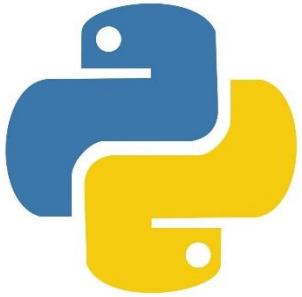
Python语言编程及实践



人生苦短，我用Python

李伟斌

liweibin@lzu.edu.cn

兰州大学，草种创新与草地农业生态系统全国重点实验室

第 2 章 变量和简单的数据类型

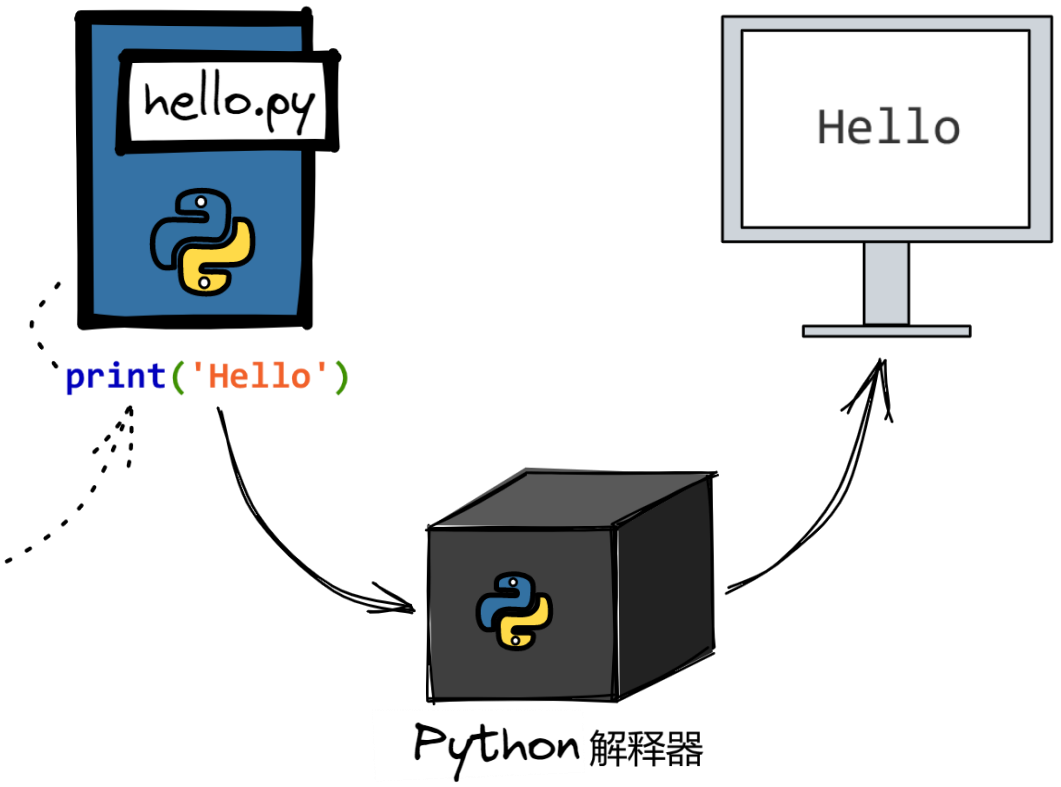
- 2.1 运行解释
- 2.2 变量
- 2.3 字符串
- 2.4 数
- 2.5 注释
- 2.6 Python 之禅
- 2.7 小结

