修改前的值为: 大乔  修改后的值为: 刘禅  获取列表集合中的所有元素值：  0 . 庄周  1 . 蔡文姬  2 . 刘禅  3 . 牛魔  4 . 混子瑶  ========================================  获取添加以后的列表集合中的所有元素值：  0 . 庄周  1 . 蔡文姬  2 . 刘禅  3 . 牛魔  4 . 混子瑶  5 . 太乙真人  6 . 百里守约  7 . 百里玄策  ========================================  请输入要查找的英雄名称:刘禅  刘禅在列表中存在！  获取删除以后的列表集合中的所有元素值：  0 . 庄周  1 . 蔡文姬  2 . 刘禅  3 . 牛魔  4 . 混子瑶  5 . 太乙真人  6 . 百里玄策  ======================================== |

### 5.4.5.List[] 与 list[:] 的区别

“ [ ] ”标识，是典型的引用操作，传址操作

List\_1 = [ 1, 2, 3, 4]

List\_2 = list\_1

“ [：] ”标识，是典型的赋值操作，传值操作

List\_1 = [ 1, 2, 3, 4]

List\_2 = list\_1[：]



## 5.5.元组类型(tuple)

### 5.5.1.元组类型中存放的数据元素不能改变

元组中的元素值不能改变，可以理解为不可变列表集合

### 5.5.2.元组类型对象的定义

元组对象=(元素1,……,元素n)

除了元素值不能更改外，其他操作与list列表一致。

### 5.5.3.示例

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#声明元组对象* mytup=('后羿','貂蝉','妲己','安琪拉','王昭君')   print('获取索引位置为1的元素值：',mytup[1])   print('获取所有的元素值:')  for i,item in enumerate(mytup):  print(i,'.',item)   *#mytup[0]='露拉' #报错，元组的元素值不能改变* pass |

## 5.6.字典（Dictionary）类型

### 5.6.1.使用键值对方式存放元素

表示方式：键:值 相当于:键=值

字典当中的元素是通过键来存取的，而不是通过偏移存取。

字典用“{ }”标识。字典由索引(key)和它对应的值value组成，是一个典型的“k-v值 ”数据结构。

### 5.6.2.字典对象的创建方式

字典对象={‘键1’:值1,……,‘键n’:值n}

字典对象名称 = { }

字典对象名称 = { key1 : value1, key2 : value2, …. , keyN : valueN }

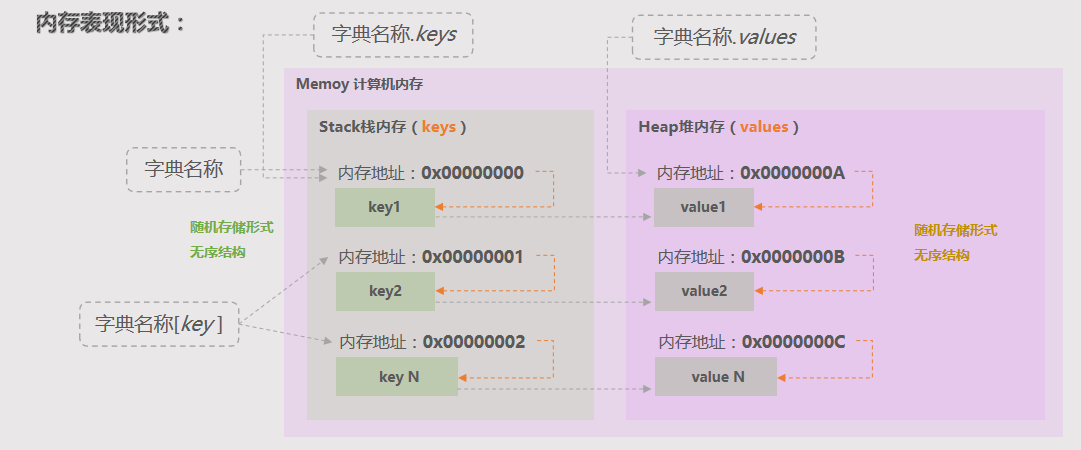
例如：

studict={‘sid’:1,’sname’:’张飞’,’sex’:’男’,’address’:’北京’}

### 5.6.3.字典（Dictionary）k-v 值在内存中的表现形式

k-v结构一般情况下在操作访问的时候都会使用key索引进行每个元素的读取操作。

由于key索引键会被频繁访问，因此索引键key存放在Stack栈内存中，而value值则存储在Heap堆内存中。



### 5.6.4.字典类型常用的方法

#### 1.获取键集合

键变量=字典对象.keys()

#### 2.获取值集合：

值列表对象=字典对象.values()

