python

[写在前面]

初　　心：整理资料，温故而知新
资　　源：马哥 python 视频（李良赠）
Version1：2017 年 3 月 12 日
作　　者：黄晓云

一、基础

1. python 简介

1.1 python 是一种面向对象、解释型计算机程序语言。
1.2 python 优势：
①语法简洁而清晰。通过文本编辑来强制标记代码块，要求程序员在使用 python 编写程序时要能够做到整个代码框架清晰优美；
②python 本身有非常强大的类库。我们可以用非常少的语言调用各类库实现功能，另 python 能够调用 C、C++模块来实现编写。

2. 编程语言

2.1 程序是由**指令和代码**组成，程序也是由**数字结构和算法**组成。即：
程序=指令+代码=数据结构+算法
2.2 变成语言的发展：机器代码———>编程语言-------->高级语言
2.3 python 是一种较为高级的编程语言，python 已非常接近人类的自然语言；python 是一种脚本语言，同样适用于开发大型应用程序。
2.4 无论一种语言多么高级，都是需要许多的**机器代码+编译器/解释器**来完成的。
计算机是帮助人类解决问题的，但是人类使用的是自然语言，而计算机是机器代码，如果将计算机能够解决问题的能力抽象出来，让它越来越接近人类的思维方式，这样使用起来就越来越方便了。但这样的沟通需要一个介质，我们将这个介质称为编译器或解释器，如同声传译。
2.5 编程语言分为面向过程和面向对象。
1）**面向过程的语言：**如何组织代码解决问题，以指令为中心，由指令处理数据。
2）**面向对象的语言：**以数据为中心，所有的处理代码围绕数据展开，考虑如何设计数据结构组织数据，并提供对此类数据所允许的处理操作。
2.6 **在 python 中一切皆对象**，所以我们在用 python 编程时可以按照面向过程来编写程序，类似 bash shell，顺序执行，if 判断条件，循环执行，无需考虑数据结构。

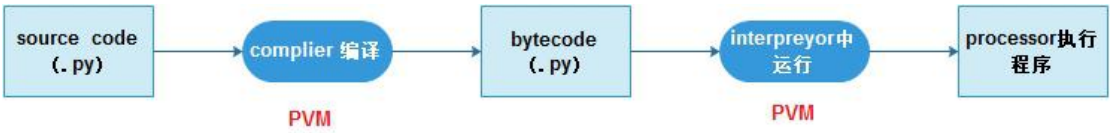
3. python 可以用来作为三种工具来使用：

3.1 当 shell 脚本自动化实现工具，但这个并不是 python 的应用领域，真正实现 linux 的自动化，bash shell 更易于实现。（不建议）
3.2 文本处理，如网页抓取内容的分析，正则表达式的高级处理，python 强大的类库比 awk、sed、grep 要更强大的多。
3.3 “胶水语言”，用来调用 python 或非 python 编译的库。
3.4 框架开发，web，Django

4. Python 与 java、C、C++

python 编译为字节码 bytecode,在类文件的描述上接口统一时，可以与 java 互换执行，前提是有各种可以实现调用的库，像 java 一样，python 可以一次编译到处在不同的环境执行。Python 比起 C 、C++，性能表现上会差大约 30%，但是 python 有丰富的类库和各种便捷的开发工具，使得 python 在开发过程中要快的多。

5. Python 程序的执行过程：



第一次执行时，python 的源代码 source code(.py)文件由编译器 complier 编译成字节码 bytecode(.py)然后在解释器 interpretor 中运行，由 PVM 读取 bytecode，在 processer 中运行。

备注： Python 的源代码都表示为.py 的文件； Python 的字节码表示为.pyc 的文件； Python 的解释器和编译器是 PVM； complier（编译器），interpreter（解释器），precesser（处理器）

6. Python 的性能优化工具

6.1 pypy 是 python 的自实现工具，即用 python 实现的 python 解释器，是 python 语言的动态编译器。
6.2 shed, skin 是 python 编译器，能够将 python 代码转化成优化的 C++代码。

7. 执行，编写 python 代码

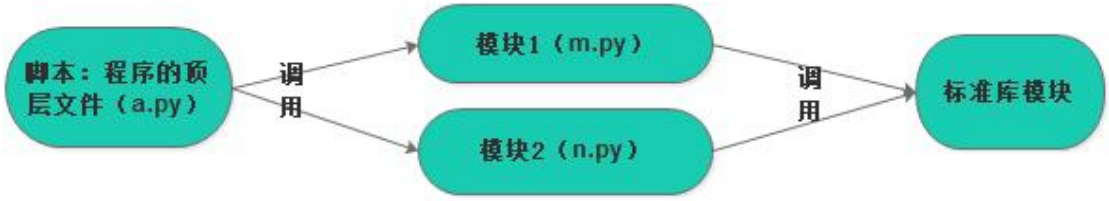
7.1 python 工具

交互的解释器中执行，直接运行 python 即可。
1) 通过 vi/vim 编译器实现；
2) 通过 python 自带的工具 idle；
IDLE 是 python 自带的工具，IDLE 本身也是使用 python 实现的应用程序，如果我们加载 IDLE 后，IDLE 也要自己装在很多 python 库中才可以使用。
Python 的 IDLE 工具：
① Eclipe 和 PyDev
② python win
③ komodo
④ winware
⑤ pycharm（windows 用，何旭东推荐）
3) 通过集成开发工具实现 python 的开发
4)

7.2 python 程序文件

手动开发 python 程序文件时， 在交互式解释器下任何代码都可以直接执行，但执行完成后难以再次执行。如果一段代码我们需要反复多次执行，最好是尽可能的将代码保存在程序文件中，让解释器读取程序文件来运行。

Python 程序文件通常为.py 的文件，在 python 编程中包含了一系列 python 语句的文件，这些文件统统被称为“模块”，能够直接运行模块的文件被称作脚本（即程序的顶层文件）。



7.3 python 源程序文件

```
[root@localhost python-test]# cat p.py
#!/usr/bin/python
import platform
print platform.uname( )
[root@localhost python-test]# chmod +x p.py
[root@localhost python-test]# ./p.py
('Linux', 'localhost.localdomain', '2.6.32-504.el6.x86_64', '#1 SMP Wed Oct 15
04:27:16 UTC 2014', 'x86_64', 'x86_64')
```

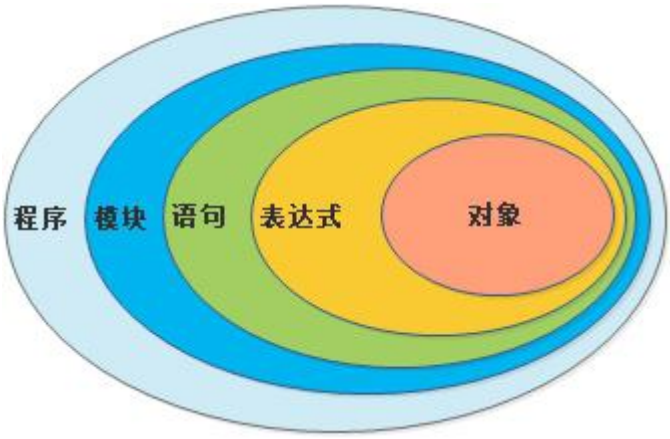
备注：
#!/usr/bin/python 是解释器的路径，即执行脚本时通知内容要启动的解释器；
import 用来导入模块；
print 用于实现程序输出；
chmod +x p.py 添加执行权限；
./py 执行。

注意：
我在本目录删除了.py 文件，但是在系统中仍然存在，需要找到后再次删除。

```
[root@localhost python-test]# ll
total 4
-rwxr-xr-x. 1 root root 58 Mar 12 21:49 p.py
[root@localhost python-test]# rm -rf *
[root@localhost python-test]# ll
total 0
[root@localhost python-test]# find / -type f -name "platform.py"
/usr/lib64/python2.6/platform.py
[root@localhost python-test]# find / -type f -name "platform.py" |xargs rm -f
[root@localhost python-test]# find / -type f -name "platform.py"
[root@localhost python-test]#
```

8. Python 程序文件结构

8.1 python 程序文件可以分解成模块、语句、表达式、对象，程序由模块构成，模块包含结构，语句包含表达式，表达式建立并处理对象。表达式是某事，而语句是做某事。如：“3+4”是某事，而“print3+4”是做某事。



