

# Tutorial 2: Python 基础类型与运算

方嘉聪

jiacong\_fang@stu.pku.edu.cn

2025 年 3 月 6 日

# 目录

## 课程安排

### 第一次作业

## Python 环境介绍

命令行交互模式

Jupyter Notebook

文本编辑器/IDE

## 数据类型与数据运算

变量的赋值与使用

数据类型与运算

常见报错及处理

推荐资源

## 作业安排

作业一已经发布:

- ▶ 主要内容: 基础语法, 包含 0304 与 0311 两次课的内容;
- ▶ 其中 5-8 题与字符串基本操作与格式化输出相关;
- ▶ 截止时间: 2025-03-20 17:00:00

请大家确认已经加入了 Openjudge. <http://csc6.openjudge.cn>

如果还没有加入请按照教学网上的教程及时加入, 并将用户名改为 学号 + 姓名.

# 目录

## 课程安排

第一次作业

## Python 环境介绍

命令行交互模式

Jupyter Notebook

文本编辑器/IDE

## 数据类型与数据运算

变量的赋值与使用

数据类型与运算

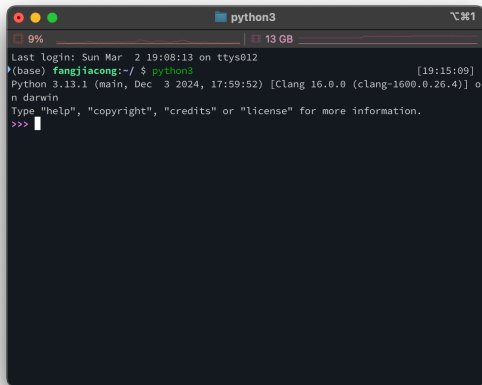
常见报错及处理

推荐资源

## Python 交互模式

Python 的命令行交互模式 (REPL: Read-Eval-Print Loop) 是一个**即问即答**的 Python 环境。

优势: 及时反馈, 便于学习, 不用额外创建文件



```
python3
9% 13 GB
Last login: Sun Mar  2 19:08:13 on ttys012
(base) fangjiacong:~/ $ python3
Python 3.13.1 (main, Dec  3 2024, 17:59:52) [Clang 16.0.0 (clang-1600.0.26.4)] o
n darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

## Python 交互模式 (Cont.)

如何使用? Demo Here.

### 1. 打开终端:

- ▶ Windows: 按 Win + R, 输入 cmd 回车;
- ▶ MacOS: 搜索 Terminal app, 并打开.

### 2. 启动 Python:

- ▶ Windows: 输入 python 回车;
- ▶ MacOS: 输入 python3 回车.

### 3. 如果启动成功, 则会弹出来一系列的版本信息, 然后看到下面这种符号, 意味着 python 正在等待你的输入。

```
1 Python 3.13.1 .... # 后面是一堆版本信息
2 >>>
```

## Python 交互模式 (Cont.)

让我们来看一些小练习 (Python 是最好用的内置计算器! )