#### 如何访问字典

字典对象名称[ key ] #访问key对应的value值

字典对象名称.keys() #访问当前字典所有key索引键

字典对象名称.values() #访问当前字典所有valeus值

### 5.6.5.示例如下：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#创建字典对象* studict={'sid':1,'sname':'张飞','sex':'男','address':'北京','classname':'class1班'}   *#获取键和值的列表集合* keys=studict.keys()  values=studict.values()  '''  #获取所有的键名和对应的值  for key,value in zip(keys,values):  print(key,":",value)  '''  for key in keys:  print(key,":",studict[key])   print('学生信息为：')  print('编号:',studict['sid'])  print('姓名:', studict['sname'])  print('性别:', studict['sex'])  print('地址:', studict['address'])  print('班级:', studict['classname'])  pass |

结果如下：

|  |
| --- |
|  |

#### 复杂的列表字典

字典列表集合对象=[

{..........},

{..........},

{..........},

....

{..........}

]

例如：

Stulist=[

{'sid':1,'sname':'张飞','sex':'男','address':'北京','classname':'class1班'},

{'sid':2,'sname':'马超','sex':'男','address':'北京','classname':'class1班'},

{'sid':3,'sname':'关羽','sex':'男','address':'北京','classname':'class1班'},

{'sid':4,'sname':'赵云','sex':'男','address':'北京','classname':'class1班'}

]

或者

stus={

‘liststudent’:[

{'sid':1,'sname':'张飞','sex':'男','address':'北京','classname':'class1班'},

{'sid':2,'sname':'马超','sex':'男','address':'北京','classname':'class1班'},

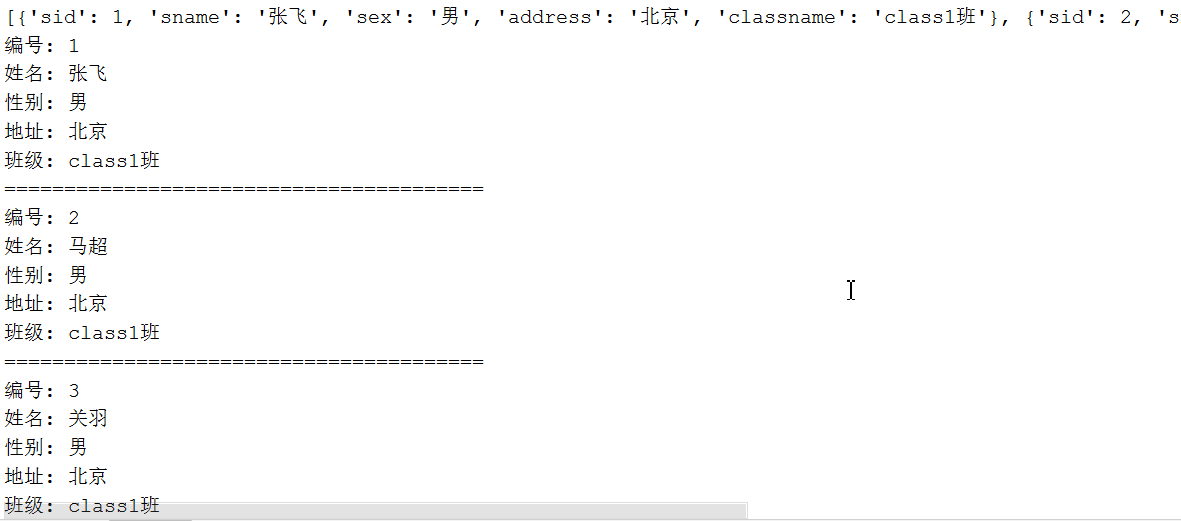
{'sid':3,'sname':'关羽','sex':'男','address':'北京','classname':'class1班'},

{'sid':4,'sname':'赵云','sex':'男','address':'北京','classname':'class1班'}

]

}

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#复杂的字典列表* stulist = [  {'sid': 1, 'sname': '张飞', 'sex': '男', 'address': '北京', 'classname': 'class1班'},  {'sid': 2, 'sname': '马超', 'sex': '男', 'address': '北京', 'classname': 'class1班'},  {'sid': 3, 'sname': '关羽', 'sex': '男', 'address': '北京', 'classname': 'class1班'},  {'sid': 4, 'sname': '赵云', 'sex': '男', 'address': '北京', 'classname': 'class1班'}  ]   print(stulist)   for stu in stulist:  print('编号:', stu['sid'])  print('姓名:', stu['sname'])  print('性别:', stu['sex'])  print('地址:', stu['address'])  print('班级:', stu['classname'])  print('='\*40) |