8.2 在 python 中一切皆对象；在 python 中存储的任何一段代码都要存储为对象结构。

9. Python 数据结构

9.1 程序=结构+算法

程序其内部最重要的元素就是数据结构。

9.2 数据结构

数据结构是通过某种方式（如元素进行编号）组织在一起的数据元素的集合。这些数据的元素可以是数字或者字符，甚至是其他的数据结构。

9.3 python 中最基本的数据结构是序列。

序列是可以被索引的；

序列中的每个元素被分配一个序号，即元素的位置，也称为索引；

索引从 0 开始。

9.4 python 包含 6 种内建的数据序列。

列表

元组

字符串

Unicode 字符串

Buffer 对象

Xrange 对象

附加：安装 ipython

0.1 ipython 简介

IPython 是 Python 的交互式 Shell，提供了**代码自动补完，自动缩进，高亮显示**，执行 Shell 命令等非常有用的特性。特别是它的代码补完功能，例如：在输入 zlib.之后按下 Tab 键，IPython 会列出 zlib 模块下所有的属性、方法和类。完全可以取代自带的 bash

0.2 安装 python+ipython

终于开始我的 Python 之旅，一开始发现系统自带的 python 版本为 2.6.6，于是偷了个懒，并没有升级我的 python 版本。然而在实际操作中，被各种路径闹得烦不胜烦，遂通过万能的百度爸爸得知存在 ipython 这一款神器，于是果断查阅相关资料进行安装。安装中遇见一些问题，觉得可能以后重新安装还会遇见，于是决定记录一下。

1）安装 IPython 需求

已经安装好 gcc 等编译工具。系统自带 Python 的版本和安装位置如图所示：

2）安装步骤

① 安装开发库文件

编译 Python 只要有 gcc 编译器就足够了，但一些扩展模块需要额外的库，否则一些 Python 模块将不可用（比如 Python 的 zlib 模块需要 zlib-devel，ssl 模块需要 openssl-devel）。用户可以根据需要，选择性的安装这些扩展模块。这里我们安装 zlib、ssl 和 sqlite3 的库

```
yum install zlib-devel openssl-devel sqlite-devel
```

文件。

② 下载 Python
目前最新版本为 2.7.8。

```
wget https://www.python.org/ftp/python/2.7.8/Python-2.7.8.tgz
tar zxvf Python-2.7.8.tgz
```

③ 编译安装
为了方便管理，将 Python 安装到/usr/local/python 路径下。

```
cd Python-2.7.8
./configure --prefix=/usr/local/python
make
make install
```

④ 替换系统自带 Python
安装后，Python 2.7.8 的可执行文件位于/usr/local/python/bin。先将系统自带的 Python 重命名为 python2.6，再创建新的 Python 到/usr/bin 目录下的符号链接。

```
mv /usr/bin/python /usr/bin/python2.6   #其实不要这句也行，加上了预防万一
ln -sf /usr/local/python/bin/python2.7 /usr/bin/python
```

再执行 python --version，可以看到已经是 2.7.8 版了，如图所示：

⑤ 修正 yum

现在，系统默认调用 Python 2.7.8，执行 yum 会出现错误。

解决方法：将/usr/bin/yum 文件开头的第一行代码：

#!/usr/bin/python

修改为：

#!/usr/bin/python2.6

⑥ 安装 setuptools 和 pip

玩 Python 不能没有 setuptools 和 pip。这里使用 pip 提供的安装脚本，自动安装 setuptools 和 pip。

```
wget https://bootstrap.pypa.io/get-pip.py
python get-pip.py
```

因为之前更改了 Python 的安装目录，Python 扩展模块的可执行文件会被安装到/usr/local/python/bin 目录下。为了能够直接运行这些文件，我们要么创建到/usr/bin 目录下的符号链接，要么将/usr/local/python/bin 目录添加到 PATH 环境变量中。为了避免不必要的冲突，这里采用创建符号链接的方式。

```
ln -sf /usr/local/python/bin/pip /usr/bin/pip
ln -sf /usr/local/python/bin/easy_install /usr/bin/easy_install
```

⑦ 安装 IPython

有了 pip，安装 IPython 简直 So easy。（同样将 IPython 的可执行文件链接到/usr/bin 目录下）

```
pip install ipython
ln -s /usr/local/python/bin/ipython /usr/bin/ipython
```

⑧、安装 readline 扩展模块

没有 readline 支持的 IPython 弱得没法用。这里安装 readline：

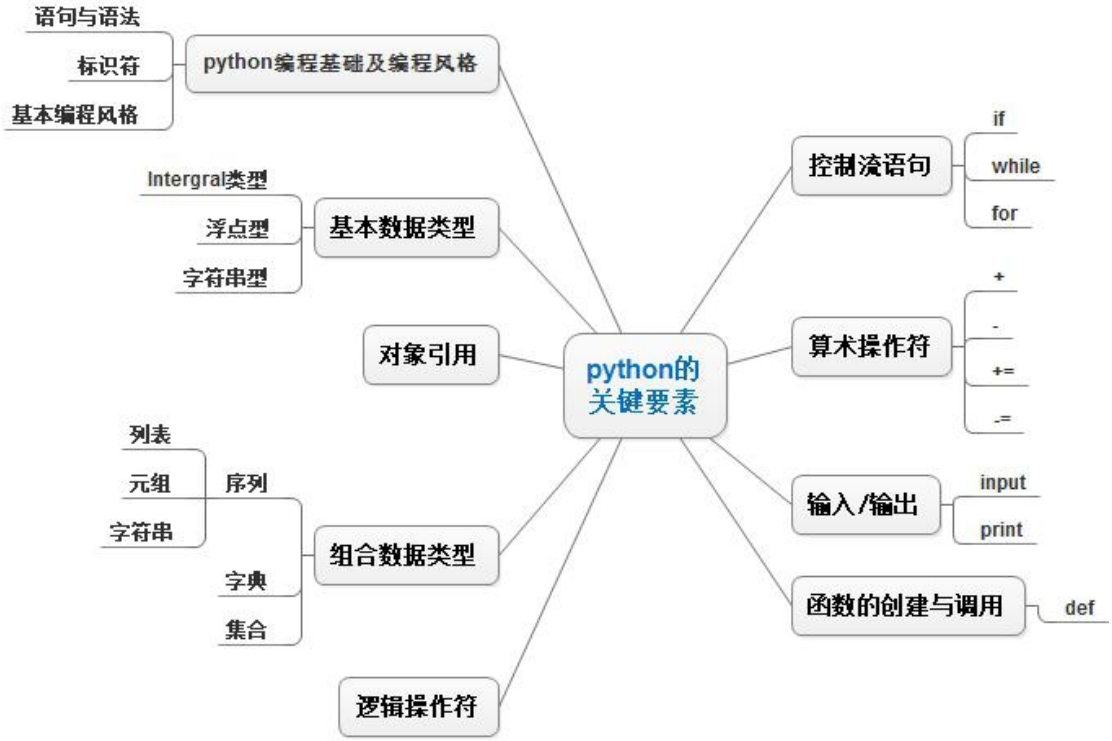
```
yum install readline-devel
yum install patch      #相信我，不安装这个就无法编译 readline
pip install readline
```

之后，IPython 的高亮显示和代码补完等功能就可以用了，如图所示。当然，现在的 IPython 只具备了最基本的功能，还有一些模块没有安装，所以无法使用诸如并行计算、Qt Console、Notebook 等功能。不过这不是我们这篇文章要讨论的内容了。以后有时间会写一篇安装使用 IPython 其他模块和功能的文章。

```
[root@localhost Python-2.7.8]# ipython
Python 2.7.8 (default, Mar 12 2017, 23:38:54)
Type "copyright", "credits" or "license" for more information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]:

In [1]: exit()
```

10. Python 的关键要素

基本数据类型
对象引用
组合数据类型
逻辑操作符
控制流语句 if　for 循环
算术操作符
输入/输出
函数的创建与调用



10.1 python 基本类型

1）任何程序语句都必须能够表示基本数据项；
2）Python 中的基本数据类型有 intergral 类型和浮点类型，以及字符串三种类型；
①intergral 类型分为整形和布尔型
A. 整形，可变类型，在 python 中被保存为对象

```
In [1]: num=1
In [2]: id(num)
Out[2]: 32498008
```

其中 num=1 定位了 num 所引用的对象在内存中存储的地址；
所变化的是 num 变量，指向内存的引用对象发生了变化；
Id 是 python 的内置函数。
• 变量分为可变类型和不可变类型，其中正数为不可变类型。
B. 布尔型只有两种，使用字符串表示 False.True，但是不能加引号
• 在 python 中所有的字符串都要用引号引起来，不区分单双引号，但是必须要有引号；
• 数值不加引号
② 浮点类型
浮点数
复数
十进制数
③ 字符串
A. 使用引号引起来

```
In [8]: name = "Jerry"

In [9]: print name
Jerry

In [10]: print name[1]
e

In [11]: print name[0]
J
```

B. 字符串在 python 中都是序列，可以使用索引、下标来引用
*序列引用从 0 开始，0,1,2,……，字符串也是不可变对象；
*type 可以看 python 中的数据类型

```
In [13]: type(name)
Out[13]: str

In [14]: type(num)
Out[14]: int
```

