

# 变量和简单的数据类型

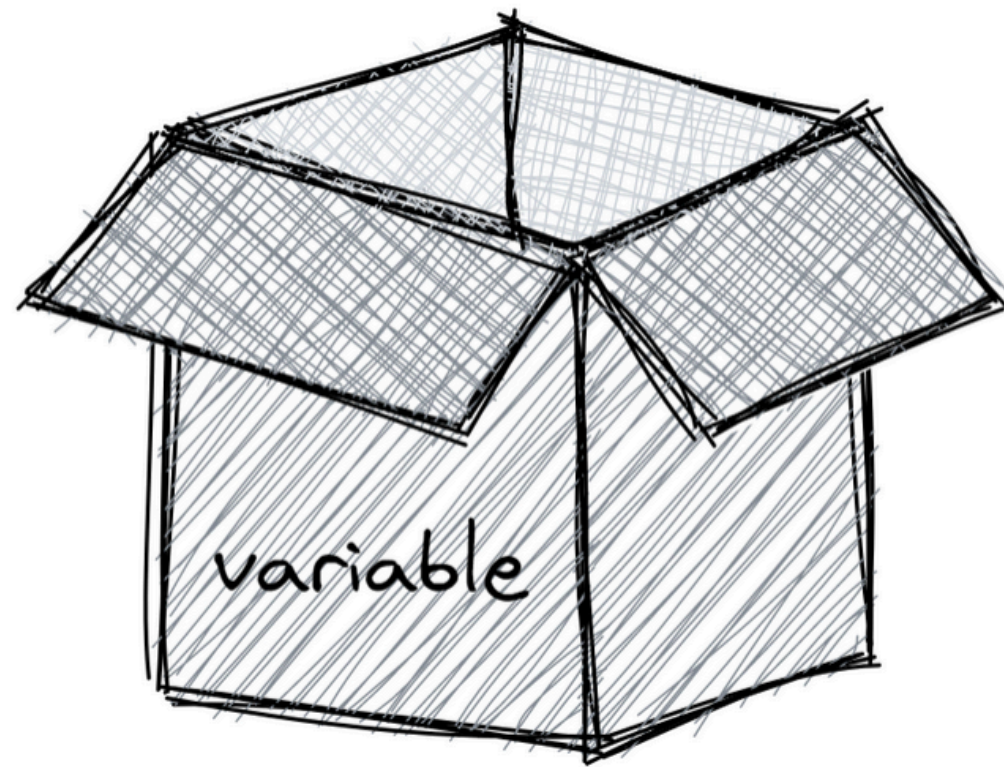
刘俊波 2025.7.13

# 目录

- 变量
- 简单的数据类型
- Input()输入函数

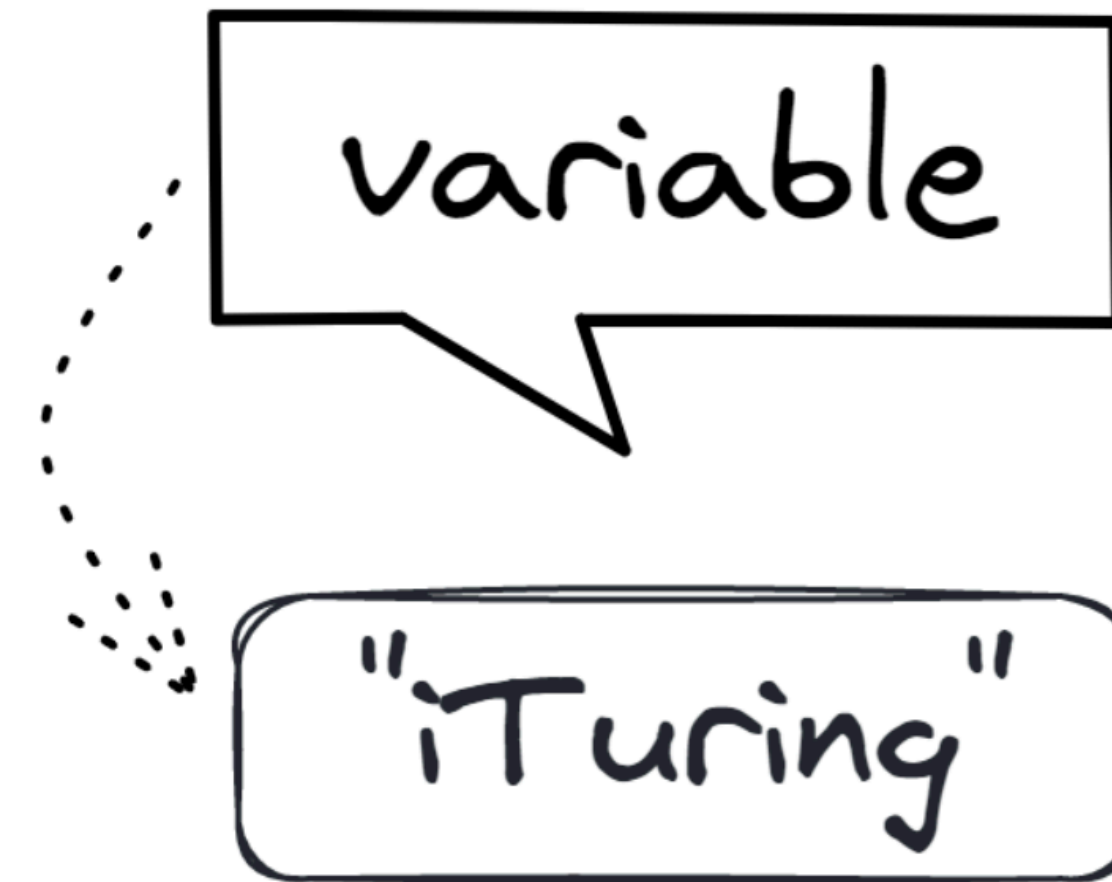
# 变量是什么？

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！

# 变量命名规则

- 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）

# 动手试一试

在完成下面的每个练习时，都编写一个独立的程序。在保存每个程序时，使用符合标准 Python 约定的文件名：使用小写字母和下划线，如 `simple_message.py` 和 `simple_messages.py`。