## 5.7.列表集合和字典的综合示例

构建一个学生的列表集合，用于添加学生字典对象，学生字典对象的数据项包括：

编号、姓名、性别、地址、年龄、班级

分别定义不同的函数对列表中的数据进行操作

添加学生addStu()

列表显示：showAll()

更新学生:update()

删除学生：delById()

查找学生:findStuBySname()

def 函数名(参数)：

代码

pass

调用函数

函数名(参数)

|  |
| --- |
| '''  添加学生信息到列表集合中的函数  编号、姓名、性别、地址、年龄、班级  传递列表对象  '''  def addStu(slist:list):  stu={}#定义学生字典  stu['sid']=input('输入编号：')  stu['sname'] = input('输入姓名：')  stu['sex'] = input('输入性别：')  stu['address'] = input('输入地址：')  stu['age'] = input('输入年龄：')  stu['classname'] = input('输入班级：')  slist.append(stu)  print('添加完成!')  pass  '''  显示所有的学生信息  showAll  '''  def showAll(slist:list):  if slist==None or len(slist)==0:#判断列表集合是否是空值或者列表集合中是否存在元素  print('没有任何学生信息存在！')  return  print('编号\t\t 姓名\t\t 性别\t\t 地址\t\t 年龄\t\t 班级')  print('-'\*80)  for stu in slist:  print('%s\t\t %s\t\t %s\t\t %s\t\t %s\t\t %s'  %(stu['sid'],stu['sname'],stu['sex'],stu['address'],stu['age'],stu['classname']))  print('-' \* 80)  pass  '''  根据学生的编号更新学生的信息update()  需要在列表中查找该编号对应的学生字典在列表中的索引位置  '''  def update(slist:list):  if slist==None or len(slist)==0:  print('没有任何学生信息存在！')  return  sindex=-1;#编号对应的学生字典在列表中的索引位置  sid=input('请输入要修改的学生编号:')  for i,stu in enumerate(slist):  if stu['sid']==sid:  sindex=i#将学生字典的索引位置赋值给sindex变量  break  if sindex!=-1:  print('要修改的学生信息为：')  stu=slist[sindex]  print('%s\t\t %s\t\t %s\t\t %s\t\t %s\t\t %s'  % (stu['sid'], stu['sname'], stu['sex'], stu['address'], stu['age'], stu['classname']))  print('输入新的学生信息：')  stu['sname'] = input('输入姓名：')  stu['sex'] = input('输入性别：')  stu['address'] = input('输入地址：')  stu['age'] = input('输入年龄：')  stu['classname'] = input('输入班级：')  print('更新成功！')  else:  print('没有该编号的学生信息存在！')  pass  '''  删除学生：delById()  根据学生编号删除学生字典元素  需要先查找学生编号对应的学生字典在列表中的索引位置  再删除该编号对应的学生字典元素  '''  def delById(slist:list):  if slist==None or len(slist)==0:  print('没有任何学生信息存在！')  return  sindex=-1;#编号对应的学生字典在列表中的索引位置  sid=input('请输入要删除的学生编号:')  for i,stu in enumerate(slist):  if stu['sid']==sid:  sindex=i#将学生字典的索引位置赋值给sindex变量  break  if sindex!=-1:  print('要删除的学生信息为：')  stu=slist[sindex]  print('%s\t\t %s\t\t %s\t\t %s\t\t %s\t\t %s'  % (stu['sid'], stu['sname'], stu['sex'], stu['address'], stu['age'], stu['classname']))  slist.remove(stu)#删除学生字典对象元素  print('删除成功！')  else:  print('没有该编号的学生信息存在！')  pass  '''  查找学生:findStuBySname()  在循环中显示找到的所有姓名相同的学生信息  '''  def findStuBySname(slist: list):  if slist==None or len(slist)==0:  print('没有任何学生信息存在！')  return  sindex=-1;#编号对应的学生字典在列表中的索引位置  searcherName=input('请输入要查找的学生姓名:')  for i,stu in enumerate(slist):  if stu['sname']==searcherName:  sindex=i#将学生字典的索引位置赋值给sindex变量  print('%s\t\t %s\t\t %s\t\t %s\t\t %s\t\t %s'  % (stu['sid'], stu['sname'], stu['sex'], stu['address'], stu['age'], stu['classname']))  if sindex==-1:  print('没有找到该姓名的学生信息！')  pass  if \_\_name\_\_ == '\_\_main\_\_':  stulist=[]  while True:  addStu(stulist)#调用添加学生的函数  choose=input('是否继续输入1--继续，0--退出:')  if choose=='1':  continue  else:  break  #调用显示所有学生信息的函数  showAll(stulist)  #更新数据  while True:  update(stulist)#调用修改学生的函数  choose=input('是否继续输入1--继续，0--退出:')  if choose=='1':  continue  else:  break  showAll(stulist)  # 删除数据  while True:  delById(stulist) # 调用删除学生的函数  choose = input('是否继续输入1--继续，0--退出:')  if choose == '1':  continue  else:  break  #查找数据  while True:  findStuBySname(stulist) # 调用查找学生的函数  choose = input('是否继续输入1--继续，0--退出:')  if choose == '1':  continue  else:  break  showAll(stulist)  pass |