```
1 >>> 2025
2 ?
3 >>> 2 + 2025
4 ?
5 >>> 2 ** 10 + 10 ** 3 # ** 表示乘方
6 ?
7 >>> 7 / 3
8 ?
9 >>> 7 // 3
10 ?
11 >>>
```

# Jupyter Notebook

**Jupyter Notebook** 是一个开源的交互式编程环境(一个可以实时跑 Python 代码的 “Word”), 可以方便的将代码、文本、图片和可视化内容整合在一个文档中, 具体来说

1. **交互编程**: 你可以逐行或逐段运行代码, 且立即查看结果
2. **笔记与代码结合**: 你可以在代码旁边添加注释、说明和公式 (支持 Markdown)
3. **教学与学习**: 在上机课上我们有时将使用其展示代码运行



# Jupyter Notebook

**Jupyter Notebook** 是一个开源的交互式编程环境(一个可以实时跑 Python 代码的 “Word”), 可以方便的将代码、文本、图片和可视化内容整合在一个文档中, 具体来说

1. **交互编程**: 你可以逐行或逐段运行代码, 且立即查看结果
  2. **笔记与代码结合**: 你可以在代码旁边添加注释、说明和公式 (支持 Markdown)
  3. **教学与学习**: 在上机课上我们有时将使用其展示代码运行
- ▶ (推荐) VS Code 中的配置参考 [教学网-上机资料](#) 或 [该教程](#).
  - ▶ PyCharm 需要使用 Professional/Education 版本, 可以在 [官网](#) 申请 Educational License.

## 文本编辑器/IDE

代码编辑器和集成开发环境 (**IDE**) 类似于 “Word” 或 “备忘录”，但是专门用于编写代码。

- ▶ **Vim/NeoVim**: 轻量级编辑器，适合高手，但学习曲线较陡。
- ▶ **PyCharm**: 专业的 Python IDE, 功能全面, 但占用内存较大。
- ▶ **VS Code**: 轻量级编辑器，丰富插件，支持多种语言。
- ▶ **记事本/文本编辑器**: 虽然可以用，但缺乏代码编辑的专用功能，效率较低。

## 文本编辑器/IDE

代码编辑器和集成开发环境 (**IDE**) 类似于 “Word” 或 “备忘录”，但是专门用于编写代码。

- ▶ **Vim/NeoVim**: 轻量级编辑器，适合高手，但学习曲线较陡。
- ▶ **PyCharm**: 专业的 Python IDE, 功能全面, 但占用内存较大。
- ▶ **VS Code**: 轻量级编辑器，丰富插件，支持多种语言。
- ▶ **记事本/文本编辑器**: 虽然可以用，但缺乏代码编辑的专用功能，效率较低。

注：编辑器本身并不运行代码，而会将代码“发送”给 Python 解释器 (Python Kernel/Launcher) 执行。

# 目录

## 课程安排

第一次作业

## Python 环境介绍

命令行交互模式

Jupyter Notebook

文本编辑器/IDE

## 数据类型与数据运算

变量的赋值与使用

数据类型与运算

常见报错及处理

推荐资源

## 变量的存储

在我们写下 `a = 3` 时，发生了什么？

## 变量的存储

在我们写下 `a = 3` 时，发生了什么？

1. 在内存中开辟一块空间，存储 3。我们把 3 称为值/对象。
2. 给 3 这个内存空间起一个名字 a。我们把 a 称为变量。

## 变量的存储

在我们写下 `a = 3` 时，发生了什么？

1. 在内存中开辟一块空间，存储 3。我们把 3 称为值/对象。
2. 给 3 这个内存空间起一个名字 a。我们把 a 称为变量。

直观上理解：

- ▶ 把内存想象成我们现在在的房间，里面有很多盒子，
- ▶ 进行 `a = 3` 操作，就是找到一个空盒子，把 3 放进去，
- ▶ 然后往盒子上贴一个标签 a，
- ▶ 以后我们就可以通过 a 来找到这个盒子。

## 变量的被赋值和使用

以下面这段简单代码为例:

```
1  a = 3    # a 首次被赋值 => 定义
2  b = a    # a 被引用而后赋值给 b
3  a = 4    # a 被重新赋值
```

1. **赋值**: 变量在 = 左边是 被赋值. 标签 **a** 贴在 3 的盒子上。
2. **定义**: 变量首次被赋值称为被定义, 先给 3 找一个盒子, 然后贴上标签 **a**。
3. **引用**: 变量在 = 右边, 参与表达式运算, 被函数调用时被引用. 通过标签 **a** 找到 3 的盒子, 再进行函数调用/表达式运算/赋值。



## 变量的被赋值和使用

```
1  a = 3    # a 首次被赋值 => 定义
2  b = a    # a 被引用而后赋值给 b
3  a = 4    # a 被重新赋值
```

1. **赋值**: 变量在 = 左边是 被赋值. 标签 **a** 贴在 3 的盒子上。

## 变量的被赋值和使用

```
1  a = 3    # a 首次被赋值 => 定义
2  b = a    # a 被引用而后赋值给 b
3  a = 4    # a 被重新赋值
```

1. **赋值**: 变量在 = 左边是 被赋值. 标签 **a** 贴在 3 的盒子上。
2. **定义**: 变量首次被赋值称为被定义, 先给 3 找一个盒子, 然后贴上标签 **a**。

## 变量的被赋值和使用

