

Python基础知识（一）

陈赞

对外经济贸易大学金融学院
yunchen@uibe.edu.cn

2022年09月26日

目录










- Python简介
- Python的安装
- Python的简单使用
- 变量
- 简单数据类型：数字、空值、布尔型
- 列表和元组

一、Python简介

关于Python

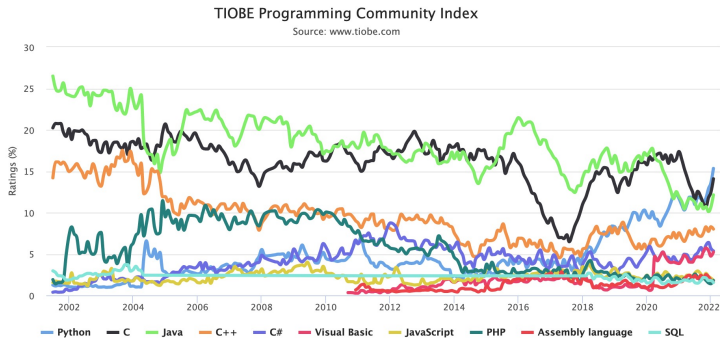
- Python是一门面向对象的高级编程语言，其功能强大，越来越多的人开始学习和使用Python来实现简单和复杂的数据挖掘与数据分析任务
- 创始人为荷兰人吉多·范罗苏姆（Guido van Rossum），曾经就职于Google、Dropbox等公司。
 - 1989年圣诞节，为了在阿姆斯特丹打发时间，决心开发一个新的脚本解释程序Python（大蟒蛇）！正式发布时间1991。
 - 取名：电视喜剧"Monty Python's Flying Circus"（蒙提·派森的飞行马戏团）。
- Python目前有两个版本，Python2（2000年发布）和Python3（2008年发布），最新版分别为2.7.18和3.10.2，Python 2.7在2020年正式停止官方支持

关于Python

Feb 2022	Feb 2021	Change	Programming Language		Ratings	Change
1	3	▲		Python	15.33%	+4.47%
2	1	▼		C	14.08%	-2.26%
3	2	▼		Java	12.13%	+0.84%
4	4			C++	8.01%	+1.13%
5	5			C#	5.37%	+0.93%
6	6			Visual Basic	5.23%	+0.90%
7	7			JavaScript	1.83%	-0.45%
8	8			PHP	1.79%	+0.04%
9	10	▲		Assembly language	1.60%	-0.06%

- TIOBE编程排行榜：<https://www.tiobe.com/tiobe-index/>

关于Python



- Python已经成为了一门简洁、优美、容易使用的编程语言，它的风靡程度超过了史上所有编程语言。
- 全世界的诸多大公司都在使用Python完成各种各样的任务。

Python语言的特点

- **免费+开源，简单易上手**: Python语言简约、语法清晰、易于读写、效率极高。同样一个指令，C语言要用1000行，Java要用100行，而Python可能只需20行就完成了
- **软件包、模块丰富**: Python语言的核心只包含数字、字符串、列表、字典、文件等常见类型和函数，Python标准库提供了文本处理、数据库接口、图形系统等额外的功能。**除了标准库外，还有许多其他高质量的库**
- **可移植性强**: 由于它的开源本质，Python已经被移植在许多平台上。Python程序无需修改就可在很多平台上面运行（Linux、Windows、Android、Macintosh，etc.）
- **运行速度较慢**，和C程序相比非常慢，因为Python是解释型语言
- **代码加密困难**，python的开源性使得Python语言**不能加密**

Python的一些应用案例

- 软件开发：游戏开发、创建Web应用程序、网站开发
- 数据获取（网络爬虫）、数据处理（大数据）、数值计算与数据可视化
- 量化投资、计量分析
- 图像处理、自然语言处理
- 人工智能、机器学习与深度学习

Python与其他软件/语言的比较

- **Excel**：大众性软件，数据展现功能强大，但数据处理能力弱，工作表数据存储有限；
- **Matlab**：工具箱富有，科学计算功能和图形展现功能强大，矩阵式存储和调用方式不适用于时序数据的整理；
- **Eviews**：计量模型实施方便，但数据处理能力弱；
- **Stata**：一种开源式统计分析软件，程序命令式操作，占磁盘空间小，简单易学，但对海量数据的处理能力弱；
- **SAS**：数据存储能力强大，常用于数据处理和统计分析领域，稳定性好。但内部处理机理与编程语言与众不同，学习有难度。

如何学好Python?