备注：str 表示字符串，int 表示整数

• python 内置函数小结：

dir 内置函数

id 引用对象在内存中存储位置

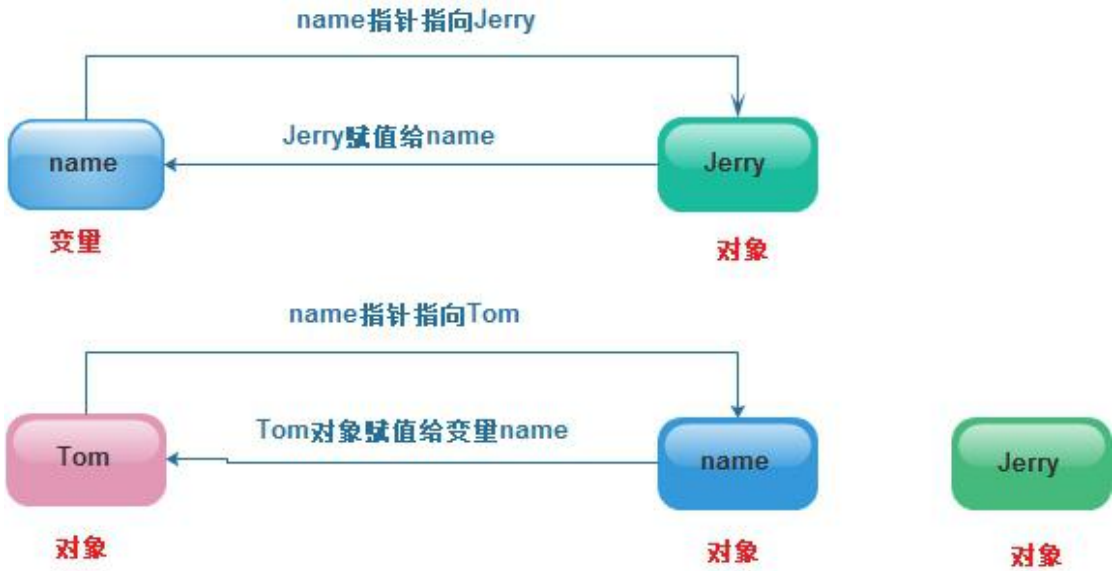
type 数据类型

10.2 对象引用（变量）

1）python 将所有数据都存为内存对象，在内存中是由特定地址的。

python 中所有的变量是指向对象的引用；

python 中变量名本身也是一个对象，也需要存放。



备注：此时，Jerry 仍存在于系统中，只是没有被变量 name 引用。

• python 中，当一个对象没有任何变量引用，即各种内存对象的引用计数为 0 时，会自动由垃圾收集器回收。

3）“=”用于将变量名与内存中的某对象绑定，如果对象事先存在，就直接绑定，否则就由“=”创建引用对象。

```
In [8]: name="Tom"

In [9]: id(name)
Out[9]: 140411931508448

In [10]: print name
Tom

In [11]: test="Tom"

In [12]: id(test)
Out[12]: 140411931508448

In [13]: print test
Tom
```

备注：

test 变量引用的对象 Tom 与 name 变量引用的对象 Tom 的 id 相同，说明变量 test 和 name 引用的对象都是 Tom。

*在 python 中字符串是不可变的；

*在 python 中引用相同的对象（字符串），会同时指向同一个字符串对象，这个可以节省内存空间。

*内建函数 type()用于返回给数据项的数据类型。

4）变量的命名规则

只能包含字母、数字和下划线，且不能以数字开头；

区分字母大小写；

禁止使用保留字；

•保留字即系统中关键字，程序自己为了程序自身正常运行，预留了一些用于其控制结构等以及内置的某些功能的字符串，通常称为保留字或关键字。

python2 与 python3 的保留字有所不同。

5）命名惯例

① 以单一下划线开头变量名（_x）不会被 from module import * 语句导入。（特殊变量尽量不要这样使用）

② 前后有下划线的变量名（_x_）是系统定义的变量名，对 python 解释器有特殊意义。

③ 以两个下划线开头，但是没有下划线结尾的变量名（__x）是类的本地变量。

④ 交互式模式下，变量名“_”用于保存最后表达式的结果。

```
In [14]: 1+1
Out[14]: 2

In [15]: _
Out[15]: 2
```

注意：变量名没有类型，对象才有。

```
In [16]: print name
Tom

In [17]: type (name)
Out[17]: str
```

注意：str 表示变量 name 引用的对象 Tom 的类型是字符串。

10.3 组合数据类型

1）数据结构：通过某种方式，例如对元素进行编号，组织在一起的数据元素的组合。
2）python 常用的组合数据类型
① 序列类型
* 列表，使用[]创建，如[“Call”,”me”,”Yvonne”.]

```
In [20]: name=["Please","call","me","Yvonne"]

In [21]: print name
['Please', 'call', 'me', 'Yvonne']

In [22]: name[0]
Out[22]: 'Please'

In [23]: name[0][0]
Out[23]: 'P'

In [24]: name[3][0]
Out[24]: 'Y'

In [26]: type(name)
Out[26]: list
```

* 列表是可变对象，支持原处修改。
备注：元素可以修改，元组是不允许修改的。

② 元组：使用()创建，如(‘one’,’two’)

③ 字符串也属于序列类型

字符串可以用作切片引用

```
In [27]: name[0:1]
Out[27]: ['Please']

In [28]: name[0:2]
Out[28]: ['Please', 'call']

In [29]: print test
Tom

In [30]: test[0:2]
Out[30]: 'To'

In [31]: test[0:1]
Out[31]: 'T'
```

备注：name[0:2] 可以显示字符串的前两个字符；

name[0:1] 只会显示第一个字符。

***步进和切片**

步进：一次迈出多少字符，这就是字串的切片操作；切片本身会创建新的内存对象。

因为字串本身是不可变的，所以切来的片又生成了新的内存对象，原来的片不发生任何改变。

```
#从 0 开始到 2
In [32]: test[:2]
Out[32]: 'To'

#从 2 开始到最后
In [33]: test[2:]
Out[33]: 'm'

#步进，一步为 2
In [34]: test[0:2]
Out[34]: 'To'

#步进，一步为 4
In [35]: name[0:4]
Out[35]: ['Please', 'call', 'me', 'Yvonne']

#步进，一步为 3
In [36]: name[0:3]
Out[36]: ['Please', 'call', 'me']

#切片，显示 0-4，步进为 2
In [37]: name[0:4:2]
Out[37]: ['Please', 'me']
```

3）集合类型

set 集合，一堆数据杂乱无章的放在一起，称为集合。

4）映射类型

字典：类似键值对，也类似 bash shell 中的关联数组，不使用下标来引用，而是使用自己指定的索引字符来引用。

*python 中组合数据类型也是对象，因此可以嵌套。如：['hello', 'world', (1, 2, 3)]

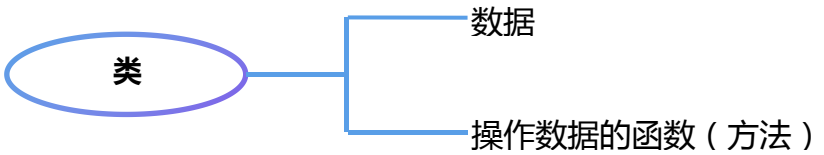
*实质上列表和元组并不真正存数据，而是存放对象引用。

*python 对象可以具有其可以被调用的特定方法（函数）。

*元组、列表以及字符串数据类型是有大小的，即：其长度可以使用内置函数 len()测量

```
In [40]: print(ll)
['hello', 'world', (1, 2, 3)]

In [41]: len(ll)
Out[41]: 3

In [42]: print(name)
['Please', 'call', 'me', 'Yvonne']

In [43]: len(name)
Out[43]: 4

In [44]: print(test)
Tom

In [45]: len(test)
Out[45]: 3
```

备注：此处是说 name 所引用的对象包含了 4 个元素。
列表是可变序列；
列表['Please', 'call', 'me', 'Yvonne']中存放的并不是“please”等字符串，而是引用了内存中的字符串“please”（这些字符串都是存在内存中的）。



当我们用类实例出一个对象时，类本身只是描述一个抽象的属性，只有把抽象属性实例出来才叫对象。而对象本身只包含数据，而没有方法；方法是属于类的，所以我们可以对对象调用方法，方法是来自于他所包含的类的，而不是来自于对象本身。
所以类中的方法可以对同一个类的不同对象都能进行处理，这个我们成为方法。
* python 中一切皆对象，而对象是由类实例出来的。

```
In [13]: type(name)
Out[13]: str

In [14]: type(num)
Out[14]: int
```

备注：str 本身可以理解为一个内置的类型，通过内置的类实例出了一个对象，而这个对象时支持某些方法调用的，都是有特定的方法，而这些方法是可以对当前对象完成某些处理的。

