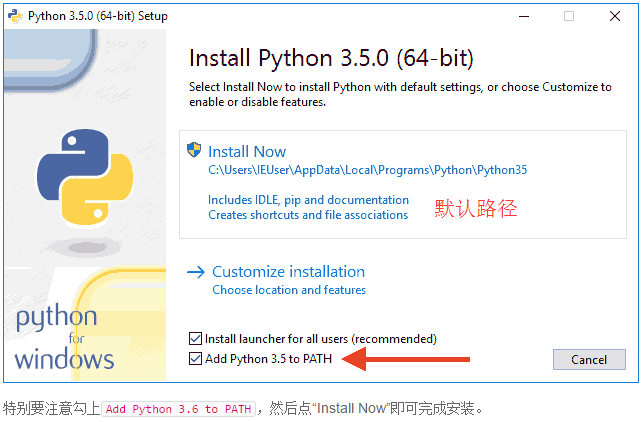
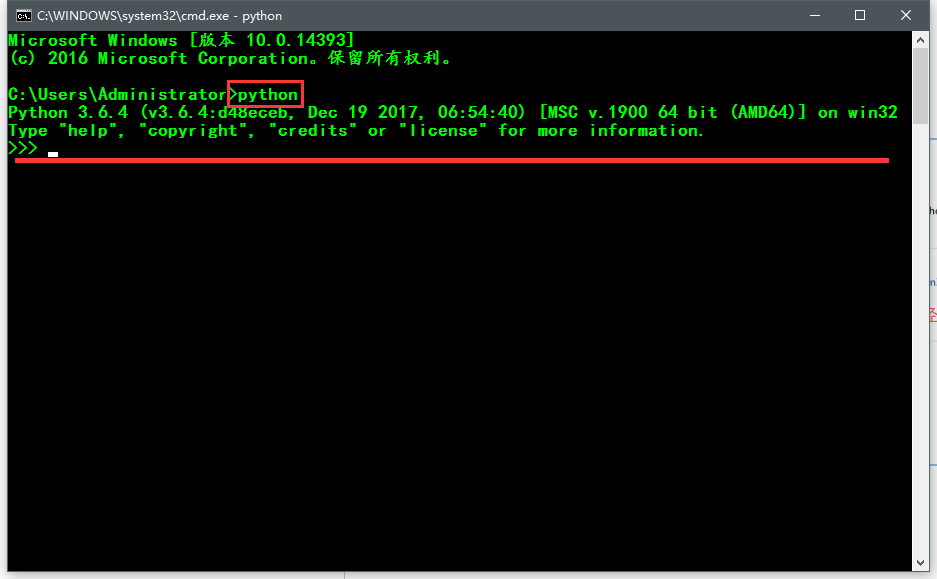
**1.安装python**

1.<https://www.python.org/> 下载安装包。



2.打开cmd，输入python，说明安装成功。



**2.python解释器**



**3.第一个python程序**

Python文件都已.py后缀结尾，可使用命令行 python xxx.py 执行文件程序

**4.输入和输出**

input:输入函数input()

output:输出函数print()

**5.python基础**

0. ’#’开头是注释

语句以’:’结尾时，缩进的语句视为代码块

Python程序是大小写敏感的，如果写错，程序会报错

默认使用四个空格缩进

1.数据类型和变量

整数:与数学写法一致

浮点数:普通浮点数与数学小数写法一致，太大或太小的浮点数必须用科学计数法表示，把10用e代替，如: 1.23x109就是1.23e9

字符串: ①字符串是以单引号'或双引号"括起来的任意文本，如果'本身也是一个字符，那就可以用""括起来

②如果字符串内部既包含'又包含"怎么办？可以用转义字符\来标识，转义字符\可以转义很多字符，比如\n表示换行，\t表示制表符，字符\本身也要转义，所以\\表示的字符就是\

③如果字符串里面有很多字符都需要转义，就需要加很多\，为了简化，Python还允许用r''表示''内部的字符串默认不转义，如果字符串内部有很多换行，用\n写在一行里不好阅读，为了简化，Python允许用'''...'''的格式表示多行内容,只有在交互式命令行内输入有效，如果写成程序并存为.py则不需要...

