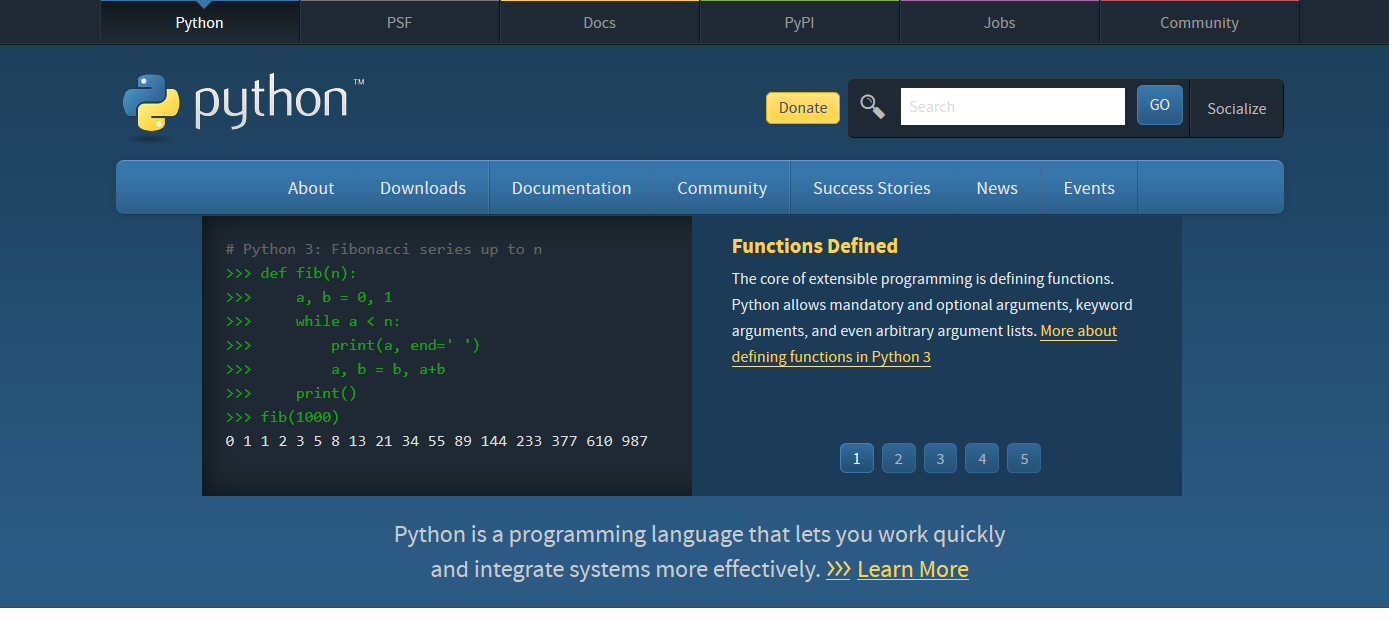
Python介绍及基础语法

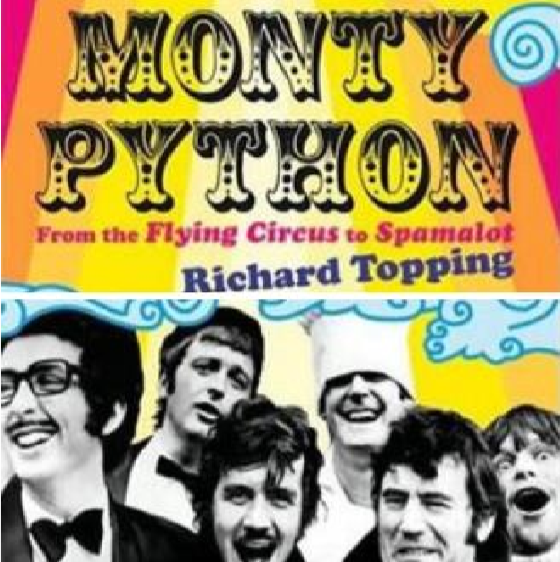
* 1. Python介绍
     1. 简介

Python是当今世界最流行的程序语言之一，它通俗易懂可读性强且拥有优秀法结构。Python是一种解释型、面向对象的语言。由吉多·范罗苏姆（Guido van Rossman）于1989 年发明，1991 年正式公布。Python官网：www.python.org，如图下。如果你是一个编程初学者，你会发现Python编程并没有那么枯燥，甚至可以体会到Python的优雅与简洁之美，希望本次课程将会是你 IT 生涯的启蒙导师。

Python 突出的简洁性、易读性和扩展性，使得 Python 应用于科学研究的机构日益增多，这里也包括一些全球顶尖的大学也在采用 Python 教授程序设计课程。除此之外，Python 在数据科学、人工智能、云计算、图形处理与互联网应用等领域同样占尽了风头。



Python 单词是“大蟒蛇”癿意思。但是龟叔不是喜欢蟒蛇才起返个名字，而是正在追剧：英国电规喜剧片《蒙提·派森的飞行马戏团》(Monty Python and the Flying Circus)。



1.1.2 特点：

1. 可读性强！

可读性远比听上去重要的多得多。一个程序会被反复的修改，可读性强意味着让你 可以在更短时间内学习和记忆，直接提高生产率。

1. 简洁，简洁，简洁！

研究证明，程序员每天可编写的有效代码数是有限的。完成同样功能只用一半的代 码，其实就是提高了一倍的生产率。

1. Python 是由 C 语言开发，但是不再有 C 语言中指针等复杂数据类型！

Python 的简洁性让开发难度和代码幅度大幅降低，开发任务大大简化。程序员再也不需要关注复杂的语法，而是关注任务本身。

1. 入门级语言！

只适合菜鸟？准确的说拉近了高手与初学者之间的距离。Python简洁的语法结构，学习门槛低，编程极易上手，无论老鸟还是菜鸟都站在同一个起跑线上。

1. 解释性与交互性！

与典型的 Java 编译型语言相比，Python属于解释型语言。一方面，Python编写一条程序语句，即可解释执行返回一个结果。当程序出错时更容易跟踪与定位；另一方面：Python这种交互式模式为人机互动提供了更广阔的可能空间。

1. 优秀的模块化思维！

将代码组织为一个或若干模块，模块组织成为包、甚至库。试想当你编写程序的时候如果已经有针对科学计算、爬虫、数据分析、可视化、机器学习等模块或第三方库可以直接拿来使用，编程效率会极大的提高。