2.1 运行解释

- 文件扩展名：结尾的 `.py` 用于指出文件内容是 Python 代码
- Python 解释器：读取整个文件，确定其中每一行的含义并执行

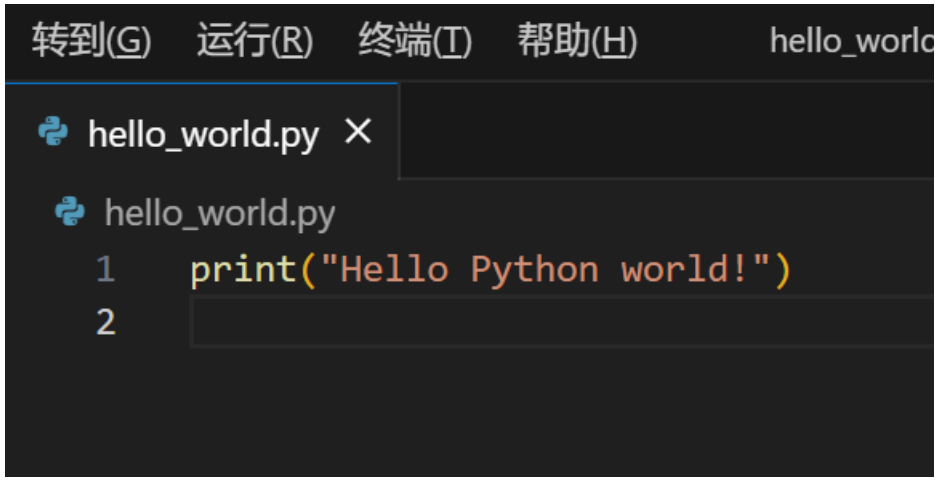
例如，当解释器看到 `print`，就会将括号中的内容打印到屏幕上。

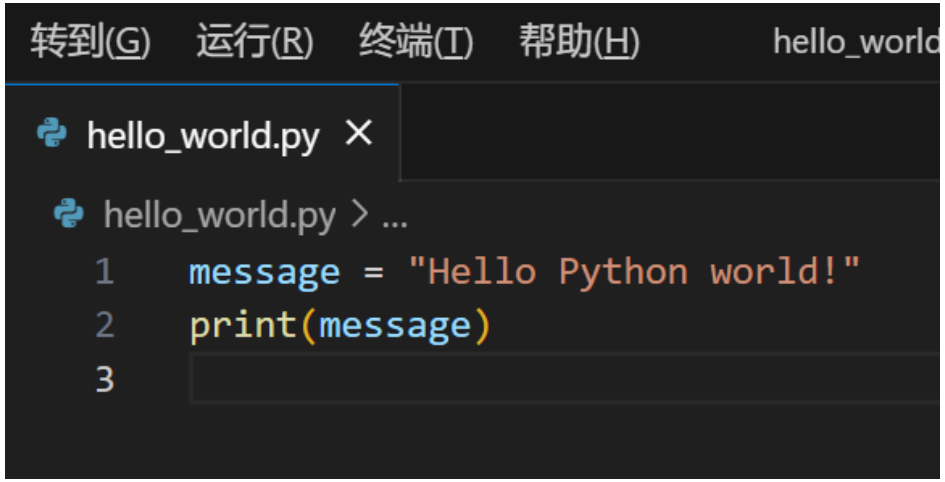
- 语法高亮：用不同的颜色，区分出程序代码中的不同部分



2.2 变量

修改我们在上一章中写的代码：





修改后的代码中：
添加了一个名为 `message` 的**变量（variable），**指向的值为文本 `"Hello Python world!"`

每个变量都指向一个**值（value），**这个值与变量相关联

2.2 变量

输出仍然是一样的：

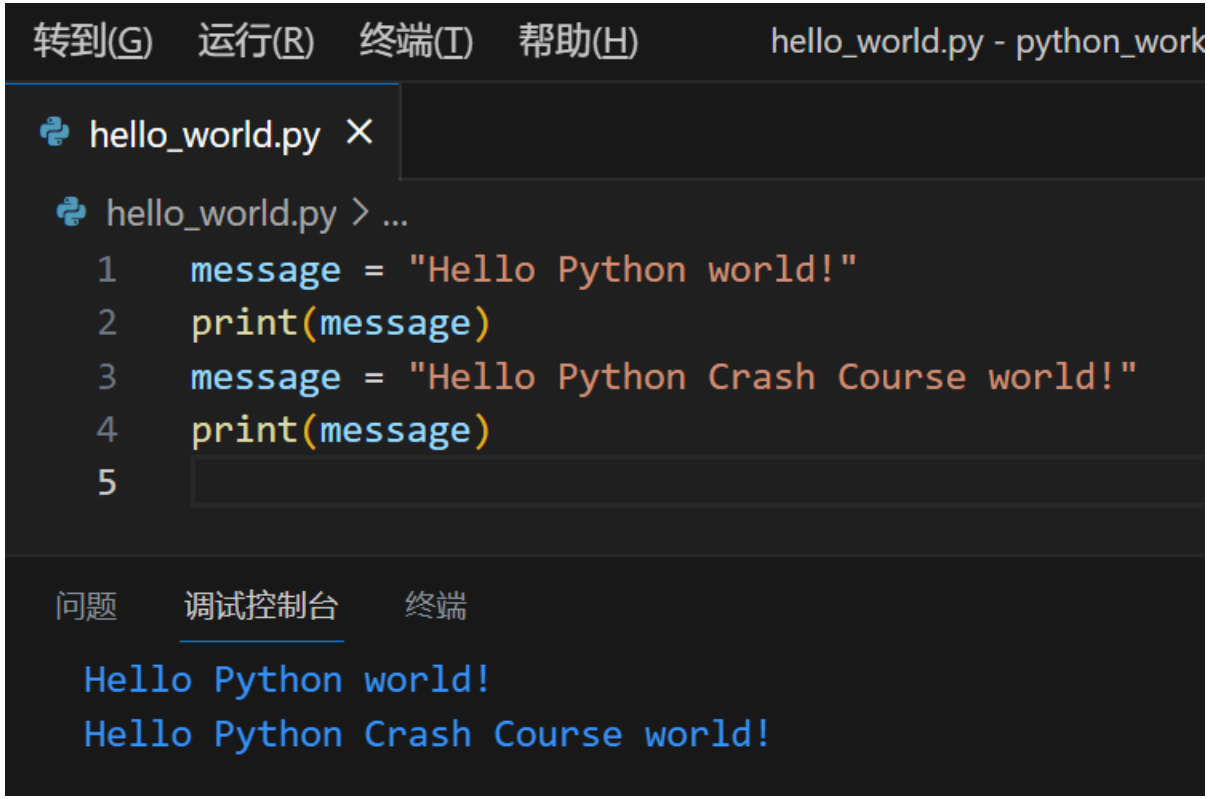
2.2 变量

再添加两行代码：

```
message = "Hello Python
Crash Course world!"
print(message)
```

运行程序

可以看到两行**不一样**的输出



我们可以在程序中随时修改变量的值，Python 将会始终记录最新值

2.2.1 变量：命名和使用

- 只能**包含字母、数字和下划线**，不能以数字开头

```
message
message1
message_1
```

```
message!
1message
```

2.2.1 变量：命名和使用

- 只能**包含字母、数字和下划线**，不能以数字开头
- 不能包含空格，**用下划线来分隔单词**

```
message_1
```

```
message 1
```

2.2.1 变量：命名和使用

- 只能**包含字母、数字和下划线**，不能以数字开头
- 不能包含空格，**用下划线来分隔单词**
- 变量名应使用**既简短又具有描述性的英文单词**，不建议用拼音