布尔值: ①布尔值和布尔代数的表示完全一致，一个布尔值只有True、False两种值，要么是True，要么是False，在Python中，可以直接用True、Fa lse表示布尔值（请注意大小写），也可以通过布尔运算计算出来

②布尔值可以用and、or和not运算

Ⅰ:and运算是与运算，只有所有都为True，and运算结果才是True

Ⅱ:or运算是或运算，只要其中有一个为True，or运算结果就是True

Ⅲ:not运算是非运算，它是一个单目运算符，把True变成False，False变成True

空值:空值是Python里一个特殊的值，用None表示。None不能理解为0，因为0是有意义的，而None是一个特殊的空值

变量:变量不仅可以是数字，还可以是任意数据类型，变量在程序中就是用一个变量名表示了，变量名必须是大小写英文、数字和\_的组合，且不能用数字开头，在Python中，等号=是赋值语句，可以把任意数据类型赋值给变量，同一个变量可以反复赋值，而且可以是不同类型的变量

常量: 所谓常量就是不能变的变量，比如常用的数学常数π就是一个常量。在Python中，通常用全部大写的变量名表示常量，但事实上PI仍然是一个变量，Python根本没有任何机制保证PI不会被改变，所以，用全部大写的变量名表示常量只是一个习惯上的用法，如果你一定要改变变量PI的值，也没人能拦住你

注: ①:整数和浮点数在计算机内部存储的方式是不同的，整数运算永远是精确的（包括除法），而浮点数运算则可能会有四舍五入的误差。

②整数的除法为什么也是精确的。在Python中，有两种除法，一种除法是/，/除法计算结果是浮点数，即使是两个整数恰好整除，结果也是浮点数；还有一种除法是//，称为地板除，两个整数的除法仍然是整数，整数的地板除//永远是整数，即使除不尽。要做精确的除法，使用/就可以，因为//除法只取结果的整数部分，所以Python还提供一个余数运算%，可以得到两个整数相除的余数，无论整数做//除法还是取余数，结果永远是整数，所以，整数运算结果永远是精确的

2.字符串和编码

在最新的Python 3版本中，字符串是以Unicode编码的，也就是说，Python的字符串支持多语言

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

Python对bytes类型的数据用带b前缀的单引号或双引号表示，如:x=b’lyc’; 要注意区分'ABC'和b'ABC'，前者是str，后者虽然内容显示得和前者一样，但bytes的每个字符都只占用一个字节,以Unicode表示的str通过encode()方法可以编码为指定的bytes，例如:’abc’.encode(‘编码’)，含有中文的str无法用ASCII编码，因为中文编码的范围超过了ASCII编码的范围，Python会报错，在bytes中，无法显示为ASCII字符的字节，用\x##显示；如果我们从网络或磁盘上读取了字节流，那么读到的数据就是bytes。要把bytes变为str，就需要用decode()方法，如：b’abc’.decode(‘编码’)，如果bytes中包含无法解码的字节，decode()方法会报错，如果bytes中只有一小部分无效的字节，可以传入errors='ignore'忽略错误的字节；要计算str包含多少个字符，可以用len()函数，len()函数计算的是str的字符数，如果换成bytes，len()函数就计算字节数，1个中文字符经过UTF-8编码后通常会占用3个字节，而1个英文字符只占用1个字节

在操作字符串时，我们经常遇到str和bytes的互相转换。为了避免乱码问题，应当始终坚持使用UTF-8编码对str和bytes进行转换。

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

|  |
| --- |
| #!/usr/bin/env python3  # -\*- coding:utf-8 -\*- |

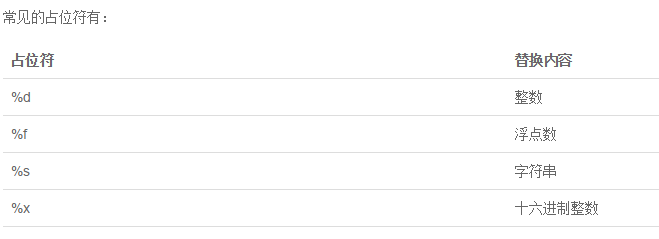
第一行注释是为了告诉Linux/OS X系统，这是一个Python可执行程序，Windows系统会忽略这个注释；

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

申明了UTF-8编码并不意味着你的.py文件就是UTF-8编码的，必须并且要确保文本编辑器正在使用UTF-8 without BOM编码，如果.py文件本身使用UTF-8编码，并且也申明了# -\*- coding: utf-8 -\*-，打开命令提示符测试就可以正常显示中文