1. 开源软件！

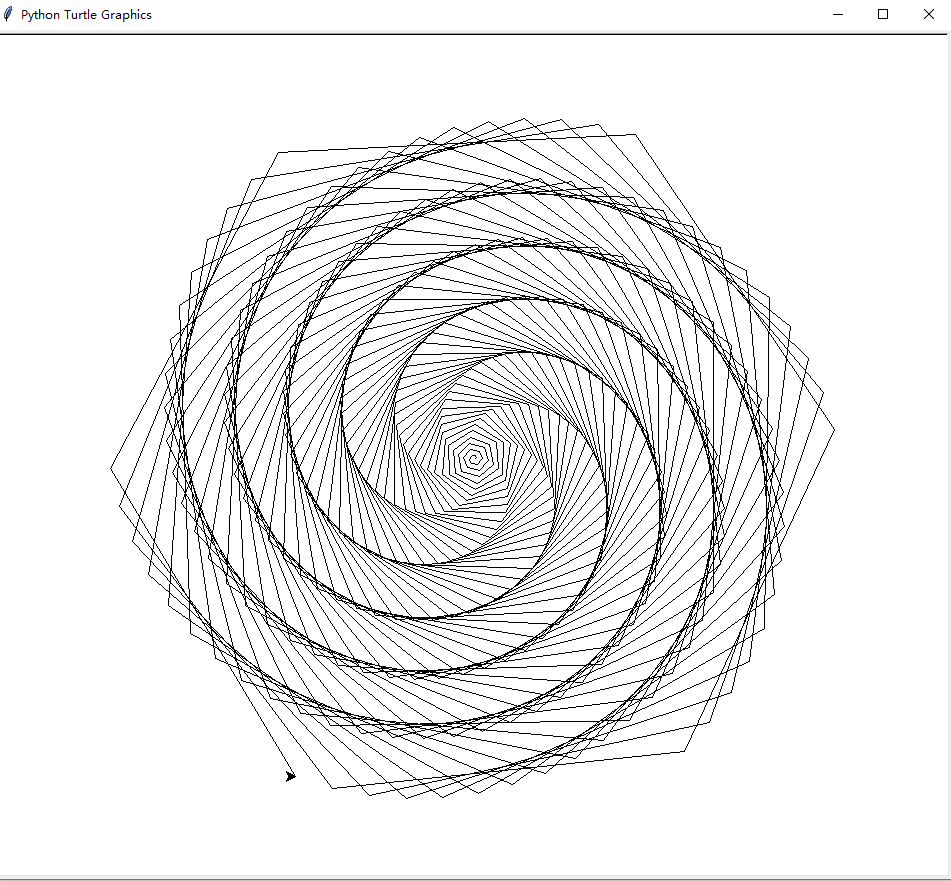
Python是纯粹的开源语言，源代码遵循 GPL 许可，这些特性使其更受大众欢迎，软件更容易移植到其他的平台，如Mac、Linux等，因此 Python 拥有丰富的第三方资源库是不足为奇的。

1. 标准脚本语言！

脚本程序是指只有需要被调用的时候才会被动态的解释执行。 Python允许混合使用C 、 Java 与 Python代码，通过增强扩展性来解决一些特殊的问题，例如 Python 程序中允许调用一段由 Java 编写的程序模块（库），甚至这段 Java 模块可以是保密的。以上这些陈述都充分体现了 Python 的可扩展性和作为脚本语言的动态灵活性。

\*GPL 许可协议(GNU General Public License)：只要软件中包含有其他 GPL 协议的产品或代码，那么该软件就必须也采用 GPL 许可协议且开源及免费。即复制自由、传播自由，允许以各种形式进行传播，并且收费传播，修改自由。

例子1.1



完成这样的螺旋线，代码只有几行：

1. **import** turtle
2. t=turtle.Pen()
3. **for** x **in** range(0,360):
4. t.forward(x)
5. t.left(59)

1. 面向对象

2. 免费和开源

3. 可移植性和跨平台

Python 会被编译成与操作系统相关的二进制代码，然后再解释执行。这种方式和 java 类似，大大提高了执行速度，也实现了跨平台。

4. 丰富的库（丰富的标准库，多种多样的扩展库）

5. 可扩展性。可嵌入到 C和 C++语言。胶水式语言。

应用范围：

1. 科学计算

2. 人工智能

3. WEB 服务端和大型网站后端。

YouTube、Gmail 等应用基于python 开发。

4. GUI 开发（图形用户界面开发）

5. 游戏开发

6. 移动设备

7. 嵌入式设备

8. 系统运维

9. 大数据

什么时候不应该用python·

Python 是解释执行。性能较低。

因此，一些影响性能的功能可以使用 C/C++/JAVA/GO（GO 是一种新语言，写起来像 Python，性能像 C）去开发。不过，也不用太担心，Python 解释器会越来越快。

1.2 IDLE开发环境使用入门

1.2.1 IDLE介绍

1. IDLE 是 Python 癿官方标准开发环境，Python 安装完后同时就安装了IDLE。

2. IDLE 已经具备了 Python 开发的几乎所有功能（语法智能提示、不同颜色显示不同类型等等），也不需要其他配置，非常适合初学者使用。

3. IDLE 是Python 标准发行版内置的一个简单小巧的 IDE，包括了交互式命令行、编辑器、调试器等基本组件，足以应付大多数简单应用。

4. IDLE 是用纯 Python 基于Tkinter编写，最初级的作者正是 Python 之父 Guido van Rossum。

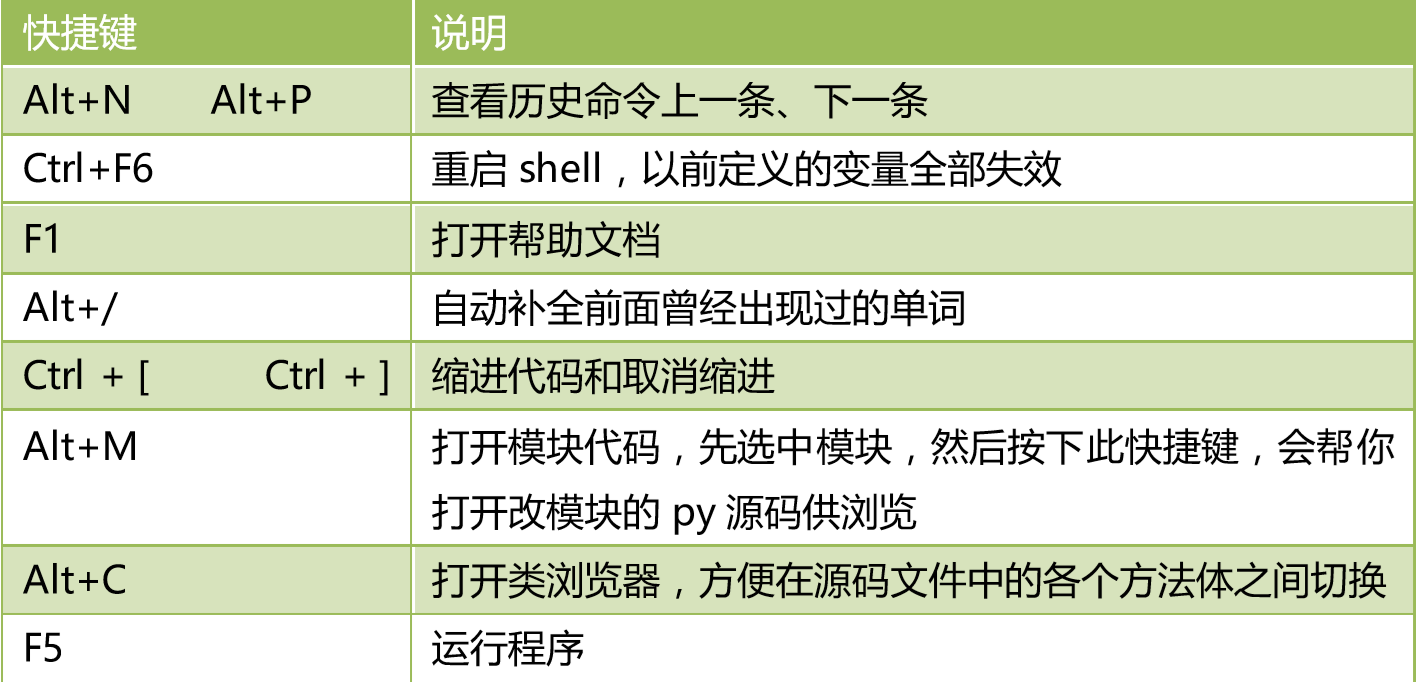
1.2.2 IDLE实操

1. 交互模式

启动IDLE ，默认就是进入交互模式。

2. 编写和执行 Python 源文件

1.2.3 快捷键



* 1. 第一个python源程序

1. **print**('Hello,world!')

