



# 《Python统计计算》

( 2021年秋季学期 )

翟祥  
北京林业大学

E-mail: zhaixbh@126.com

# 课程内容（编程基础）

---

- ❑ 第1章 基础知识
- ❑ 第2章 Python数据结构
- ❑ 第3章 选择与循环
- ❑ 第4章 字符串与正则表达式
- ❑ 第5章 函数设计与使用
- ❑ 第6章 文件操作
- ❑ 第7章 异常处理

# 课程内容（统计计算）

---

- ❑ 第8章 数据库编程（爬虫）
- ❑ 第9章 统计计算理论
- ❑ 第10章 numpy与scipy数据处理
- ❑ 第11章 matplotlib/seaborn绘制图表
- ❑ 第12章 pandas统计计算
- ❑ 第13章 statsmodels模型构建
- ❑ 第14章 sklearn机器学习

# 教材



商品精选

广告

官方直售

Python进阶编程

编写更高效、优雅的Python代码

已有1人评价

机械工业出版社官方旗舰店

¥90.30

Python进阶编程：编写更高效、优雅的Python代码

已有1人评价

入门+实战+项目

Python编程入门三剑客

适合Python3.X各版本

python编程入门三剑客

¥137.00

Python编程三剑客：Python编程从入门到实践

已有90+人评价

机械工业出版社官方旗舰店

¥90.30

Python进阶编程：编写更高效、优雅的Python代码

10条评价

机械工业出版社官方旗舰店

关注 加入购物车

Python一行流

像专家一样写代码

¥89.00

Python一行流：像专家一样写代码

已有10万+人评价

机械工业出版社官方旗舰店

¥89.00

Python一行流：像专家一样写代码

已有10万+人评价

Python编程从入门到实践

第2版

¥92.00

Python编程从入门到实践 第2版

50万+条评价

人民邮电出版社

自营 放心购

关注 电子书 加入购物车

机械工业出版社官方旗舰店

¥92.00

Python编程从入门到实践 第2版

50万+条评价

人民邮电出版社

自营 放心购

关注 电子书 加入购物车

入门+实战+项目

Python编程入门三剑客

适合Python3.X各版本

python编程入门三剑客

¥137.00

Python编程三剑客：Python编程从入门到实践

1000+条评价

机械工业出版社官方旗舰店

¥137.00

Python编程三剑客：Python编程从入门到实践

1000+条评价

Python学习手册

（原书第2版）

¥219.00

Python学习手册（原书第2版）

2万+条评价

机械工业出版社官方旗舰店

¥219.00

Python学习手册（原书第2版）

2万+条评价

2021新版

Python

零基础学Python

¥69.50

零基础学Python（Python3.9全彩版）

20万+条评价

机械工业出版社官方旗舰店

¥69.50

零基础学Python（Python3.9全彩版）

20万+条评价

Effective Python

编写高质量Python代码

¥129.00

Effective Python：编写高质量Python代码

2万+条评价

机械工业出版社官方旗舰店

¥129.00

Effective Python：编写高质量Python代码

2万+条评价

综合

销量

评论数

出版时间

价格

配送至

北京昌平区百善镇

☐ 京东物流
 ☐ 货到付款
 ☐ 仅显示有货
 ☐ 新品
 ☐ PLUS专享
 ☐ 拍拍二手

共12万+ 件商品 1/100

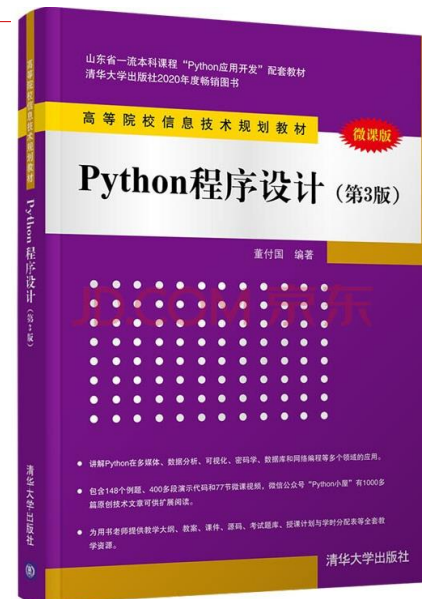
<

>

4

# 教材

□ 董付国, Python程序设计 (第3版), 清华大学出版社, 2020年06月



□ 嵩天等著, Python语言程序设计基础 (第2版), 高等教育出版社, 2017年2月



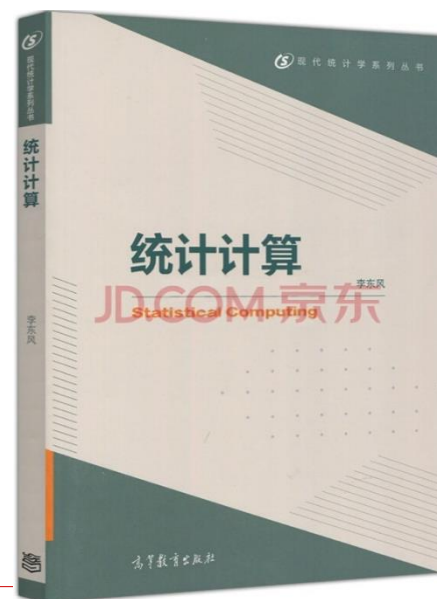
- ❑ 张若愚, Python科学计算(第2版), 清华大学出版社, 2016年04月
- ❑ [意] 阿尔贝托·博斯凯蒂, 数据科学导论: Python语言实现, 机械工业出版社, 2016年08月
- ❑ [美] Peter Harrington著, 李锐, 李鹏, 曲亚东, 王斌 译, 机器学习实战, 人民邮电出版社, 2013年06月

# 教材

□ 高惠璇, 统计计算, 北京大学出版社, 1995年07月



□ 李东风, 统计计算, 高等教育出版社, 2017年3月



# Python网络资源

---



- Python 基础教程
  - ◆ <http://www.w3cschool.cc/python/python-tutorial.html>
- Python 2.7入门指南
  - ◆ <http://www.pythondoc.com/pythontutorial27/index.html>
- Python 3.10入门指南
  - ◆ [The Python Tutorial — Python 3.10.0 documentation](#)
- PythonTab:Python中文开发者社区门户
  - ◆ <http://www.pythontab.com/>
- Python中文官方文档
  - ◆ <http://python.usyiyi.cn/>
- Python官方文档
  - ◆ <https://docs.python.org/>



## 第1章 Python基础知识



什么是Python?

# Python是什么？

---



## Python 官方网站的描述

Python is a programming language that lets you work more quickly and integrate your systems more effectively. You can learn to use Python and see almost immediate gains in productivity and lower maintenance costs.

## 某种通用定义

**Python**是一种简单易学，功能强大的编程语言。它有高效率的高层数据结构，能够简单、有效地实现面向对象编程。

**Python**语法简洁，支持动态输入，是解释性语言。在大多数平台上，对于众多领域，**Python**都是一个理想的开发语言，特别适合于应用程序的快速开发。

---

面向对象

程序设计语言

解释型

---

复杂的应用程序粘合在一起

尤其是C/C++

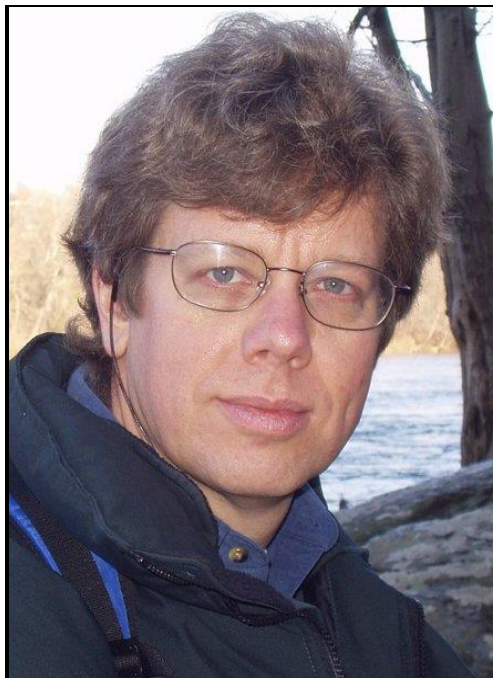
“胶水”语言

---

# 1.0 Python是一种怎样的语言

---

- ❑ Python是一门跨平台、开源、免费的解释型高级动态编程语言，同时也支持伪编译将Python源程序转换为字节码来优化程序和提高运行速度，并且支持使用py2exe工具将Python程序转换为扩展名为“.exe”的可执行程序，可以在没有安装Python解释器和相关依赖包的Windows平台上运行。
- ❑ Python支持命令式编程、函数式编程，完全支持面向对象程序设计，语法简洁清晰，并且拥有大量的几乎支持所有领域应用开发的成熟扩展库。
- ❑ Python就像胶水一样，可以把多种不同语言编写的程序融合到一起实现无缝拼接，更好地发挥不同语言和工具的优势，满足不同应用领域的需求。



## Python的创始人为荷兰的Guido

1989年，Guido为了打发圣诞节的无趣，决心开发一个新的脚本解释程序，做为ABC 语言的一种继承。之所以选中Python作为该编程语言的名字，是因为他是一个叫Monty Python的喜剧团体的爱好者。

Python是ABC语言的后代，Guido参与到ABC语言的开发，Python设计理念是能够像C语言那样能够全面调用计算机的功能接口，又可以像shell那样可以轻松的编程。

1991年，第一个Python编译器(同时也是解释器)诞生。它是用C语言实现的，并能够调用C库(.so文件)。

**Guido Van Rossum (GvR)**

<http://www.python.org/~guido/>

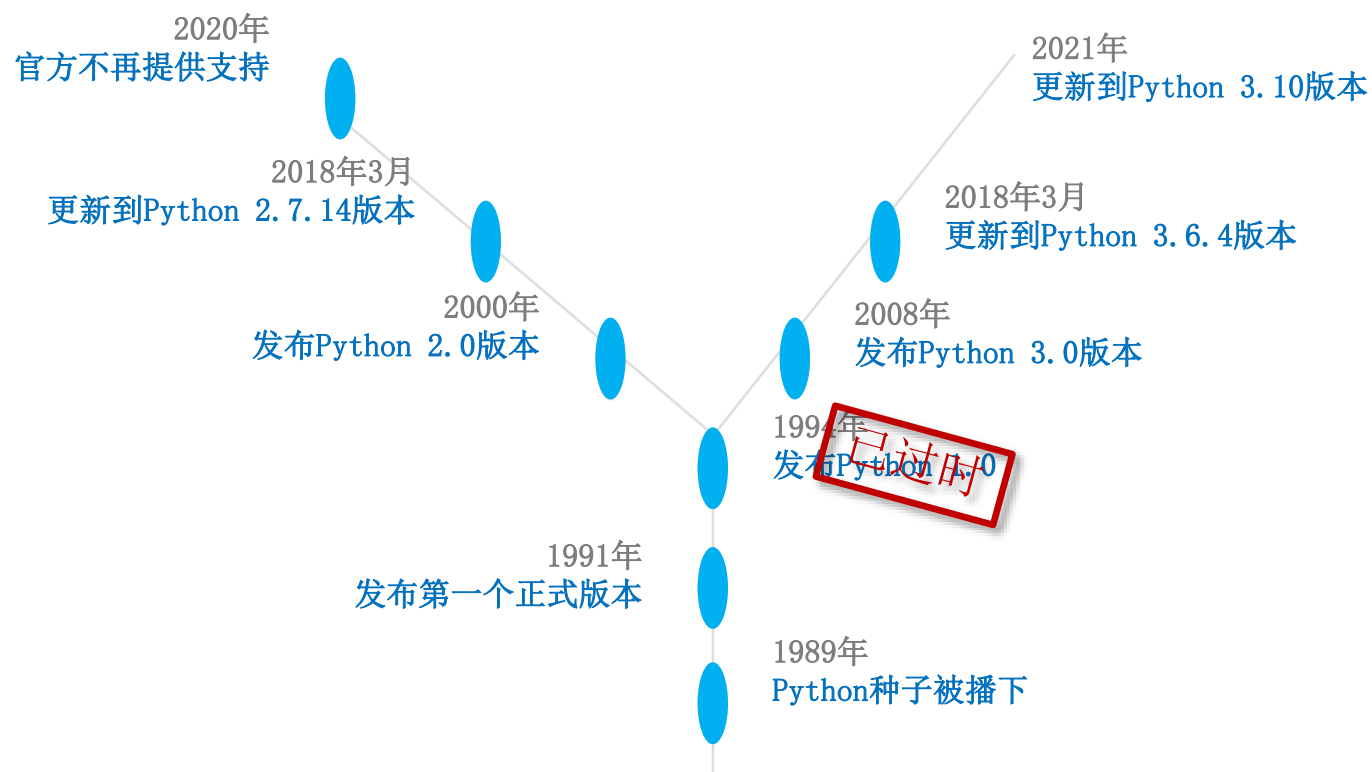
- ❑ ***Computer Programming for Everybody***
  - ❑ 1999年，Guido向DARPA 阐述Python语言的特性：
    - ❑ 简单、直观、强大
    - ❑ 开源，以便任何人都可以为它做贡献
    - ❑ 代码像纯英语那样容易理解
    - ❑ 适用于短期开发的日常任务
  - ❑ 其它包括可扩展性、丰富的库、面向对象、高级语言、可移植性等等特点
-



## Python的特点

- 易于学习
- 易于阅读
- 易于维护
- 强大标准库
- 互动模式
- 可移植
- 可扩展
- 数据库操作
- GUI编程
- 可嵌入

十大  
特点



# Python缺点

---



- ❑ 强制缩进
  - ❑ 构架选择太多
  - ❑ 性能问题
  - ❑ 单行语句和命令行输出问题
-

# 1.1 如何选择Python版本

---

- ❑ 2.x: 一边鄙视一边用
- ❑ 3.x: 必然的趋势
- ❑ 多版本共存与切换简便方法: 更改系统环境变量path
- ❑ 查看已安装版本的方法 (在所启动的**IDLE**界面也可以直接看到):

```
>>> import sys
>>> sys.version
'2.7.8 (default, Jun 30 2014, 16:08:48) [MSC v.1500 64 bit (AMD64)]'
>>> sys.version_info
sys.version_info(major=2, minor=7, micro=8, releaselevel='final', serial=0)
```

# 1.1 如何选择Python版本

---

- ❑ 安装好Python以后，在“开始”菜单中启动“IDLE (Python GUI)”即可启动Python解释器并可以看到当前安装的Python版本号。
- ❑ 如果您喜欢的话，也可以启动“Python (command line)”来开始美妙的Python之旅。
- ❑ 在“IDLE (Python GUI)”和“Python (command line)”两种界面中，都以三个大于号“>>>”作为提示符，您可以在提示符后面输入要执行的语句。

# 1.1 如何选择Python版本

---

- 除了在启动主界面上查看已安装的Python版本之外，还可以使用下面的命令随时进行检查。

```
>>> import sys
>>> sys.version
'3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:25) [MSC v.1900 64 bit (AMD64)]'
>>> sys.winver
'3.5'
>>> sys.version_info
sys.version_info(major=3, minor=5, micro=1, releaselevel='final', serial=0)
```

# 1.1 如何选择Python版本

---

- 有时候可能需要同时安装多个不同的版本，例如同时安装Python 2.7.11和Python 3.10，并根据不同的开发需求在两个版本之间进行切换。
- 多版本并存一般不影响在IDLE环境中直接运行程序，只需要启动相应版本的IDLE即可。在命令提示符环境中运行Python程序时，如果无法正确运行，可以尝试在调用Python主程序时指定其完整路径，或者通过修改系统Path变量来实现不同版本之间的切换。



**Python都能应用在哪些领域？**



Web  
开发

大数据处理

云计算

游戏  
开发

人工智能

自动化运维  
开发

爬虫

---

## Python的应用领域

- Web开发领域

豆瓣 **douban**



# Python的应用领域

- 操作系统管理、服务器运维的自动化脚本



# Python的应用领域

- 科学计算



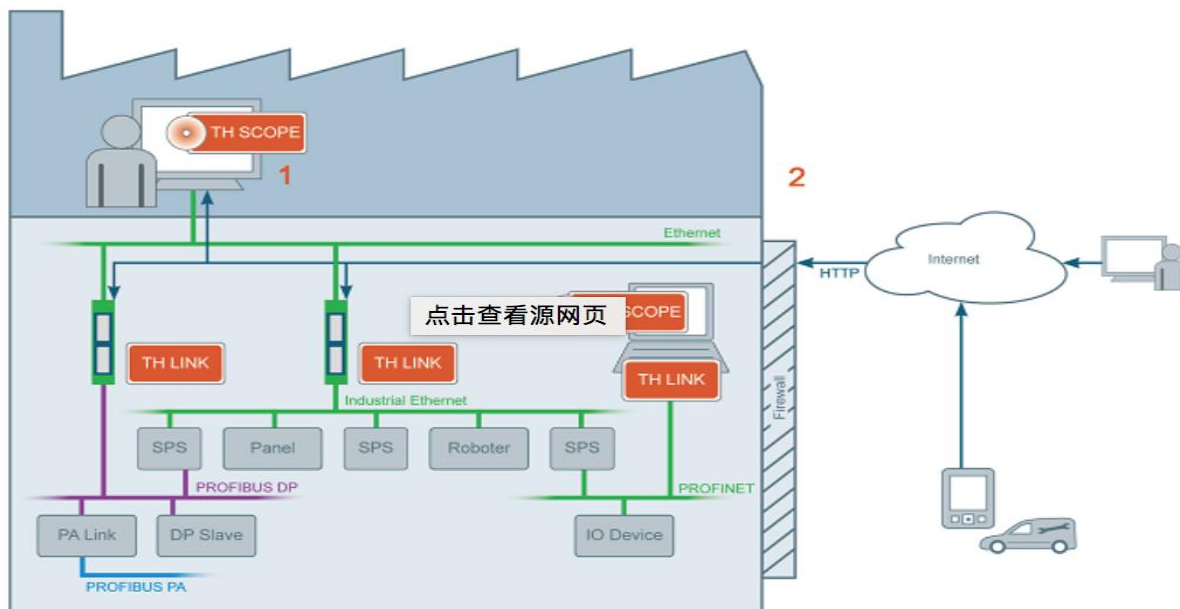
# Python的应用领域

- 桌面应用



# Python的应用领域

- 服务器软件（网络软件）



# Python的应用领域

- 游戏



# Python的版本

## 历史版本

版本号	发布时间	拥有者	GPL 兼容
0.9.0~1.2	1991~1995	CWI	是
1.3~1.5.2	1995~1999	CNRI	是
1.6	2000	CNRI	否
2.0	2000	BeOpen.com	否
1.6.1	2001	CNRI	否
2.1	2001	PSF	否
2.0.1	2001-06-22	PSF	是
2.2~2.7.11	2001~2015	PSF	是
2.7.12	2016-06	PSF	是
2.7.13	2016-12	PSF	是
3.x	2008~至今	PSF	是









# Python的发展

Jan 2016	Jan 2015	Change	Programming Language	Ratings	Change
1	2	▲	Java	21.465%	+5.94%
2	1	▼	C	16.036%	-0.67%
3	4	▲	C++	6.914%	+0.21%
4	5	▲	C#	4.707%	-0.34%
5	8	▲	Python	3.854%	+1.24%
6	6		PHP	2.706%	-1.08%
7	16	▲	Visual Basic .NET	2.582%	+1.51%
8	7	▼	JavaScript	2.565%	-0.71%
9	14	▲	Assembly language	2.095%	+0.92%
10	15	▲	Ruby	2.047%	+0.92%
11	9	▼	Perl	1.841%	-0.42%
12	20	▲	Delphi/Object Pascal	1.786%	+0.95%
13	17	▲	Visual Basic	1.684%	+0.61%
14	25	▲	Swift	1.363%	+0.62%
15	11	▼	MATLAB	1.228%	-0.16%
16	30	▲	Pascal	1.194%	+0.52%
17	82	▲	Groovy	1.182%	+1.07%
18	3	▼	Objective-C	1.074%	-5.88%
19	18	▼	R	1.054%	+0.01%
20	10	▼	PL/SQL	1.016%	-1.00%

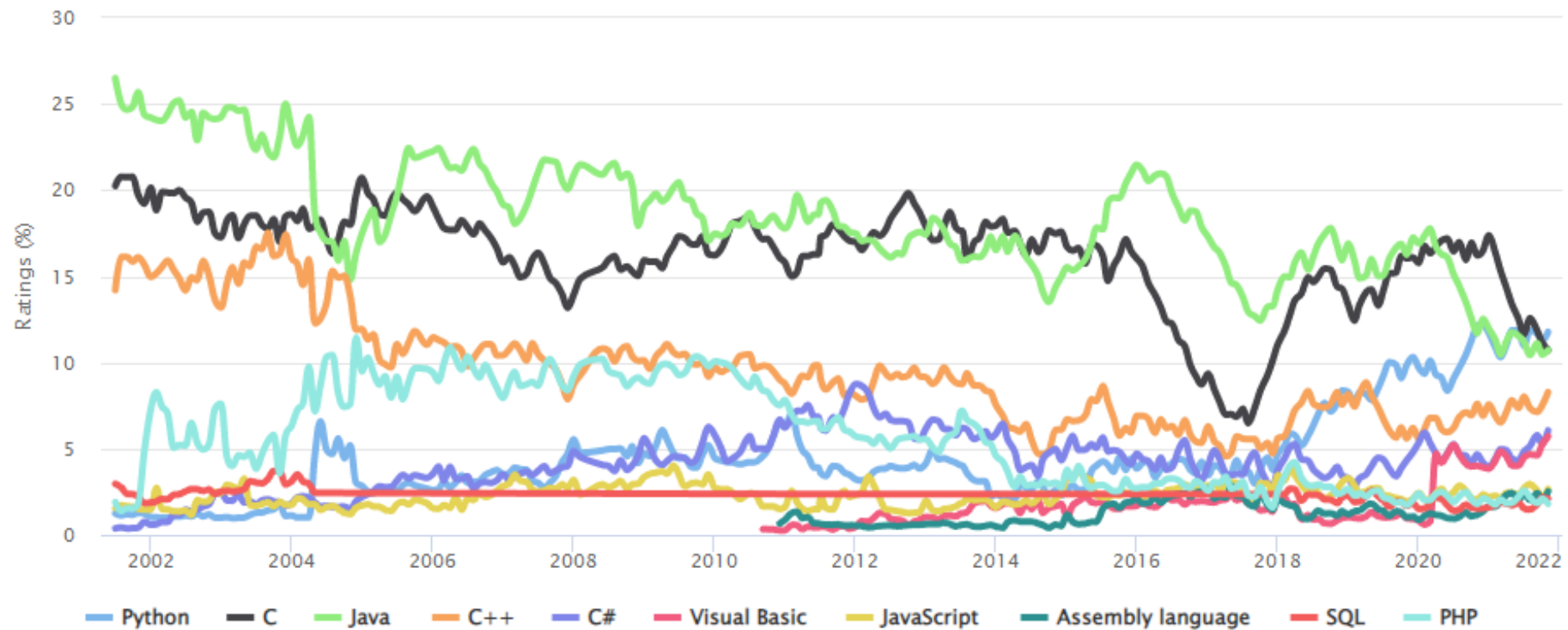
Dec 2017	Dec 2016	Change	Programming Language	Ratings	Change
1	1		Java	13.268%	-4.59%
2	2		C	10.158%	+1.43%
3	3		C++	4.717%	-0.62%
4	4		Python	3.777%	-0.46%
5	6	▲	C#	2.822%	-0.35%
6	8	▲	JavaScript	2.474%	-0.39%
7	5	▼	Visual Basic .NET	2.471%	-0.83%
8	17	▲	R	1.906%	+0.08%
9	7	▼	PHP	1.590%	-1.33%
10	18	▲	MATLAB	1.569%	-0.25%
11	13	▲	Swift	1.566%	-0.57%
12	11	▼	Objective-C	1.497%	-0.83%
13	9	▼	Assembly language	1.471%	-1.07%
14	10	▼	Perl	1.437%	-0.90%
15	12	▼	Ruby	1.424%	-0.72%
16	15	▼	Delphi/Object Pascal	1.395%	-0.55%
17	16	▼	Go	1.387%	-0.55%
18	25	▲	Scratch	1.374%	+0.19%
19	20	▲	PL/SQL	1.368%	-0.13%
20	14	▼	Visual Basic	1.347%	-0.62%

# Python的发展

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	^	Python	7.653%	+4.67%
4	3	v	C++	7.394%	+1.83%
5	8	^	Visual Basic .NET	5.308%	+3.33%
6	4	v	C#	3.295%	-1.48%
7	6	v	PHP	2.775%	+0.57%
8	7	v	JavaScript	2.131%	+0.11%
9	-	^^	SQL	2.062%	+2.06%
10	18	^^	Objective-C	1.509%	+0.00%
11	12	^	Delphi/Object Pascal	1.292%	-0.49%
12	10	v	Ruby	1.291%	-0.64%
13	16	^	MATLAB	1.276%	-0.35%
14	15	^	Assembly language	1.232%	-0.41%
15	13	v	Swift	1.223%	-0.54%
16	17	^	Go	1.081%	-0.49%
17	9	vv	Perl	1.073%	-0.88%
18	11	vv	R	1.016%	-0.80%
19	19		PL/SQL	0.850%	-0.63%
20	14	vv	Visual Basic	0.682%	-1.07%

Nov 2021	Nov 2020	Change	Programming Language		Ratings	Change
1	2	▲		Python	11.77%	-0.35%
2	1	▼		C	10.72%	-5.49%
3	3			Java	10.72%	-0.96%
4	4			C++	8.28%	+0.69%
5	5			C#	6.06%	+1.39%
6	6			Visual Basic	5.72%	+1.72%
7	7		 JavaScript page	JavaScript	2.66%	+0.63%
8	16			Assembly language	2.52%	+1.35%
9	10	▲		SQL	2.11%	+0.58%
10	8	▼		PHP	1.81%	+0.02%
11	21			Classic Visual Basic	1.56%	+0.83%
12	11	▼		Groovy	1.51%	-0.00%

Source: [www.tiobe.com](http://www.tiobe.com)



Position	Programming Language	Ratings
21	(Visual) FoxPro	1.06%
22	SAS	0.98%
23	Prolog	0.77%
24	Scratch	0.77%
25	COBOL	0.68%
26	Lua	0.58%
27	PL/SQL	0.58%
28	Objective-C	0.55%
29	Rust	0.54%
30	Lisp	0.49%
31	Dart	0.42%
32	Ada	0.41%
33	Kotlin	0.40%
34	D	0.40%

## Very Long Term History

To see the bigger picture, please find below the positions of the top 10 programming languages of many years back. Please note that these are *average* positions for a period of 12 months.

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Python	2	5	6	8	20	27	-	-
Java	3	1	1	1	2	14	-	-
C++	4	3	3	3	3	2	2	5
C#	5	4	4	7	12	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	20	-	-
PHP	8	6	5	5	10	-	-	-
Assembly language	9	10	-	-	-	-	-	-
SQL	10	-	-	-	36	-	-	-
Fortran	20	25	29	21	25	5	3	8
Ada	28	29	18	16	18	10	5	2
Lisp	31	27	13	13	16	7	4	3
(Visual) Basic	-	-	8	4	4	3	8	6

## Programming Language Hall of Fame

The hall of fame listing all "Programming Language of the Year" award winners is shown below. The award is given to the programming language that has the highest rise in ratings in a year.

Year	Winner
2020	🏆 Python
2019	🏆 C
2018	🏆 Python
2017	🏆 C
2016	🏆 Go
2015	🏆 Java
2014	🏆 JavaScript
2013	🏆 Transact-SQL
2012	🏆 Objective-C
2011	🏆 Objective-C
2010	🏆 Python
2009	🏆 Go
2008	🏆 C
2007	🏆 Python
2006	🏆 Ruby
2005	🏆 Java
2004	🏆 PHP
2003	🏆 C++

# 1.2 Python安装与简单使用

---



<https://www.python.org/>

<https://www.python.org/doc/>

<http://bugs.python.org/>

默认编程环境：IDLE



# 1.2 Python安装与简单使用



- 安装好以后，默认以IDLE为开发环境。如果使用交互式编程模式，那么直接在IDLE提示符“>>>”后面输入相应的命令并回车执行即可，如果执行顺利的话，马上就可以看到执行结果，否则会抛出异常。

```
>>> 3+5
```

```
8
```

```
>>> import math
```

```
>>> math.sqrt(9)
```

```
3.0
```

```
>>> 3*(2+6)
```

```
24
```

```
>>> 2/0
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#18>", line 1, in <module>
```

```
    2/0
```

```
ZeroDivisionError: integer division or modulo by zero
```

# 1.2 Python安装与简单使用



- 在IDLE界面中使用菜单“File”==>“New File”创建一个程序文件，输入代码并保存为文件（务必要保证扩展名为“.py”，如果是GUI程序可以保存为“.pyw”文件。如果您保存为其他扩展名的文件，一般并不影响在IDLE中直接运行，但是在“命令提示符”环境中运行时需要显式调用Python主程序，并且在资源管理器中直接双击该文件时可能会无法关联Python主程序从而导致无法运行）后，使用菜单“Run”==>“Check Module”来检查程序中是否存在语法错误，或者使用菜单“Run”==>“Run Module”运行程序，程序运行结果将直接显示在IDLE交互界面上。

## 1.2 Python安装与简单使用



□也可以通过在资源管理器中双击扩展名为“.py”或“.pyc”的Python程序文件直接运行；在有些情况下，可能还需要您在命令提示符环境中运行Python程序文件。在“开始”菜单的“附件”中单击“命令提示符”，然后执行Python程序。例如，假设有程序HelloWorld.py内容如下。

```
def main():  
    print('Hello world')  
main()
```

# 1.2 Python安装与简单使用

---



```
>>> ===== RESTART =====  
=  
>>>  
Hello world  
>>>
```

## 1.2 Python安装与简单使用

---



C:\> 命令提示符

```
C:\Python35>python helloworld.py  
Hello world
```

```
C:\Python35>helloworld.py  
Hello world
```

```
C:\Python35>
```

# 1.2 Python安装与简单使用

在实际开发中，如果您能够熟练使用集成开发环境IDLE提供的一些快捷键，将会大幅度提高您的编写速度和开发效率。在IDLE环境下，除了撤销（Ctrl+Z）、全选（Ctrl+A）、复制（Ctrl+C）、粘贴（Ctrl+V）、剪切（Ctrl+X）等常规快捷键之外，其他比较常用的快捷键如下表所示。

快捷键	功能说明
Alt+p	浏览历史命令（上一条）
Alt+n	浏览历史命令（下一条）
Ctrl+F6	重启Shell，之前定义的对象和导入的模块全部失效
F1	打开Python帮助文档
Alt+/ 	自动补全前面曾经出现过的单词，如果之前有多个单词具有相同前缀，则在多个单词中循环选择
Ctrl+]	缩进代码块
Ctrl+[	取消代码块缩进
Alt+3	注释代码块
Alt+4	取消代码块注释。

## 1.2 使用pip管理第三方包

---

- ❑ Python2中,pip本身需要单独安装
- ❑ 下载地址: <https://pypi.python.org/pypi/pip>
- ❑ 安装方法: 命令行下用 `python get-pip.py`
  
- ❑ Pip管理第三方包用法示例
  - `pip install NumPy`
  - `pip list`
  - `pip install --upgrade SomePackage`
  - `pip uninstall SomePackage`
- ❑ Python 2.7.9和Python 3.4.0之后的安装包中已默认包含pip。

## □ Anaconda

◆ <https://www.anaconda.com/download/>



# Anaconda简介

---

- ❑ Anaconda Python 是 Python 科学计算包的合集，包含了常用科学计算、数据分析、自然语言处理、绘图等包，所有的包几乎都是最新的，容量适中。
- ❑ 使用了conda和pip包管理工具，安装第三方包非常方便，避免了管理各个库之间依赖性的麻烦。
- ❑ 安装后就默认安装了python、IPython、集成开发环境Spyder和众多的包和模块。支持Python2和Python3，包括免费版、协作版、企业版等。

Anaconda Python 是完全免费的企业级的Python发行大规模数据处理、预测分析和科学计算工具。

---

# Anaconda下载地址



https://www.continuum.io/downloads

## Anaconda for Windows

PYTHON 2.7	PYTHON 3.5
<b>WINDOWS 64-BIT GRAPHICAL INSTALLER</b> 335M	<b>WINDOWS 64-BIT GRAPHICAL INSTALLER</b> 345M
Windows 32-bit Graphical Installer 281M	Windows 32-bit Graphical Installer 283M

Behind a firewall? Use these zipped Windows installers.

### Windows Anaconda Installation

1. Download the graphical installer.
2. Double-click the .exe file to install Anaconda and follow the instructions on the screen.
3. Optional: [Verify data integrity with MD5.](#)

支持windows、OS X、linux等各种平台，支持python2.7和Python3.5

# Anaconda版本选择



← → ↻ <https://repo.continuum.io/archive/.winzip/>

Index of /archive/.winzip

Filename	Size	Last Modified	MD5
../	-		<directory>
<a href="#">Anaconda2-4.0.0-Windows-x86.zip</a>	279.2M	2016-03-29 11:33:14	3d5c75c977b05bc2220e4eab34109e
<a href="#">Anaconda2-4.0.0-Windows-x86_64.zip</a>	332.5M	2016-03-29 11:32:59	c488eaf27937c80a6c9274c101489f
<a href="#">Anaconda3-4.0.0-Windows-x86.zip</a>	281.3M	2016-03-29 11:33:47	27e29f6bb165ba5cb65fcd87d6fc30
<a href="#">Anaconda3-4.0.0-Windows-x86_64.zip</a>	343.4M	2016-03-29 11:33:32	98d05d3bbc40eb29e25dd09ea183f62
<a href="#">Anaconda2-2.5.0-Windows-x86.zip</a>	294.4M	2016-02-04 23:57:00	0108cb1bc3ba05ac93351cc40bad78
<a href="#">Anaconda2-2.5.0-Windows-x86_64.zip</a>	346.6M	2016-02-04 23:56:46	cc12a97226cbea2e4b42e859833f0c
<a href="#">Anaconda3-2.5.0-Windows-x86.zip</a>	294.4M	2016-02-04 23:57:33	fc365365b99520cb3e4ac33d202008
<a href="#">Anaconda3-2.5.0-Windows-x86_64.zip</a>	359.1M	2016-02-04 23:57:18	d5dc2b1ea6b53173fc64fb043f9d90
<a href="#">Anaconda2-2.4.1-Windows-x86.zip</a>	285.9M	2015-12-08 15:00:53	694e10d87fa8539651a3091504591f
<a href="#">Anaconda2-2.4.1-Windows-x86_64.zip</a>	351.9M	2015-12-08 15:00:52	abaa1e10bf2d0207f89256f8a494a22
<a href="#">Anaconda3-2.4.1-Windows-x86.zip</a>	297.3M	2015-12-08 15:00:58	10f9efb5d7337629b1626df4a59460
<a href="#">Anaconda3-2.4.1-Windows-x86_64.zip</a>	361.6M	2015-12-08 15:00:55	5226daf4e3e95d58059b79cb26519f
<a href="#">Anaconda2-2.4.0-Windows-x86.zip</a>	319.6M	2015-11-03 11:14:18	c95355c6e64bbf9576c00ea04e2e12e
<a href="#">Anaconda2-2.4.0-Windows-x86_64.zip</a>	385.8M	2015-11-03 11:14:03	96290082c04804cae0f8dc0ab9133ea
<a href="#">Anaconda3-2.4.0-Windows-x86.zip</a>	314.7M	2015-11-03 11:14:54	db093c41c17fe4b715363f07b2a

建议安装Python3，但是支持一般较滞后

# Python使用

---

- 控制台
  - 交互界面Ipython
  - 使用Ipython notebook
  - 安装包、更新包-使用pip或conda工具
-

## 1.3 Python语法基础

# 目录页

---



- 01** 基本语法
  - 02** 变量和数据类型
  - 03** 标识符和关键字
  - 04** 简单数值类型
  - 05** 运算符
  - 06** 位运算
  - 07** 运算符优先级
-

# 语法基础



## 01 基本语法

02 变量和数据类型

03 标识符和关键字

04 简单数值类型

05 运算符

06 位运算

07 运算符优先级

## 注释

# Python中的单行注释以#开头！

# 第一个注释

```
print (“Hello, Python!”) # 第二个注释
```



## 注释

多行注释可以使用**三引号**作为开头和结束符号

```
"""
```

```
print(value, ..., sep=' ', end='\n',  
file=sys.stdout, flush=False)
```

```
"""
```

## 行与缩进

python最具特色的就是使用**缩进来表示代码块**

if True:

print ("True")

else:

print ("False")

print ("False")

if True:

print ("True")

else:

print ("False")

print ("hello")



## 语句换行

Python 通常是一行写完一条语句，但如果语句很长，我们需要换行，这时可以使用圆括号来实现。

```
str = ('Python是一种面向对象、解释型计算机程序设计语言，'  
      '由Guido van Rossum于1989年底发明。'  
      '第一个公开发行人版发行于1991年，'  
      '源代码同样遵循 GPL(GNU General Public License)协议。')
```

## 语句换行

Python 通常是一行写完一条语句，但如果语句很长，我们需要换行，这时可以使用\来实现。

```
str = 'Python是一种面向对象、解释型计算机程序设计语言，'\n      '由Guido van Rossum于1989年底发明。'\n      '第一个公开发行人版发行于1991年，'\n      '源代码同样遵循 GPL(GNU General Public License)协议。'
```

## 语句换行

需要注意的是，在 [], {}, 或 () 中的语句，不需要使用圆括号进行换行。

```
total = ['item_one', 'item_two', 'item_three',  
        'item_four', 'item_five']
```

# 语法基础



**01** 基本语法

**02** 变量和数据类型

**03** 标识符和关键字

**04** 简单数值类型

**05** 运算符

**06** 位运算

**07** 运算符优先级

## 变量和赋值

### 超市购物

现实生活中，大家去超市买东西的时候，往往都需要一个菜篮子，用来进行存储物品，等到所有的物品都购买完成后，在收银台进行结账即可。



## 变量和赋值

Python中的变量用来存储数据，变量可以理解为去超市购物的菜篮子，其类型和值在赋值的那一刻被初始化

•

```
num1 = 100
```

```
num2 = 87
```

```
result = num1 + num2
```

num1和num2变量就好比一个小菜篮子，它们分别存储的数据是100和87。result变量存储的数据是num1和num2这两个“菜篮子”中的数据累计之和。



## 变量的类型

1. 数字类型
  - 整型
  - 复数
  - 浮点型
2. 布尔类型
3. 字符串类型
4. 列表类型
5. 元组类型
6. 字典类型
7. 集合



# 语法基础



- 01 基本语法
- 02 变量和数据类型
- 03 标识符和关键字**
- 04 简单数值类型
- 05 运算符
- 06 位运算
- 07 运算符优先级

# 标识符

## 水果名称

现实生活中，人们常用一些名称来标记事物，例如，现实生活中每种水果都有一个名称来标识。



# 标识符

若希望在程序中表示一些事物，需要开发人员自定义一些符号和名称，这些符号和名称叫做**标识符**。

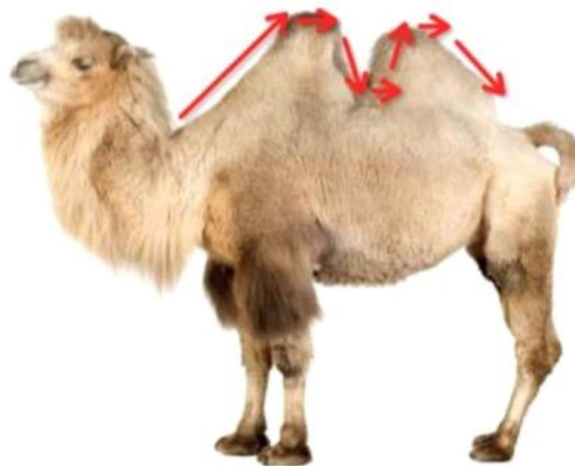
## 命名规则

- 标识符由字母、下划线和数字组成，且数字不能开头。
- Python中的标识符是区分大小写的。
- python中的标识符不能使用关键字

## 标识符

为了规范命名标识符，关于标识符的命名提以下建议：

1. 见名之意
2. 驼峰式



如：  
`userName`      `userLoginFlag`

# 关键字

**关键字**指的是具有特殊功能的标识符。

```
>>> help()           # 进入帮助系统  
help> keywords       # 查看所有的关键字列表  
help> return          # 查看return这个关键字的说明  
help> quit            # 退出帮助系统
```

# 语法基础



- 01 基本语法
- 02 变量和数据类型
- 03 标识符和关键字
- 04 简单数值类型**
- 05 运算符
- 06 位运算
- 07 运算符优先级

# 基本数据类型

---

## □ Python中5种基本的数据类型：

- string-字符串
- Integer、float-整数与浮点数
- bool(True/False)-布尔值
- time/datetime-日期时间
- Complex-复数
- None-空值

## □ 类型识别及转换

- type()
- 类型转换



# Python字符串简介

---

- 在Python中用引号引起来的字符集称之为字符串，比如：  
'hello'、"my Python"、"2+3"等都是字符串。本课程中需要掌握的内容包括：
  - Python中字符串中使用的引号可以是单引号、双引号跟三引号
  - 特殊字符(换行符、制表符等)可以通过转义字符\'\'进行表示，原样输出引号内字符串可以使用在引号前加r 的形式来表示-r''
  - 字符串可以进行运算，其中子字符串的运算经常被使用：子字符串是指字符串的子集
  - 采用格式化输出可以方便地将字符串打印到屏幕上

# 整数与浮点数

---

- ❑ Python可以处理任意大小的整数，当然包括负整数，在程序中的表示方法和数学上的写法一模一样
  - **int** (有符号整数): 通常被简称为整数，是正或负整数，不带小数点。
  - **long** (长整数 ): 可表示任意大小的整数，后面跟着一个大写或小写的L。
  - **float** (浮点实数值) : 或浮点数，表示实数，并写入一个小数点分隔的整数部分和小数部分。浮点数也可以是科学记数法，无穷大**inf**也为浮点数。

## 整型

整数类型（int）简称整型，它用于表示整数。

– 二进制：0b10100

– 八进制：0o10111

– 十进制：64

– 十六进制：0x14

## 浮点型

- 浮点型（Float）用于表示**实数**。
- 浮点型字面值可以用十进制或科学计数法表示

- 

**<实数>E或者e<整数>**

E或e表示基是10，后面的整数表示指数，指数的正负使用+或-表示

-

- ❑ 布尔值用于逻辑判断，只有两个可能结果:True/False
- ❑ 整型、浮点型的“0”和复数0+0j也可以表示False，其余整型、浮点型、复数数值都被判断为True
- ❑ bool值往往是比较运算的结果，其运算包括and/or/not
  - And: 与运算，只有所有都为True运算结果才是True。
  - or: 或运算，只要其中有一个为True运算结果就是True
  - Not: 非运算，它是一个单目运算符，把True 变成False， False 变成True

## 布尔类型

- 布尔型数据只有两个取值：**True**和**False**
- bool值没有操作
- 所有的空都是False，所有的非空都是True

```
a = 0 # 整数0, False
print(bool(a)) s = "" # 空字符串, False
print(bool(s)) lst = [] # 空列表, False
print(bool(lst)) dic = {} # 空字典, False
print(bool(dic)) a = None # None表示空, False
print(bool(a))
```

## 布尔类型

### – 以下对象的布尔值都是False：

- None
- False (布尔型)
- 0 (整型0)
- 0.0 (浮点型0)
- 0.0+0.0j (复数0)
- "" (空字符串)
- [] (空列表)
- () (空元组)
- {} (空字典)

## 复数类型

复数类型，用于表示数学中的复数，例如， $5+3j$

。 两大特点

( 1 ) 复数由实数部分和虚数部分构成，表示为：

`real+imagj` 或 `real+imagJ`

( 2 ) 复数的实数`real`和虚数`imag`都是浮点型



# 复数、空值

---

- ❑ 复数`complex`中的虚部用`j`或`J`表示，可进行复数加减乘除运算、取实部、取虚部、取共轭等运算
- ❑ `None`代表空值，类型为`NoneType`，支持的运算较少，也没有任何内建方法。`None`和任何其他的数据类型比较永远返回`False`，`not None`返回`True`。要注意`None`与`NaN`不是一种类型。

# 日期时间



- ❑ Python 程序能用很多方式处理日期和时间，转换日期格式是一个常见的功能。
- ❑ Python 提供了一个 `time` 和 `datetime` 模块可以用于格式化日期和时间。
  - 时间间隔是以秒为单位的浮点小数
  - 时间戳都以1970年1月1日午夜（历元）经过了多长时间来表示
  - 常用日期格式化符号包括：

%y 两位数的年份表示（00-99）	%H 24小时制小时数（0-23）
%Y 四位数的年份表示（000-9999）	%I 12小时制小时数（01-12）
%m 月份（01-12）	%M 分钟数（00=59）...
%d 月内中的一天（0-31）	%S 秒（00-59）

# 类型判断与类型转换

---

- 类型判断：使用`type()`可判断对象类型，函数、类、各种数据类型的变量在python中都是对象，因此都可以使用`type()`进行判断
- 类型转换：常用的类型转换函数包括`int`、`float`、`complex`、`str`、`bool`，使用时需要保证对象是可转换成相应类型的。

## 数字类型转换

函数	说明
<code>int(x [,base ])</code>	将x转换为一个整数
<code>float(x )</code>	将x转换到一个浮点数， 默认六位小数
<code>complex(real [,imag ])</code>	创建一个复数

# 语法基础



- 01 基本语法
- 02 变量和数据类型
- 03 标识符和关键字
- 04 简单数值类型
- 05 运算符**
- 06 位运算
- 07 运算符优先级

## 算术运算符

运算符	相关说明
+	加：两个对象相加
-	减：得到负数或一个数减去另一个数
*	乘：两个数相乘或是返回一个被重复若干次的字符串
/	除：x除以y
%	取余：返回除法的余数
**	幂：返回x的y次幂
//	取整除：返回商的整数部分

## 赋值运算符

赋值运算符只有一个，即`=`，它的作用是把等号右边的值赋给左边。例如，`x=1`

```
anInt=12
```

```
anFloat=2.2
```

```
anStr='string'
```

```
aList=['a','a','a']
```

```
anArray=(1,2,3)
```

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

为多个变量赋同一个值：`x=y=z=1`

增量赋值

```
x+=1 x=x+1
```

将多个值赋值给多个变量

```
a, b = 1, 2
```

变量交换

```
X= 2
```

```
y =3
```

```
x,y = y,x
```

## 复合赋值运算符

运算符	相关说明	实例
<code>+=</code>	加法赋值运算符	<code>c+=a</code> 等效于 <code>c=c+a</code>
<code>-=</code>	减法赋值运算符	<code>c-=a</code> 等效于 <code>c=c-a</code>
<code>*=</code>	乘法赋值运算符	<code>c*=a</code> 等效于 <code>c=c*a</code>
<code>/=</code>	除法赋值运算符	<code>c/=a</code> 等效于 <code>c=c/a</code>
<code>%=</code>	取模赋值运算符	<code>c%=a</code> 等效于 <code>c=c%a</code>
<code>**=</code>	幂赋值运算符	<code>c**=a</code> 等效于 <code>c=c**a</code>
<code>//=</code>	取整除赋值运算符	<code>c//=a</code> 等效于 <code>c=c//a</code>



## 复合赋值运算符

```
a = 21
b = 10
c = 0
```

```
c = a + b
print ("1 - c 的值为:", c)
```

```
c += a
print ("2 - c 的值为:", c)
```

```
c *= a
print ("3 - c 的值为:", c)
```

```
c /= a
print ("4 - c 的值为:", c)
```

```
c = 2
c %= a
print ("5 - c 的值为:", c)
```

```
c **= a
print ("6 - c 的值为:", c)
```

```
c //= a
print ("7 - c 的值为:", c)
```

1 - c 的值为:	31
2 - c 的值为:	52
3 - c 的值为:	1092
4 - c 的值为:	52.0
5 - c 的值为:	2
6 - c 的值为:	2097152
7 - c 的值为:	99864

# 比较运算符

运算符	相关说明
<code>==</code>	检查两个操作数的值是否相当
<code>!=</code>	检查两个操作数的值是否相等
<code>&gt;</code>	检查左操作数的值是否大于右操作数的值
<code>&lt;</code>	检查左操作数的值是都小于右操作数的值
<code>&gt;=</code>	检查左操作数的值是否大于或等于右操作数的值
<code>&lt;=</code>	检查左操作数的值是否小于或等于右操作数的值

# 逻辑运算符

运算符	逻辑表达式	描述
and	x and y	布尔“与”，如果x为False，x and y返回False，否则它返回y的计算值
or	x or y	布尔“或”，如果x为True，它返回True，否则返回y的计算值
not	not x	布尔“非”，如果x为True，返回False，如果x为False，它返回True

# 逻辑运算符

```
a = 10
b = 20

if ( a and b ):
    print ("1 - 变量 a 和 b 都为 true")
else:
    print ("1 - 变量 a 和 b 有一个不为 true")

if ( a or b ):
    print ("2 - 变量 a 和 b 都为 true，或其中一个变量为 true")
else:
    print ("2 - 变量 a 和 b 都不为 true")
```

```
a = 0
if ( a and b ):
    print ("3 - 变量 a 和 b 都为 true")
else:
    print ("3 - 变量 a 和 b 有一个不为 true")

if ( a or b ):
    print ("4 - 变量 a 和 b 都为 true，或其中一个变量为 true")
else:
    print ("4 - 变量 a 和 b 都不为 true")

if not( a and b ):
    print ("5 - 变量 a 和 b 都为 false，或其中一个变量为 false")
else:
    print ("5 - 变量 a 和 b 都为 true")
```

- 1 - 变量 a 和 b 都为 true
- 2 - 变量 a 和 b 都为 true，或其中一个变量为 true
- 3 - 变量 a 和 b 有一个不为 true
- 4 - 变量 a 和 b 都为 true，或其中一个变量为 true
- 5 - 变量 a 和 b 都为 false，或其中一个变量为 false

## 成员运算符

运算符	描述	实例
in	如果在指定的序列中找到值返回 True，否则返回 False。	x 在 y 序列中，如果 x 在 y 序列中返回 True。
not in	如果在指定的序列中没有找到值返回 True，否则返回 False。	x 不在 y 序列中，如果 x 不在 y 序列中返回 True。

## 成员运算符

```
3 a = 10
4 b = 20
5 list = [1, 2, 3, 4, 5 ];
6
7 if ( a in list ):
8     print ("1 - 变量 a 在给定的列表中 list 中")
9 else:
10    print ("1 - 变量 a 不在给定的列表中 list 中")
11
12 if ( b not in list ):
13    print ("2 - 变量 b 不在给定的列表中 list 中")
14 else:
15    print ("2 - 变量 b 在给定的列表中 list 中")
16
17 # 修改变量 a 的值
18 a = 2
19 if ( a in list ):
20    print ("3 - 变量 a 在给定的列表中 list 中")
21 else:
22    print ("3 - 变量 a 不在给定的列表中 list 中")
```



# 语法基础



- 01 基本语法
- 02 变量和数据类型
- 03 标识符和关键字
- 04 简单数值类型
- 05 运算符
- 06 位运算**
- 07 运算符优先级

## 位运算符

运算符	描述
&	按位与运算符：参与运算的两个值,如果两个相应位都为1,则该位的结果为1,否则为0
	按位或运算符：只要对应的二个二进位有一个为1时，结果位就为1。
^	按位异或运算符：当两对应的二进位相异时，结果为1
~	按位取反运算符：对数据的每个二进制位取反,即把1变为0,把0变为1。~x 类似于 -x-1
<<	左移动运算符：运算数的各二进位全部左移若干位，由"<<"右边的数指定移动的位数，高位丢弃，低位补0。
>>	右移动运算符：把">>"左边的运算数的各二进位全部右移若干位，">>"右边的数指定移动的位数



## 按位与

参与运算的两个数各对应的二进位进行“与”的操作。只有对应的两个二进位都是1时，结果位就为1，否则结果位为0

	0	0	0	0	1	0	0	1
按位与 ( & )	0	0	0	0	0	0	1	1
运算结果	0	0	0	0	0	0	0	1

## 按位或

按位或指的是参与运算的两个数各对应的二进位进行“或”的操作。只要对应的两个二进位有一个为1时，结果位就为1

按位或 (|)

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

运算结果

0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

## 按位异或

- 按位异或就是将参与运算的两个数对应的二进制位进行比较，如果一个位为1，另一个位为0，则结果为1，否则，结果位为0。

	0	0	0	0	1	0	0	0
按位异或 ( ^ )	0	0	0	0	0	1	0	0
运算结果	0	0	0	0	1	1	0	0

## 按位取反

- 按位取反就是将二进位的每一位进行取反；
- 0取反为1，1取反为0

9的二进制

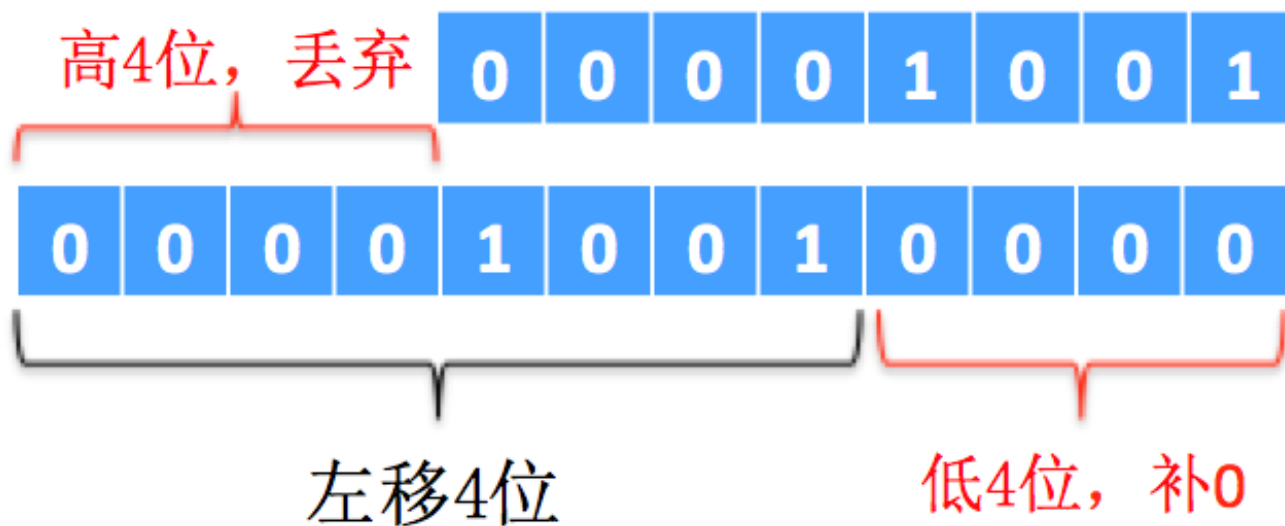
0	0	0	0	1	0	0	1
---	---	---	---	---	---	---	---

按位取反

1	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---

## 按位左移

按位左移指的是二进制全部左移n位，高位丢弃，低位补0。



## 按位右移

按位右移指的是将二进制全部右移n位，移出的位丢弃，移进的位补符号位。



# 位运算

```
a = 60          # 60 = 0011 1100
b = 13          # 13 = 0000 1101
c = 0

c = a & b;       # 12 = 0000 1100
print ("1 - c 的值为:", c)

c = a | b;       # 61 = 0011 1101
print ("2 - c 的值为:", c)

c = a ^ b;       # 49 = 0011 0001
print ("3 - c 的值为:", c)

c = ~a;          # -61 = 1100 0011
print ("4 - c 的值为:", c)

c = a << 2;      # 240 = 1111 0000
print ("5 - c 的值为:", c)

c = a >> 2;      # 15 = 0000 1111
print ("6 - c 的值为:", c)
```

1 - c 的值为:	12
2 - c 的值为:	61
3 - c 的值为:	49
4 - c 的值为:	-61
5 - c 的值为:	240
6 - c 的值为:	15

# 语法基础



- 01** 基本语法
- 02** 变量和数据类型
- 03** 标识符和关键字
- 04** 简单数值类型
- 05** 运算符
- 06** 位运算
- 07** 运算符优先级



# 运算符优先级

运算符	描述
**	指数（最高优先级）
~ + -	按位翻转，一元加号和减号（最后两个的方法名为+@ 和 -@）
* / % //	乘，除，取模和取整除
+ -	加法减法
>> <<	右移，左移运算符
&	位 'AND'
^	位运算符
<= < > >=	比较运算符
<> == !=	等于运算符
= %= /= //= -= += *= **=	赋值运算符
is is not	身份运算符
not or and	逻辑运算符

假设：

$a = 20$  ,  $b = 10$  ,  $c = 15$

$d = 5$  ,  $e = 0$

那么

$e = (a + b) * c / d = 90$

$e = ((a + b) * c) / d = 90$

$e = (a + b) * (c / d) = 90$

$e = a + (b * c) / d = 50$

## 1.4.1 Python的对象模型

---

- ❑ 对象是python语言中最基本的概念
- ❑ python中处理的每样东西都是对象。
  - ◆ 内置对象可直接使用，如数字、字符串、列表、del等
  - ◆ 非内置对象需要导入模块才能使用，如正弦函数`sin(x)`，随机数产生函数`random( )`等

# 1.4.1 Python的对象模型



## □ 常用内置对象

对象类型	示例
数字	1234, 3.14, 3+4j
字符串	'swfu', "I'm student", "Python "
列表	[1, 2, 3]
字典	{1:'food' ,2:'taste', 3:'import'}
元组	(2, -5, 6)
文件	f=open('data.dat', 'r')
集合	set('abc'), {'a', 'b', 'c'}
布尔型	True, False
空类型	None
编程单元类型	函数、模块、类

## 1.4.2 Python变量

---

在Python中，不需要事先声明变量名及其类型，直接赋值即可创建各种类型的对象变量。例如语句

```
>>> x = 3
```

创建了整型变量x，并赋值为3，再例如语句

```
>>> x = 'Hello world.'
```

创建了字符串变量x，并赋值为'Hello world.'。

这一点适用于Python任意类型的对象。

## 1.4.2 Python变量



- 虽然不需要在使用之前显式地声明变量及其类型，但是Python仍属于强类型编程语言，Python解释器会根据赋值或运算来自动推断变量类型。
- 每种类型支持的运算也不完全一样，因此在使用变量时需要程序员自己确定所进行的运算是否合适，以免出现异常或者意料之外的结果。
- 同一个运算符对于不同类型数据操作的含义和计算结果也是不一样的，后面会进行介绍。
- Python还是一种动态类型语言，也就是说，变量的类型是可以随时变化的。

## 1.4.2 Python变量

---



```
>>> x = 3
>>> print(type(x))
<class 'int'>
>>> x = 'Hello world.'
>>> print(type(x))
<class 'str'>
>>> x = [1, 2, 3]
>>> print(type(x))
<class 'list'>
>>> isinstance(3, int)
True
>>> isinstance('Hello world', str)
True
```

## 1.4.2 Python变量

---

- ❑ 内置函数`type()`用来返回变量类型，内置函数`isinstance()`用来测试对象是否为指定类型的实例。代码中首先创建了整型变量`x`，然后又分别创建了字符串和列表类型的变量`x`。当创建了字符串类型的变量`x`之后，之前创建的整型变量`x`自动失效，创建列表对象`x`之后，之前创建的字符串变量`x`自动失效。可以将该模型理解为“状态机”，在显式修改其类型或删除之前，变量将一直保持上次的类型。

## 1.4.2 Python变量

- 在大多数情况下，如果变量出现在赋值运算符或复合赋值运算符（例如+=、\*=等等）的左边则表示创建变量或修改变量的值，否则表示引用该变量的值，这一点同样适用于使用下标来访问列表、字典等可变序列以及其他自定义对象中元素的情况。例如下面的代码：

```
>>> x = 3 #创建整型变量
>>> print(x**2)
9
>>> x += 6 #修改变量值
>>> print(x) #读取变量值并输出显示
9
>>> x = [1, 2, 3] #创建列表对象
>>> print(x)
[1, 2, 3]
>>> x[1] = 5 #修改列表元素值
>>> print(x) #输出显示整个列表
[1, 5, 3]
>>> print(x[2]) #输出显示列表指定元素
```



## 1.4.2 Python变量



- 字符串和元组属于不可变序列，这意味着不能通过下标的方式来修改其中的元素值，例如下面的代码试图修改元组中元素的值时抛出异常。

```
>>> x = (1, 2, 3)
>>> print(x)
(1, 2, 3)
>>> x[1] = 5
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    x[1] = 5
TypeError: 'tuple' object does not support item assignment
```

# 1.4.2 Python变量



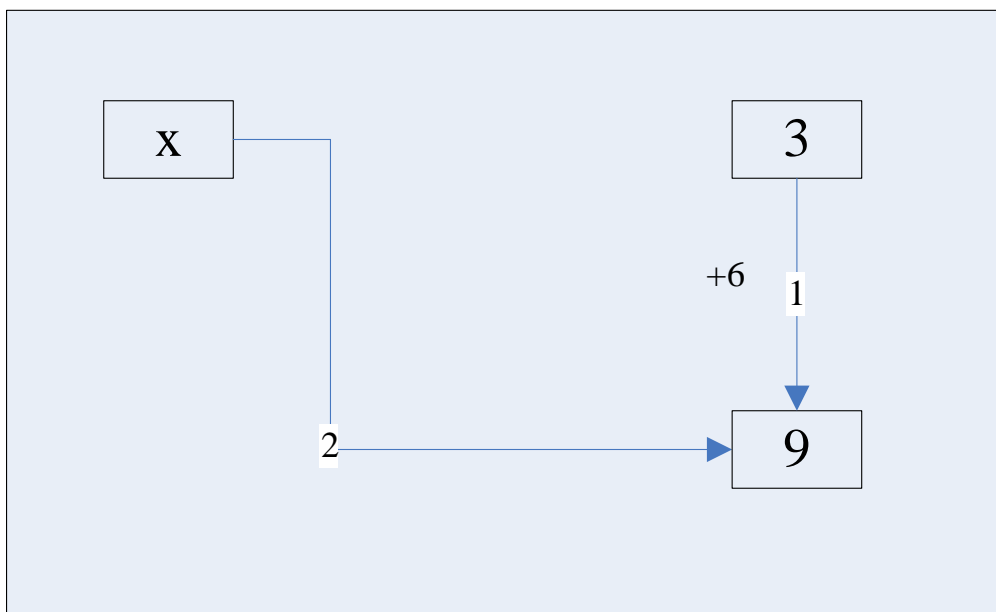
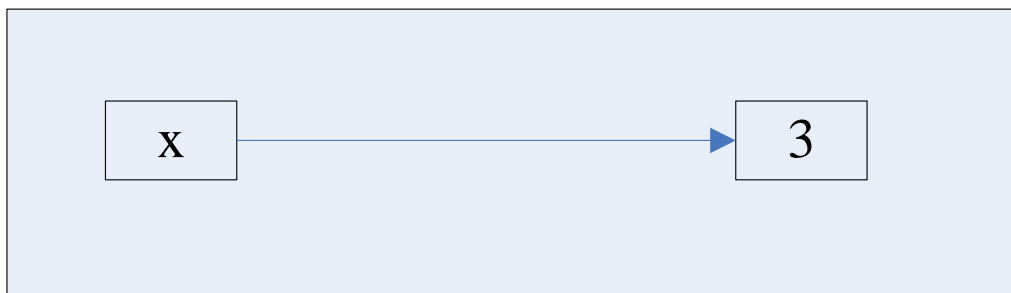
- 在Python中，允许多个变量指向同一个值，例如：

```
>>> x = 3
>>> id(x)
1786684560
>>> y = x
>>> id(y)
1786684560
```

- 然而，需要注意的是，继续上面的示例代码，当为其中一个变量修改值以后，其内存地址将会变化，但这并不影响另一个变量，例如接着上面的代码再继续执行下面的代码：

```
>>> x += 6
>>> id(x)
1786684752
>>> y
3
>>> id(y)
1786684560
```

# 1.4.2 Python变量



## 1.4.2 Python变量



- Python采用的是基于值的内存管理方式，如果为不同变量赋值为相同值，这个值在内存中只有一份，多个变量指向同一块内存地址，前面的几段代码也说明了这个特点。再例如下面的代码：

```
>>> x = 3
>>> id(x)
10417624
>>> y = 3
>>> id(y)
10417624
>>> y = 5
>>> id(y)
10417600
>>> id(x)
10417624
```

## 1.4.2 Python变量

---

- Python具有自动内存管理功能，对于没有任何变量指向的值，Python自动将其删除。Python会跟踪所有的值，并自动删除不再有变量指向的值。因此，Python程序员一般情况下不需要太多考虑内存管理的问题。尽管如此，显式使用`del`命令删除不需要的值或显式关闭不再需要访问的资源，仍是一个好的习惯，同时也是一个优秀程序员的基本素养之一。

## 1.4.2 Python变量



□最后，在定义变量名的时候，需要注意以下问题：

- ◆ 变量名必须以字母或下划线开头，但以下划线开头的变量在Python中有特殊含义，本书后面第6章会详细讲解；
- ◆ 变量名中不能有空格以及标点符号（括号、引号、逗号、斜线、反斜线、冒号、句号、问号等等）；
- ◆ 不能使用关键字作变量名，可以导入keyword模块后使用 `print(keyword.kwlist)` 查看所有Python关键字；
- ◆ 不建议使用系统内置的模块名、类型名或函数名以及已导入的模块名及其成员名作变量名，这将会改变其类型和含义，可以通过 `dir(__builtins__)` 查看所有内置模块、类型和函数；
- ◆ 变量名对英文字母的大小写敏感，例如 `student` 和 `Student` 是不同的变量。

- ```
9999999999999999999999999999999970000000000000000000000  
000000000000299999999999999999999999999999999999L
```

119

## 1.4.3 数字

---

- ❑ 十进制整数如，0、-1、9、123
- ❑ 十六进制整数，需要16个数字0、1、2、3、4、5、6、7、8、9、a、b、c、d、e、f来表示整数，必须以0x开头，如0x10、0xfa、0xabcdef
- ❑ 八进制整数，只需要8个数字0、1、2、3、4、5、6、7来表示整数，必须以0o开头，如0o35、0o11
- ❑ 二进制整数、只需要2个数字0、1来表示整数，必须以0b开头如，0b101、0b100



## 1.4.3 数字

---

□浮点数又称小数

15.0、0.37、-11.2、1.2e2、314.15e-2

## 1.4.3 数字

### □ Python内置支持复数类型。

```
>>> a = 3+4j
>>> b = 5+6j
>>> c = a+b
>>> c
(8+10j)
>>> c.real #查看复数实部
8.0
>>> c.imag #查看复数虚部
10.0
>>> a.conjugate() #返回共轭复数
(3-4j)
>>> a*b #复数乘法
(-9+38j)
>>> a/b #复数除法
(0.6393442622950819+0.03278688524590165j)
```

## 1.4.4 字符串

---

- ❑ 用单引号、双引号或三引号括起来的符号系列称为字符串
- ❑ 单引号、双引号、三单引号、三双引号可以互相嵌套，用来表示复杂字符串。
- ❑ 'abc'、'123'、'中国'、"Python"
- ❑ 字符串属于不可变序列
- ❑ 空串表示为""或""
- ❑ 三引号"""或"""表示的字符串可以换行，支持排版较为复杂的字符串；三引号还可以在程序中表示较长的注释。

## 1.4.4 字符串

---

### 1. 字符串合并

```
>>> a='abc' + '123'      #生成新对象
```

### 2. 字符串格式化

```
>>>a = 3.6674
```

```
>>>'%7.3f' % a
```

```
' 3.667'
```

```
>>> "%d:%c"%(65,65)
```

```
'65:A'
```

```
>>> """My name is %s, and my age is %d"""%('Dong Fuguo',38)
```

```
'My name is Dong Fuguo, and my age is 38'
```

## 1.4.4 字符串

---

### 3. 转义字符

- **\n**: 换行符
- **\t**: 制表符
- **\r**: 回车
- **\'**: 单引号
- **\"**: 双引号
- **\\**: 一个\
- **\ddd**: **3**位八进制数对应的字符
- **\xhh**: **2**位十六进制数对应的字符

字符串界定符前面加字母**r**表示原始字符串，其中的特殊字符不进行转义，但字符串的最后一个字符不能是**\**。

# 1.4.5 操作符和表达式

| 运算符示例                                         | 功能说明                    |
|-----------------------------------------------|-------------------------|
| <code>x+y</code>                              | 算术加法，列表、元组、字符串合并        |
| <code>x-y</code>                              | 算术减法，集合差集               |
| <code>x*y</code>                              | 乘法，序列重复                 |
| <code>x/y</code>                              | 除法（在Python 3.x中叫做真除法）   |
| <code>x//y</code>                             | 求整商                     |
| <code>-x</code>                               | 相反数                     |
| <code>x%y</code>                              | 余数（对实数也可以进行余数运算），字符串格式化 |
| <code>x**y</code>                             | 幂运算                     |
| <code>x&lt;y; x&lt;=y; x&gt;y; x&gt;=y</code> | 大小比较（可以连用），集合的包含关系比较    |
| <code>x==y; x!=y</code>                       | 相等（值）比较，不等（值）比较         |
| <code>x or y</code>                           | 逻辑或（只有x为假才会计算y）         |
| <code>x and y</code>                          | 逻辑与（只有x为真才会计算y）         |
| <code>not x</code>                            | 逻辑非                     |
| <code>x in y; x not in y</code>               | 成员测试运算符                 |
| <code>x is y; x is not y</code>               | 对象实体同一性测试（地址）           |
| <code> 、^、&amp;、&lt;&lt;、&gt;&gt;、~</code>    | 位运算符                    |
| <code>&amp;、 、^</code>                        | 集合交集、并集、对称差集            |

## 1.4.5 操作符和表达式

---

- Python中的除法有两种，“/”和“//”分别表示除法和整除运算，并且Python 2.x和Python 3.x对“/”运算符的解释也略有区别。Python 2.x将“/”解释为普通除法，而Python 3.x将其解释为真除法。例如，在Python 3.5.1中运算结果如下：

```
>>> 3/5
0.6
>>> 3//5
0
>>> 3.0/5
0.6
>>> 3.0//5
0.0
>>> 13//10
1
>>> -13//10
-2
```

## 1.4.5 操作符和表达式

---

□而上面的表达式在Python 2.7.11中运算结果如下：

```
>>> 3/5
```

```
0
```

```
>>> 3//5
```

```
0
```

```
>>> 3.0/5
```

```
0.6
```

```
>>> 3.0//5
```

```
0.0
```

```
>>> 13//10
```

```
1
```

```
>>> -13//10
```

```
-2
```



## 1.4.5 操作符和表达式

---

- 在Python中，除去前面已经介绍过的字符串格式化用法之外，该运算符还可以对整数和浮点数计算余数。但是由于浮点数的精确度影响，计算结果可能略有误差。

```
>>> 3.1%2
```

```
1.1
```

```
>>> 6.3%2.1
```

```
2.0999999999999996
```

```
>>> 6%2
```

```
0
```

```
>>> 6.0%2
```

```
0.0
```

```
>>> 6.0%2.0
```

```
0.0
```

```
>>> 5.7%4.8
```

```
0.90000000000000004
```

## 1.4.5 操作符和表达式

- “\*” 运算符是Python运算符中比较特殊的一个，它不仅可以用于数值乘法，还可以用于列表、字符串、元组等类型，当列表、字符串或元组等类型变量与整数进行“\*”运算时，表示对内容进行重复并返回重复后的新对象。

```
>>> 3*2 #整数相乘
6
>>> 2.0*3 #浮点数与整数相乘
6.0
>>> (3+4j)*2 #复数与整数相乘
(6+8j)
>>> (3+4j)*(3-4j) #复数与复数相乘
(25+0j)
>>> "a"*10 #字符串重复
'aaaaaaaaaa'
>>> [1, 2, 3]*3 #列表重复
[1, 2, 3, 1, 2, 3, 1, 2, 3]
>>> (1, 2, 3)*3 #元组重复
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

## 1.4.5 操作符和表达式

- 在Python中，单个任何类型的对象或常数属于合法表达式，使用运算符连接的变量和常量以及函数调用的任意组合也属于合法的表达式。

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a+b
>>> c
[1, 2, 3, 4, 5, 6]
>>> d = list(map(str, c))
>>> d
['1', '2', '3', '4', '5', '6']
>>> import math
>>> list(map(math.sin, c))
[0.8414709848078965, 0.9092974268256817, 0.1411200080598672, -
 0.7568024953079282, -0.9589242746631385, -0.27941549819892586]
>>> 'Hello' + ' ' + 'world'
'Hello world'
>>> 'welcome ' * 3
'welcome welcome welcome '
>>> ('welcome,'*3).rstrip(',')+'!'
'welcome, welcome, welcome!'
```

## 1.4.5 操作符和表达式

---

- 在Python中逗号 “,” 并不是运算符，而只是一个普通分隔符。

```
>>> 'a' in 'b', 'a'
```

```
(False, 'a')
```

```
>>> 'a' in ('b', 'a')
```

```
True
```

```
>>> x = 3, 5
```

```
>>> x
```

```
(3, 5)
```

```
>>> 3 == 3, 5
```

```
(True, 5)
```

```
>>> x = 3+5, 7
```

```
>>> x
```

```
(8, 7)
```

## 1.4.6 常用内置函数

---

- ❑ 内置函数不需要导入任何模块即可使用
- ❑ 执行下面的命令可以列出所有内置函数

```
>>> dir(__builtins__)
```

# 1.4.6 常用内置函数

| 函数                                              | 功能简要说明                                                                      |
|-------------------------------------------------|-----------------------------------------------------------------------------|
| <code>abs(x)</code>                             | 返回数字x的绝对值                                                                   |
| <code>all(iterable)</code>                      | 如果对于可迭代对象中所有元素x都有 <code>bool(x)</code> 为True，则返回True。对于空的可迭代对象也返回True       |
| <code>any(iterable)</code>                      | 只要可迭代对象中存在元素x使得 <code>bool(x)</code> 为True，则返回True。对于空的可迭代对象，返回False        |
| <code>bin(x)</code>                             | 把数字x转换为二进制串                                                                 |
| <code>callable(object)</code>                   | 测试对象是否可调用。类和函数是可调用的，包含 <code>__call__()</code> 方法的类的对象也是可调用的                |
| <code>chr(x)</code>                             | 返回ASCII编码为x的字符                                                              |
| <code>cmp(x, y)</code>                          | 比较大小，如果 $x < y$ 则返回负数，如果 $x == y$ 则返回0，如果 $x > y$ 则返回正数。Python 3.x不再支持该函数   |
| <code>dir()</code>                              | 返回指定对象的成员列表                                                                 |
| <code>eval(s[, globals[, locals]])</code>       | 计算字符串中表达式的值并返回                                                              |
| <code>filter(function or None, sequence)</code> | 返回序列中使得函数值为True的那些元素，如果函数为None则返回那些值等价于True的元素。如果序列为元组或字符串则返回相同类型结果，其他则返回列表 |

# 1.4.6 常用内置函数



|                                                                                                   |                                                               |
|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| <code>float(x)</code>                                                                             | 把数字或字符串x转换为浮点数并返回                                             |
| <code>help(obj)</code>                                                                            | 返回对象obj的帮助信息                                                  |
| <code>hex(x)</code>                                                                               | 把数字x转换为十六进制串                                                  |
| <code>id(obj)</code>                                                                              | 返回对象obj的标识（地址）                                                |
| <code>input([提示内容字符串])</code>                                                                     | 接收键盘输入的内容，返回字符串。Python 2.x和Python 3.x对该函数的解释不完全一样，详见后面的1.4.8节 |
| <code>int(x[, d])</code>                                                                          | 返回数字的整数部分，或把d进制的字符串x转换为十进制并返回，d默认为十进制                         |
| <code>isinstance(object, class-or-type-or-tuple)</code>                                           | 测试对象是否属于指定类型的实例                                               |
| <code>len(obj)</code>                                                                             | 返回对象obj包含的元素个数，适用于列表、元组、集合、字典、字符串等类型的对象                       |
| <code>list([x])</code> 、 <code>set([x])</code> 、 <code>tuple([x])</code> 、 <code>dict([x])</code> | 把对象转换为列表、集合、元组或字典并返回，或生成空列表、空集合、空元组、空字典                       |
| <code>map(函数,序列)</code>                                                                           | 将函数映射至序列中每个元素，返回列表或map对象                                      |
| <code>max(x)</code> 、 <code>min(x)</code> 、 <code>sum(x)</code>                                   | 返回序列中的最大值、最小值或数值元素之和                                          |

## 1.4.6 常用内置函数

|                                                |                                                                      |
|------------------------------------------------|----------------------------------------------------------------------|
| <code>open(name[, mode[, buffering]])</code>   | 以指定模式打开文件并返回文件对象                                                     |
| <code>ord(s)</code>                            | 返回1个字符s的编码                                                           |
| <code>pow(x, y)</code>                         | 返回x的y次方，等价于 <code>x**y</code>                                        |
| <code>range([start, ] end [, step] )</code>    | 返回一个等差数列（Python 3.x中返回一个range对象），不包括终值                               |
| <code>reduce(函数, 序列)</code>                    | 将接收2个参数的函数以累积的方式从左到右依次应用至序列中每个元素，最终返回单个值作为结果                         |
| <code>reversed(列表或元组)</code>                   | 返回逆序后的迭代器对象                                                          |
| <code>round(x [, 小数位数])</code>                 | 对x进行四舍五入，若不指定小数位数，则返回整数                                              |
| <code>str(obj)</code>                          | 把对象obj转换为字符串                                                         |
| <code>sorted(列表[, cmp[, key[reverse]]])</code> | 返回排序后的列表。Python 3.x中的 <code>sorted()</code> 方法没有 <code>cmp</code> 参数 |
| <code>type(obj)</code>                         | 返回对象obj的类型                                                           |
| <code>zip(seq1 [, seq2 [...]])</code>          | 返回 <code>[(seq1[0], seq2[0] ...), (...)]</code> 形式的列表                |



## 1.4.6 常用内置函数

- `ord()` 和 `chr()` 是一对功能相反的函数，`ord()` 用来返回单个字符的序数或Unicode码，而 `chr()` 则用来返回某序数对应的字符，`str()` 则直接将其任意类型参数转换为字符串。

```
>>> ord('a')
97
>>> chr(65)
'A'
>>> chr(ord('A')+1)
'B'
>>> str(1)
'1'
>>> str(1234)
'1234'
>>> str([1, 2, 3])
'[1, 2, 3]'
>>> str((1, 2, 3))
'(1, 2, 3)'
>>> str({1, 2, 3})
'set([1, 2, 3])'
```

## 1.4.6 常用内置函数

- `max()`、`min()`、`sum()` 这三个内置函数分别用于计算列表、元组或其他可迭代对象中所有元素最大值、最小值以及所有元素之和，`sum()` 只支持数值型元素的序列或可迭代对象，`max()` 和 `min()` 则要求序列或可迭代对象中的元素之间可比较大小。例如下面的示例代码，首先使用列表推导式生成包含10个随机数的列表，然后分别计算该列表的最大值、最小值和所有元素之和。

```
>>> import random
>>> a = [random.randint(1,100) for i in range(10)]
>>> a
[72, 26, 80, 65, 34, 86, 19, 74, 52, 40]
>>> print(max(a), min(a), sum(a))
86 19 548
```

- 如果需要计算该列表中的所有元素的平均值，可以直接使用下面的方法：

```
>>> a = [72, 26, 80, 65, 34, 86, 19, 74, 52, 40]
>>> sum(a)*1.0/len(a) #Python 2.7.11
54.8
>>> sum(a)/len(a) #Python 3.5.1
```

54.8

## 1.4.6 常用内置函数

---

- 使用`dir()`函数可以查看指定模块中包含的所有成员或者指定对象类型所支持的操作，而`help()`函数则返回指定模块或函数的说明文档。

## 1.4.7 对象的删除

---

- ❑ 在Python中具有自动内存管理功能，Python解释器会跟踪所有的值，一旦发现某个值不再有任何变量指向，将会自动删除该值。尽管如此，自动内存管理或者垃圾回收机制并不能保证及时释放内存。显式释放自己申请的资源是程序员的好习惯之一，也是程序员素养的重要体现之一。
- ❑ 在Python中，可以使用del命令来显式删除对象并解除与值之间的指向关系。删除对象时，如果其指向的值还有别的变量指向则不删除该值，如果删除对象后该值不再有其他变量指向，则删除该值。

## 1.4.7 对象的删除

---

```
>>> x = [1, 2, 3, 4, 5, 6]
>>> y = 3
>>> z = y
>>> print(y)
3
>>> del y #删除对象
>>> print(y)
NameError: name 'y' is not defined
>>> print(z)
3
>>> del z
>>> print(z)
NameError: name 'z' is not defined
>>> del x[1] #删除列表中指定元素
>>> print(x)
[1, 3, 4, 5, 6]
>>> del x #删除整个列表
>>> print(x)
NameError: name 'x' is not defined
```

## 1.4.7 对象的删除

---

- del命令无法删除元组或字符串中的指定元素，而只可以删除整个元组或字符串，因为这两者均属于不可变序列。

```
>>> x = (1, 2, 3)
```

```
>>> del x[1]
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#62>", line 1, in <module>
```

```
    del x[1]
```

```
TypeError: 'tuple' object doesn't support item deletion
```

```
>>> del x
```

```
>>> print(x)
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#64>", line 1, in <module>
```

```
    print(x)
```

```
NameError: name 'x' is not defined
```

---

## 1.4.8 基本输入输出

---

用Python进行程序设计，输入是通过`input( )`函数来实现的，`input( )`的一般格式为：

```
x=input('提示：')
```

该函数返回输入的对象。可输入数字、字符串和其它任意类型对象。

## 1.4.8 基本输入输出

- 尽管形式一样，Python 2.x和Python 3.x对该函数的解释略有不同。在Python 2.x中，该函数返回结果的类型由输入值时所使用的界定符来决定，例如下面的Python 2.7.11代码：

```
>>> x = input("Please input:")
Please input:3 #没有界定符，整数
>>> print type(x)
<type 'int'>

>>> x = input("Please input:")
Please input:'3' #单引号，字符串
>>> print type(x)
<type 'str'>

>>> x = input("Please input:")
Please input:[1,2,3] #方括号，列表
>>> print type(x)
<type 'list'>
```



## 1.4.8 基本输入输出

- 在Python 3.x中，只提供了input()函数用来接收用户的键盘输入。在Python 3.x中，不论用户输入数据时使用什么界定符，input()函数的返回结果都是字符串，需要将其转换为相应的类型再处理。例如下面的Python 3.5.1代码：

```
>>> x = input('Please input:')
Please input:3
>>> print(type(x))
<class 'str'>
>>> x = input('Please input:')
Please input:'1'
>>> print(type(x))
<class 'str'>
>>> x = input('Please input:')
Please input:[1,2,3]
>>> print(type(x))
<class 'str'>
>>> x = raw_input('Please input:')
NameError: name 'raw_input' is not defined
```

## 1.4.8 基本输入输出

---

- Python 2.x和Python 3.x的输出方法也不完全一致。  
在Python 2.x中，使用`print`语句进行输出，而Python 3.x中使用`print()`函数进行输出。

## 1.4.8 基本输入输出

---

- 默认情况下，Python将结果输出到IDLE或者标准控制台，在输出时也可以进行重定向，例如可以把结果输出到指定文件。在Python 2.7.11中使用下面的方法进行输出重定向：

```
>>> fp = open(r'C:\mytest.txt', 'a+')
>>> print >>fp, "Hello, world"
>>> fp.close()
```

- 而在Python 3.5.1中则需要使用下面的方法进行重定向：

```
>>> fp = open(r'D:\mytest.txt', 'a+')
>>> print('Hello, world!', file = fp)
>>> fp.close()
```

## 1.4.8 基本输入输出

---

- 另外一个重要的不同是，对于Python 2.x而言，在print语句之后加上逗号“,”则表示输出内容之后不换行，例如：

```
>>> for i in range(10):
```

```
    print i,
```

```
0 1 2 3 4 5 6 7 8 9
```

- 在Python 3.x中，为了实现上述功能则需要使用下面的方法：

```
>>> for i in range(10, 20):
```

```
    print(i, end=' ')
```

```
10 11 12 13 14 15 16 17 18 19
```

# Python2.x和Python3.x

## 1. print函数替代了print语句

```
>>> print(3,4)
(3, 4)
```

Python2.x

```
>>> print(3,4)
3 4
```

Python3.x

# Python2.x和Python3.x

## 2. python3.x默认使用UTF-8编码

```
>>> str = "我爱北京天安门"  
>>> str  
'\xe6\x88\x91\xe7\x88\xb1\xe5\x8c\x97\xe4\xba\xac\xe5\xa4\xa9\xe5\xae\x89\xe9\x97\xa8'
```

Python2.x

```
>>> str = "我爱北京天安门"  
>>> str  
'我爱北京天安门'
```

Python3.x

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

# Python2.x和Python3.x

## 3. 除法运算符/

```
>>> 1 / 2
```

```
0
```

```
>>> 1.0 / 2.0
```

```
0.5
```

Python2.x

```
>>> 1/2
```

```
0.5
```

Python3.x

# Python2.x和Python3.x

## 4. 异常

– 捕获异常的语法是  
`except exc,var .`

Python2.x

– 被抛出捕获异常的语法  
变更为  
`except exc as var`

Python3.x



## Python2.x和Python3.x

### 5. 八进制字面量表示0b\0o\64\0x

```
>>> 0o1000
512
>>> 01000
512
```

Python2.x

```
>>> 01000
File "<stdin>", line 1
    01000
      ^
SyntaxError: invalid token
>>> 0o1000
512
```

Python3.x

# Python2.x和Python3.x

## 6. 不等运算符

```
>>> 1!=2
True
>>> 1<>2
True
```

Python2.x

```
>>> 1!=2
True
>>> 1<>2
File "<stdin>", line 1
    1<>2
      ^
SyntaxError: invalid syntax
```

Python3.x

# Python2.x和Python3.x

## 7. 数据类型

– 有long类型

Python2.x

– 没有long，只有int

Python3.x

## 1.4.9 模块的使用

---

- ❑ Python默认安装仅包含部分基本或核心模块，但用户可以安装大量的扩展模块，**pip**是管理模块的重要工具。
- ❑ 在Python启动时，仅加载了很少的一部分模块，在需要时由程序员显式地加载（可能需要先安装）其他模块。
- ❑ 减小运行的压力，仅加载真正需要的模块和功能，且具有很强的可扩展性。
- ❑ 可以使用

```
>>> import sys
```

```
>>> sys.modules.items()
```

显示所有预加载模块的相关信息。

## 1.4.9 模块的使用

---

### □ import 模块名

```
>>>import math
```

```
>>>math.sin(0.5)          #求0.5的正弦
```

```
>>>import random
```

```
>>>x=random.random( )    #获得[0,1) 内的随机小数
```

```
>>>y=random.random( )
```

```
>>>n=random.randint(1,100) #获得[1,100]上的随机整数
```

### □ 可以使用**dir**函数查看任意模块中所有的对象列表，如果调用不带参数的**dir()**函数，则返回当前脚本的所有名字列表。

### □ 可以使用**help**函数查看任意模块或函数的使用帮助。

## 1.4.9 模块的使用

---

❑ `from 模块名 import 对象名[ as 别名]` #可以减少查询次数，提高执行速度

❑ `from math import *` #谨慎使用

```
>>> from math import sin
```

```
>>> sin(3)
```

```
0.1411200080598672
```

```
>>> from math import sin as f #别名
```

```
>>> f(3)
```

```
0.141120008059867
```

## 1.4.9 模块的使用

---

- ❑ 在2.x中可以使用`reload`函数重新导入一个模块，在3.x中，需要使用`imp`模块的`reload`函数
- ❑ Python首先在当前目录中查找需要导入的模块文件，如果没有找到则从`sys`模块的`path`变量所指定的目录中查找。可以使用`sys`模块的`path`变量查看python导入模块时搜索模块的路径，也可以向其中`append`自定义的目录以扩展搜索路径。
- ❑ 在导入模块时，会优先导入相应的`pyc`文件，如果相应的`pyc`文件与`py`文件时间不相符，则导入`py`文件并重新编译该模块。

# 1.5 Python代码规范

---



## (1) 缩进

- ❑ 类定义、函数定义、选择结构、循环结构，行尾的冒号表示缩进的开始
- ❑ python程序是依靠代码块的缩进来体现代码之间的逻辑关系的，缩进结束就表示一个代码块结束了。
- ❑ 同一个级别的代码块的缩进量必须相同。
- ❑ 一般而言，以4个空格为基本缩进单位，可以通过下面的方法进行代码块的缩进和反缩进：

**Format→Indent Region/Dedent Region**



# 1.5 Python代码规范

---

## (2) 注释

一个好的、可读性强的程序一般包含**30%**以上的注释。常用的注释方式主要有两种：

- ❑ 以#开始，表示本行#之后的内容为注释
- ❑ 包含在一对三引号'''...'''或"""..."""之间且不属于任何语句的内容将被解释器认为是注释

在**IDLE**开发环境中，可以通过下面的操作快速注释/解除注释大段内容：

- ❑ **Format**→**Comment Out Region/Uncomment Region**

# 1.5 Python代码规范

---



- (3) 每个import只导入一个模块
- (4) 如果一行语句太长，可以在行尾加上\来换行分成多行，但是更建议使用括号来包含多行内容。
- (5) 必要的空格与空行
  - 运算符两侧、函数参数之间、逗号两侧建议使用空格分开。
  - 不同功能的代码块之间、不同的函数定义之间建议增加一个空行以增加可读性。
- (6) 适当使用异常处理结构进行容错，后面将详细讲解。
- (7) 软件应具有较强的可测试性，测试与开发齐头并进。

# 1.6 Python文件名

---

- ❑ `.py`: Python源文件，由Python解释器负责解释执行。
  - ❑ `.pyw`: Python源文件，常用于图形界面程序文件。
  - ❑ `.pyc`: Python字节码文件，无法使用文本编辑器直接查看该类型文件内容，可用于隐藏Python源代码和提高运行速度。对于Python模块，第一次被导入时将被编译成字节码的形式，并在以后再次导入时优先使用“`.pyc`”文件，以提高模块的加载和运行速度。对于非模块文件，直接执行时并不生成“`.pyc`”文件，但可以使用`py_compile`模块的`compile()`函数进行编译以提高加载和运行速度。另外，Python还提供了`compileall`模块，其中包含`compile_dir()`、`compile_file()`和`compile_path()`等方法，用来支持批量Python源程序文件的编译。
-

## 1.6 Python文件名（续）

---

- .pyo: 优化的Python字节码文件，同样无法使用文本编辑器直接查看其内容。可以使用“`python -O -m py_compile file.py`”或“`python -OO -m py_compile file.py`”进行优化编译。
- .pyd: 一般是由其他语言编写并编译的二进制文件，常用于实现某些软件工具的Python编程接口插件或Python动态链接库。

## 1.7 Python脚本的“\_\_name\_\_”属性

---

- 每个Python脚本在运行时都有一个“\_\_name\_\_”属性。如果脚本作为模块被导入，则其“\_\_name\_\_”属性的值被自动设置为模块名；如果脚本独立运行，则其“\_\_name\_\_”属性值被自动设置为“\_\_main\_\_”。例如，假设文件nametest.py中只包含下面一行代码：

```
print(__name__)
```

- 在IDLE中直接运行该程序时，或者在命令行提示符环境中运行该程序文件时，运行结果如下：

```
__main__
```

- 而将该文件作为模块导入时得到如下执行结果：

```
>>> import nametest  
  
nametest
```

## 1.7 Python脚本的“\_\_name\_\_”属性（续）

---

- 利用“\_\_name\_\_”属性即可控制Python程序的运行方式。例如，编写一个包含大量可被其他程序利用的函数的模块，而不希望该模块可以直接运行，则可以在程序文件中添加以下代码：

```
if __name__ == '__main__':  
    print('Please use me as a module.')
```

- 这样一来，程序直接执行时将会得到提示“Please use me as a module.”，而使用import语句将其作为模块导入后可以使用其中的类、方法、常量或其他成员。

## 1.8 编写自己的包与模块

---

- ❑ 包可以看做处于同一目录中的模块。
- ❑ 在包的每个目录中都必须包含一个 `__init__.py` 文件，该文件可以是一个空文件，仅用于表示该目录是一个包。
- ❑ `__init__.py` 文件的主要用途是设置 `__all__` 变量以及所包含的包初始化所需的代码。其中 `__all__` 变量中定义的对象可以在使用 `from ...import *` 时全部正确导入。

# 1.9 Python快速入门

---

□ 问题1：已知三角形的两边长及其夹角，求第三边长。

```
import math
x = input('输入两边长及夹角（度）： ')
a, b, theta = map(float, x.split()) #split默认分隔符是空格
c = math.sqrt(a**2 + b**2 - \
              2*a*b*math.cos(theta*math.pi/180))
print('c=', c)
```



# 1.9 Python快速入门

---

- 问题2：任意输入三个英文单词，按字典顺序输出。

```
s = input('x,y,z=')
```

```
x, y, z = s.split(',')
```

```
if x > y:
```

```
    x, y = y, x
```

```
if x > z:
```

```
    x, z = z, x
```

```
if y > z:
```

```
    y, z = z, y
```

```
print(x, y, z)
```

- 或直接写为：

```
s = input('x,y,z=')
```

```
x, y, z = s.split(',')
```

```
x, y, z = sorted([x, y, z])
```

```
print(x, y, z)
```

# 例3: Python程序框架生成器

---



```
import os
import sys
import datetime
head = '# '+'-'*20+' \n'+\
      '# Function description:\n'+\
      '# '+'-'*20+' \n'+\
      '# Author: Dong Fuguo\n'+\
      '# QQ: 306467355\n'+\
      '# Email: dongfuguo2005@126.com\n'+\
      '# '+'-'*20+' \n'
```

---

```
desFile = sys.argv[1]
if os.path.exists(desFile) or not desFile.endswith('.py'):
    print('%s already exist or is not a Python code file.!'%desFile)
    sys.exit()
fp = open(desFile, 'w')
today = str(datetime.date.today().year)+'-'
'+str(datetime.date.today().month)+'\
    '-' +str(datetime.date.today().day)
fp.write('# -*- coding:utf-8 -*-\n')
fp.write('# Filename: '+desFile+' \n')
fp.write(head)
fp.write('# Date: '+today+' \n')
fp.write('# '+'-'*20+' \n')
fp.close()
```

---

## 1.10 Python练习

---

- 问题：用户输入一个三位自然数，计算并输出其百位、十位和个位上的数字。

```
x = input('请输入一个三位数：')
```

```
.....
```

谢谢Q/A