Python 基础

续本达

清华大学 工程物理系

2025-06-24 清华

预备

- 查看是否已经安装 Python

```
python3
```

```
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

  - 退出输入 `exit()` 或者按 `Ctrl-d` 。
  - `Ctrl-d` 代表文件（标准输入）的终止。

- 如果没有 Python（可能性不大），或想升级 Python

```
apt install python3 # Debian @ WSL
emerge -vt python:3 # Gentoo Prefix @ macOS
```

黑客审美

- 当代文明的两大支柱是 实验 和 逻辑 。
- 四个原则：
  1. 复现 - 否则成伪科学
  2. 透明 - 否则玄学黑箱
  3. 一次 - 否则到处是坑
  4. 最佳工具 - 否则效率低下
- 推论：
  1. 兼容比性能优先

Premature optimization is the root of all evil. – Tony Hoare, Donald Knuth

  2. 人类时间比机器时间宝贵
  3. 使用工具进行版本控制

黑客审美

- 当代文明的两大支柱是 实验 和 逻辑 。
- 四个原则：
  1. 复现 - 否则成伪科学
  2. 透明 - 否则玄学黑箱
  3. 一次 - 否则到处是坑
  4. 最佳工具 - 否则效率低下
- 推论：
  1. 兼容比性能优先

Premature optimization is the root of all evil. – Tony Hoare, Donald Knuth

  2. 人类时间比机器时间宝贵
  3. 使用工具进行版本控制

最佳工具

- GNU 环境、POSIX 标准：用于促进程序有几十年跨度的兼容性。

Git：当代的版本控制

- 快照与差分
- 三个状态：
  - 已提交 committed（未改动 unmodified）
  - 已修改 modified
  - 待提交 staged

程序编辑器三大流派

请完成网络学堂的“课程问卷”，计划帮助大家掌握一款编辑器。

- Emacs
- Vi
- Visual Studio Code

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。
  https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。
  https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。
  https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。

https://www.python.org/

为什么用 Python

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读。
  - 书写效率高，快速写出不错的程序。
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
  - 易于与已有工具整合。
  - 促进团队分工，协作。
  - 大大丰富了 Python 生态系统的功能，进一步优化程序运行效率。
  - 符合 最佳工具 原则。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
- 在科学计算领域得到广泛欢迎和采用。
  `https://www.python.org/`

特点和用途

- Python 是一门“解释型语言”，相对于“编译型语言”更易调试。
- Python 的语法风格简明，即使对外行也易读，大大降低了程序设计的门槛
- Python 可以直接调用 Fortran, C/C++, R 等语言库，因此也叫“胶水”语言，即把不同的程序粘合在一起。
- Python 是一个通用语言，不仅在科学研究，在生活中的方方面面都会有用。
  - 操作系统生成器和管理器 (Gentoo Portage)
  - 网站 (Django)

参考资料

- Allen Downey, Think Python 2e
  简明通俗的入门书
- `http://py4e.com/`
  Python for everybody，全球知名的 Python 在线教程，新手友好。
- Learn X in Y minutes
  `https://learnxinyminutes.com/docs/python3/`
  已经掌握若干门语言的同学，可以通过此提纲快速入门

Python 环境

安装了 Python 之后，在命令行界面可以直接进入 Python 的交互模式：

```
$ python3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

IPython（Jupyter 前身）增强的额外交互功能环境

```
# apt install ipython3
$ ipython3
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]:
```

查看 Python 的基本信息

```
import sys
print(sys.version)
```

```
3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0]
```

```
print("\n".join(sys.path))
```

```
/usr/lib/python311.zip
/usr/lib/python3.11
/usr/lib/python3.11/lib-dynload
/usr/local/lib/python3.11/dist-packages
/usr/lib/python3/dist-packages
/usr/lib/python3.11/dist-packages
```

Python 之禅

```
import this
```

```
The Zen of Python, by Tim Peters

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!
```

算术基本运算

- + 加，- 减，* 乘，/ 除，// 整除，% 取余，** 乘方
- 其它运算由函数来定义

```
2+1
```

```
3
```

```
2*7, 2**7, 3/2, 3//2, 3%2
```

```
14  128  1.5  1  1
```

```
import math
math.factorial(10) # 10!
```

```
3628800
```

整数是高精度的

- 计算机上标准整数运算有界；
- 高精度整数并没有硬件的底层支持，是 Python 的软件实现，以一些性能损失为代价给予用户便利。

```
math.factorial(66)
```

```
544344939077443064003729240247842752644293064388798874532860126869671081148416000000
```

```
2**100
```

```
1267650600228229401496703205376
```

整除的基本约定

负数整除：向小的方向截断。

```
5 // 3, -5 // 3, 5.0 // 3.0, -5.0 // 3.0
```

```
1   -2   1.0   -2.0
```

```
5 % 3, -5 % 3, 5.0 % 3.0, -5.0 % 3.0
```

```
2   1   2.0   1.0
```

```
(-5 // 3) * 3 + (-5 % 3) == -5
```

```
True
```

布尔运算：真与假

Python 的设计目标是“符合直觉”

```
not True, not False
```

False True

```
True and False, False or True
```

False True

```
True + True, True * False # True 实际上是 1, False 实际上是 0
```

2 0

```
True * 8, False - 5 # 只为了说明 True 和 False 的内部表示，不要这样写代码
```

8 -5

条件判断

- 等号写两次用于判断，写一次用于赋值。

```
1 == 1, 2 == 1
```

True False

```
1 != 1, 2 != 1
```

False True

```
1 < 10, 1 > 10, 2 <= 2, 2 >= 2
```

True False True True

数据类型

int 指整型，没有上限；float 是浮点型，一般为双精度；str 是字符串

```
type(1)
```

```
<class 'int'>
```

```
type(1.5)
```

```
<class 'float'>
```

```
type('Hello')
```

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

```
print(type("a"), type("你好")) # 单个字符和汉字都是字符串
```

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

字符串

与高精度整数一样，字符串也没有硬件的对应，是 Python 的软件实现。这极大方便了使用 Python 进行文本处理。

```
"今天" + "要下雨"
```

今天要下雨

```
"1" + "2"
```

12

标准输入输出

- 标准输出默认与屏幕连接，`print()` 默认向标准输出写
- 标准输入默认与键盘连接，`input()` 默认从标准输入读

```
q = input() # 下面由现场输入
print(q)

frog
```

Python 的注释

- 注释使用 '#' 号引出，多行注释则多用几次 '#' 号

```
# 在这个程序中，我们将使用计算球谐函数对任意
# 球面上的连续函数进行拟合
```

正式入门 Python：输出 Hello World!

```
print("Hello World!")
```

```
Hello World!
```

创建脚本文件

世界上本没有脚本，把输入的命令记录下来就成了脚本。Python 脚本一般以 `.py` 结尾。

使用编辑器创建 `hello.py`，写入以下内容，第一行是标注脚本由什么来解释。

```
#!/usr/bin/env python3

print("Hello World!")
```

给予脚本可执行权限并执行

```
chmod +x hello.py
./hello.py
```

变量创建与使用

- Python 是“弱类型语言”，创建变量可不指定类型。

```
message = "This is a new era. 新时代"
print(message)
```

```
This is a new era. 新时代
```

- 变量在使用中可以改换类型

```
message = 1
type(message)
```

```
<class 'int'>
```

字符串函数

字符串的操作相对于硬件调用，是一项高级功能。Python 有强大的字符串处理工具。

```
len('123456'), len('654321') # 字符串长度
```

6 6

```
s = "我正在上课。"
s[0], s[2:4], s[-1] # 字符串可以取子串
```

我 在上 。

```
"啊" * 10 + "哈" * 20
```

啊啊啊啊啊啊啊啊啊哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈哈

字符串与变量的联合操作

```
"{} 乘以 {} 等于 {}".format(3, 5, 3*5) # 把其它类型的值嵌入字符串
```

```
3 乘以 5 等于 15
```

f-string 用来把变量值嵌入字符串

```
b=50
f"b 的取值是 {b}"
```

```
b 的取值是 50
```

None 值

- None 是一个特殊的值，代表空集、无、无法表达或者非法的结果

```
None
```

```
None
```

```
x = None # None 可以被赋值
1 is None, x is None # 可以被判断
```

False True

- 使用场景：判断某个结果是否是 None 有助于我们了解操作是否成功

```
bool(None) # None 也可以当作“假”被判断
```

```
False
```

不要害怕英文提示

- 人生第一次遇到英语不是“屠龙之技”的场景，值得庆贺。
- 下一个场景：阅读英语科技文献。
- 再下一个场景：与国际同行讨论争吵。

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-f40f0b3453ee> in <module>
----> 1 message + 1

TypeError: can only concatenate str (not "int") to str
```

程序结构

head to `Python-Constructs.slides`.

- 顺序结构
- 选择结构
- 循环结构