10.4 逻辑操作符

逻辑运算是任何程序设计语言的基本功能。

1）python 提供了四组逻辑运算

① 身份操作符 is

is ：判断左端对象引用是否相同于右端对象引用，也可以与 none 进行。

判断两个对象引用是否是同一个。

```
In [46]: hxy="yvonne"

In [47]: huangxy="yvonne"

In [48]: hxy is huangxy
Out[48]: True

In [49]: name is hxy
Out[49]: False
```

备注：说明 hxy 和 huangxy 引用了同一个对象。

```
In [51]: print hxy
yvonne

In [52]: print test
Tom

In [53]: hxy is test
Out[53]: False

In [54]: type(hxy) is type(test)
Out[54]: True
```

备注：对象引用可能不同，但对象所引用的对象类型是相同的。

10.5 控制流语句

1）控制流语句是过程式编程语言的基本控制机制。

2）python 的常见控制流语句

① if

语法

if 条件表达式：

模块

② **while**
语法
while 条件表达式：
模块
③ for ... in ...
便利方式来利用对象中的所有元素
③ try
异常捕获

10.6 算术操作符

1）python 提供了完整的算术操作符集。
2）很多的 python 数据类型也可以用增强的赋值操作符，如**+=，-=**
同样的功能，使用增强型赋值操作符的性能较好。
3）python 的 int 类型是不可变的，因此增强型赋值的实际过程是创建了一个新的对象来存储结果后，将变量名执行了重新绑定。

备注：m+1 是将 m 指向了新的对象。

```
In [55]: m=58

In [56]: print m
58

In [57]: m=m+1

In [58]: print m
59
```

10.7 输入/输出

1）显示中具有实际功能的程序必须能读取输入（如键盘或文件中）以及产生输出，并写到终端或文件中。
2）python 的输入输出
① 输出
python3：print()函数
python2：print 语句

② 输入
input()
raw_input()

```
In [59]: raw_input("Please input a num:")
Please input a num:88
Out[59]: '88'

In [60]: a=raw_input("Please input a name:")
Please input a name:yvonne

In [62]: print(a)
yvonne
```

备注：现在将 raw_input 对象赋值给了 a，后续调用 a 引用的对象就很方便了。

③ python 解释器提供了三种标准文件对象，分别为**标准输入、标准输出和标准错误**。
他们在 sys 模块中分别以 sys.stdin , sys.stdout , sys.stderr 形式提供。
④ python 的 print 语句实现打印——一个对程序员友好的标准输出流接口。
⑤ 从技术角度来讲，print 是把一个或多个对象转换为其基本表达形式，然后发送给这个标准输出或另外一个类似文件的流。
A. 在 python 中，打印与文件的流的概念联系紧密。
文件写入方法是把字符串写入到任意文件；
print 默认把对象打印到 stdout 流，并添加了一些自动的格式化。
B. 实质上，print 语句只是 python 的人性化特征的具体体现，他提供了一些默认的格式设置。
C. print 接受一个逗号分隔的对象列表，并为行尾自动添加一个换行符，如果不需要则在最后一个元素添加逗号。
⑥ print 语句输入/输出
print “字串 格式化 1，格式化 2，……” %（变量 1，变量 2，……）
print “string %fromat1, %format2.....” %(avariable,vaarible2....)

字符	输出格式
d,i	十进制整数或长整数
u	无符号整数或长整数
o	八进制整数或长整数
x	十六进制整数或长整数
X	十六进制整数（大写字母）
f	浮点数，如[-]m.dddddd
e	浮点数，如[-]m.ddddde±xx
E	浮点数，如[-]m.dddddE±xx
g,G	指数小于-4 或更高精度时使用%e 或%E，否则使用%f
s	字符串或任意对象，格式化代码使用 str()生成字符串
r	用 repr() 生成的字符串
c	单个字符
%	字面量%

```
In [69]: num=7.9

In [70]: print num
7.9

In [71]: print"The num is %f" %num
The num is 7.900000

In [72]: print"The num is %d" %num
The num is 7
```

备注：以上为只有一个对象的情况。

当引用两个变量时，需要有两个对象引用。

```
In [72]: print"The num is %d" %num
The num is 7

In [73]: num1=15

In [76]: print"The nums are %d and %f" % (num,num1)
The nums are 7 and 15.000000

In [77]: num2=22

In [78]: num3=33

In [79]: print num1
15
```

数据类型在 python 中是需要转换的，可以为显示的，也可以转换为隐示的。

⑦ 查看内置函数

```
#查看内置函数：
In [100]: dir(str)
Out[100]:
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__doc__',
 '__eq__',

#查看内置函数帮助：
In [101]: help(str)
Help on class str in module __builtin__:
```

⑧ 对类施加方法

```
In [102]: name="tom"

In [103]: name.upper()
Out[103]: 'TOM'

In [104]: print name
tom
```

备注：对变量 name 引用的对象 tom 施加方法。

每一个对象都属于某一个类，而类只要有方法，我们就可以对此对象施加属于类的方法。一个类所能够接受的操作方法就决定了这个类所实例出来的对象所支持的操作或运算。

⑨ %后面可以使用的修饰符（如果有，则只能按如下顺序）

% [(name)][flags][width][.precision] typecode

说明：

[(name)] 位于括号中的一个属于后面的字典的键名，用于造出一个具体项。

[flags] 下面标志中的一个或多个：

- \- 表示左对齐，默认为右对齐
- \+ 表示包含数字符号，正数也会带+
- 0 表示一个零填充

[width] 一个指定最小宽度的数字

[. precision] 一个小数点，同时按照精度分割字段的宽度

[. precision] 一个数字，指定要打印字符串中最大的字符串个数，浮点数中小数点后的位数，或者整数的最小位数。

d={'x':'32','y':27.4096,'z':65}

* python 也可以用 history 查看历史记录。In [118]: %history

删除变量 In [120]: del test

10.8 函数的创建与调用

1）函数是实现模块化编程的基本组件
2）python 使用 def 语句定义函数

```
def Function Name (arguments): suite
```

函数名　　参数　　主体部分

python 中同一代码块的各式代码缩进保持一致。

```
In [3]: def printName(name):
   ...:         print name
   ...:

In [4]: printName('Tony')
Tony
```

对象引用，这种引用机制是传递了引用机制来实现任何一个对象的内置方法决定了它所能支持的运算。

3）函数可以参数化，通过传递不同的参数来调用。
4）每个 python 函数都有一个返回值，默认为 none，也可以使用“return value”明确定义返回值。
5）def 语句会创建一个函数对象，并同时创建一个指向函数的对象引用。

A. 函数也是对象，可以存储在组合数据类型中，也可以作为参数传递给其他函数。
B. callable() 可用于测试函数是否可调用。

```
In [25]: def printName(name):
    ...:         print name
    ...:

In [26]: printName('test')
test

In [27]: print name
yvonne

In [29]: callable(name)
Out[29]: False

In [30]: callable(printName)
Out[30]: True
```

6）python 有许多内置函数：　dri(), id(), type(), str(), help(), len(), callable(), range()

```
In [31]: range(10)
Out[31]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

7）python 标准库拥有众多内置模块，这些模块拥有大量函数

A. python 模块实际上是包含 python 代码的.py 文件。其拥有自定义的函数、类及变量等。

B. 导入模块使用 import 语句进行，后跟模块名称（不能指定模块文件名的后缀.py）。

C. 导入一个模块后，可以访问其内部包含的任意函数、类及变量。

```
In [32]: import random

In [33]: x=random.choice(['Yvonne','hxy','lanzhi'])

In [34]: print x
lanzhi

In [35]: print x
lanzhi
```

备注： random.random　随机生成

random.choice　随机选择

```
In [38]: print(random.random())
0.231548382884
```

10.9 python 编程基础及编程风格

1）语句和语法

①注释

可以从一行的任何地方开始（只要是#都表示注释符，都被解释器忽略）

② 续行

\ 续行

‘’’ 闭合操作符，单一语句跨多行

③ 代码组

A. 缩进相同的一组语句构成一个代码块

B. 行首以关键词开始，如 if, while, 以冒号结束

C. python 使用缩进来分隔代码组，同一代码组的代码行必须严格左对齐，否则会造成语法错误。

④ 同一行放置多个语句

；以分号作为分隔符

⑤ 模块

A. 每一个 python 脚本文件都可以被当做是一个模块，模块名称就是脚本名称。

B. 模块的代码可以是一段直接执行的脚本，也可以是一些类似库函数的代码，从而可以由别的模块执行导入（import）

注：模块中通常都是可调用的代码段，而不是可执行的语法。