**练习 2.1：简单消息** 将一条消息赋给变量，并将其打印出来。

**练习 2.2：多条简单消息** 将一条消息赋给变量，并将其打印出来；再将变量的值修改为一条新消息，并将其打印出来。

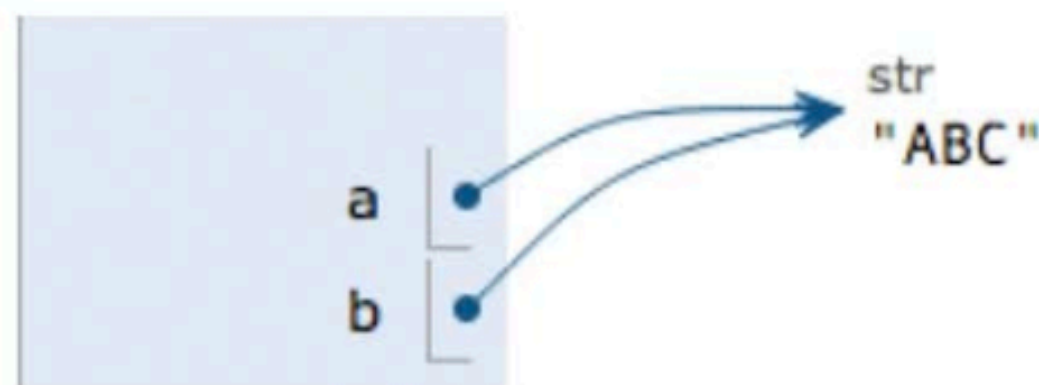


# 思考print(b)输出什么?

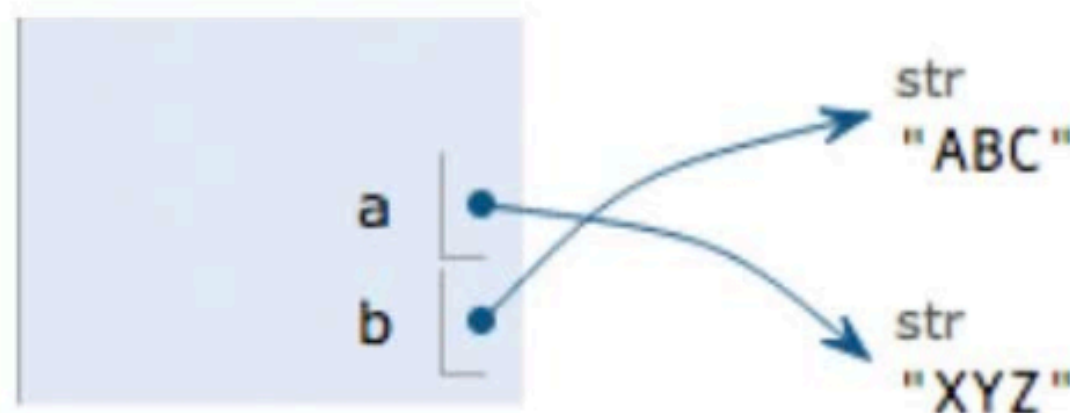
执行 `a = 'ABC'`，解释器创建了字符串'ABC' 和变量 `a`，并把 `a` 指向'ABC'：



执行 `b = a`，解释器创建了变量 `b`，并把 `b` 指向 `a` 指向的字符串'ABC'：



执行 `a = 'XYZ'`，解释器创建了字符串'XYZ'，并把 `a` 的指向改为'XYZ'，但 `b` 并没有更改：



```
a = 'ABC'
b = a
a = 'XYZ'
print(b)
```