- 基础知识很重要，熟记最基本的用法，了解大概的功能
- 课后投入时间学习和训练，课堂上讲解的只是Python知识中“很小”的一块[**课堂时间有限**]
- 编程学习光看不行，一定要动手敲代码，养成写代码的好习惯
- 日常学习研究中尽量用上Python[**一段时间不用会忘记**]，遇到问题，学会利用搜索引擎！

二、Python的安装

Python环境

- Python官方网站: <https://www.python.org>
- **Anaconda**官方网站: <https://www.anaconda.com>
 - 一款开源的python发行版本, 包含了许多python相关的科学计算包
 - 集成开发环境IDE (Integrated Development Environment): Jupyter Notebook, Spyder, IPython etc.
 - IDE涵盖了代码创建、测试和debug等功能, 提高用户程序开发等效率
- **PyCharm**官方网站: <https://www.jetbrains.com/pycharm/>
- 其他IDE: Sublime Text, Wing, PyScripter, etc.

三、Python的简单使用

Python界面

- 安装好python后，在命令行输入：

```
python
```

进入Python解释器的页面，它可用于执行语句和查看结果。

- IPython是一款增强的Python交互式的解释器，其作用是提高编写、测试、调试Python代码的速度。命令行下输入：

```
ipython
```

即可进入ipython解释器。和Python解释器相比，它具有代码自动补全、自动缩进等功能

Python界面

- **Jupyter Notebook**：是一款使用方便，兼容多种语言的交互式编程平台。利用它，我们可将代码、计算结果、程序说明及笔记整合在同一个网页页面，并及时更新互动。命令行下输入：

```
jupyter notebook
```

- **Anaconda** 是一个很好用的Python集成开发环境，它集成了很多科学计算需要使用的python第三方工具包。Anaconda默认使用的编辑器是spyder，可以在命令行下输入：

```
spyder
```

- 除了使用命令行进入这些Python环境，也可以采用点按钮的方式。（课堂演示）

编写简单的Python程序：print函数

- **print()**: 用于在python控制台打印内容，是python里很常见的一个操作。基本格式：

```
print()  
print('something')
```

- 注意：这里要用英文字符的括号（编程过程中注意中英文输入法）。如果是字符，要带引号(' ', " ", "'' ''"), 数字就不用。

```
>>> print("Hello, Python!")  
Hello, Python!  
>>> print(100)  
100
```

- 运行与保存、执行.py文件（课堂演示）

编写简单的Python程序：print函数

- `print()` 函数更详细的用法：<https://docs.python.org/3/library/functions.html#print>
- 可同时打印多个表达式，条件是用逗号分隔它们

```
print('a', 'b', 'c', 3)
```

- 可自定义分隔符

```
print("I", "love", "Python", sep="_")
```

- 可自定义结束字符串，以替换默认的换行符

```
print('Hello,', end='')  
print('Python!')
```

编写简单的Python程序：input函数

- **input()**：用户输入函数（python内置函数），接受一个标准输入数据（键盘输入）。
函数语法：

```
input([prompt])
```

prompt表示提示信息，通常为字符串，[]表示参数可选，即什么也不提供

- input()函数以字符串类型返回结果

```
>>> input()
I love python
'I love python'
>>> input("Please Enter Your Name:")
Please Enter Your Name:Yun
'Yun'
>>> a = input("How old are you:")
How old are you:28
>>> a
'28'
```

四、变量

变量(Variable)

- 变量是可以赋值的标签，变量**指向**特定的值
- 每个变量在使用前都必须赋值，变量赋值以后该变量才会被创建

```
>>> a = 3
>>> a
3
>>> b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
```

变量的命名和使用

- 变量的命名规则：
 - 变量名只能包含字母、数字和下划线。变量名能以字母或下划线开头，但不能以数字开头。如：`message_1` (ok)，`1_message` (wrong)。
 - 变量名不能包含空格，但可使用下划线来分隔其中的单词。如：`my_name` (ok)，`my name` (wrong)
 - 不要将Python关键字和函数名用作变量名，即不要使用Python保留用于特殊用途的单词。如：`print`, `True`, `False`, `for`, `if`, `class`, `else`, `in`, `break`, `not`, `pass`, `import`, etc.
 - 变量名应既简短又具有描述性。如：`my_name`比`m_n`好
 - 大小写敏感
- 注意：尽量使用小写的变量名，大写字母在变量名中有特殊含义

课堂练习- 变量与简单py程序

- 以下关于变量的命名说法错误的是：

- A. 3abc是合法的变量名
- B. ABC和abc代表同一个变量
- C. 不建议用if、print作为变量名
- D. 学生成绩单变量s_s是符合规则的

- 编程练习：

- 编写一个独立的程序，其功能是将姓名（如小王）复制给变量`name`，并打印“你好，小王！”
- 将其存储为`simple_greeting.py`
- 分别在spyder和anaconda prompt中运行该程序

五、简单数据类型：数字、空值、布尔型

数字类型

Python语言提供整数、浮点数、复数三种数字类型。

- 整数integer, int():

```
>>> type(3)
<class 'int'>
>>> int(3.6)
3
```

- 浮点数floating-point number, float():

```
>>> type(3.0)
<class 'float'>
>>> float(3)
3.0
>>> z = 1e3
>>> type(z)
<class 'float'>
>>> z
1000.0
```

数字类型

- 复数complex number, complex():

```
>>> z = 3 + 4j
>>> type(z)
<class 'complex'>
>>> z.real
3.0
>>> z.imag
4.0
>>> complex(4.3)
(4.3+0j)
```

- 尝试int、float、complex对不同字符串的转换

```
>>> complex("3.1")
(3.1+0j)
>>> float("3.1")
3.1
>>> int("3.1")
```

数字类型

- 三种类型存在一种逐渐"扩展"或"变宽"的关系：整数 < 浮点数 < 复数。它们之间可进行混合运算，生成结果为"最宽"类型

```
>>> a = 1 + 2.0
>>> type(a)
<class 'float'>
>>> b = 3 + 4j
>>> c = a + b
>>> type(c)
<class 'complex'>
```

- 运算符：+、-、*、/、//、%、**
- 运算符优先级，圆括号

空值None

- None表示没有值，也就是空值。常用来填补数据缺失
- None是值，Nonetype是类型

```
>>> a = None
>>> type(a)
<class 'NoneType'>
```

- None是NoneType数据类型的唯一值，也就是说我们不能再创建其它NoneType 类型的变量，但是可以将None 赋值给任何变量

布尔型 Boolean（简称 bool）

- 布尔型只有两个值：True，False，它通常在关系比较中用到（后续在语句章节我们会详细介绍）
- 在数组运算中，Boolean值的True和False分别对应于1和0。

```
>>> type(True)
<class 'bool'>
>>> True * 1
1
>>> False * 1
0
```

- 对于值为0的任何数字（浮点数、复数）、空集（空字符串、空集合、空列表、空元组和空字典）和None的布尔值都是False。

```
>>> bool(0)
False
>>> bool(3)
True
```

math模块（提前了解下）

- math模块包含了许多数学运算的函数和数学常量

```
import math
dir(math)
help(math)
help(math)
```

- 数学常量：math.pi, math.e, math.inf, math.nan（非浮点数标记，Not a Number）
- 数学函数：fabs、ceil、floor、factorial、pow、sqrt、log、exp、isnan，使用方法math.函数名(参数)

课堂练习- print函数与字符串

- 计算二元一次方程 $ax^2 + bx + c = 0$ 的解，代入具体的值查看结果。要求程序能输出如下结果：

$1 * (x ** 2) + 4 * x + 3 = 0$ 的根分别为-1.0 -3.0

- 对于一个五位整数，提取它的个、十、百、千、万位数字。要求程序能输出如下结果：

12306 的个十百千万位数字分别为：6 0 3 2 1

六、列表与元组

列表

- **数据结构**是以某种方式（如通过编号）组合起来的数据元素（如数、字符乃至其他数据结构）集合
- Python中，最基本的数据结构为序列（sequence）。序列中的每个元素都有编号，即其位置或索引，其中第一个元素的索引为0，第二个元素的索引为1，依此类推。
- Python内置了多种序列：列表、元组、字符串（'hello'）
- 序列的通用操作包括：索引(indexing)、切片(slicing)、序列的加法与乘法、初始化、成员资格、长度、最小值、最大值等基本操作
- 下面我们以列表为例，介绍序列等通用操作，这些操作对元组和字符串等结构都适用

列表

- 在Python中，列表是一个有序的序列。列表用一对[]生成，中间的元素用,隔开，其中的元素不需要是同一类型，同时列表的长度也不固定。

```
>>> a = [1, 2.0, 'hello']
>>> a
[1, 2.0, 'hello']
>>> type(a)
<class 'list'>
```

- 空列表可以用[] 或者list() 生成：

```
>>> empty_list1 = []
>>> empty_list2 = list()
>>> empty_list1
[]
>>> empty_list2
[]
```

列表（序列）的常规操作

- 用len()查看列表长度，min()查看最小值，max()查看最大值

```
>>> numbers = [1, 2, 3, 4, 5]
>>> len(numbers)
5
>>> max(numbers)
5
>>> min(numbers)
1
```

- 列表加法，相当于将两个列表按顺序连接；列表与整数相乘，相当于将列表重复相加

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
>>> a * 2
[1, 2, 3, 1, 2, 3]
```

列表（序列）的常规操作：索引和切片

- 索引(indexing): 访问序列的元素。序列中的所有元素都有编号，从0开始递增。使用负数索引时，Python将从右（最后一个元素）往左数，-1是最后一个元素的位置。

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8]
>>> a[1]
2
>>> a[-1]
8
```

- 通过索引直接对列表进行修改

```
>>> a = [1, 2, 3, 4]
>>> a[1] = 999
>>> a
[1, 999, 3, 4]
```

列表（序列）的常规操作：索引和切片

- 切片(slicing)：访问特定范围内的元素，语法seq[start:end:step]。

其中第一个索引指定的元素包含在切片内，第二个索引指定的元素不包含在切片内，step表示切片的步长，默认为1。

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 8]
>>> a[:3]
[1, 2, 3]
>>> a[2:]
[3, 4, 5, 6, 7, 8]
>>> a[2:-1]
[3, 4, 5, 6, 7]
>>> a[::2]
[1, 3, 5, 7]
>>> a[::-1]
[8, 7, 6, 5, 4, 3, 2, 1]
```

不改变列表的方法

- 方法是与对象（列表、数、字符串等）联系紧密的函数，其调用方式：

`object.method(arguments)`

方法调用与函数调用很像，只是在方法名前加上了对象和句点

- 关于方法，注意两点：（1）是否修改对象的内容；（2）是否有返回值

不改变列表的方法

- `l.count(ob)`: 返回列表中元素`ob`出现的次数。

```
>>> a = [11, 12, 13, 12, 11]
>>> a.count(11)
2
```

- `l.index(ob)`: 返回列表中元素`ob`第一次出现的索引位置，如果`ob`不在`l`中会报错。

```
>>> a.index(12)
1
>>> a
[11, 12, 13, 12, 11]
```

改变列表的方法

- `l.append(ob)`: 将元素`ob`添加到列表`l`的最后。

```
>>> a = [10, 11, 12]
>>> a.append(11)
>>> a
[10, 11, 12, 11]
>>> a.extend([11, 12])
>>> a
[10, 11, 12, 11, 11, 12]
```

- `l.extend(lst)`: 将序列`lst`的元素依次添加到列表`l`的最后, 作用相当于`l += lst`。

```
>>> a = [10, 11, 12, 11]
>>> a.extend([1, 2])
>>> a
[10, 11, 12, 11, 1, 2]
```

改变列表的方法

- `l.insert(idx, ob)` : 在索引`idx`处插入`ob`，之后的元素依次后移。

```
>>> a = [10, 11, 12, 13, 11]
>>> a.insert(3, 'a')
>>> a
[10, 11, 12, 'a', 13, 11]
```

- `l.remove(ob)` : 将列表中第一个出现的`ob`删除，如果`ob`不在`l`中会报错。