格式化字符串

Python中，采用的格式化方式和C语言是一致的，用%实现，%运算符就是用来格式化字符串的。在字符串内部，%s表示用字符串替换，%d表示用整数替换，有几个%?占位符，后面就跟几个变量或者值，顺序要对应好。如果只有一个%?，括号可以省略。



如果你不太确定应该用什么，%s永远起作用，它会把任何数据类型转换为字符串；有些时候，字符串里面的%是一个普通字符怎么办？这个时候就需要转义，用%%来表示一个%

另一种格式化字符串的方法是使用字符串的format()方法，它会用传入的参数依次替换字符串内的占位符{0}、{1}……，不过这种方式写起来比%要麻烦得多

3.使用list和tuple

list

Python内置的一种数据类型是列表：list。list是一种有序的集合，可以随时添加和删除其中的元素。用len()函数可以获得list元素的个数。用索引来访问list中每一个位置的元素，记得索引是从0开始的。当索引超出了范围时，Python会报一个IndexError错误，所以，要确保索引不要越界，记得最后一个元素的索引是len(classmates)-1。如果要取最后一个元素，除了计算索引位置外，还可以用-1做索引，直接获取最后一个元素，以此类推，-2、-3可以获取倒数第2个、倒数第3个。

list是一个可变的有序表，所以，list.append(‘’)可以往list中追加元素到末尾。list.insert(1,‘’)也可以把元素插入到指定的位置，比如索引号为1的位置。要删除list末尾的元素，用pop()方法。要删除指定位置的元素，用pop(i)方法，其中i是索引位置。要把某个元素替换成别的元素，可以直接赋值给对应的索引位置

list里面的元素的数据类型也可以不同，list元素也可以是另一个list，如果一个list中一个元素也没有，就是一个空的list，它的长度为0

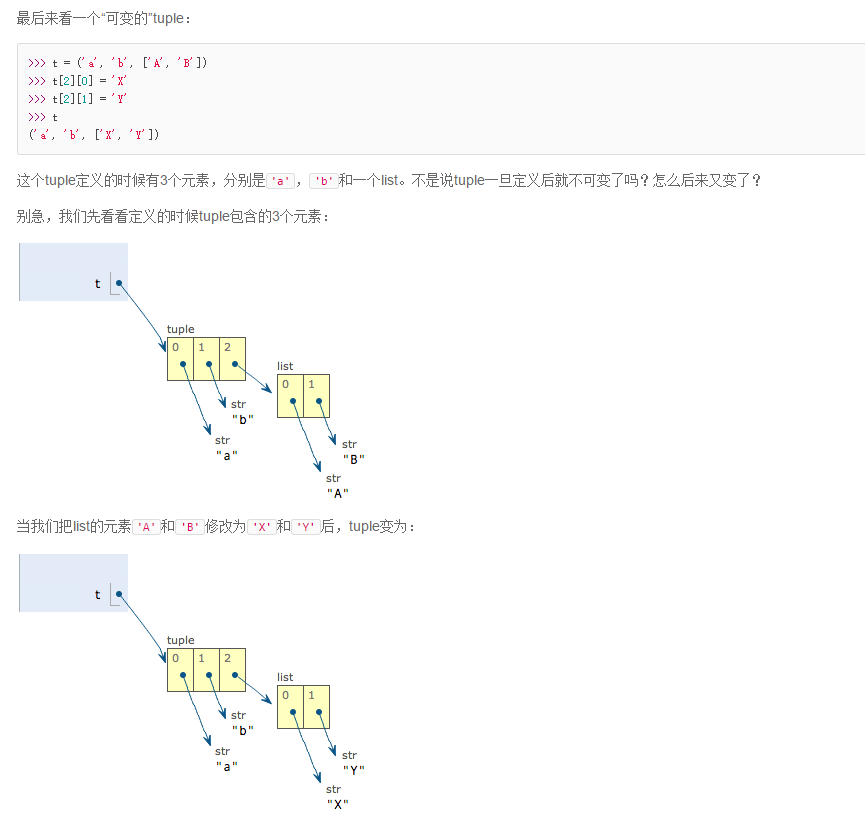
tuple

另一种有序列表叫元组：tuple。tuple和list非常类似，但是tuple一旦初始化就不能修改，它没有append()，insert()这样的方法。其他获取元素的方法和list是一样的。不可变的tuple有什么意义？因为tuple不可变，所以代码更安全。如果可能，能用tuple代替list就尽量用tuple。tuple的陷阱：当你定义一个tuple时，在定义的时候，tuple的元素就必须被确定下来，如果要定义一个空的tuple，可以写成()。