`message` 

`msg` 

`xiaoxi` 

`m` 

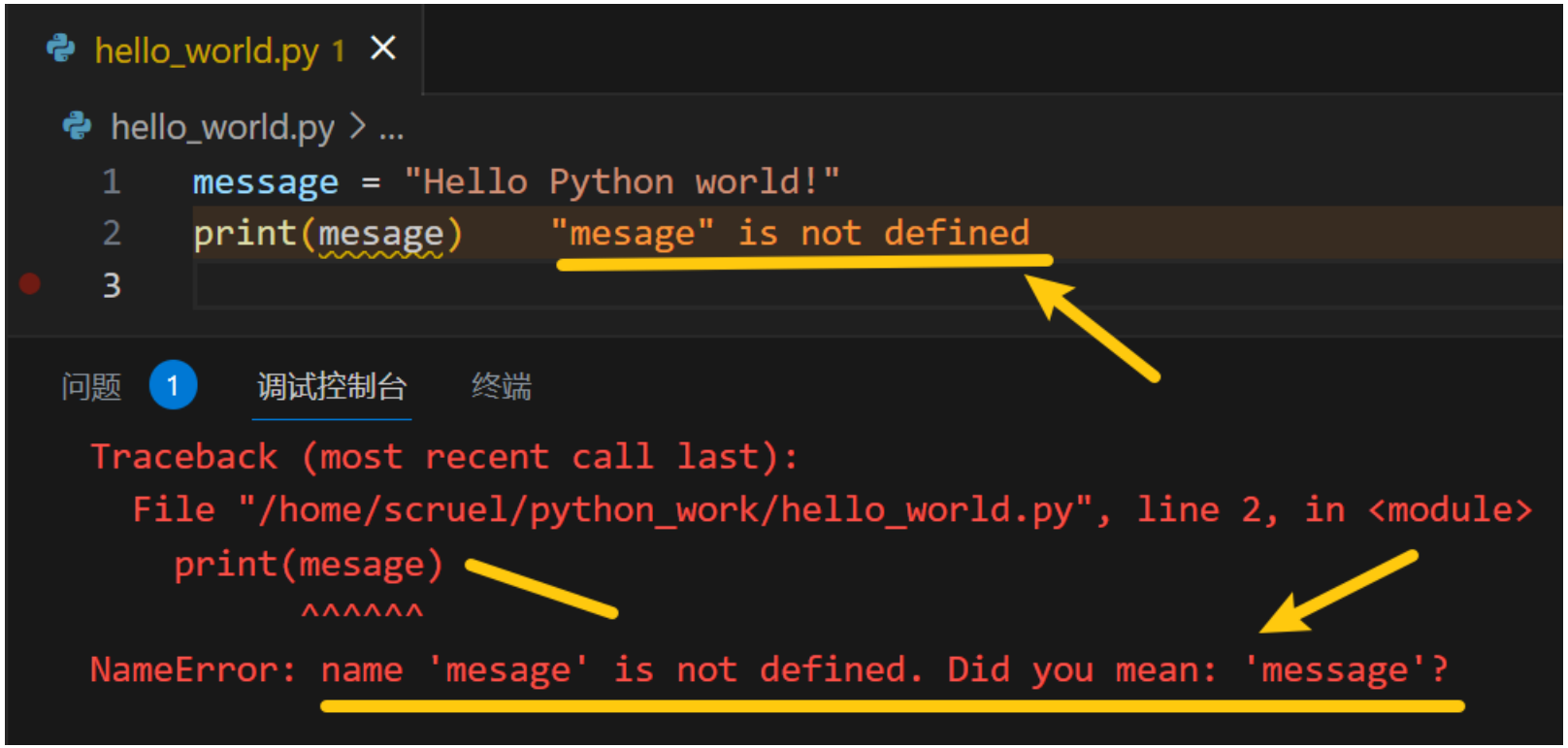
2.2.1 变量：命名和使用

- 只能**包含字母、数字和下划线**，不能以数字开头
- 不能包含空格，**用下划线来分隔单词**
- 变量名应使用**既简短又具有描述性的英文单词**，不建议用拼音
- 变量名区分大小写，命名普通的变量一般使用小写
- **慎用**小写字母 l 和大写字母 O，因为它们容易被人错看成数字
- Python 的关键字和函数名不能用作变量名（见附录 A.4）