# 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，或者是单引号
- 如果字符串里包含单引号，则用双引号引起，例如 `"It's my fault"` `'John say "Hi All" '`
- 使用三个单引号或者三个双引号，可以创建跨行字符串

```
"""I want to say to you today,  
My friends """
```

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

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



都是英文括号哦！

每个方法后面都跟着一对括号



# 字符串删除空白

可以使用以下方法来删除空白：

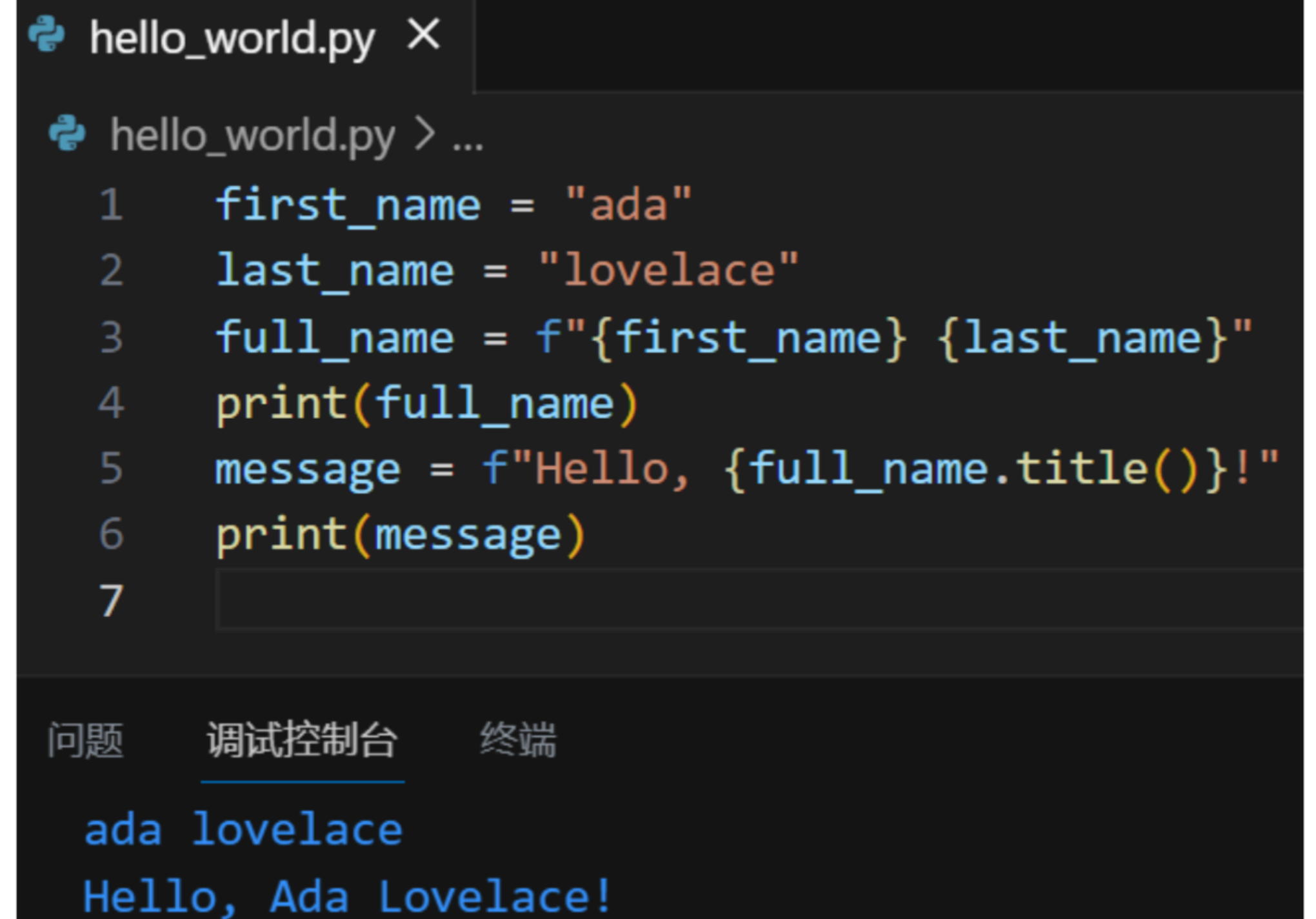
- **lstrip** () ： 移除左端的空白
- **rstrip** () ： 移除右端的空白
- **strip** () ： 移除两端的空白