要定义一个只有1个元素的tuple，不能使用t=(1)，因为这样定义的不是tuple，是1这个数！这是因为括号()既可以表示tuple，又可以表示数学公式中的小括号，这就产生了歧义，因此，Python规定，这种情况下，按小括号进行计算，计算结果自然是1。所以，只有1个元素的tuple定义时必须加一个逗号,，来消除歧义：t=(1,), Python在显示只有1个元素的tuple时，也会加一个逗号,，以免你误解成数学计算意义上的括号

tuple所谓的“不变”是说，tuple的每个元素，指向永远不变。

”可变”的tuple原理↓。



4.条件判断

If(条件):。。。：

If(条件):。。。else。。。：

If(条件):。。。elif(条件):。。。elif(条件):。。。else。。。：

注意：每个条件分支的冒号:

if判断条件还可以简写: if x:print(‘’);

只要x是非零数值、非空字符串、非空list等，就判断为True，否则为False。

5.循环

For…in::

依次把list或tuple中的每个元素迭代出来

例如: for x in list 🡪 就会依次遍历list的元素并代入x

range()函数，可以生成一个整数序列

While::

只要条件满足，就不断循环，条件不满足时退出循环。

Break:

在循环中，break语句可以提前退出循环。

Countinue:

通过continue语句，跳过当前的这次循环，直接开始下一次循环。

6.使用dict和set

Dict:

Python内置了字典：dict的支持，dict全称dictionary，在其他语言中也称为map，使用键-值（key-value）存储，具有极快的查找速度。

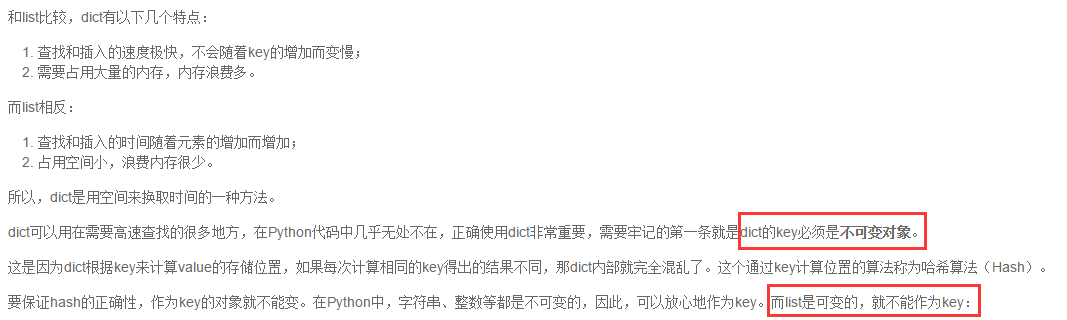
声明方式：dict = {‘key’:value}

一个key只能对应一个value，所以，多次对一个key放入value，后面的值会把前面的值冲掉。

取值：dict[‘key’]、dict.get(‘key’)

要避免key不存在的错误，有两种办法，一是通过in判断key是否存在; 二是通过dict提供的get()方法，如果key不存在，可以返回None，或者自己指定的value; 注意：返回None的时候Python的交互环境不显示结果。

要删除一个key，用pop(key)方法，对应的value也会从dict中删除。



Set:

是一组key的集合，但不存储value。由于key不能重复，所以，在set中，没有重复的key。

要创建一个set，需要提供一个list作为输入集合。

set可以看成数学意义上的无序和无重复元素的集合，因此，两个set可以做数学意义上的交集、并集等操作。

通过add(key)方法可以添加元素到set中，可以重复添加，但不会有效果；通过remove(key)方法可以删除元素。

set和dict的唯一区别仅在于没有存储对应的value，但是，set的原理和dict一样，所以，同样不可以放入可变对象，因为无法判断两个可变对象是否相等，也就无法保证set内部“不会有重复元素”。试试把list放入set，看看是否会报错。

不可变对象：