将源代码保存到：E:\python\first-programme.py

在 IDLE 中单击 F5 或者 run-->run module 执行这个源程序

第一个 Python 程序中需要注意的小要点：

1. 不要在程序中，行开头处增加空格。空格在 Python 中有缩进的含义。

2. 符号都是英文符号，不是中文。比如：(,”

下面我们全程讲解第一个程序编码过程。

1. **Windows 命令行编写代码（Console）**

通过上节基础性的学习，各位可能感觉在命令行（Console）控制台中编写代码觉得有点枯燥，感觉效率不是很高。不过这是为了照顾零基础初学者，让大家都打下个扎实的基础， 后面我们会慢慢过渡到使用 IDE（Integrated Development Environment）集成开发环境来高效率地编写 Python 代码。这里来看一个简单的例子，读者参考上节的步骤，进入 Console 界面，开始敲入以下代码。

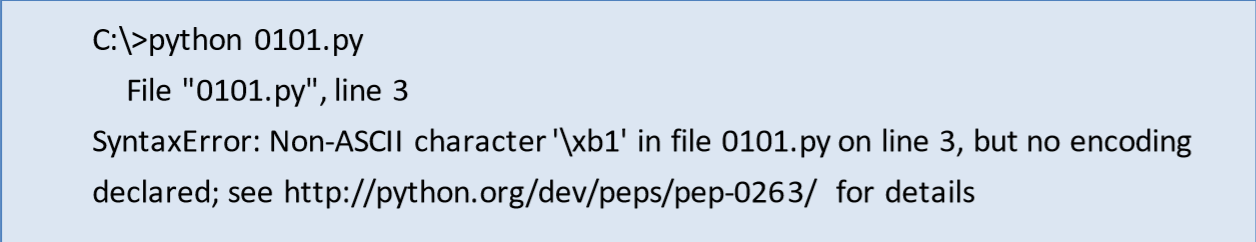
1. #! /usr/bin/evn python
2. #直接给变量赋值，无需先定义变量类型
3. var01="First console python code"
4. var02="Python execute successfully"
6. #打印输出函数
7. **print**(var01)
8. **print**(var02)

Console 的两种执行方式如下：

第一种方式：进入 Console 界面后，一边编写代码，一边执行。 

第二种方式：将 Python 代码以文本编辑器编写好并保存，然后在 Console 当中进入 Python 代码所在文件夹，只需要输入 Python 文件名，即可解释执行 Python 代码。 命令格式为：python filename.py，其中 filename 是 Python 文件的名字。

实际上，如果一字不差的执行书中代码，不会顺利出来上述结果的，可能会提示出如下错误信息：

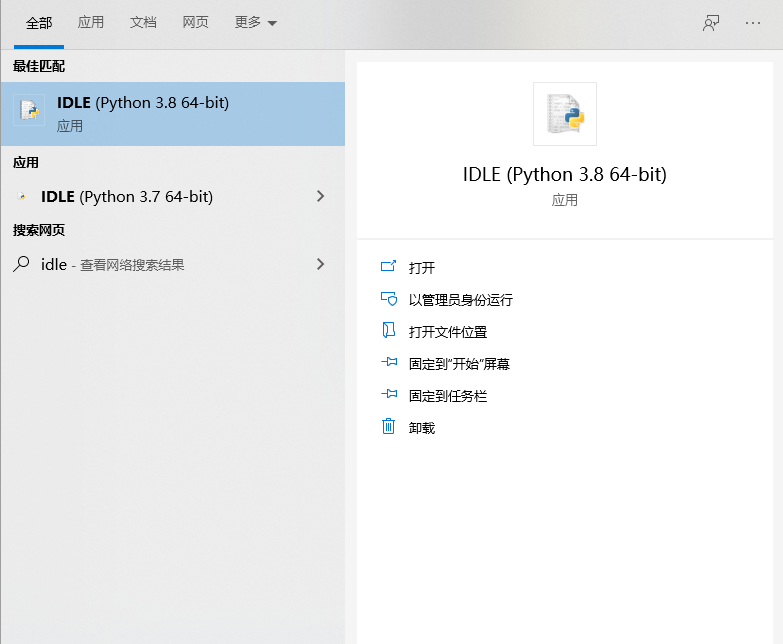


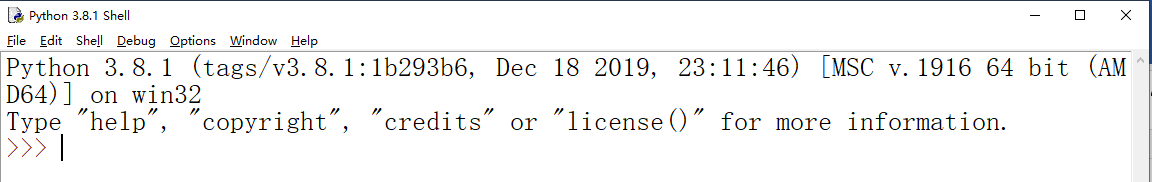
在未来的程序学习之旅，调试程序也是需要学习和锻炼的一项基本功。看懂上述错误的原因了吗？这里是因为程序代码中有中文注释。但是，Python 中默认采用 ASCII 编码格式， 无法处理汉字。因此，需要添加一行支持中文的代码说明：#-\*- coding:utf-8 -\*-。

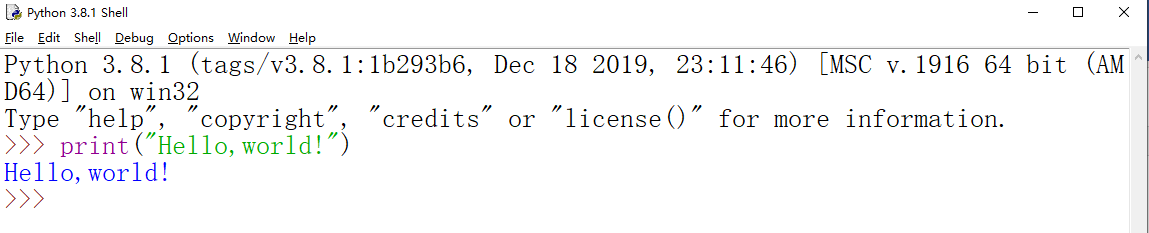
改进后的程序代码如下面程序所示，这时程序就不会再出现错误提示了。

1. #! /usr/bin/evn python
2. #-\*- coding:utf-8 -\*-
3. #直接给变量赋值，无需先定义变量类型
4. var01="First console python code"
5. var02="Python execute successfully"
7. #打印输出函数
8. **print**(var01)
9. **print**(var02)
10. **IDLE编写代码**