```
[root@localhost python-test]# cat shisi1.py

#!/usr/local/python/bin/python
#
name="Jerry"

def printName(a):
        print a

num=3
print num
[root@localhost python-test]# python shisi1.py
3
```

```
[root@localhost python-test]# cat shisi1.py

#!/usr/local/python/bin/python
#
name="Jerry"

def printName(a):
        print a

num=3
print num
print name
[root@localhost python-test]# python shisi1.py
3
Jerry
```

2）标识符

① 标识符是计算机语言中允许作为名字的有效字符串集合

A. 其中有一部分是关键字，它是语言的标识符，因此是保留字，不能用于其它用途。

B. python 还有标为“内建”的标识符集合，虽不是保留字，仍不推荐使用这些特别的名字。

② python 标识符

A. 第一个字符只能使用字母或下划线

B. 余下的字符可以使用字母、数字或下划线

C. 区分字符大小写

D. python 的内置保留字：

False, None, True, and, as, class, continue, def, del, elif, finally, for, from, for, from, global, if, is, lambda, nonlocal, not, not, or, return, try, while, with, yield, assert, else, import, pass, break, except, in, raise

3）python 的基本编程风格

① 注释

A. 既不能少注释，亦要避免过多注释

B. 一般写代码是一定要注释的；一个好程序员是一定要写注释的。

② 文档

python 允许通过_doc_动态获取文档字串

```
In [14]: str. _doc_
或者可以用 print 来表示：
In [15]: print str._doc_
```

③ 缩进：统一缩进 4 个字符串

④ 标识符名称：标识符的名称要见名之意，用完整的单词表示

⑤ python 风格指南……/wiki/PythonStyleGuide 及其译文

4）python 命令惯例

① 以单一下划线开头的变量（_x）不被 from moddle import * 语句导入

② 前后有下划线的变量名（_ _x_ _）是系统变量名，对解释器有特殊意义。
解释器会自动的将某些操作运算转化成内置的调用。

```
In [16]: a="tom"

In [17]: b="jerry"

In [18]: print a+b
tomjerry

In [19]: print a.__add__(b)
tomjerry

In [20]: c="yvonne"

In [21]: print a+b+c
tomjerryyvonne
```

③ 以两个下划线开头，但没有下划线结尾的变量名（_ _x）是类的本地变量。
④ 交互模式下，只有单个下划线的变量名（_）用于保存最后表达式的结果。

5）python 主文件程序
① 主程序
A. 无论当前模块是被别的模块导入还是作为脚本直接运行，都会执行这行代码。
B. 注意：所有的模块都有能力执行代码
a. 最高级别的 python 语句（没有缩进的）在模块被导入时就会执行，无论是否真的需要执行。
b. 妥当的做法：除了那些真正需要执行的代码以外，所有的功能代码都通过函数建立。因此：
a). 仅在主程序模块中编写大量的顶级可执行代码；
b). 用于被导入的模块只应该存在较少的顶级代码。
C. _ _name_ _ 指示模块应该如何被加载
a. 如果模块是被导入，_ _name_ _的值是模块名称
b. 如果模块是直接执行，_ _name_ _的值是'_ _main_ _'

11. python 对象类型及其运算

11.1 python 对象的相关术语

1） python 程序中保存的所有数据都是围绕对象这个概念展开的。
① 程序中存储的所有数据都是对象。
② 每个对象都有一个身份，一个类型和一个值。
A. "身份" 指的就是这个对象在内存中的存储位置。

```
In [24]: name="Jerry"

In [25]: print name
Jerry

In [26]: id(name)
Out[26]: 140710718866656
```

B. “一个类型”，任何对象都是由所谓的类实例化而来。

```
In [27]: type(name)
Out[27]: str
```

C. “一个值”每一个对象内部都存有一个数值，即数据。所以每个对象都应该有一个值，但值根据对象类型的不同，它的值不同，有的是整数，有的是列表。

D. 例如：hxy=”Huang Xiaoyun”会以 Huang Xiaoyun 创建一个字符串对象，其身份是指向它在内存中所处位置的指针（其在内存中的地址），而 hxy 就是引用这个具体位置的名称。

③ 对象的类型也称对象的类别。用于描述对象的内部表示以及它支持的方法和操作。

一般而言，对象所属的类型决定了对象所支持的操作。如：In [28]: num=5 此时 num 所引用的对象 5 可以支持参与加减乘除等操作；若是把 5 换成字符串，如：In [31]: num="yvonne" 此时 num 所引用的对象 yvonne 支持参与的加法操作实际上是字符串的链接。请看一下操作：

```
In [32]: num1=5

In [33]: num2=6

In [34]: num=num1+num2

In [35]: print num
11

In [36]: name1="yvonne"

In [37]: name2="and"

In [38]: name3="spencer"

In [39]: name=name1+name2+name3

In [40]: print name
yvonneandspencer
```

注：python 属于强类型的运算，只要运算不符合，就会报错

④ 创建特定类型的对象时，有时也将对象称为该类型的实例。

class 自定义类型
type 内置类型
instance 实例化
⑤ 在实例被创建后，其身份和其类型就不可变。
A. 如对象值是可修改的，则称为可变对象。如果对象值是不可修改的，则为不可变对象。
B. 列表、字典都属于可变对象；数字、字符、元组都属于不可变对象。
⑥ 如果某个对象包含其它对象的引用，则将其称为容器。如列表，可以包含数值、字符、其他列表等。
⑦ 大多数对象都拥有大量特有的数据属性和方法。
A. 属性：与对象相关的值（对象内部的可用变量，就叫做对象的属性）
B. 方法：被调用时将在对象上执行某些操作的函数。
C. 使用点（.）运算符可以访问属性和方法。

类
- 数据　　表现为 变量 -------->通过实例化对象时赋值得来
- 方法　　实质是　函数 ------> 处理数据的代码段

注意：一般来讲，方法是公共的，数据时私有的。

11.2 对象的身份和类型

1) python 中内置函数 id()可以返回一个对象的身份，即该对象在内存中的位置。
① is 运算符用于比较两个对象的身份。两个对象的之间的比较有三种方法：
A. 比较两个对象是否为同一个对象（存储数据是否相同-----身份比较）；
B. 比较两个对象的值是否相等（值比较）；
C. 比较两个对象的类型是否相同（类型比较）。

```
#值相等
In [41]: num1=5

In [42]: num2=4+1

In [43]: num1==num2
Out[43]: True
```

```
#引用了同一个对象
In [44]: num1 is num2
Out[44]: True
```

备注：值相同，对象未必一样。

```
#类型一样
In [45]: type(name) is    type(name1)
Out[45]: True

In [46]: print name
yvonneandspencer

In [47]: print name1
yvonne

In [48]: name is name1
Out[48]: False
```

② 对象类型本身也是一个对象，称为对象的类。

A. 该对象的定义是唯一的，且对于某类型的所有实例都是相同的

B. 所有类型对象都有一个指定的名称，可用于执行类型检查，如 list, dict

```
In [51]: if a is b:
    ...:         statements
    ...:         if a==b:
    ...:             statements
    ...:             if type(a) is type(b):
    ...:                 statements
    ...:

In [52]:
```

11.3 python 的核心数据类型

序号	对象类型	例子
1	数字	3001, 3.14,300000
2	字符串	“yvonne”, ”Huang xiaoyun”, ”www.yvonne.com”
3	列表	[‘yvonne’,’hxy’,’lanzhi’]
4	字典	{‘course’:’linux’,’hxy’:’girl’}
5	元组	[32,’eggs’,’food’]
6	文件	myfile=open(‘/tmp/tfile’,’r’)
7	集合	set(a,b,c),{‘a’,’b’,’c’}
8	其他类型	类型，None,布尔型
9	编程简单模型	函数，模块类
10	与实现相关的类型	编译的代码堆栈跟踪

1）数字：整数、浮点数、复数

int(整型)，long(长整型)， float(浮点型)， complex(复数)， bool(布尔型)
2）字符型：str, unicode
3）列表：list
4）字典：dict
5）元组：tuple
6）文件：file
7）集合：set(集合)，frozenset(不可变集合)
8）其他类型：类类型，None（空）
9）其他文件类工具：pips（管道），fifos(先进先出管道)，sockets(套接字文件)
* 类型转换常用的内置函数：str(), repr() 或 format()，用于将非字符串转换成字符串，其中 str() 的输出和 print 的结果是一样的，是序列化的结果；
repr() 主要功能是表示程序中某个对象的精确值；
format 是利用特定格式转化成字符串。

```
In [52]: print(num)
11

In [53]: str(num)
Out[53]: '11'

In [54]: repr(num)
Out[54]: '11'

In [55]: format(name)
Out[55]: 'yvonneandspencer'

In [56]: format(num)
Out[56]: '11'

In [57]: type(num)
Out[57]: int

In [58]: num1=str(num)