*要养成**良好的命名习惯***

2.2.2 变量：避免命名错误

- 尽量避免拼写错误，容易产生错误，可以利用工具检查
- 利用 VS Code 的提示和运行后展示的 Traceback 信息排错

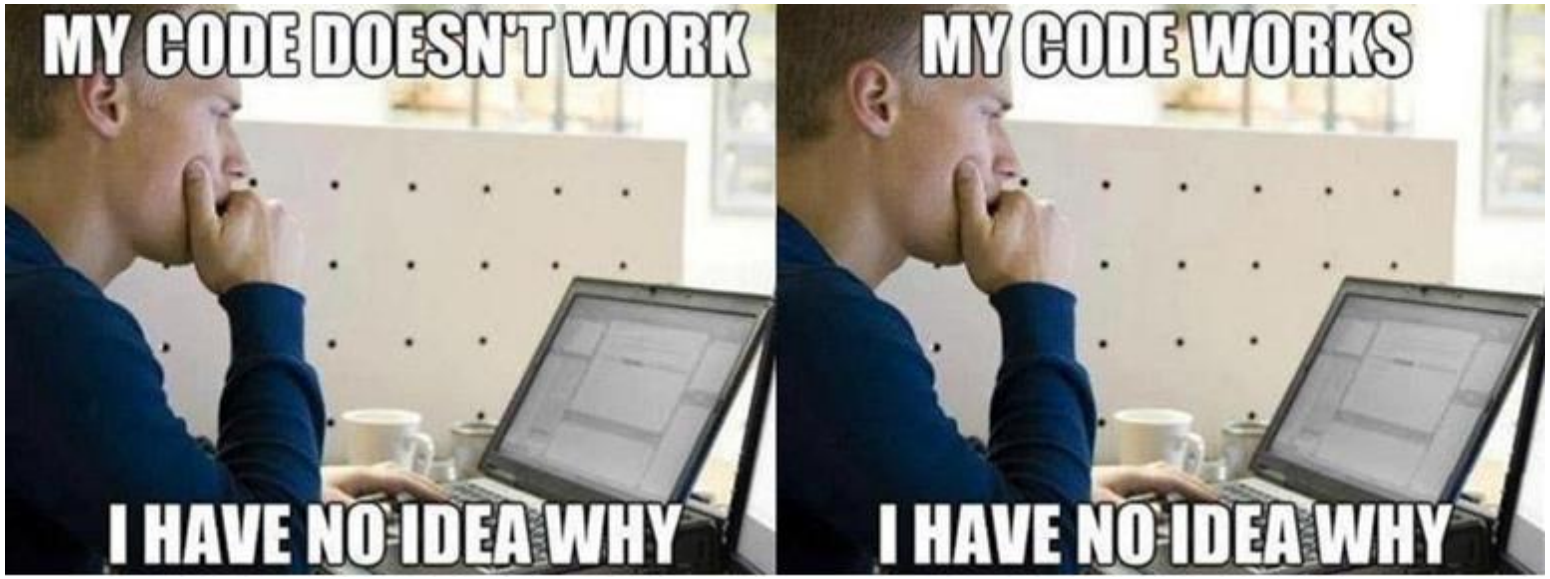


名称错误 (NameError)

2.2.2 变量：避免命名错误

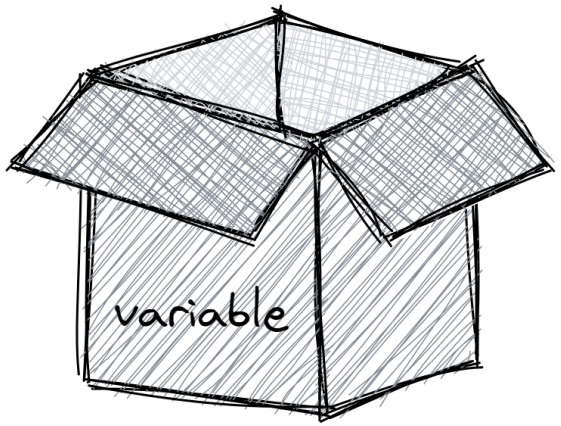
Be Nice:

不要害怕犯错，更不要因为错误显而易见，就对他人冷嘲热讽~



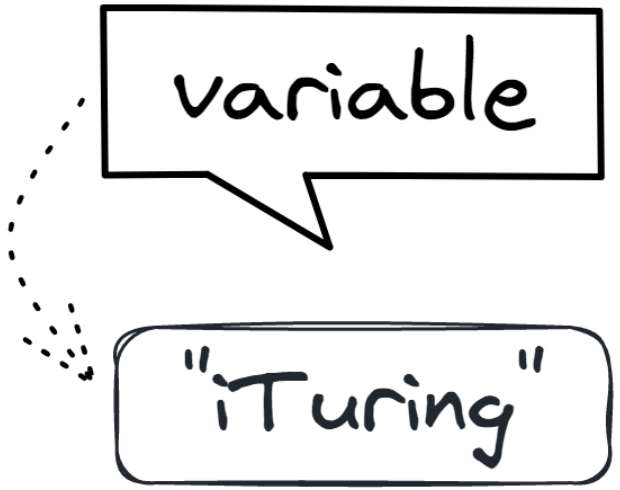
2.2.3 变量是标签

变量是用于存储值的盒子?



好像不太恰当

变量是可以被赋值的标签
(变量指向特定的值)



这个定义好得多!

2.3 字符串

字符串（string）就是一系列字符

- 在 Python 中，用英文引号引起的都是字符串
- 引号可以是双引号

```
"Hello Python world!"
```

2.3 字符串

字符串（string）就是一系列字符

- 在 Python 中，用英文引号引起的都是字符串
- 引号可以是双引号，也可以是单引号

```
'Hello Python world!'
```

2.3 字符串

字符串（string）就是一系列字符

- 在 Python 中，用英文引号引起的都是字符串
- 引号可以是双引号，也可以是单引号
- 还有一种特殊的写法，使用三个双引号

```
"""Hello Python world!"""
```

2.3 字符串

字符串（string）就是一系列字符

- 在 Python 中，用英文引号引起的都是字符串
- 引号可以是双引号，也可以是单引号
- 还有一种特殊的写法，使用三个单引号或三个双引号
- 三个引号可以创建跨行字符串

```
"""Hello Python world!
Hello iTuring!"""
```

2.3 字符串

字符串（string）就是一系列字符

- 在 Python 中，用英文引号引起的都是字符串
- 引号可以是双引号，也可以是单引号
- 还有一种特殊的写法，使用三个单引号或三个双引号
- 三个引号可以创建跨行字符串，一个引号的写法则不能跨行

```
"Hello Python world!
Hello iTuring!"
```

2.3 字符串

Python 提供了一些途径，帮助我们修改字符串，比如大小写：

2.3.1 字符串：大小写修改

方法/函数（method/function）：Python 可对数据执行的操作。

```
name = "ada lovelace"
print(name.title())
```

name 后面的**点号（dot）**让 Python 对 name 变量执行指定的操作。

2.3.1 字符串：大小写修改

方法/函数（method/function）：Python 可对数据执行的操作。

```
name = "ada lovelace"
print( name.title ( ) )
```

都是英文括号哦！

name 后面的**点号（dot）**让 Python 对 name 变量执行指定的操作。

每个方法后面都跟着一对括号

2.3.1 字符串：大小写修改

方法/函数（method/function）：Python 可对数据执行的操作。

```
name = "ada lovelace"
print(name.title( ))
```

`title` 方法所需的额外信息

name 后面的**点号（dot）**让 Python 对 name 变量执行指定的操作。

每个方法后面都跟着一对括号

括号内包含方法需要的额外信息

2.3.1 字符串：大小写修改

方法/函数（method/function）：Python 可对数据执行的操作。

```
name = "ada lovelace"
print(name.title( ))
```

print 函数所需的额外信息

name 后面的**点号（dot）**让 Python 对 name 变量执行指定的操作。

每个方法后面都跟着一对括号

括号内包含方法需要的额外信息

2.3.1 字符串：大小写修改

方法/函数（method/function）：

Python 可对数据执行的操作。

```
name = "ada lovelace"
print(name.title())
```

`title()`：以首字母大写的方式显示每个单词

这个方法没有所需的额外信息，因此它后面的括号内是空的。

`"ada lovelace"` ➡ `Ada Lovelace`

关于函数和方法

刚才遇到了两个关键词：

- 方法（method）
- 函数（function）

```
name.title()
```

```
print()
```

关于函数和方法

刚才遇到了两个关键词：

- 方法（method）
- 函数（function）

它们的书写方式略有不同：

```
实例名.方法名(...)
函数名(...)
```

目前为止，我们还不用知道它们之间的异同，但你可以把方法视为特殊的函数。

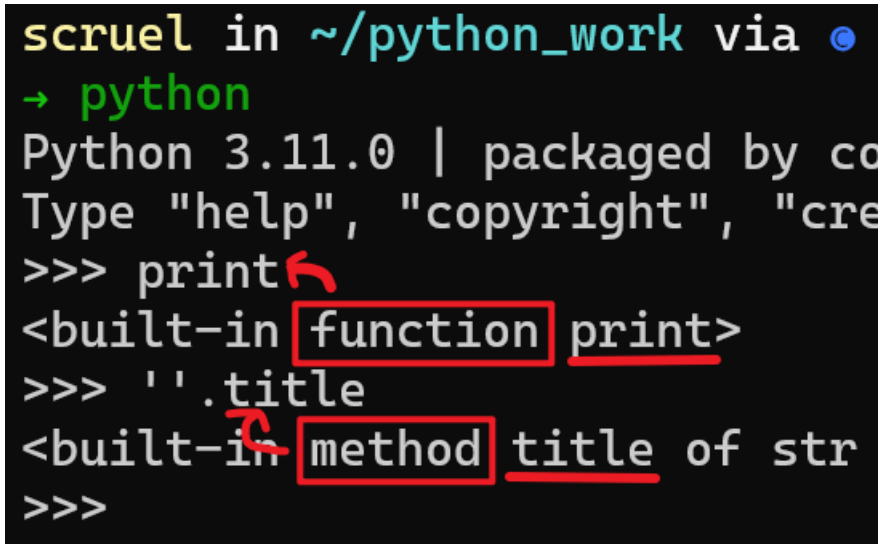
关于函数和方法

刚才遇到了两个关键词：

- 方法（method）
- 函数（function）

目前为止，我们还不用知道它们之间的异同，但你可以把方法视为特殊的函数。

如果你想要力求精确，可以通过去掉函数后面的括号，然后交给 Python 解释器执行代码，来告诉你它到底是什么：



2.3.1 字符串：大小写修改

- upper()：将字符串全部改为大写。

```
name = "Ada Lovelace"
print(name.upper())
```

"Ada Lovelace" → ADA LOVELACE

2.3.1 字符串：大小写修改

- upper()：将字符串全部改为大写。
- lower()：将字符串全部改为小写。

```
name = "Ada Lovelace"
print(name.lower())
```

"Ada Lovelace" → ada lovelace

2.3.1 字符串：大小写修改

- upper()：将字符串全部改为大写
- lower()：将字符串全部改为小写
- title()：每个单词的首字母大写

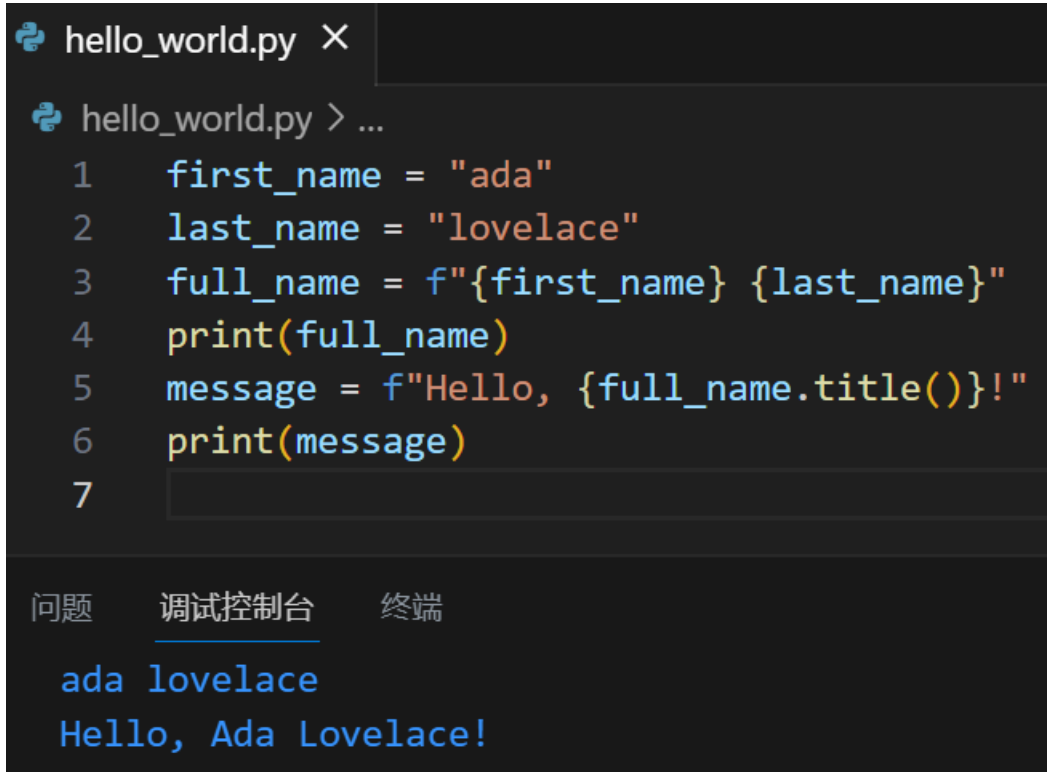
```
name = "ada lovelace"
print(name.title())
name = "Ada Lovelace"
print(name.upper())
print(name.lower())
```

2.3.2 字符串：f-字符串

我们可以在 f-字符串中，使用花括号来引用代码中定义的变量

f 是 format 的简写

```
first_name = "ada"
last_name = "lovelace"
full_name = f"{first_name} {last_name}"
print(full_name)
message = f"Hello, {full_name.title()}!"
print(message)
```



2.3.3 字符串：添加空白

我们可以使用空格来添加空白：

```
print('Python Rust')
```

```
Python Rust
```

2.3.3 字符串：添加空白

或使用特殊的字符来添加空白：

- `\t`：在字符串中表示制表符。

```
print('Python Rust')
```

```
Python Rust
```

```
print('Python\tRust')
```

```
Python	Rust
```

看着比空格要长一些

2.3.3 字符串：添加空白

或使用特殊的字符来添加空白：

- \t：在字符串中表示制表符。
- \n：在字符串中表示换行符。

```
print('Python\nRust')
```

```
Python
Rust
```

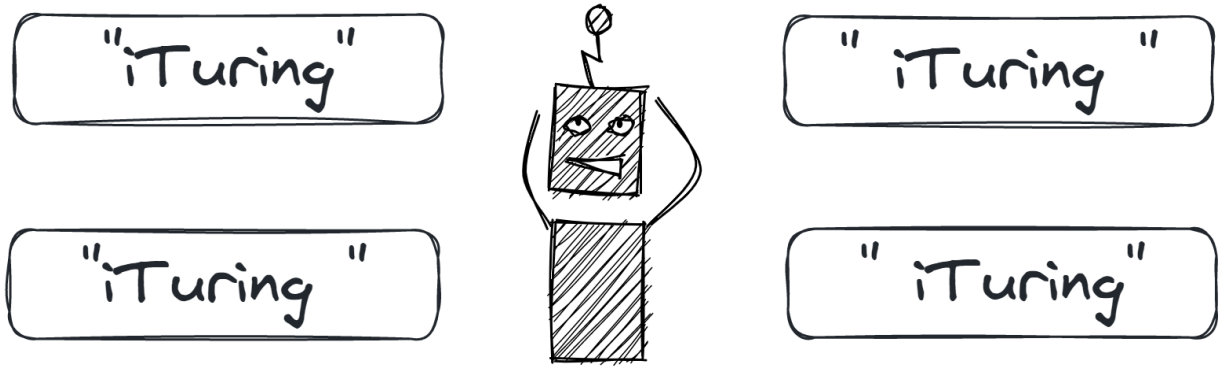
2.3.3 字符串：添加空白

- 直接使用空格
- \t：在字符串中表示制表符。
- \n：在字符串中表示换行符。

```
print('Python Rust')
print()
print('Python\tRust')
print()
print('Python\nRust')
```

2.3.4 字符串：删除空白

额外的空白可能令人迷惑，毕竟很难看出来有什么区别：



字符串空白？
Python 认得！

hello_world.py > ...

```
1  name = "iTuring"
2  print(f"Hi, {name}!")
3  name = "iTuring"
4  print(f"Hi, {name}!")
5  name = " iTuring"
6  print(f"Hi, {name}!")
7
```

问题　调试控制台　终端

```
Hi, iTuring!
Hi, iTuring!
Hi,  iTuring!
```

2.3.4 字符串：删除空白

可以使用以下方法来删除空白：

- lstrip()：移除左端的空白
- rstrip()：移除右端的空白
- strip()：移除两端的空白

```
' iTruing'.lstrip()
'iTruing '.rstrip()
' iTruing '.strip()
```

2.3.4 字符串：删除空白

可以使用以下方法来删除空白：

- `lstrip()`：移除左端的空白
- `rstrip()`：移除右端的空白
- `strip()` ：移除两端的空白

```
'iTruing'
'iTruing'
'iTruing'
```

2.3.4 字符串：删除空白

可以使用以下方法来删除空白：

- lstrip()：移除左端的空白
- rstrip()：移除右端的空白
- strip()：移除两端的空白

```
' iTruing'.lstrip()
'iTruing '.rstrip()
' iTruing '.strip()
```

hello_world.py > ...

```
1  name = "iTuring"
2  print(f"Hi, {name}!")
3  name = "iTuring"
4  print(f"Hi, {name.rstrip()}!")
5  name = " iTuring"
6  print(f"Hi, {name.lstrip()}!")
7  name = " iTuring "
8  print(f"Hi, {name.strip()}!")
9
```

问题　调试控制台　终端

```
Hi, iTuring!
Hi, iTuring!
Hi, iTuring!
Hi, iTuring!
```

2.3.5 字符串：删除前缀

还有一个常见的字符串处理任务是删除前缀

比如我们有时需要**删除**网址的**前缀** `'https://'` 部分

```
'https://www.ituring.com.cn'
```

2.3.5 字符串：删除前缀

还有一个常见的字符串处理任务是删除前缀

比如我们有时需要**删除**网址的**前缀** `'https://'` 部分

```
'www.ituring.com.cn'
```

2.3.5 字符串：删除前缀

还有一个常见的字符串处理任务是删除前缀

比如我们有时需要**删除**网址的**前缀** `'https://'` 部分

`removeprefix()`：移除字符串中指定的前缀

```
url = 'https://www.ituring.com.cn'
url = url.removeprefix('https://')
print(url)
```

运行结果

```
www.ituring.com.cn
```

2.3.6 字符串：避免语法错误

语法错误：包含非法的 Python 代码时发生的错误

例如，单引号和双引号的错误包裹，时常会引起语法错误：

```
message = 'Life's pathetic, let's pythonic.'
print(message)
```

运行结果

```
  File "<stdin>", line 1
    message = 'Life's pathetic, let's pythonic.'
                    ^
SyntaxError: unterminated string literal (detected at line 1)
```

2.3.6 字符串：避免语法错误

语法错误：包含非法的 Python 代码时发生的错误

我们可以这样纠正，将两边的单引号改为双引号：

```
message = "Life's pathetic, let's pythonic."
print(message)
```

运行结果

```
Life's pathetic, let's pythonic.
```

2.3.6 字符串：避免语法错误

语法错误：包含非法的 Python 代码时发生的错误（时常会遇到）

- 解释器无法正确地确定字符串的结束位置，所以会给出错误
- 语法错误较难定位（不借助工具时）
  - 利用编辑器的语法高亮
  - 注意编辑器的错误提示
  - 查看运行后的错误提示
  - 参考书籍附录 C 的建议

字符串的简单拼接

除了用 f-字符串来生成字符串，我们能对字符串“做运算”：

可以使用 + 号来拼接字符串：

```
"Hello" + " iTuring"
```

```
"Hello iTuring"
```

还可以使用 * 号来生成重复的字符串：

```
"Hello " * 2
```

```
"Hello Hello "
```

字符串的简单拼接

试着输出一下你的 Ohhhh！

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

```
1
1.5
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

```
print(1)
print(1.5)
```

运行结果

```
1
1.5
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

```
1000000
1000000.5
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

```
1_000_000
1_000_000.5
```

可在数中添加下划线使大数更易读

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

可在数中添加下划线使大数更易读

```
print(1000000)
print(1_000_000)

print(1000000.5)
print(1_000_000.5)
```

运行结果

```
1000000
1000000
1000000.5
1000000.5
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

数的基本运算：

加（+）

```
print(3 + 3)
print(2.5 + 2)
print(1 + 2)
print(1.5 + 1.5)
```

运行结果

```
6
4.5
3
3.0
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

数的基本运算：

加（+）减（-）

```
print(3 - 3)
print(2.5 - 2)
print(1 - 2)
print(1.5 - 1.5)
```

运行结果

```
0
0.5
-1
0.0
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

数的基本运算：

加（+）减（-）乘（*）

```
print(3 * 3)
print(2.5 * 2)
print(1 * 2)
print(1.5 * 1.5)
```

运行结果

```
9
5.0
2
2.25
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

数的基本运算：

加（+）减（-）乘（*）除（/）

```
print(3 / 3)
print(2.5 / 2)
print(1 / 2)
print(1.5 / 1.5)
```

运行结果

```
1.0
1.25
0.5
1.0
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

数的基本运算：

加（+）减（-）乘（*）除（/）

双斜杠（//）：表示整数除法

```
print(3 // 3)
print(2.5 // 2)
print(1 // 2)
print(1.5 // 1.5)
```

运行结果

```
1
1.0
0
1.0
```

2.4 数：整数和浮点数

- 整数（integer）：不带小数点的数
- 浮点数（float）：带小数点的数

数的基本运算：

加（+）减（-）乘（*）除（/）

双斜杠（//）：表示整数除法

双乘号（**）：表示乘方运算

```
print(3 ** 3)
print(2.5 ** 2)
print(1 ** 2)
print(1.5 ** 1.5)
```

运行结果

```
27
6.25
1
1.8371173070873836
```

2.4 补充：布尔值（Boolean）

用于表示逻辑真或假

- `True`：表示真
- `False`：表示假

```
print(True)
print(False)
```

运行结果

```
True
False
```

2.4 同时给多个变量赋值

Python 支持在一行代码中给多个变量赋值：

```
x, y, z = 0, 0, 0
```

上面的代码中，我们将 x、y、z 都初始化为了零。

书写的顺序就是分配的顺序

2.4 常量（constant）

常量：程序内一直保持不变的变量

Python 没有内置的常量支持，我们给出的是一个约定俗成的惯例

- 使用**全大写字母（单词由下划线分割）**来将某个变量视为常量

```
MAX_CONNECTIONS = 5000
```

请不要在程序中修改任何常量！

2.5 注释（Comment）

注释：让你能够使用自然语言在程序中添加说明

- Python 中使用（#）标识注释
- 解释器会忽略注释的内容

```
# 向大家问好
print("Hello Python people!")
```

2.5 注释（Comment）

注释：让你能够使用自然语言在程序中添加说明

- Python 中使用（#）标识注释
- 解释器会忽略注释的内容

```
# 向大家问好
# print("Hello Python people!")
```

2.5 该编写什么样的注释

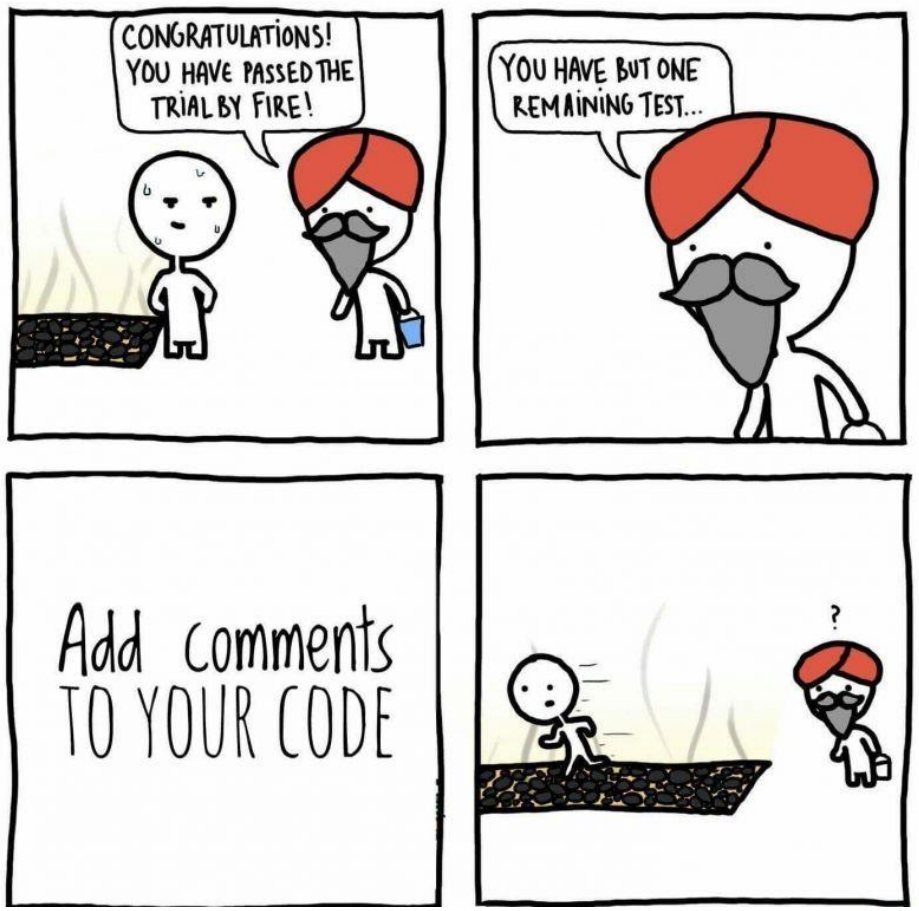
90% of all code comments:


```
#                    _ooOoo_
#                   o8888888o
#                   88" . "88
#                   (| -_- |)
#                    O = /O
#                ____/`---'____
#              .   ' | |// `.
#               / ||| : |||//
#             / _||||| -:- |||||-
#               | |  - /// | |
#             | _| ''---/'' | |
#              .-__ `-` ___/-. /
#           ___`. .' /--.-- `. . __
#        ."" '< `.___<|>_/___.' >'"".
#       | | : `- `.;` _ /`;.`/ - ` : | |
#         `-. _ __ /__ _/ .-` / /
# ======`-.____`-.________/___.-`____.-'======
#                    `=---='
#
# .............................................
#           佛祖保佑                永无BUG
```

2.5 该编写什么样的注释

- 阐述代码的目的
- 阐述难懂的细节
- 要力求**清晰简洁**
- 便于个人回忆和团队合作

- 避免添加没必要的注释
- 不要怕麻烦而不写注释



2.6 Python 之禅

Python 社区的理念（由 Tim Peters 撰写）

```
>>> import this
The Zen of Python, by Tim Peters
```

- Beautiful is better than ugly.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.
- There should be one -- and preferably only one -- obvious way to do it.
- Now is better than never.

2.7 小结

- 学习了如何使用变量，如何创建描述性变量名，以及如何消除名称错误和语法错误
- 学习了字符串是什么，以及如何调整字符串的显示方式，使用空白来显示整洁的输出，还学习了如何删除字符串中多余的字符。
- 学习了如何使用整数和浮点数，了解了一些使用数值数据的方式。
- 学习了如何编写说明性注释，了解了让代码尽可能简单的理念

在下一章中，我们将学习关于列表的相关知识。

课后拓展

- 学习并使用用于获取帮助信息的内置函数 `help()`
- 了解如何养成良好的编码习惯
- 了解什么是转义字符，利用它解决 2.3.6 节中出现的错误
- 完成书中习题，粗略地浏览一下 Python 之禅中的其他指导原则

可选拓展

- 学习用于获取变量类型的内置函数 `type()`
- 了解数值运算的运算优先级及括号的用法
- 了解位运算以及其他的数类型
- 了解精度丢失及其原因