运行效果：

|  |
| --- |
| 输入编号：1  输入姓名：庄周  输入性别：男  输入地址：岐山  输入年龄：22  输入班级：class1  添加完成!  是否继续输入1--继续，0--退出:1  输入编号：2  输入姓名：蔡文姬  输入性别：女  输入地址：洛阳  输入年龄：22  输入班级：class1  添加完成!  是否继续输入1--继续，0--退出:1  输入编号：3  输入姓名：庄周  输入性别：女  输入地址：洛阳  输入年龄：23  输入班级：class3  添加完成!  是否继续输入1--继续，0--退出:0  编号 姓名 性别 地址 年龄 班级  --------------------------------------------------------------------------------  1 庄周 男 岐山 22 class1  2 蔡文姬 女 洛阳 22 class1  3 庄周 女 洛阳 23 class3  --------------------------------------------------------------------------------  请输入要修改的学生编号:4  没有该编号的学生信息存在！  是否继续输入1--继续，0--退出:0  编号 姓名 性别 地址 年龄 班级  --------------------------------------------------------------------------------  1 庄周 男 岐山 22 class1  2 蔡文姬 女 洛阳 22 class1  3 庄周 女 洛阳 23 class3  --------------------------------------------------------------------------------  请输入要删除的学生编号:8  没有该编号的学生信息存在！  是否继续输入1--继续，0--退出:0  请输入要查找的学生姓名:庄周  1 庄周 男 岐山 22 class1  3 庄周 女 洛阳 23 class3  是否继续输入1--继续，0--退出:1  请输入要查找的学生姓名:蔡文姬  2 蔡文姬 女 洛阳 22 class1  是否继续输入1--继续，0--退出:1  请输入要查找的学生姓名:貂蝉  没有找到该姓名的学生信息！  是否继续输入1--继续，0--退出:0  编号 姓名 性别 地址 年龄 班级  --------------------------------------------------------------------------------  1 庄周 男 岐山 22 class1  2 蔡文姬 女 洛阳 22 class1  3 庄周 女 洛阳 23 class3  -------------------------------------------------------------------------------- |

## 5.8.集合 Set

### 5.8.1.Set集合是一个无序集合

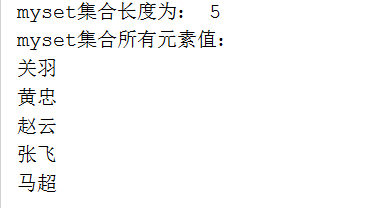
集合中的元素不能有重复，元素没有索引位置存在。不能通过索引位置取值或者设置值。

### 5.8.2.Set集合的定义语法

Set对象={元素1,……,元素n}

### 5.8.3.示例

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  *#创建Set对象* myset={'马超','关羽','赵云','张飞','黄忠','关羽'}   print('myset集合长度为：',len(myset))  print('myset集合所有元素值：')  for ms in myset:  print(ms)  pass |



## 5.9.数据类型转换

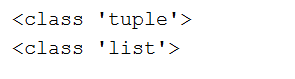
当需要对数据内置的类型进行转换，就需要了解Python数据类型的转换知识。

使用各种转换函数，转换函数返回一个新的对象，表示转换的值。

语法格式：转换后变量 = 转换函数名称（待转换数据或变量）

示例：

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  mytu=('庄周','后羿','盘古','女娲')  print(type(mytu))   mlist=list(mytu)*#将元组类型转为列表类型* print(type(mlist))    pass |