**我们讲了上述 Python 代码执行方式后，我们现在介绍一下 Python 官方自带的原生编辑器。这款原生编辑器比起 Console 界面编程方式在用户体验方面舒适方便了一些。 Python 自带的一款集成开发环境：IDLE。它可以让初学者方便的掌握基本的 Python 代 码编写与调试工作。我们现在开始了解如何操作这款工具吧。**







读者在使用 Python 的 IDLE 时会发现，每次编辑完代码完成后，按回车键后并不会马上执行程序并输出结果，而是第二次按回车键时候才执行了程序结果。这是因为编辑器认为可能还会输入下一条代码，除非连续按二次回车，系统才会知道没有代码编辑了，需要解释执行代码（当然，如果输入 print 语句，会马上执行）。 通常，我们会在一个文本编辑器中编写 Python 代码，Python 的 IDLE 自带有编辑器，我 们选中要编码的 Python 文件，鼠标右键打开：Edit with IDLE，然后就可以在界面里敲入代码了。代码编写完成之后，单击【菜单】|【Run】|【Run Module】（或F5），就可在新弹出的 Python 3.7.13 shell 窗口查看程序运行结果了。



* 1. 程序基本格式

1. 恰恰当的空格，缩进问题
2. 逻辑行首的空白（空格和制表符）用来决定逻辑行的缩进层次，从而用来决定语句的分组。
3. 语句从新行的第一列开始
4. 缩进风格的统一：
5. 每个缩进层次使用 单个制表符 或四个空格（IDE 会自动将制表符设置成 4 个空格）
6. Python 用缩进而不是{}表示程序块
7. Python 区分大小写
8. 注释
9. 行注释

每段注释前加#。当解释器看到#，则忽略这一行#后边的内容

1. 段注释

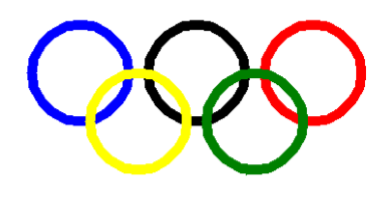
使用连续三个单引号(''')。当解释器看到'''，则会扫描到下一个'''，然后忽略他们之间的内容。

1.4开始学习图形化程序设计

为了让初学者更加容易接受编程，我们这里先从海龟画图开始讲解。这样，大家在不接触其他编程概念时，就能开始做出一些简单的效果。提高兴趣，寓教于乐。

1. **import** turtle             #导入 turtle 模块
2. turtle.showturtle()        #显示箭头
3. turtle.write("施想")        #写字符串
4. turtle.forward(300)      #前进 300 像素
5. turtle.color("red")       #画笔颜色改为 red
6. turtle.left(90)           #箭头左转 90 度
7. turtle.goto(0,50)        #去坐标（0,50）
8. turtle.goto(0,0)
9. turtle.penup()          #抬笔。返样，路径就不会画出来
10. turtle.goto(0,300)
11. turtle.pendown()       #下笔。这样，路径就会画出杢
12. turtle.circle(100)       #画圆

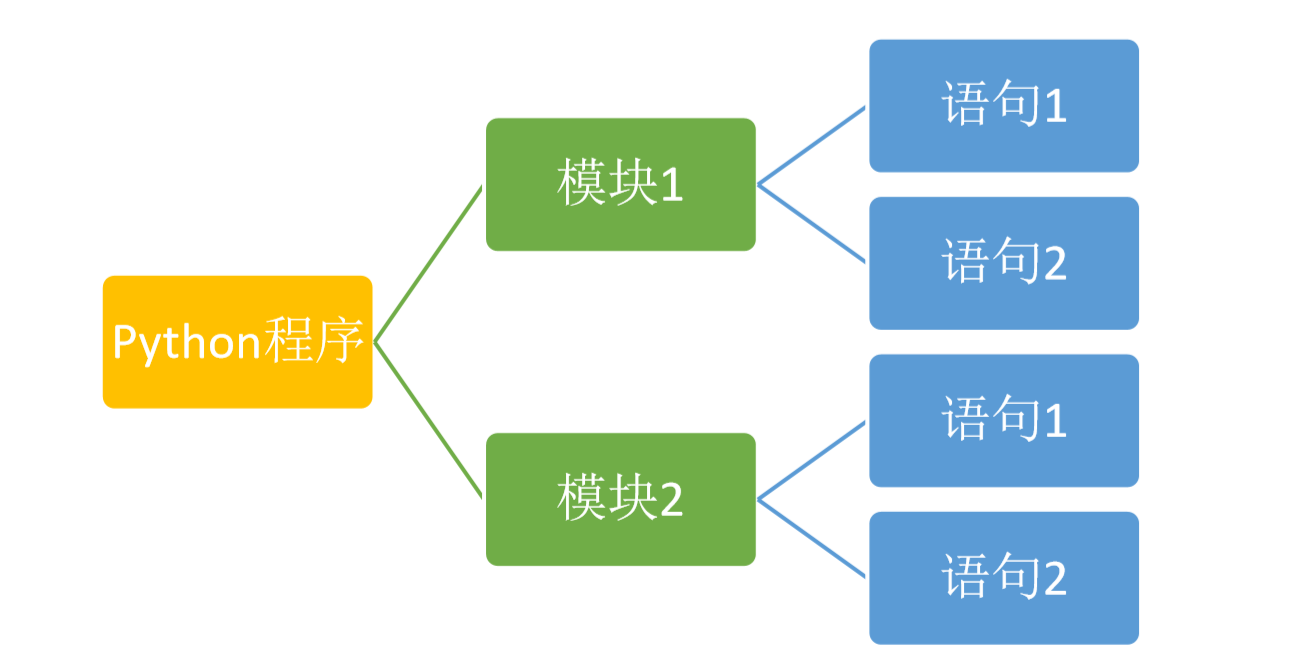
课后作业1：完成奥运五环画图程序



第二章 编程基本概念

2.1基本元素

2.1.1 Python 程序的构成



1. Python程序由模块组成。一个模块对应python源文件，一般后缀名是：.py。

2. 模块由语句组成。运行Python程序时，按照模块中语句的顺序依次执行。

3. 语句是Python程序的构造单元，用于创建对象、变量赋值、调用函数、控制语句等。

2.1.2 Python文件的创建和执行

前面使用的交互式环境，每次只能执行一条语句；为了编写多条语句实现复杂的逻辑，本章开始我们通过创建Python文件，并执行该文件。

在IDLE环境中，我们可以通过File-->new创建Python文件，并可以编辑该文件内容。我们也可以通过File-->save/save as保存文件。一般保存成扩展名为py的文件。

需要执行编辑好的文件，可以用快捷键F5或者点击Run-->Run module。

2.1.3使用注释#

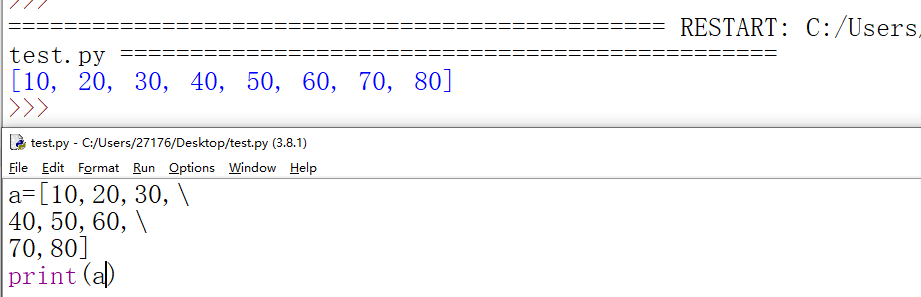
注释是程序中会被Python解释器忽略的一段文本。程序员可以通过注释记录任意想写的内容，通常是关于代码的说明。