In [59]: print num1
11

In [60]: type(num1)
Out[60]: str

In [61]: type(repr(num))
Out[61]: str

In [62]: type(format(num))
Out[62]: str

In [65]: format(num) is str(num)
Out[65]: False

In [66]: repr(num) is num
Out[66]: False

In [68]: type(str(num)) is    type(format(num))
Out[68]: True
```

疑问：以下 In [72]: id(format(num))和 In [70]: id(str(num))的 id 为何一样？

```
In [69]: id(num)
Out[69]: 30249064

In [70]: id(str(num))
Out[70]: 140710718197152

In [71]: id(repr(num))
Out[71]: 140710718197232

In [72]: id(format(num))
Out[72]: 140710718197152
```

* 类型转换：
str() , repr() , format , 将非字符型数据转换为字符；
int() , 转换为整数；
float() , 转为浮点数；
list(s) , 将字符串转换为列表（将字符串中每一个元素当做列表中的元素，空格也是一个元素）。
tuple(s) , 将字符串转换成元组，（切割字符串中的每一个元素为元组）
set(s) , 将字符串 s 转换为集合对（可变动）。
dict(d) , 创建字典，（必须是键值对元组序列。key,value）.
frozenset(s), 将字符串 s 转换为不可变集合。
chr(x) , 将整数转换成单个字符。
ord(x) , 将字符转换为整数值。
hex(x) , 将整数转换成二进制字串。
bin(x) , 将整数转换成二进制字串。
oct(x) , 将整数转换成八进制字串。

```
In [6]: name=1

In [7]: print name
1

In [8]: name="Hello world!"

In [9]: type(name)
Out[9]: str

In [10]: list(name)
Out[10]: ['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!']

In [18]: list_name=list(name)

In [19]: print(list_name)
['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!']

In [20]: type(list_name)
Out[20]: list

In [21]: type(list(name))
Out[21]: list

In [22]: type(name)
Out[22]: str

In [12]: print name
Hello world!

In [23]: tuple(name)
Out[23]: ('H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd', '!')

In [24]: set(name)
Out[24]: {' ', '!', 'H', 'd', 'e', 'l', 'o', 'r', 'w'}
-----------------------------
In [13]: num=2

In [14]: chr(num)
Out[14]: '\x02'

In [15]: bin(num)
Out[15]: '0b10'
```

```
In [17]: hex(num)
Out[17]: '0x2'
```

11.4 数字类型操作

1）python 的数字字面量：布尔型，整数，浮点数，复数。

A. 布尔型（所有数字类型均为不可变）

True:1

False:0

2）数字操作

操作 1	描述	操作 2	描述
x+y	加法	x**y	乘方(x^y)
x-y	减法	x%y	取模（xmody）
x*y	乘法	-x	一元减法
x/y	除法（传统除法，真除法）	+x	一元加法
x//y	截断除法（floor 除法）	【红色字体】	不懂

备注：以上为数字类型的算术运算。

3）比较运算（移位和位移）

操作 1	描述	操作 2	描述
x<<y	左移	x|y	按位或
x>>y	右移	x^y	按位异或
x&y	按位与	~x	按位求反

```
按位异或，相同则为 0，不同则为 1
In [35]: 1^0
Out[35]: 1

In [36]: 1^1
Out[36]: 0
```

```
In [28]: num7=2

In [29]: num7<<3
Out[29]: 16

In [30]: num8=81

In [31]: num8>>2
Out[31]: 20
```

```
In [32]: True & False
Out[32]: False

In [33]: 255 & 101
Out[33]: 101

In [34]: 255 | 101
Out[34]: 255
```

11.5 用于表示数据的内置类型

类型分类	类型名称	描述
None	type（None）	null 或对象 None
数字	int	整数
	Long	任意精度的整数（仅在 python2 中）
	float	浮点数
	complex	复数
	bool	布尔值（True 或 False）
序列	str	字符串
	unicode	unicode 字符串（仅在 python2 中）
	list	列表
	tuple	元组
	xange	xange()函数创建的整数范围（在 python3 中为 range）
映射	dict	字典
集合	set	可变集合
	frozenset	不可变集合

11.6 序列类型

1）序列表示索引为非页整数的有序对象集合，包括字符串、列表和元组。
A. 字符串是字符的；
B. 列表和元组是任意 python 对象的序列。
2）字符串和元组属于不可变序列，而列表则支持插入、删除和替换元素。
3）所有序列都支持迭代。
备注： 在 python 中有一个库，是专门和数值相关的，
math,用来实现算术的高级算术运算。

```
In [51]: import math

In [52]: math.pi
Out[52]: 3.141592653589793

In [56]: print (num)
2

# 相当于 2 开平方
In [57]: math.sqrt(num)
Out[57]: 1.4142135623730951

In [58]: num2=81

In [59]: math.sqrt(num2)
Out[59]: 9.0
```

4）字符类型

A. 字符串字面量：把文本放入单引号、双引号或三引号中（前后符号需要一致，单双引号必须在同一行，三引号可以跨行），另外，三引号中所引用的字串包括空格、回车等所有的控制符号。三引号可以使用三个单引号，也可以使用三个双引号。

B. python 中不支持 unicode，所以需要前面加 u 才能成为 unicode 类型，如果要使用 unicode 编码，则在字符串前面加 u 进行标识。如下：

```
In [61]: print name
Hello world!

In [62]: name0=u"Hello world!"

In [63]: type(name)
Out[63]: str

In [64]: type(name0)
Out[64]: unicode
```

C. 文档字串：模块、类或函数的第一条语句是一个字符的话，该字符串就成为文档字符，可以使用_ _dco_ _属性引用。

备注：加上()表示调用函数，不加()表示引用函数对象。因为函数本身也是作为对象存在内存中的。()小括号叫调用运算符。

```
In [67]: def printName():
    ...:            "test function"
    ...:            print"hello word!"
    ...:

In [68]: printName()
hello word!

In [70]: printName().__doc__
hello word!
```

11.7 适用于所有序列的操作和方法

* 因为字串是一种序列，那么所有适用于序列的操作和方法都适用于字串。
1）s[i]　索引运算术
2）s[i:j]　切片运算符
　s[i:j:strige] 扩展切片运算符
3）min(s) 和 max(s) 只适用于能够对元素排序的序列。
注意：是按照其在 ASC 码表中次序比较的。

```
In [71]: n="9875489"

In [72]: min(n)
Out[72]: '4'

In [73]: max(n)
Out[73]: '9'
```

4）sum(s)只适用于数字序列，求其中各项之和。

```
In [75]: m=[2,4,6,8,10]

In [76]: sum(m)
Out[76]: 30

In [77]:

In [77]: a=range(1,10)

In [78]: b=range(1,11)

In [79]: s=sum([item for item in a if item in b])

In [80]: print s
45
```

☺ 是不是很有趣呢？O(∩_∩)O~~

```
In [81]: print(name)
Hello world!

In [82]: all(name)
Out[82]: True
```

5）适用于所有序列的操作和方法

项目	描述	项目	描述
s[i]	返回一个序列的元素 i	min	s 中的最小值
s[i:j]	返回一个切片	max	s 中的最大值
s[i:j:stride]	返回一个扩展切片	all(s)	检查 s 中的所有项是否为 True
lens(s)	s 中的元素数	any(s)	检查 s 中的任意项是否为 True
sum(s[initial])	s 中各项的和		

6）适用于可变序列的操作和方法

项目	描述	项目	描述
s[i]=v	项目赋值	del s[i]	项目删除
s[i:j]=t	切片赋值	del s[i:j]	切片删除
s[i:j:stride]=t	扩展切片赋值	del s[i:j:stride]	扩展切片删除

11.8 适用于字串的操作

1）python 提供两种字符串对象类型

A. 字节字符串：字节（8bit 数据）序列

B. unicode 字符串：unicode 字符（16bit 数据）序列

a. python 可以使用 32bit 数据保存 unicode 字符，但此为可选特性。

方法	描述
s.captitalize	首字符变大写
s.index(sub[,start [,end]])	找到指定字符串 sub 出现的位置，否则报错
s.join(t)	使用 s 作为分隔符，链接序列中的字符串
s.lower	转换为小写形式
s.replace(old,new,[maxreplace])	替换一个子字符串
split([sep[maxsplit]])	使用 sep 作为分隔符对一个字符串进行划分，maxsplit 是划分的最大次数
s.strip([chrs])	删除 chrs 开头的和结尾的空白或字符
s.upper	将一个字符串转换为大写形式

操练：

① 将字符串转换为列表

```
In [83]: str1="str1 change to list"

In [84]: print(str1)
str1 change to list

In [85]: list1=list(str1)

In [86]: print(list1)
['s', 't', 'r', '1', ' ', 'c', 'h', 'a', 'n', 'g', 'e', ' ', 't', 'o', ' ', 'l', 'i', 's', 't']
```

备注：以空格’’为分隔符，将 list1 合并为字符串。

②将列表转换为字符串

```
In [93]: ''.join(list1)
Out[93]: 'str1 change to list'
```

③ 替换

```
In [95]: name="I am yvonne."