```
>>> a = [10, 11, 12, 13, 11]
>>> a.remove(11)
>>> a
[10, 12, 13, 11]
>>> a.remove(9)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: list.remove(x): x not in list
```

改变列表的方法

- `l.pop(idx)`: 会将索引`idx`处的元素删除, 并返回这个元素。

```
>>> a = [10, 11, 12, 13, 11]
>>> a.pop(2)
12
>>> a
[10, 11, 13, 11]
```

- `l.sort()`: 会将列表中的元素按照一定的规则排序

```
>>> a = [10, 1, 11, 13, 11, 2]
>>> a.sort()
>>> a
[1, 2, 10, 11, 11, 13]
```

- `l.reverse()`: 会将列表中的元素从后向前列。

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> a.reverse()
>>> a
[6, 5, 4, 3, 2, 1]
>>> a[::-1]
```

使用列表*

- 切片赋值

```
>>> a = [1, 2, 3, 4]
>>> a[:2] = [9, 9]
>>> a
[9, 9, 3, 4]
>>> a[:2] = []
>>> a
[3, 4]
```

- Python提供了删除列表中元素的方法del

```
>>> a = [1002, 'a', 'b', 'c']
>>> del a[0]
>>> a
['a', 'b', 'c']
>>> del a[1:]
>>> a
['a']
```

使用列表*

- 如何复制列表？ 看一个例子：

```
>>> a = [1, 2, 3]
>>> b = a
>>> a[0] = 3
>>> a
[3, 2, 3]
>>> b
[3, 2, 3]
```

使用列表*

- 复制列表可以创建一个包含整个列表的切片：

```
>>> digits
[0, 1, 2, 3]
>>> digits_cp1 = digits
>>> digits_cp2 = digits[:]
>>> digits.append(4)
>>> digits
[0, 1, 2, 3, 4]
>>> digits_cp1
[0, 1, 2, 3, 4]
>>> digits_cp2
[0, 1, 2, 3]
```

- 搜索下关于赋值、浅复制、深度复制的内容

使用列表*

- 赋值、copy、deepcopy

```
>>> a = [1, 2, [3, 4]]
>>> b = a
>>> c = a.copy()
>>> import copy
>>> d = copy.deepcopy(a)
>>> a[0] = 3
>>> a[2].append(5)
>>> a
[3, 2, [3, 4, 5]]
>>> b
[3, 2, [3, 4, 5]]
>>> c
[1, 2, [3, 4, 5]]
>>> d
[1, 2, [3, 4]]
```

元组

- 元组是Python中一类不可修改值的有序数据
- 元组用圆括号表示，中间的元素用逗号隔开，元素可以是不同类型的数据

```
>>> mi12 = ("164.3mm", "74.6mm")
>>> mi12[0]
'164.3mm'
>>> mi12[0] = 3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

- 功能：元组不能更改，可以防止错误赋值所导致重要对象的改变
- 常用情况：表达固定数据项、函数多返回值、多变量同步赋值等

元组

- 定义空元组和一个元素的元组：

```
>>> a = ()
>>> a
()
>>> type(a)
<class 'tuple'>
>>> b = tuple()
>>> b
()
>>> type(b)
<class 'tuple'>
>>> c = (1,)
>>> c
(1,)
>>> type(c)
<class 'tuple'>
>>> d = (1)
>>> d
1
>>> type(d)
<class 'int'>
```

元组

- 元组的常用方法，用法与列表一样，并且这些方法不能改变元组的值

```
>>> a = (10, 11, 12, 13, 14)
>>> a.count(10)
1
>>> a.index(12)
2
>>> len(a)
5
>>> max(a)
14
>>> min(a)
10
```

- 元组的索引与切片与列表一样：

```
>>> t = (10, 11, 12, 13, 14)
>>> t[0]
10
>>> t[1:3]
(11, 12)
```

课堂练习- 列表元组的使用

- 使用list函数将你的名字（姓名）转换成列表，统计其中字母'h'出现的次数
- 索引列表[1, 3, [4, 5], (1, 3)]中所有的3
- 对于一个列表a，它存储了1-10000的数字，但这些数字在列表中的位置不确定，如何索引得到999