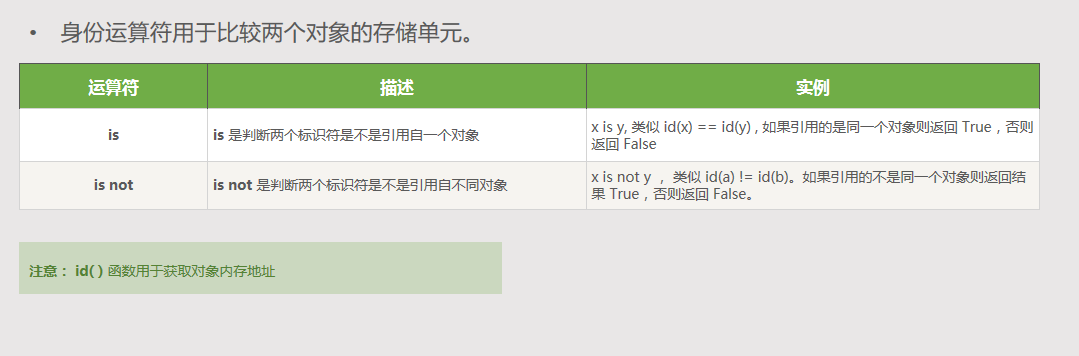


## 5.10.运算符及表达式

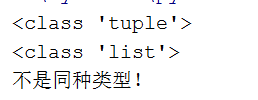
### 5.10.1逻辑运算符



### 5.10.2.Python 身份运算符



|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  mytu=('庄周','后羿','盘古','女娲')  print(type(mytu))   mlist=list(mytu)*#将元组类型转为列表类型* print(type(mlist))    if id(mytu) is id(mlist):  print('是同一种对象类型')  else:  print('不是同种类型！')   pass |



### 5.10.2.Python 运算符优先级



### 5.11.作业

1. 完成课堂所示的案例，截图到文档中
2. 使用字典列表，在列表中添加字典类型的数据，字典的数据格式如下：

(1)使用列表的方法添加如下字典类型数据到listemp列表中

emp={‘id’:1,’name’,’xxxx’,’sex’:xxx,’address’:’xxxsxs’,’sal’:10000}

(2)emp中的字典数据使用键盘输入对应的值

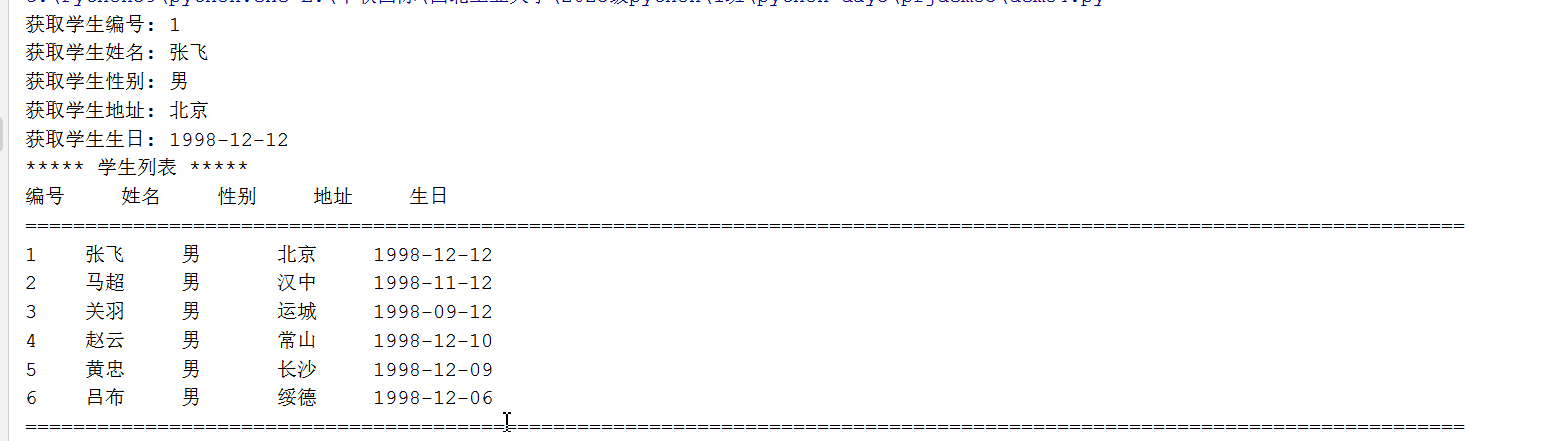
例如：emp[‘id’]=input(‘输入员工编号:’)

(3)使用列表的函数append添加字典数据到列表，并提示是否继续，直到输入n结束

listemp=[]

listemp.append(emp)

(4)使用循环输出员工的信息，并按照课堂案例所示进行呈现



1. 从键盘输入员工姓名，查找列表中是否存在该员工的信息。并输出显示
2. 从键盘输入员工的编号，并从列表中删除改员工的信息，循环显示剩下的员工列表信息。

(7)从键盘输入员工的编号，并将指定编号的员工信息中的薪资和地址进行修改，并使用循环显示修改后的所有员工信息。