# 在字符串中使用变量

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)
```



The screenshot shows a code editor with a file named 'hello\_world.py'. The code defines variables for first and last names, concatenates them into a full name, and then prints the full name and a formatted greeting. The output in the terminal shows the full name 'ada lovelace' and the formatted greeting 'Hello, Ada Lovelace!'.

```
hello_world.py X
hello_world.py > ...
1 first_name = "ada"
2 last_name = "lovelace"
3 full_name = f"{first_name} {last_name}"
4 print(full_name)
5 message = f"Hello, {full_name.title()} !"
6 print(message)
7

问题 调试控制台 终端
ada lovelace
Hello, Ada Lovelace!
```

# 动手试一试

在完成下面的每个练习时，都编写一个独立的程序，并将其保存到名称类似于 `name_cases.py` 的文件中。如果遇到困难，请休息一会儿或参阅附录 C 提供的建议。

**练习 2.3：个性化消息** 用变量表示一个人的名字，并向其显示一条消息。显示的消息应非常简单，如下所示。

Hello Eric, would you like to learn some Python today?

**练习 2.4：调整名字的大小写** 用变量表示一个人的名字，再分别以全小写、全大写和首字母大写的方式显示这个人名。

**练习 2.5：名言 1** 找到你钦佩的名人说的一句名言，将这个名人的姓名和名言打印出来。输出应类似于下面这样（包括引号）。

Albert Einstein once said, “A person who never made a mistake never tried anything new.”

**练习 2.6：名言 2** 重复练习 2.5，但用变量 `famous_person` 表示名人的姓名，再创建要显示的消息并将其赋给变量 `message`，然后打印这条消息。

2.7和2.8参考电子书

# 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

# 动手试一试

**练习 2.9：数字 8** 编写 4 个表达式，分别使用加法、减法、乘法和除法运算，但结果都是数字 8。为了使用函数调用 `print()` 来显示结果，务必将这些表达式用括号括起来。也就是说，你应该编写 4 行类似于这样的代码：

```
print(5+3)
```

输出应为 4 行，其中每行都只包含数字 8。

**练习 2.10：最喜欢的数** 用一个变量来表示你最喜欢的数，再使用这个变量创建一条消息，指出你最喜欢的数是什么，然后将这条消息打印出来。

# 常量

- 常量：在程序中一直保持不变的变量
- 约定使用全大写字母（单词由下划线分割）
- `MAX_CONNECTIONS = 5000`



# 注释

- 注释：方便在程序中添加说明，解释器忽略注释内容
- # 标识注释 `# this is comment`
- 多行注释？  
`"""This is  
a  
comment"""`

# 用户输入

`input ()` 函数：让程序等待用户输入

函数的接受一个参数：向用户显示的提示 (**prompt**)

```
name = input ( "Please enter your name: " )  
print ( f" \n Hello,  {name}!" )
```

**运行结果**

Please enter your name: