



Chap1 Approach Python

第1章 走近Python

Department of Computer Science and Technology
Department of University Basic Computer Teaching
Nanjing University

Rank	Change	Language	Share	Trend
1		Python	29.88 %	+4.1 %
2		Java	19.05 %	-1.8 %
3		Javascript	8.17 %	+0.1 %
4		C#	7.3 %	-0.1 %
5		PHP	6.15 %	-1.0 %
6		C/C++	5.92 %	-0.2 %
7		R	3.74 %	-0.2 %
8		Objective-C	2.42 %	-0.6 %
9		Swift	2.28 %	-0.2 %
10	↑	TypeScript	1.84 %	+0.3 %



编程语言流行指数 ([PyPL](#))

Python指数——TIOBE

3

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1	1		Java	17.358%	+1.48%
2	2		C	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%
5	7	▲	C#	5.927%	+3.08%
6	5	▼	Visual Basic .NET	5.862%	-1.23%
7	6	▼	JavaScript	2.060%	-0.79%
8	8		PHP	2.018%	-0.25%
9	9		SQL	1.526%	-0.37%
10	20	▲▲	Swift	1.460%	+0.54%

Python指数——IEEE Spectrum

4

Choose a Ranking

IEEE Spectrum | Trending

Jobs | Open | Custom

Create custom ranking

Language Types

Web | Enterprise

Mobile | Embedded

(Click to hide)

Language Ranking: **Jobs**

Rank	Language	Type	Score
1	Python	Web, Enterprise, Mobile, Embedded	100.0
2	Java	Web, Mobile, Embedded	97.5
3	C	Mobile, Embedded	96.0
4	C++	Mobile, Embedded	85.7
5	JavaScript	Web	83.1
6	C#	Web, Mobile, Embedded	77.2
7	HTML, CSS	Web	75.6
8	Swift	Mobile, Embedded	69.4
9	Matlab	Embedded	69.4
10	SQL	Embedded	69.3

The IEEE Spectrum Top Programming Languages app synthesizes 11 metrics from 9 sources to arrive at an overall ranking of language popularity. The sources cover contexts that include social chatter, open-source code production, and job postings judged by searching “X programming” in Google Search, Google Trends, Twitter, Github, Stack Overflow etc.

Python应用实例



1.1

PYTHON简介

1.1.1 PYTHON的历史和特性



Guido van Rossum

python 的诞生

- 第1个Python编译器/解释器于1991年诞生
- Python名称来自Guido挚爱的电视剧Monty Python's Flying Circus
- Python介于C和Shell之间、功能全面、易学易用、可扩展

Python的哲学

优雅

Python在其表达方式和语法形式等多个方面均体现其优雅

明确

拥有传统编译型程序语言所有强大通用的功能

简单

拥有简单脚本语言和解释型程序语言的易用性

- 胶水语言 Glue Language

- 很容易和其他著名的程序语言连接 (C/C++)，集成封装

- 脚本语言 Script Language

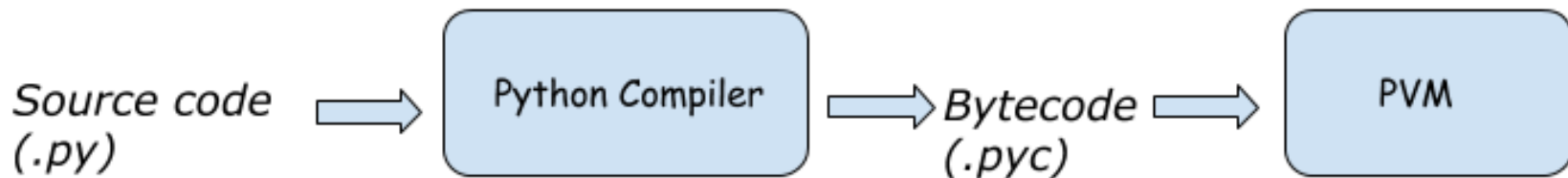
- 高级脚本语言，比脚本语言只能处理简单任务强大

- 面向对象语言 Object-Oriented Language

- 完全支持继承、重载、派生、多继承

Python运行机制

11



From: <https://www.codecompiled.com/>

Python的应用（一）



Web
开发

Python定义了WSGI标准应用接口来协调http服务器与基于Python的Web程序之间的沟通



大数据

Python提供各种库，具有极其便捷强大的数据处理和统计功能

Python的应用 (二)

13



人工
智能

基于丰富的Python第三方库可以便捷高效地实现人工智能的各个阶段任务



多媒体

可用于计算机游戏三维场景制作和各种专业领域应用

1.1.2 PYTHON的版本

Python的2个版本

A blue rectangular box with a white border and a white wavy top edge, containing the text "Python 2.x". An orange triangle is visible behind the top right corner of the box.