In [96]: name.replace('yvonne','Huang Xiaoyun')
Out[96]: 'I am Huang Xiaoyun.'

In [97]: print name
I am yvonne.

In [98]: name.replace('yvonne','Huang Xiaoyun')
Out[98]: 'I am Huang Xiaoyun.'
```

```
In [99]: questiion="Tom,where is you mather?"

In [100]: questiion.replace('m','M')
Out[100]: 'ToM,where is you Mather?'
```

备注：如果替换单个字符 M ，会将所有的 M 替换掉。
M、’,’ 、‘.’ 为对字符串进行划分。

④ 删除字串两端的指定字符

```
In [109]: nn="Marry and I both like roller stating but people more like Marry"

In [110]: nn.strip('Marry')
Out[110]: ' and I both like roller stating but people more like '
```

备注：默认()是删除两端的空格。

2) 支持运算：索引、协议，min().max(),len()
支持操作：对象的自有方法，如 islower, isupper, coptitalize, replace, slip, strip
3) 如果用到某个内置方法：
dir(str) 获取其属性方法列表，然后再用 help()针对某一特定方法获取其帮助；
如：help(str.find) 就可以查看其帮助。
4) 列表：是容器类型
内容包含任意对象的有序集合，通过索引访问其中的元素，是可变对象，支持异构、任意嵌套。支持原处修改，修改指定索引元素，指定的分片，删除语句、内置方法。

① 空列表

```
In [112]: null=[]

In [113]: print null
[]

In [114]: id(null)
Out[114]: 140356588428608

In [115]: type(null)
Out[115]: list
```

② 异构列表

```
In [116]: di=[1,'d']

In [117]: print(di)
[1, 'd']
```

③ 辅加指定列表

```
In [149]: list1=range(10)

In [150]: print list1
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [151]: list2=list(name)

In [152]: print list2
['y', 'v', 'o', 'n', 'n', 'e']

In [153]: list2.append(list1)

In [154]: print list2
['y', 'v', 'o', 'n', 'n', 'e', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]]
```

```
④ 修改
In [121]: name="hxy"

In [122]: print (name)
hxy

In [123]: name="yvonne"

In [124]: print (name)
yvonne


In [125]: num=[1,2,3,4,5]

In [126]: num[2]="yvonne"

In [127]: print num
[1, 2, 'yvonne', 4, 5]
```

5）删除分片或某些元素，只需要其分片或元素定义为空列表 [] 即可。
无论如何修改，其 id 和 name 是不会变的。

```
① 删除指定分片或元素
In [128]: print num
[1, 2, 'yvonne', 4, 5]

In [129]: num[3]=[]

In [130]: print num
[1, 2, 'yvonne', [], 5]
```

```
② 辅加指定元素和
In [132]: num.append(100)

In [133]: print num
[1, 2, 'yvonne', [], 5, 100]
```

备注：此时是嵌套了 list1 整个列表，而非 list1 列表中的元素。

6) help(list) 中比较重要的参数：

参数	说明	参数	说明
append	后面附加指定元素和列表	insert	insert object before index
count	计数指定元素出现的次数	pop	弹出一个元素赋值给新的变量
extend	后面添加指定列表中的元素	remove	移除列表中的指定元素
index	引用某元素的次数	reverse	
sort	正向排序		

```
① extend 后面添加指定列表中的元素
In [156]: list2.extend(list1)

In [157]: print list2
['y', 'v', 'o', 'n', 'n', 'e', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

② **append** 后面附加指定元素和列表
count 后面添加指定列表中的元素
index 计数指定元素出现的次数

```
In [168]: ll2=range(5)

In [169]: print ll2
[0, 1, 2, 3, 4]

In [171]: ll2.count(list1)
Out[171]: 0
#还没有引用 list1

In [172]: ll2.append(list1)

In [174]: ll2.extend(list1)

In [175]: print ll2
[0, 1, 2, 3, 4, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [176]: ll2.count(list1)
Out[176]: 1
#列表 ll2 中 list1 出现了一次

In [177]: ll2.index(list1)
Out[177]: 5
#操作过程中引用了 list1 共 5 次

In [178]: ll2.index(list1)
Out[178]: 5
```

③ **POP** 弹出一个元素赋值给变量（默认为最后一个）

```
In [182]: print ll2
[0, 1, 2, 3, 4, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [183]: ll3=ll2.pop()

In [184]: print ll3
9

In [185]: print ll2
[0, 1, 2, 3, 4, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 0, 1, 2, 3, 4, 5, 6, 7, 8]
```

④ remove　移除列表中的指定元素

```
In [186]: ll2.remove(list1)

In [187]: print ll2
[0, 1, 2, 3, 4, 0, 1, 2, 3, 4, 5, 6, 7, 8]

In [188]: ll2.remove(4)

In [189]: print ll2
[0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7, 8]
```

⑤ sort 正向排序

```
In [190]: print ll2
[0, 1, 2, 3, 0, 1, 2, 3, 4, 5, 6, 7, 8]

In [191]: ll2.sort()

In [192]: print ll2
[0, 0, 1, 1, 2, 2, 3, 3, 4, 5, 6, 7, 8]
```

7）相加操作

list1+list2　合并两个列表，返回一个新的列表，不会修改原列表；

str1+str2　两个字符串也可以合并相加；字符串*3,重复三次，生成新列表或新字符串。

8)　in　成员关系判断符，用法：

in　　intem in container

not in　　obj not in container

```
In [195]: print list1
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [196]: print list2
['y', 'v', 'o', 'n', 'n', 'e', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9], 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

In [197]: list1 in list2
Out[197]: True

In [198]: list1 not in list2
Out[198]: False
```

18. 序列类型之元组

18.1 元组

1）表达式符号:()
2）容器类型：任意对象的有序集合，通过索引访问其中的元素，不可变对象，长度固定，支持异构、嵌套。
3）常见操作：
A. 定义一个空元素 t1=()
B. 定义单个元素 t2=1 或 t2=(1)
C. 定义为元组 t3=('x','y','z') 或 t3='x','y','z'
D. 支持运算 + ,*,in ,如：t1+t2, t3*N
备注：元组本身不可变，但如果元组内嵌套了可变类型的元素，那么此类元素的修改不会返回新元组。

18.2 序列操作总结

1) 序列类型支持的操作符

操作	描述	操作	描述
s+r	连接	x in s,x not in s	从属关系
any(s)	如果 s 中的任意项为 True，则返回 True	all(s)	如果 s 中的所有项都为 True，则返回 True
s[i]	索引	for x in s	迭代
s[i:j]	切片	s*n,n*s	制作 s 的 n 个副本，n 为整数
s[i:j:stride]	扩展切片	V1,V2,....Vn=S	变量解包

操作	描述	操作	描述
len(s)	长度	min(s)	s 中的最小项
sum(s[,initial])	具有可选初始值的项的和	max(s)	s 中的最大项

19. 字典 dict

1）字典在其他编程语言中国又称作关联组或者散列表。
① 特点：通过键实现元素存取元素集合，可变类型容量，长度可靠，支持异构、嵌套。
② 格式：{key1:value1,key2:value2,.....}
③ {} 空字典
④ 字典内部可以嵌套字典，也可以嵌套元组和列表

```
① 查看字典中的某些元素切片
In [199]: dict1={'name':'yvonne','age':22,'sex':'girl'}

In [210]: print dict1
{'age': 22, 'name': 'yvonne', 'sex': 'girl'}

In [211]: dict1["name"]
Out[211]: 'yvonne'
```

```
② 修改字典中的某些切片或元素
In [212]: dict1["name"]="lanzhi"

In [213]: print dict1
{'age': 22, 'name': 'lanzhi', 'sex': 'girl'}
```

⑤ 字典复制 dict1.copy()

```
In [214]: dict2=dict1.copy()

In [215]: print dict1
{'age': 22, 'name': 'lanzhi', 'sex': 'girl'}

In [216]: print dict2
{'age': 22, 'name': 'lanzhi', 'sex': 'girl'}

In [218]: id (dict1)
Out[218]: 140356588622744

In [219]: id (dict2)
Out[219]: 140356588631496
```

备注： dict2=dict1 也可以复制，但是同赋值得来的副本会因原变量的变化而变化。
更多 dict 的内置函数见 dict._ _（+Tab）

常用参数 help(dict)	说明	语法示例
get	返回对应键的值	dict1.get("age")
clear	Remove all items	dict1.clear()
has_key	判断字典中是否包含此键值元素	dict1.has_key("age")
item	将字典转化为元组列表	dict1.items()
pop	删除最后一个元素	list1.pop()
update	合并两个字典为一个字典	dict1.update(dict2)
copy	复制	dict3=dict1.copy()
viewvalues	显示所有键的值	dict2.viewvalues()

⑥ 变量解包

注意：解包时需要两个或几个变量匹配相当

```
In [243]: print dict3
{'age': 22, 'name': 'lanzhi', 'sex': 'girl'}

In [244]: m0,m1,m2=dict3

In [245]: print m0
age

In [247]: print m1
name

In [248]: print m2
sex
```

zip 可以把两个序列或多个序列拆分成多个元组，还可以使用它构造字典。

```
In [253]: zip('abcdef','123456')
Out[253]: [('a', '1'), ('b', '2'), ('c', '3'), ('d', '4'), ('e', '5'), ('f', '6')]

In [255]: dict(zip('abcdef','123456'))
Out[255]: {'a': '1', 'b': '2', 'c': '3', 'd': '4', 'e': '5', 'f': '6'}
```

注意：我们不允许对一个键赋多个值，如果键发生冲突，以最后一个键为准。

另外大多数 python 对象都可以用到自定义的键，但他们必须是可以哈希的对象，因为他们的查找方式是把每一个键做成哈希表。所以，非可哈希的对象都不可以做成键，如字典本身是不可以哈希的，我们不能把字典本身当做键，但字典本身可以当做值。

所以当字典嵌套字典时，字典内部的字典可以当做值来用，但不能当做键来用。因为字典是非可哈希的。

一般来说，充当键的都是字符串或数字。

我们前面提到过，python 中的每一种数据类型都是由类来定义的，所以在 python 中类和类型是不加区分的。

A. 类的组成
a. 内置数字+方法，对象的类型决定了其所支持的方法
b. 常用类型：列表和字典
* 变量没有类型，对象才有
B. 点号运算：可以调用其属性和方法
属性--->变量名--->数据（使用 print 语句）
方法、操作、() （调用、执行一段代码）
操作是可调用对象。可调用对象：函数、类、对象中的方法；字符串、字典是不可调用的。
callable 判断是否可调用

22. python 的集合类型详解

1） 集合
集合是一组无序排列的，可哈希的值；支持集合关系测试（如交集、并集、差集），支持成员关系测试（in, not in）,支持迭代。但无法支持索引、切片、元素获取操作。

集合有两种类型
- 可变类型 set()
- 不可变类型 frozenset()

集合没有特定语法格式，只能通过工厂函数创建。help(set) 可以看到 set 支持的各种操作，使用的是可迭代的对象。如下：

```
In [261]: s1=set()

In [262]: s2=set(range(5))

In [263]: print s2
set([0, 1, 2, 3, 4])
```

2） 集合类型的方法和操作

项目	描述
len(s)	返回 s 中项目数
s.copy	制作 s 的一份副本
s.defferece(t)	求差集，返回所有在 s 中但不在 t 中的项目
s.intersection(t)	求交集，返回所有在 s 和 t 中的项目
s.isdisjoint(t)	如果 s 和 t 没有相同项，则返回 True
issubset(t)	如果 s 是 t 的一个子集，则返回 True
s.issuperset(t)	如果 s 是 t 的一个超集，则返回 True
s.symmetric_differece(t)	求对称差集，返回所有在 s 或 t 中，但不同在这两个集中的项
s.union(t)	求并集，返回所有在 s 或在 t 中的项

3） 集合操作

操作	描述	操作	描述
s|t	并集	s-t	差集
s&t	交集	s^t	对称差集

```
In [270]: s=set(range(5))

In [271]: t=set(range(10))

In [272]: s|t
Out[272]: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

In [273]: s&t
Out[273]: {0, 1, 2, 3, 4}

In [274]: s-t
Out[274]: set()

In [275]: t-s
Out[275]: {5, 6, 7, 8, 9}

In [276]: s^t
Out[276]: {5, 6, 7, 8, 9}
```

23. 容器数据类型总结

1）列表、元素和字典可以在使用续行符的情况下直接分布式写于多个行上，最后一行上允许使用逗号，但是空时（即没有任何一个元素时）是不允许有逗号的。
2）所有对象都有引用计数，如果为其对象指向新名称或将其放入容器中时，都会导致引用计数增加，当引用计数为 0 时，就会由垃圾收集器回收次对象，另外使用 del 删除引用或某些此应用的修改超出作用域时，也会造成引用计数减少。

```
In [277]: import sys

In [278]: name="Ierry"

In [279]: name="Mark"

In [280]: sys.getrefcount(name)
Out[280]: 2
```

3）列表和字典都支持两种类型的复制操作，浅复制和深复制：

浅复制创建一个新的对象，包含的是原始对象的引用；
深复制创建一个新的对象，对递归复制它所包含的所有对象。
用 copy 和 deepcopy 来实现。
4）python 中的所有对象都是“第一类”，这意味着使用标识符命名的所有对象都具有相同状态，于是，能够命名的对象都可以直接当数据进行处理的。
5）序列表示索引为非负整数的有序对象集合，包括字符串、列表和元组，（字符串是字符的序列，而列表和元组则是 python 任意的对象的序列），列表和元组本身是容器。
6）所有序列都支持迭代。
7）所有序列都支持操作方法。

24. python 的表达式和语句

1）表达式：由一个或多个操作数，以及 0 个以上的运算符组成的序列，就叫表达式。
可以通过计算得到一个值、方法、对象，或命名的空间等结果。
表达式包含文本盾，包含方法调用，运算符、操作数，甚至是一些简单的名称（可以是变量、类型、方法）
2）程序所执行的操作都是由语句来表达的。
语句常见的操作：
① 声明变量---->赋值语句（变量声明语句）
② 赋值---->赋值语句
③ 循环--->控制语句
④ 根据条件来分支--->条件语句
因此语句在程序中的执行顺序又称控制流或执行流，如 while, if, for, 顺序执行
3）语句是以分号（;）结尾的单行代码或语句块结尾的多行语句；
在 python2 中 print 是语句，在 python3 中 print 是函数。
4）python 中常用的表达式操作符：

操作	表达式	运算优先级
算术运算	x+y, x-y, x*y, x/y, x//y, x%y	7(*, /, //, %)，8(+,-)
逻辑运算	x or y, x and y, not x	18(not) 19(and) 20(or)
成员关系运算	x in y, x not in y	17
对象实例测试	x is y, x not is y	16
比较运算	x<y, x>y, x<=y, x>=y, x==y, x!=y	13(==,!=),14(<,<=), 15(>,>=)
按位运算	x|y, x&y, x^y	10(&) 11(^)12(|)
移位	x<<y, x>>y, -x, +x, ~x (一元运算)	5（一元操作符）9(移位)
幂运算	x**y, x^y	6
索引运算和分片运算	x[i], x[i:j], x[i:j:stride]	2
调用	x(...)	4
取属性	x.attribute	3
元组	()	1
序列	[]	1
字典	{}	1
集合	set	
三元选择表达式	x if y,else z	

匿名函数相关表达式	lamda args:expression	21
生成器函数发送协议	yield x	

5) python 中常用的语句
① 赋值语句
A. 隐式赋值 import, from, def, class, for, 函数参数等
B. 元组和列表分解赋值
当赋值符号(=)的左侧为元组或列表时，python 会按照位置把右边的对象和左边的目标自左而右逐一进行匹配，个数不同时会触发异常，此时可以切片的方式进行。
C. 多重目标赋值 如：In [291]: name3=name1=name2=name
D. 增强赋值 (+=, -=, *=, /=, //=, %=)

26. 条件测试及 if 语句---python 流程控制

26.1 python 中的比较操作

1） 所有的 python 对象都支持比较操作
A. 可用于测试相等性，相对大小等；
B. 如果是复合对象，python 会检查其所有部分，包括自动遍历各级嵌套对象，直到可以得出最终结果。
2) 测试操作符
A. == 操作符测试值得想等性
B. is 表达式测试对象的一致性
C. 其他测试操作符 >, <, >=, <=,in, not in
3) python 中不同类型的比较方法
A. 数字：通过相对大小进行比较
B. 字符串：按照字典次序逐个字符串进行比较
C. 列表和元组：自左至右比较各部分内容
D. 字典：对排序后的（键，值）列表进行比较
4）python 中真假的含义
A. 非零整数为真，否则为假
B. 非空对象为真，否则为假
C. None 则始终为假
小结：任何非空对象和非 0 整数都为真；数字 0、空对象和特殊对象 None 均为假。
比较和相等测试会递归地应用于数据结构中；返回值为 True 或 False。
5）组合条件测试
and x and y 与运算
or x or y 或运算
not 返回 True 或 False 非运算
注意：python 中 and 和 or 运算会返回真或假的对象，而不是 True 或 False
对象在本质上不是真，就是假；and 和 or 是短路操作符。

26.2 if 测试的语法结构

需要再看视频补充

26.3 python 控制流语句之 while 和 for 循环