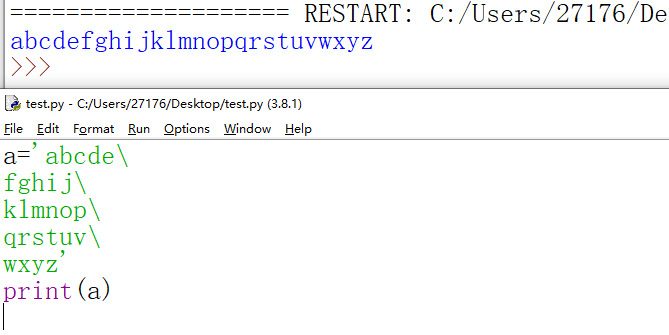
Python中的注释只有单行注释，使用#开始知道行结束的部分。

1. #注释是个好习惯，方便自己方便他人
2. a = [10,20,30]  #生成一个列表对象，变量 a 引用了这个变量

2.1.4使用\行连接符

一行程序长度是没有限制的，但是为了可读性更强，通常将一行比较长的程序分为多行。这是，我们可以使用\行连接符，把它放在行结束的地方。Python 解释器仍然将它们解释为同一行。

列表



字符串

2.1.5对象

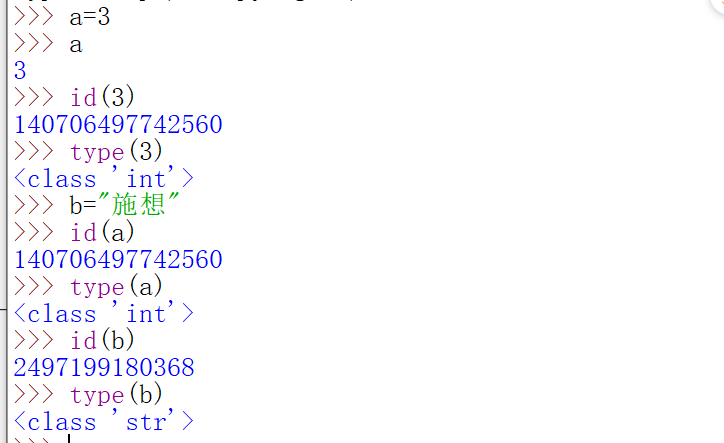
**Python 中，一切皆对象**。每个对象由：标识（identity）、类型（type）、value（值） 组成。

1. 标识用于唯一标识对象，通常对应于对象在计算机内存中的地址。使用内置函数 id(obj) 可返回对象 obj 的标识。

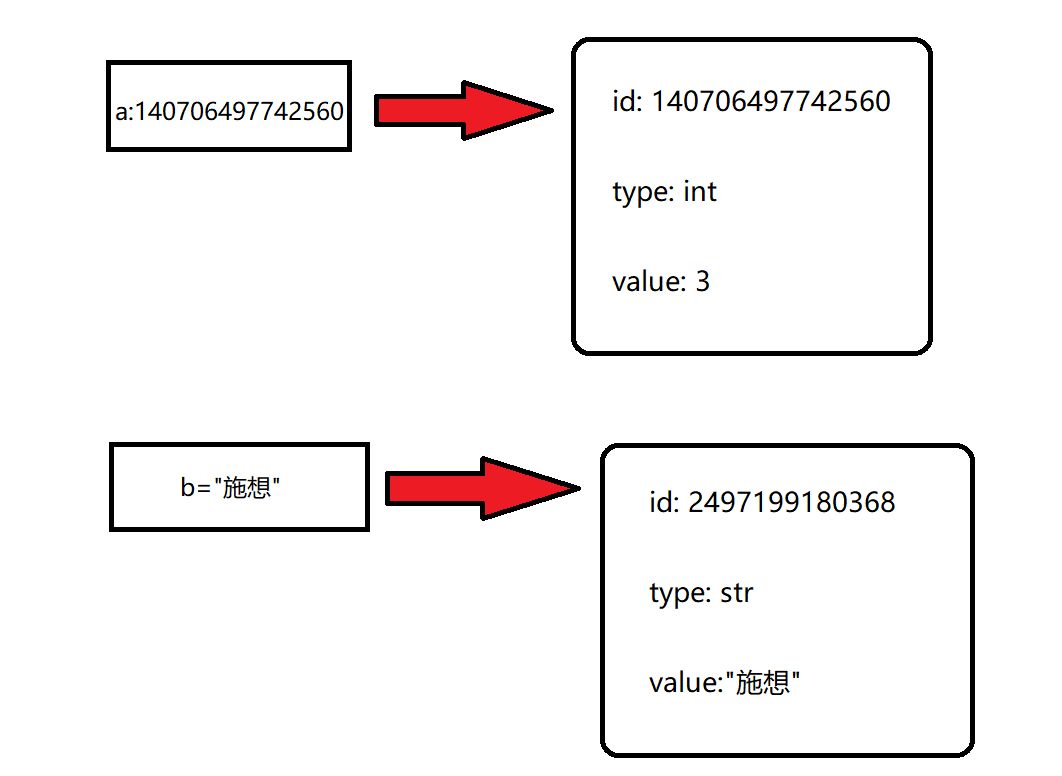
2. 类型用于表示对象存储的“数据”的类型。类型可以限制对象的取值范围以及可执行的 操作。可以使用type(obj)获得对象的所属类型。

3. 值表示对象所存储的数据的信息。使用 print(obj)可以直接打印出值。

对象的本质就是：一个内存块，拥有特定的值，支持特定类型的相关操作。



示意图：



**2.1.6 引用**

在Python中，变量也成为：对象的引用。因为，变量存储的就是对象的地址。

变量通过地址引用了“对象”。

变量位于：栈内存（压栈出栈等细节，后续再介绍）。

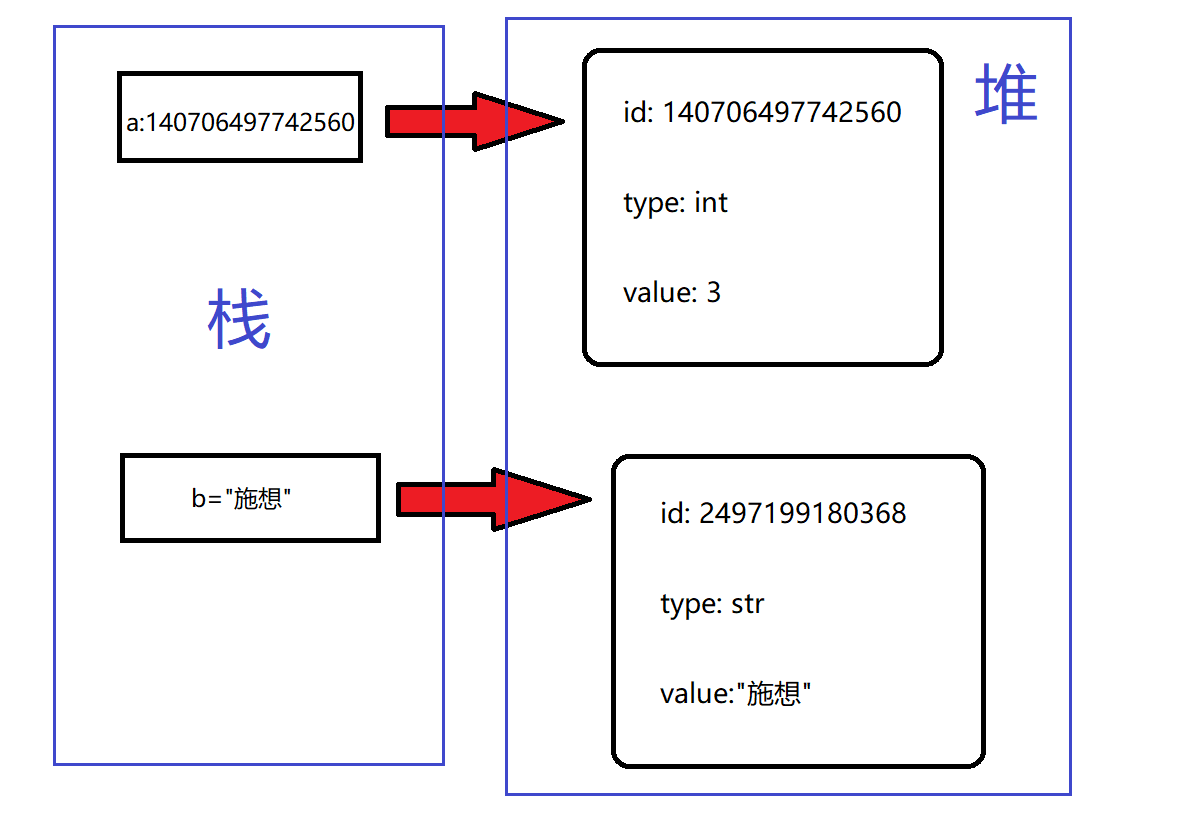
对象位于：堆内存。

**·Python是动态类型语言**

变量不需要显式声明类型。根据变量引用的对象，Python解释器自动确定数据类型。

**·Python是强类型的语言**

**每个对象都有数据类型，只支持该类型支持的操作。**

****

**2.1.7标识符**

Python语法中，标识符由数字、字母、下画线组成，所有标识符不能以数字开头，且Python标示符区分字母的大小写。以下划线开始的标识符有特殊意义，归类如下：

 \_单下画线开头：不能直接访问类属性，需要通过接口进行访问，不能采用from xxx import \*的方式。

 \_ \_双下画线开头：类的私有成员。

\_ \_xxx\_ \_双下画线开头和结尾：特殊方法专用的标识。如\_init\_()代表构造函数。

**标识符：用于变量、函数、类、模块等的名称。**标识符有如下特定的规则：

1. 区分大小写。如：sx和SX是不同的

2. 第一个字符必须是字母、下划线。其后的字符是：字母、数字、下划线

3. 不能使用关键字。比如：if、or、while等。

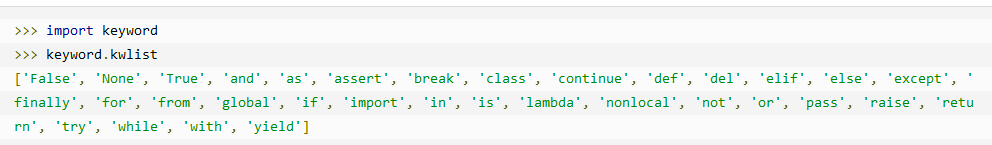
4. 以双下划线开头和结尾的名称通常有特殊含义，尽量避免这种写法。比如：\_\_init\_\_是类的构造函数。

\*使用help（）查看关键字

关键字无需可以去背，后面都会学习

Python保留字不能作为常数或变量，以及其他任何标识符名称，且都是小写字母。

Python保留字：



**2.2变量和简单赋值语句**

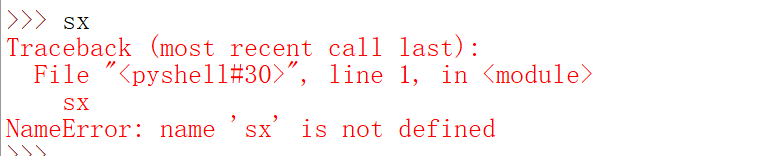
**2.2.1变量的声明和赋值**

变量的声明和赋值用于将一个变量绑定到一个对象上，格式如下：

变量名 = 表达式

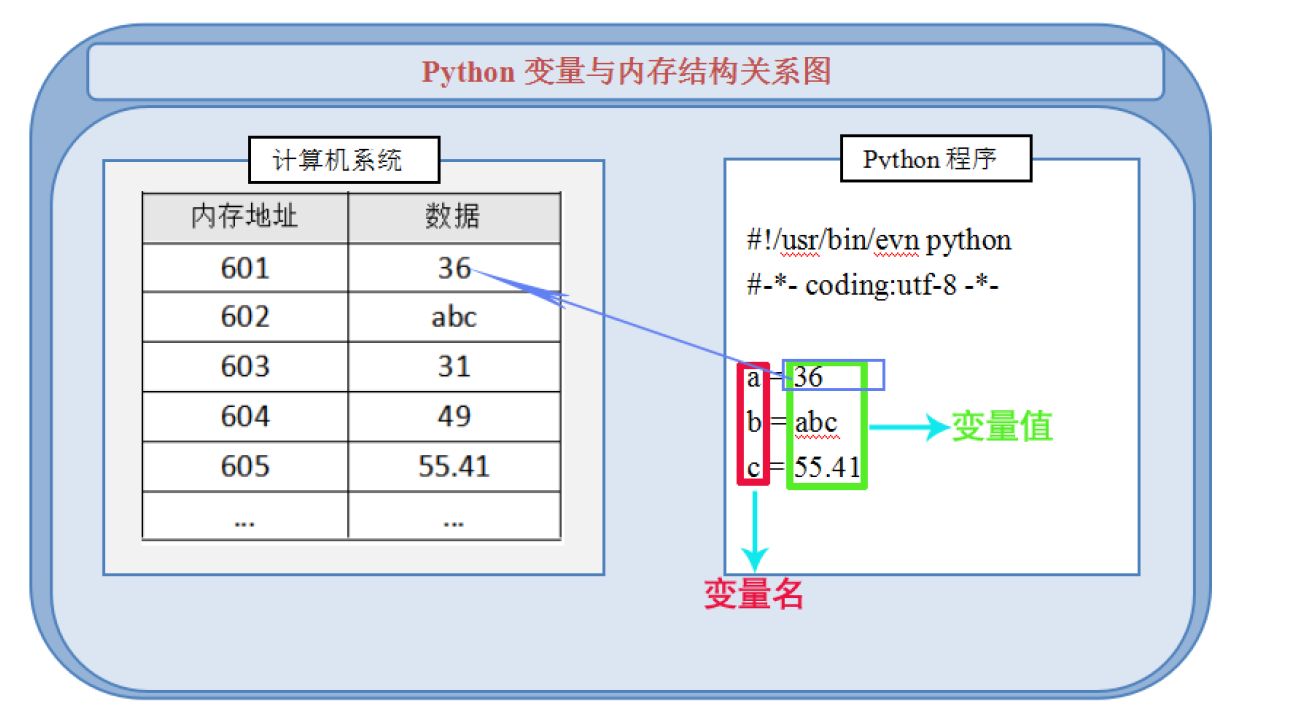
最简单的表达式就是字面量。比如：a = 123 。运行过程中，解释器先运行右边的表达式，生成一个代表表达式运算结果的对象；然后，将这个对象地址赋值给左边的变量。

【操作】变量在使用前必须先被初始化（先被赋值）



变量sx在被使用前未做赋值，因此报错：’sx’ is not defined。

变量存储在内存当中，当为变量赋值时，系统会在内存中开辟一个存储空间。变量中存放的数据都具有特定的数据类型，变量可以存储整数、小数、复数、字符等。



下面是Python变量常见类型：

 变量赋值

 数值型

 字符串

 列表

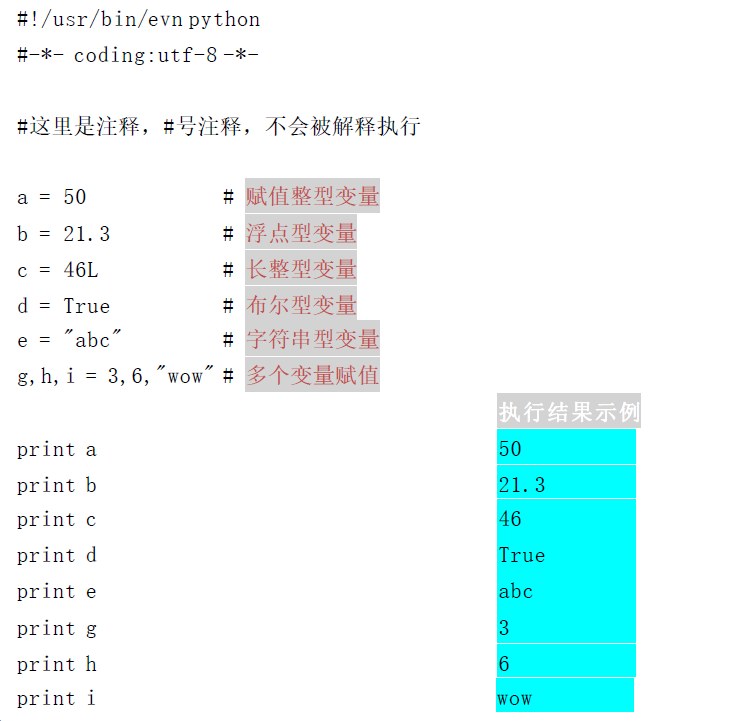
 元组

 字典

 集合

**1**．**Python**变量赋值

Python变量赋值不需要类型声明，直接赋值即可，且Python支持多个变量赋值,多变量赋值形式，比如：a = b = c = 12。



**2**．**Python**数值类型

Python数值类型（Number）是存储数值，Python支持四种不同的数值类型：

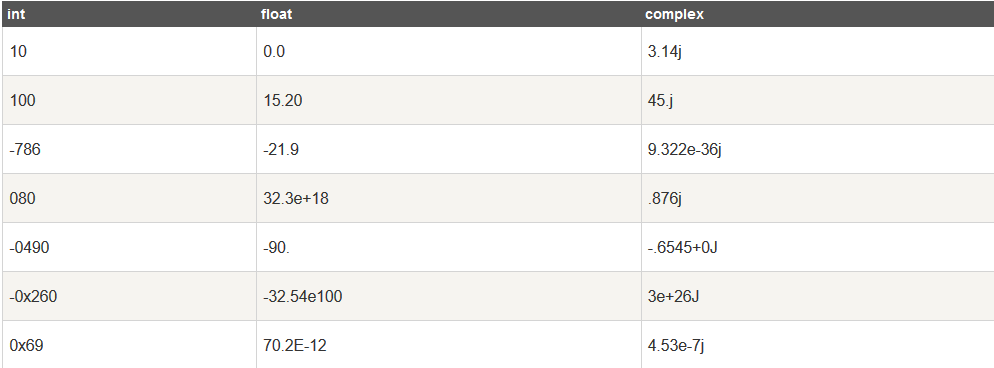
 int（有符号整型）

 float（浮点型）

 complex（复数）

 **bool** （布尔数）

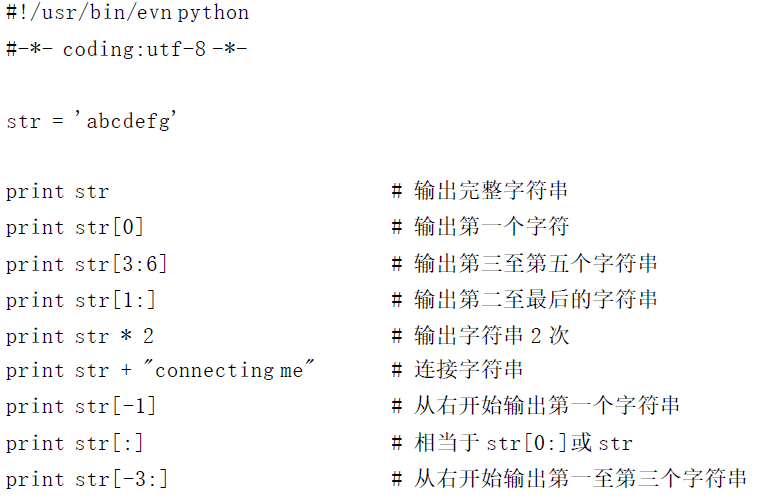
数值类型实例：



字符串类型（String）是由数字、字母、下划线组成的一串字符。字符创类型用来存储各类文本数据。Python的字符串取子串的话，有两种取值方式：

 从左到右，以0下标索引开始

 从右到左，以-1下标索引开始



温馨提示:Python 双引号与单引号的意义相同，都可表示为字符串。

字符串双引号之前加上字母u，Python会将其解释为Unicode字符串。

字符串格式化是将字符串经过处理后显示出希望的目标形式.

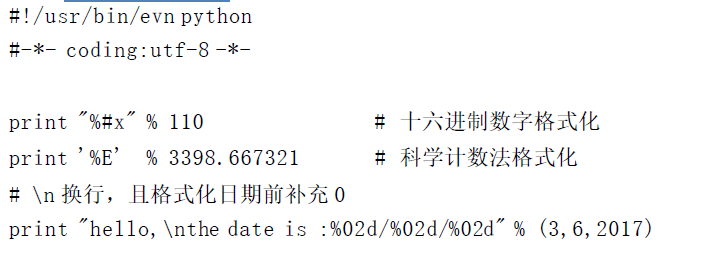
其表达式语法：format\_string % string\_to\_convert...... %左边为格式化要求，%为要格式化的内容。格式化符号相关说明如表:

Python中转义字符与其它高级程序设计语言区别不大，主要用来格式化不显示的字符，例如缩进，换行，回车等等，转义字符如下表所示：



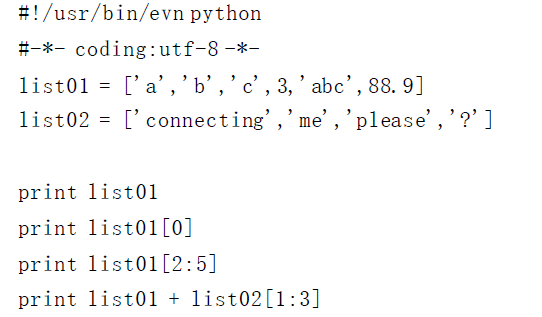


示例：



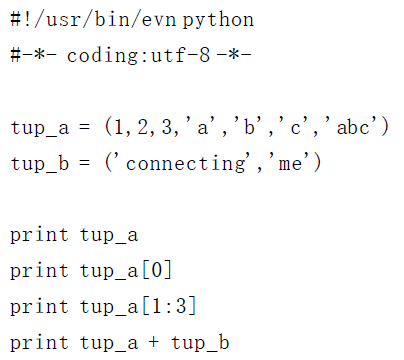
1. **Python**列表（**List**）

列表可以表示Python大多数的集合类数据结构，采用[ ]标识，是Python最常用的符合数据类型。列表支持字符、数字、字符串、列表（嵌套）等。列表实例代码：



1. **Python**元组（**Tuple**）

元组用（）标识，内部元素用逗号分隔。元组内的元素不可以修改，意味着不可以二次赋值。示例：





首先，通过对Python基础的学习，分别对List和Tuple两类数据类型做知识点梳理，我们最终采用二分法的形式对二者的区别与联系加以分析。我们这里采纳了思维导图的方式，更加容易理解，人们通常对图形的识别和印象是比较深刻，过程简单吧。

这里，将问题又划分为二大类：一类是偏向宏观方面，比如数据类型是否属于Sequence序列簇，是否是基本数据类型；另一类是更加具体的，比如数据的函数性质，操作符操作权限以及系统内存分配等。

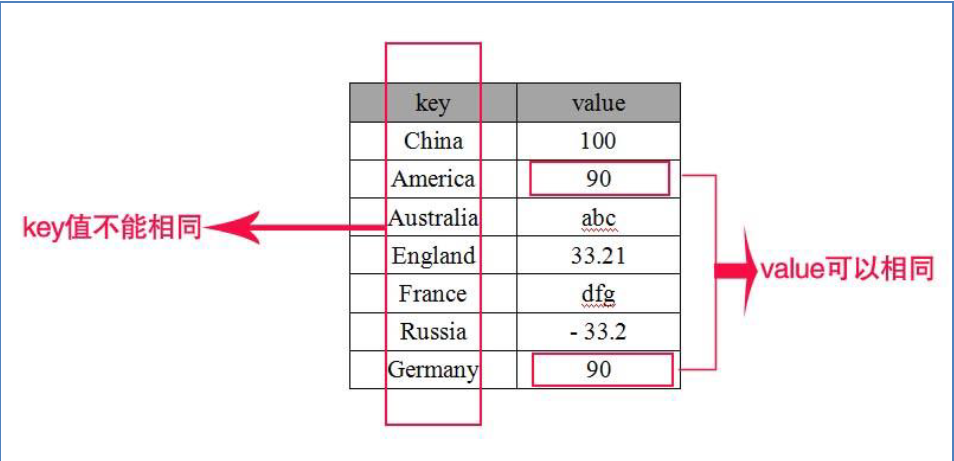
相信同学们通过上述图示的分析，对于Python当中其它数据类型的掌握和运用会更加轻车熟路。

**6**．**Python**字典（**Dictionary**）

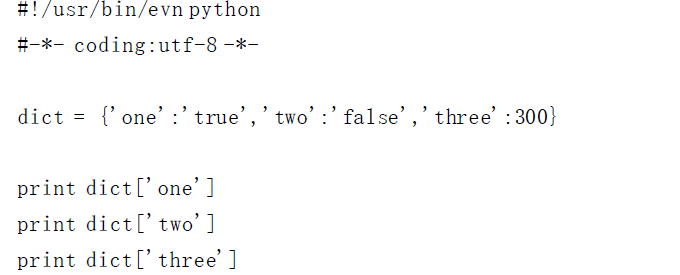
字典数据类型是有序的对象集合。字典的存取采用键-值存取的方式，即通过键取值。字典用{}标识。

关于键**-**值对存储形式在计算机科学当中应用非常广泛。程序访问键-值对的‘键’，系统会根据‘键’来获取对应的‘值’，‘键’与‘值’是一一对应关系，且‘键’不能重复。当我们的Python程序访问字典的每一个键时，根据其去查找唯一对应的值。

其工作原理如下图所示：



字典（Dictionary）相关代码实例如下图所示。

**7**．**Python**集合（**Set**）

集合（Set）是用于表示相互之间无序的一组元素。集合在算术上的运算包括交集、并集、补集等。同时，Python的集合又分为二类：

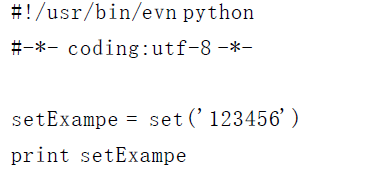
 普通集合：实现交集、并集、补集操作，通过关键字set实现。

 不可变集合：初始化后不可以改变，通过关键字frozenset实现

集合（Set）相关操作符如下图所示：



Set类型相关代码实例如下：



**8**．**Python**数据类型转换

数据类型的转换是编程当中经常会使用到的方法，我们只需要将数据类型作为函数名便可。关于一共有多少个转换函数可查阅相关问答，无需强硬的记忆。我们需要学会如何查找，如何看懂使用说明，这是需要加强的学习方法。

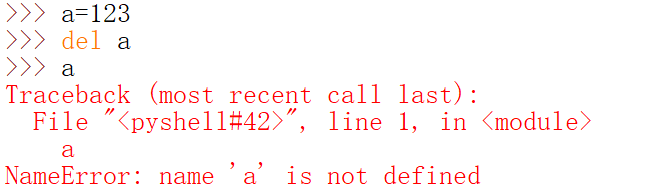
语法格式：函数名（对象）

例如：float(36) 含义：将整型数字36转化成浮点型

**2.2.2删除变量和垃圾回收机制**

可以通过del语句删除不在使用的变量。

【操作】 删除变量示例



如果对象没有变量引用，就会被垃圾回收器回收，清空内存空间。

**2.2.3链式赋值**

链式赋值用于同一个对象赋值给多个变量。

x=y=123 相当于：x=123; y=123

**2.2.4系列解包赋值**

系列数据赋值给对应相同个数的变量（个数必须保持一致）

>>> a,b,c=4,5,6 相当于：a=4;b=5;c=6

【操作】使用系列解包赋值实现变量交换



**2.2.5常量**

Python不支持常量，即没有语法规则限制改变一个常量的值。我们只能约定常量的命名规则，以及在程序的逻辑上不对常量的值作出修改。

>>> MAX\_SPEED = 120

>>> print(MAX\_SPEED)

120

>>> MAX\_SPEED = 140 #实际是可以改的。只能逻辑上不做修改。

>>> print(MAX\_SPEED)

140

**2.3 最基本内置数据类型和运算符**

**2.3.1 数据类型**

物以类聚，人以群分，数据亦是如此。数据类型是一个集合以及定义在这个集合上的一组操作。数值类型必然都是数字，字符串类型必然都是字符串。当然，还会有一些高级的数据类型，初学者可能不是很容易理解，但相信通过我们的讲解会顺利掌握。

Python语言中的各种数据类型，如下表所示。