Python 2.x

A blue rectangular box with a white border and a white wavy top edge, containing the text "Python 3.x". An orange triangle is visible behind the top right corner of the box.

Python 3.x



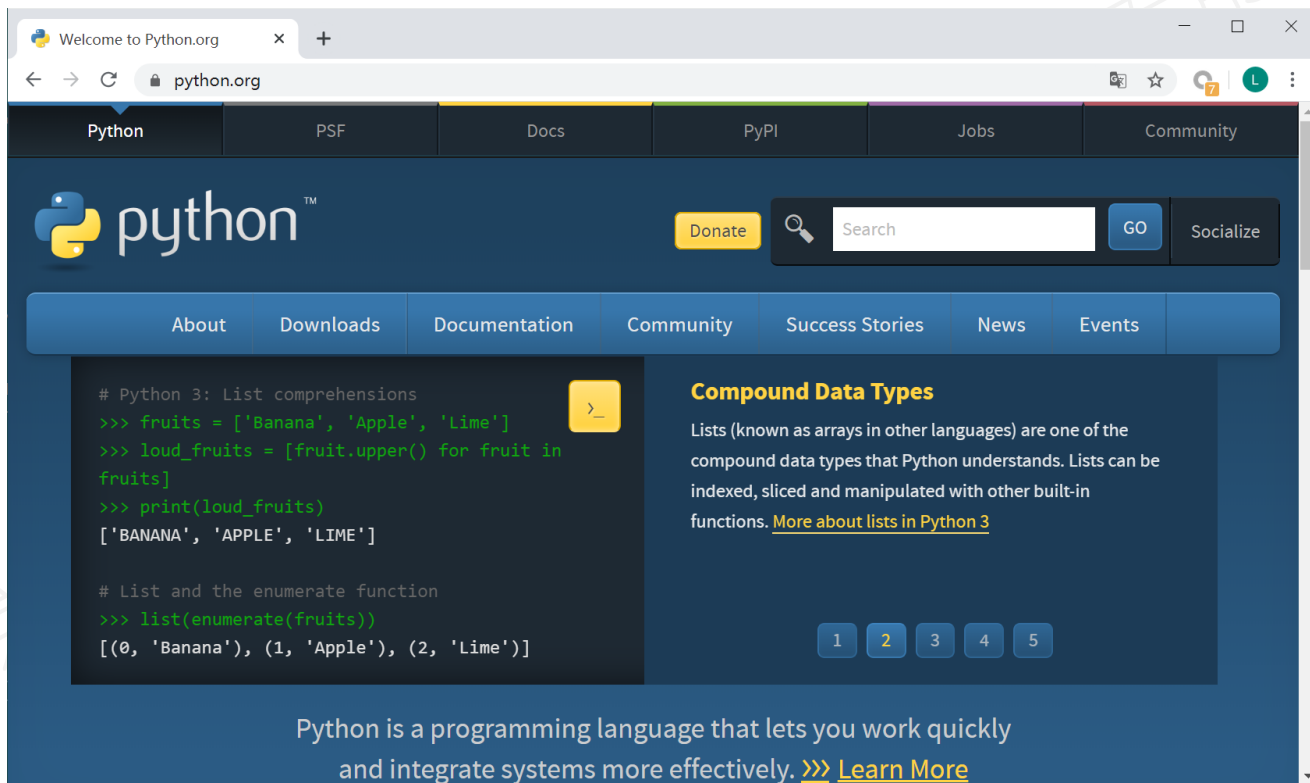
互不兼容

本课程以Python3.x为主



Python官方网站

16



Python 格言

The Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than **right** now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

```
>>> import this
```

by Tim Peters

1.2

PYTHON开发 环境和运行方式

经典 Hello World

19

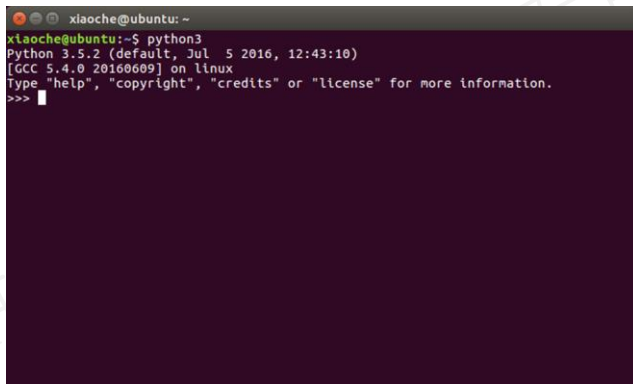
```
myString = 'Hello, World!'
```

```
print(myString)
```

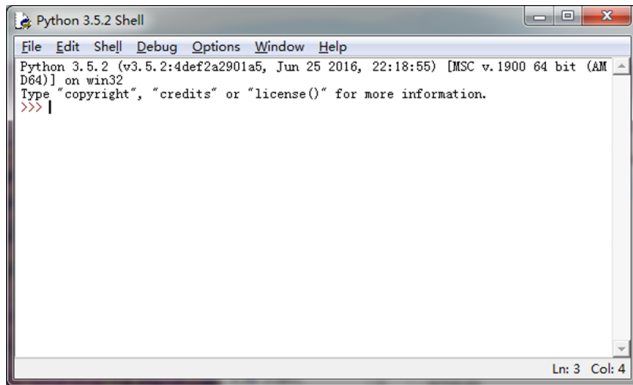
1.2.1 PYTHON开发环境

Python开发环境1

- Mac OS & Linux
 - \$ python
 - \$ python3
- Python内置IDE
 - IDLE



```
xiaoche@ubuntu: ~  
xiaoche@ubuntu:~$ python3  
Python 3.5.2 (default, Jul 5 2016, 12:43:10)  
[GCC 5.4.0 20160609] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> |
```



```
Python 3.5.2 Shell  
File Edit Shell Debug Options Window Help  
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> |
```

Python开发环境2

22



最省事方案：Python平台Anaconda

- **Anaconda**

- download the installer

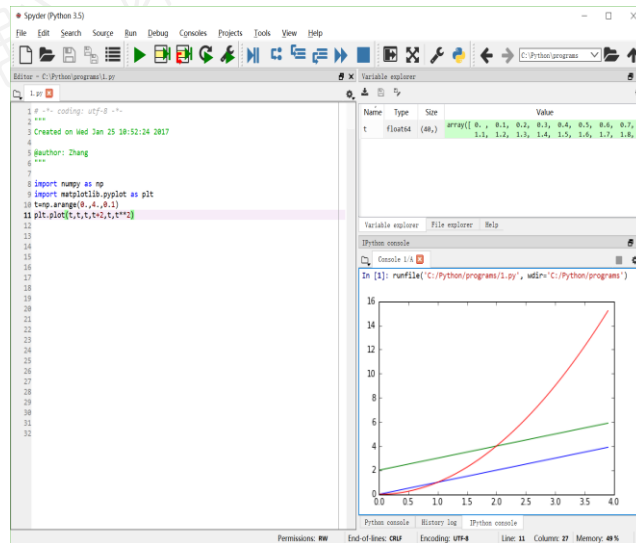
官网或清华镜像站

<https://www.anaconda.com/distribution/>

- install and use

勾选 “Add Anaconda to my PATH environment variable”

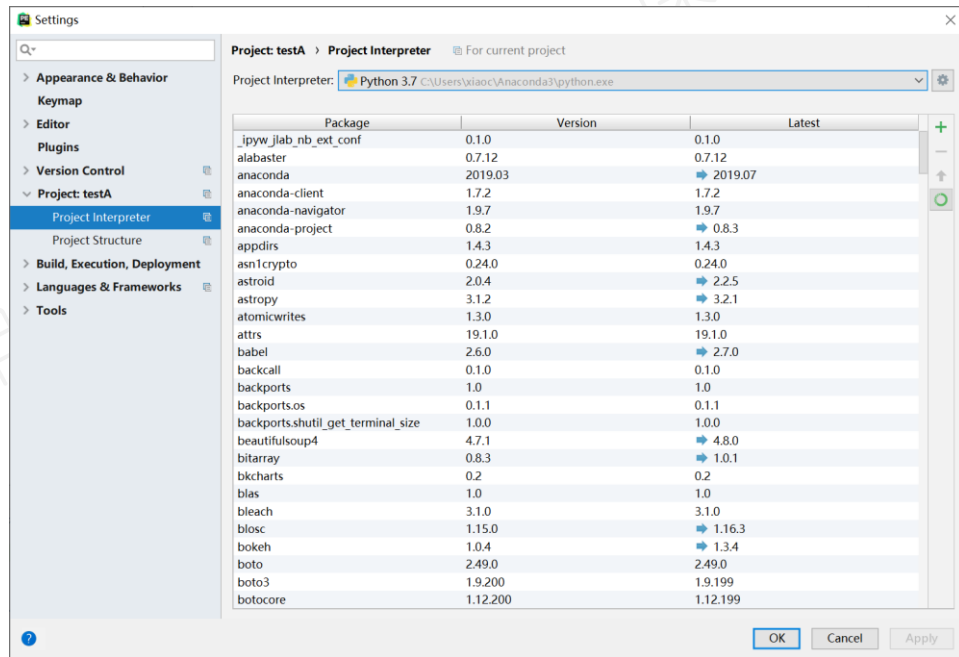
安装后一般使用Spyder Python解释器编辑和执行程序



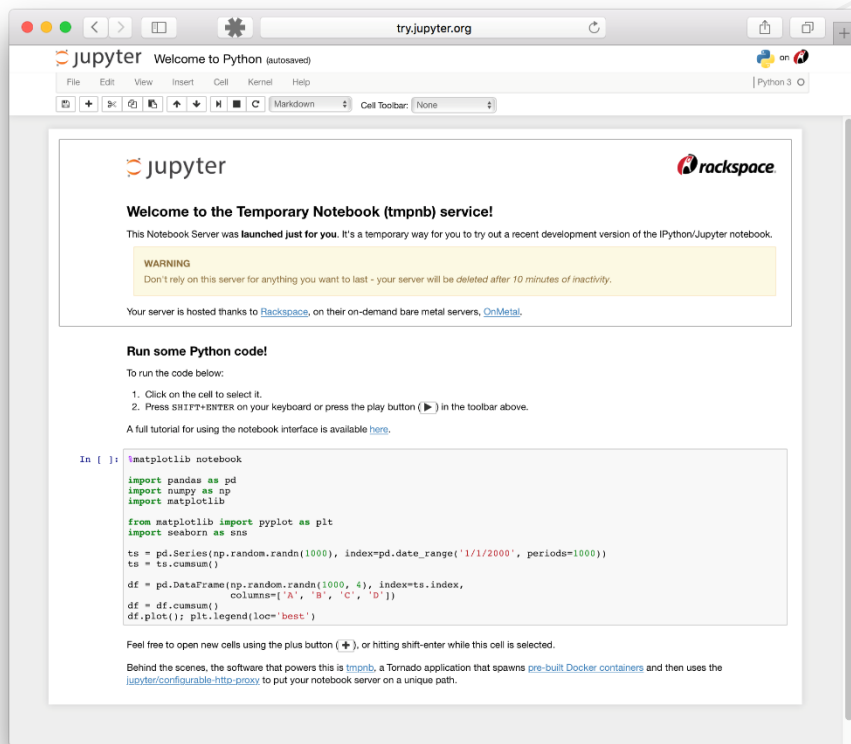
继续推荐（稳定+丰富的第三方包）

PyCharm + Anaconda interpreter

使用“Existing interpreter”，选择Anaconda的python.exe路径



Jupyter Notebook



Install packages

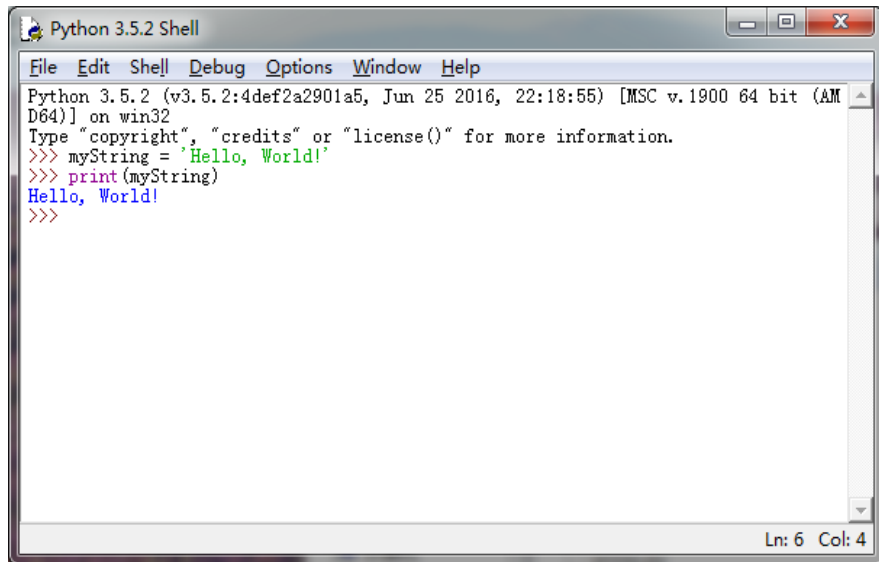
安装插件

- 用pip/conda在操作系统终端安装第三方包，在官网<https://pypi.org/>页面搜索，例如：
 > pip install jieba

1.2.2 PYTHON运行方式

Python的运行方式（一）

Shell方式



The screenshot shows a window titled "Python 3.5.2 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The text area contains the following text:

```
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:18:55) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> myString = 'Hello, World!'
>>> print(myString)
Hello, World!
>>>
```

The status bar at the bottom right indicates "Ln: 6 Col: 4".

- Shell是交互式的解释器
- 输入一行命令，解释器就解释运行出相应结果

Python的运行方式（二）

文件执行方式

- 在Python的IDE环境中，创建一个以py为扩展名的文件
- 用Python解释器在Shell中运行出结果



```
C:\Users\xiaoc>python test.py
```

经典 Hello World



```
>>> myString = 'Hello, World!'
>>> print(myString)
Hello, World!
>>> myString
'Hello, World!'
```



```
# Filename: helloworld.py
myString = 'Hello, World!'
print(myString)
```

Shell方式和文件执行方式

- 都是解释运行方式
 - 如果整个代码段较短，则优先考虑选择用Shell交互方式
 - 如果代码段较长，则建议使用文件执行方式



1.3

PYTHON程序 基本构成与风格

1.3.1 PYTHON程序基本构成

一个小程序

34



```
# Filename: prog2-1.py
```

```
# For loop on a list
```

```
num = [1, 2, 3, 4, 5]
```

```
prog = int(input('please input the value of prog: '))
```

```
for number in num:
```

```
    prog = prog * number
```

```
print('The prog is: ', prog)
```

第1行

第2行

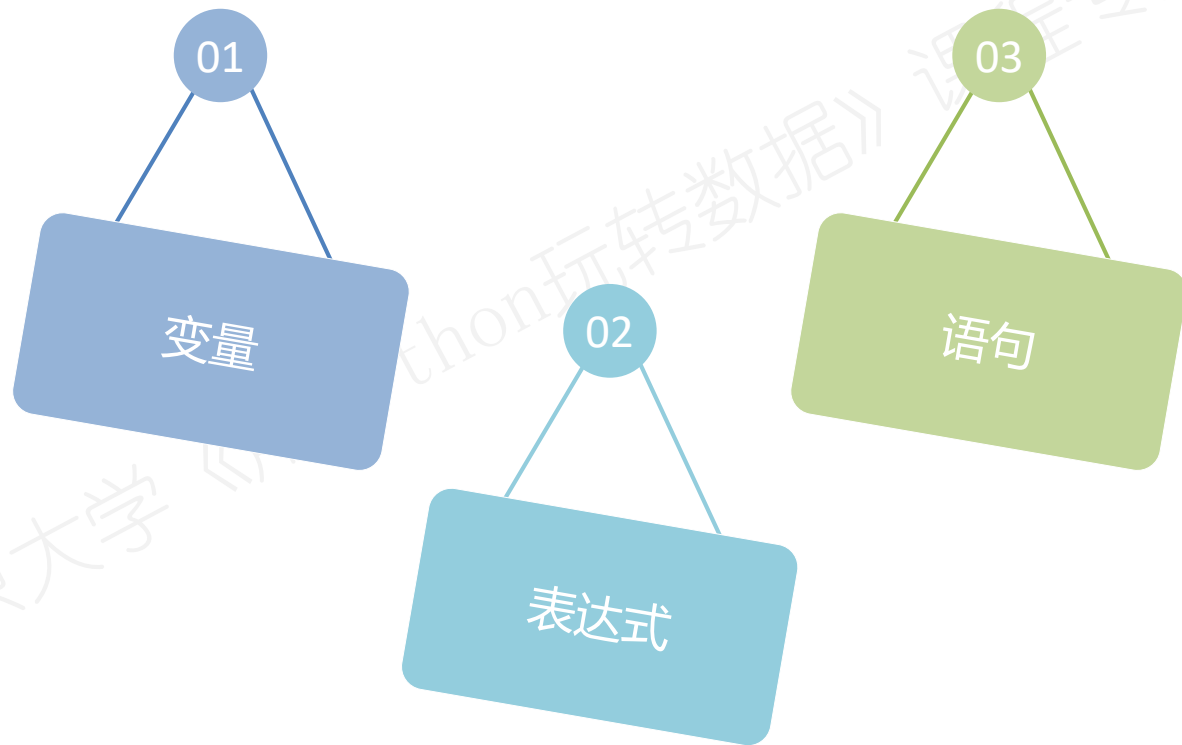
第3行

第4行

第5行

第6行

程序基本要素



Python输出: print函数

36

- Python使用print函

数实现输出:

- print(变量)
- print(字符串)



```
>>> myString = 'Hello, World!'
```

```
>>> print(myString)
```

```
Hello, World!
```

Python输入: input()函数

S
ource

```
>>> price = input('input the stock price of Apple: ')
```

```
input the stock price of Apple: 109
```

```
>>> price
```

```
'109'
```

```
>>> type(price)
```

```
<class 'str'>
```

```
>>> price = int(input('input the stock price of Apple: '))
```

```
>>> price = eval(input('input the stock price of Apple: '))
```

input()
返回的类型
是字符型

1.3.2 PYTHON程序设计风格

单行注释



```
>>> # For loop on a list          # 第1行
>>> print('The prog is: ', prog) # 第6行
```



```
"""
i = 2
i = 3**i
print(i)
"""
```

缩进

01

增加缩进
表示语句
块的开始

Python用相
同的缩进表示
同级别语句块

02

减少缩进
表示语句
块的退出

03

S
ource

```
# prog2-1.py
# For loop on a list          # 第1行
num = [1, 2, 3, 4, 5]
prog = int(input("please input the value of prog: "))
for number in num:
    prog = prog * number
print("The prog is: ", prog)
```


缩进

A speech bubble icon containing the word "Source" in orange text.

```
i = 1
s = 0
while i <= 5:
    s = s + i
    i = i + 1
print('The sum is: ', s)
```

Python 风格 (三)

42

续行

```
>>> # long statement
>>> if signal == 'red' and \
>>> # long statement
>>>     car == 'moving':
>>> if signal == 'red' and car == 'moving':
>>>     car = 'stop'
>>> elif signal == 'green' and \
>>> elif signal == 'green' and car == 'stop':
>>>     car = 'stop'
>>>     car = 'moving'
```



续行

- 无需续行符可直接换行的两种情况：
 - 小括号、中括号、花括号的内部可以多行书写
 - 三引号包括下的字符串也可以跨行书写



```
>>> # triple quotes
>>> print("Hi everybody,
welcome to Python's MOOC course.
Here we can learn something about
Python. Good luck!")
```



Python 风格 (四)

44



```
>>> x = 'Today' ; y = 'is' ; z = 'Thursday' ; print(x, y, z)  
Today is Thursday
```



一行多语句



```
>>> x = 'Today'  
>>> y = 'is'  
>>> z = 'Thursday'  
>>> print(x, y, z)  
Today is Thursday
```

1.4

PYTHON 语法基础

1.4.1 变量

变量



```
>>> # variable
>>> PI = 3.14159
>>> pi = 'circumference ratio'
>>> print(PI)
3.14159
>>> print(pi)
circumference ratio
```

标识符

- 标识符是指Python语言中允许作为变量名或其他对象名称的有效符号
 - 首字符是字母或下划线
 - 其余可以是字母、下划线、数字
 - 大小写敏感(PI和pi是不同的标识符)



Source

```
>>> # Identifier
```

```
>>> PI = 3.14159
```

```
>>> pi = 'circumference ratio'
```

```
>>> print(PI)
```

```
3.14159
```

```
>>> print(pi)
```

```
circumference ratio
```


关键字

- 关键字是Python语言的关键组成部分，不可随便作为其他对象的标识符

- 在一门语言中关键字是基本固定的集合
- 在 IDE 中常以不同颜色字体出现

```
>>> import keyword  
>>> print(keyword.kwlist)
```


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

变量的使用

- 变量第一次赋值，同时获得类型和“值”

- Python是动态的强类型语言
- 不需要显式声明，根据“值”确定类型
- 以“引用”的方式实现赋值



 **S**ource

```
>>> # Identifier
>>> PI = 3.14159
>>> pi = 'one word'
>>> print(PI)
3.14159
>>> print(pi)
one word
```

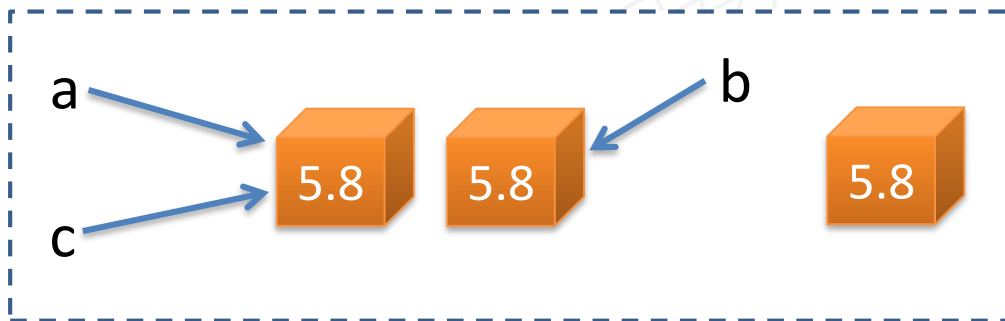


变量的管理

动态类型

- 引用计数

- $a = c$, 指向了相同的数据对象
- b 和 a 创建的是不同的5.8对象
- 单独 $\text{id}(5.8)$ 是创建的全新的对象



变量的管理

动态类型

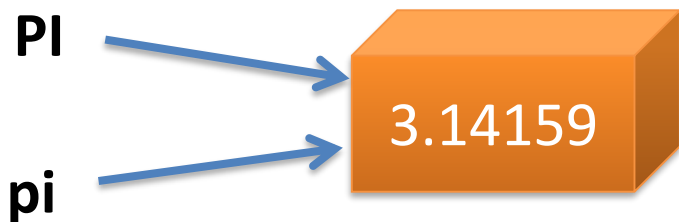
Source

```
>>> a = 5.8
>>> id(a)
1709988157600
>>> b = 5.8
>>> id(b)
1709988157648
```

Source

```
>>> id(5.8)
1709988157696
>>> c = a
>>> id(c)
1709988157600
>>> c = 3
```

变量的管理



`>>> p = 3`
`>>> q = 3`
`>>> p is q`
`True`

`>>> PI = 3.14159`
`>>> pi = 3.14159`
`>>> PI is pi`
`False`
`>>> pi = PI`
`>>> print(PI)`
`3.14159`
`>>> pi is PI`
`True`

图中的形式
用哪个语句
可以表示?

`is`运算符的
基础是`id()`
函数

1.4.2 表达式和语句

表达式

- 用运算符连接各种类型数据的式子就是表达式

算术运算符

乘方	**
正负号	+ -
乘除	* /
整除	//
取余	%
加减	+ -

位运算符

取反	~
与	&
或	
异或	^
左移	<<
右移	>>

关系运算符

小于	<
大于	>
小于等于	<=
大于等于	>=
等于	==
不等于	!=

逻辑运算符

非	not
与	and
或	or

- 运算符有优先级顺序
- 表达式必须有运算结果

Source

```
>>> # expression
>>> PI = 3.14159
>>> r = 2
>>> c_circ = 2 * PI * r
>>> print("The circle's circum is", c_circ)
```

- $2 * PI * r$ 是表达式
- 运算结果赋值给变量 `c_circ`

- 完整执行一个任务的一行逻辑代码
 - 赋值语句完成了赋值
 - `print()`函数调用语句完成了输出



```
>>> # statement  
>>> PI = 3.14159  
>>> print(PI)
```

语句和表达式

语句

完成一个任务

如，打印一份文件



表达式

任务中的一个具体组成部分

如，这份文件的
具体内容



赋值语句 增量赋值

增量赋值 操作符

+=	-=	*=	/=	%=	**=
<<=	>>=	&=	^=	=	

• $m /= 5$

即 $m = m / 5$

Source

```
>>> # Augmented assignment
```

```
>>> m = 18
```

```
>>> m /= 5
```

```
>>> m
```

3.6

赋值 链式赋值

- $b = a = a + 1$ 相当于
如下2条语句:

```
>>> a = a + 1
```

```
>>> b = a
```



```
>>> # Chained assignment
```

```
>>> a = 1
```

```
>>> b = a = a + 1
```

```
>>> b
```

```
2
```

```
>>> a
```

```
2
```

赋值 多重赋值

- 等号左右两边都以元组的方式出现
- 相当于:

>>> (PI, r) = (3.14159, 3)



```
>>> # Multiple assignment
```

```
>>> PI, r = 3.1415, 3
```

```
>>> PI
```

```
3.1415
```

```
>>> r
```

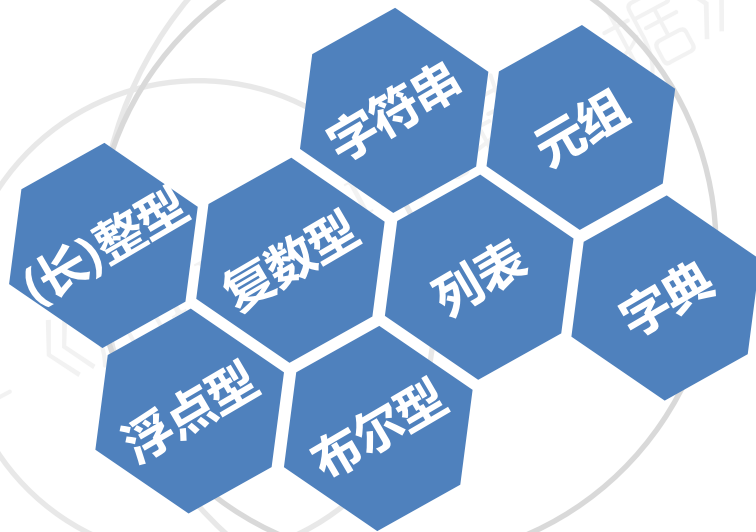
```
3
```

1.5

PYTHON 数据类型

Python数据类型

63



基本类型

01

• (长) 整型

02

• 布尔型

浮点型 •

03

04

• 复数型

- 整型和长整型并不严格区分
- Python 2支持整型值后加
“L” 即为长整型



```
>>> # integer  
>>> type(3)  
<class 'int'>
```

布尔型

- 整型的子类
- 仅有2个值: True、False
- 本质上是用整型的1、0分别存储的

Source

```
>>> # boolean
```

```
>>> x = True
```

```
>>> type(x)
```

```
<class 'bool'>
```

```
>>> int(x)
```

```
1
```

```
>>> y = False
```

```
>>> int(y)
```

```
0
```


浮点型


- 即数学中的实数
- 可以类似科学计数法表示



```
>>> # float
>>> 3.22
3.22
>>> 9.8e3
9800.0
>>> -4.78e-2
-0.0478
>>> type(-4.78e-2)
<class 'float'>
```

- $j=\sqrt{-1}$, 则 j 是虚数
- 实数+虚数 就是复数
- 虚数部分必须有 j


>>> *# complex*
>>> 2.4+5.6j
(2.4+5.6j)
>>> type(2.4+5.6j)
<class 'complex'>


>>> *# complex*
>>> 3j
3j
>>> type(3j)
<class 'complex'>
>>> 5+0j
(5+0j)
>>> type(5+0j)
<class 'complex'>