```
1 a = 3    # a 首次被赋值 => 定义
2 b = a    # a 被引用而后赋值给 b
3 a = 4    # a 被重新赋值
```

1. **赋值**: 变量在 = 左边是 被赋值. 标签 **a** 贴在 3 的盒子上。
2. **定义**: 变量首次被赋值称为被定义, 先给 3 找一个盒子, 然后贴上标签 **a**。
3. **引用**: 变量在 = 右边, 参与表达式运算, 被函数调用时 被引用. 通过标签 **a** 找到 3 的盒子, 再进行函数调用/表达式运算/赋值。

**注意**: 变量的使用遵循 先定义后使用 的原则, 必须先赋值才能引用, 否则报错 **NameError**

# 基本运算

Demo Here.  
(Refer to the attached Jupyter Notebook)

# Python 常见报错

## 1. **SyntaxError**: invalid syntax. 语法错误.

- ▶ 为括号、引号等未成对出现. 输入了非法字符 (如中文标点).
- ▶ 在 **if**, **for**, **while** 等语句后忘记加 **:**.

# Python 常见报错

1. **SyntaxError**: invalid syntax. 语法错误.
  - ▶ 为括号、引号等未成对出现. 输入了非法字符 (如中文标点).
  - ▶ 在 `if`, `for`, `while` 等语句后忘记加 `:`.
2. **IndentationError**: unexpected indent. 缩进错误.
  - ▶ 代码块内部的缩进不一致. → 不要将空格与制表符混用.

## Python 常见报错

1. **SyntaxError**: invalid syntax. 语法错误.
  - ▶ 为括号、引号等未成对出现. 输入了非法字符 (如中文标点).
  - ▶ 在 **if**, **for**, **while** 等语句后忘记加 **:**.
2. **IndentationError**: unexpected indent. 缩进错误.
  - ▶ 代码块内部的缩进不一致. → 不要将空格与制表符混用.
3. **NameError**: name '...' is not defined. 变量未定义.
  - ▶ 变量未被赋值, 未定义就被引用.

## Python 常见报错

1. **SyntaxError**: invalid syntax. 语法错误.
  - ▶ 为括号、引号等未成对出现. 输入了非法字符 (如中文标点).
  - ▶ 在 **if**, **for**, **while** 等语句后忘记加 **:**.
2. **IndentationError**: unexpected indent. 缩进错误.
  - ▶ 代码块内部的缩进不一致. → 不要将空格与制表符混用.
3. **NameError**: name '...' is not defined. 变量未定义.
  - ▶ 变量未被赋值, 未定义就被引用.
4. **TypeError**: unsupported operand type(s) for ...
  - ▶ 不同类型的数据进行了不支持的运算. 例如 **'str'** + 1.



## Python 常见报错

1. **SyntaxError**: invalid syntax. 语法错误.
  - ▶ 为括号、引号等未成对出现. 输入了非法字符 (如中文标点).
  - ▶ 在 **if**, **for**, **while** 等语句后忘记加 **:**.
2. **IndentationError**: unexpected indent. 缩进错误.
  - ▶ 代码块内部的缩进不一致. → 不要将空格与制表符混用.
3. **NameError**: name '...' is not defined. 变量未定义.
  - ▶ 变量未被赋值, 未定义就被引用.
4. **TypeError**: unsupported operand type(s) for ...
  - ▶ 不同类型的数据进行了不支持的运算. 例如 `'str' + 1`.
5. **ZeroDivisionError**: division by zero. 除零错误.
  - ▶ 除数为零. 例如 `1 / 0`.

## Python 常见报错

1. **SyntaxError**: invalid syntax. 语法错误.
  - ▶ 为括号、引号等未成对出现. 输入了非法字符 (如中文标点).
  - ▶ 在 **if**, **for**, **while** 等语句后忘记加 **:**.
2. **IndentationError**: unexpected indent. 缩进错误.
  - ▶ 代码块内部的缩进不一致. → 不要将空格与制表符混用.
3. **NameError**: name `'..'` is not defined. 变量未定义.
  - ▶ 变量未被赋值, 未定义就被引用.
4. **TypeError**: unsupported operand type(s) for ...
  - ▶ 不同类型的数据进行了不支持的运算. 例如 `'str' + 1`.
5. **ZeroDivisionError**: division by zero. 除零错误.
  - ▶ 除数为零. 例如 `1 / 0`.
6. More errors will be discussed in the future. **Google it!**

## Openjudge 评测结果



### OJ上评测结果中未通过的测试及其含义

- ☑ **Compile Error:** 语法错误, 程序语句不符合语法规则, 注意看错误提示
- ☑ **Runtime Error:** 程序运行出错。可能有多种错误, 需结合程序及测试样例查错, 例如对一个列表去索引 `a[i]`, 但变量 `i` 取到了大于该列表长度的值。有同学会困惑为什么本地运行是对的, 因为本地你可能只测试了一个最简单的样例, 未出现上述情况。碰到该错误, 首先要检查你的程序代码逻辑上是否有漏洞。
- ☑ **Wrong Answer:** 答案错了。有同学很不解, 我本地跑了一个case, 都是对的呀? 那可能你的程序都是简单的输入, 测评系统上一些“刁钻的”测试点可能你的程序就不过, 例如1不是素数2是素数, 要求输出All, 而你的输出的是ALL
- ☑ **Presentation Error:** 格式错误, 最常见的多了换行之类的。
- ☑ 总之, 上述错误是几大类错误, 而不是对应某个具体的错误, 大家需要多从自己程序上找问题, 注意输入输出格式, 多试试一些测试点, 尝试使用debug去找错误。

### ► 在线的 Python 代码可视化工具: Python Tutor

The screenshot shows the Python Tutor interface in a web browser. The main area displays a Python code snippet:

```
Python 3.11
known limitations
1 def func1(a, b):
2     n = a+b
3     print(n)
4
5 n = 0
6 a = 123
7 b = 456
8 func1(a,b)
9 print(n)
Edit this code
```

Below the code, there are navigation controls: a slider and buttons for "<< First", "< Prev", "Next >", and "Last >>". The text "Step 8 of 10" is displayed.

On the right side, there is a "Print output (drag lower right corner to resize)" box, which is currently empty. Below it, the "Frames" and "Objects" panels are visible. The "Frames" panel shows the "Global frame" with variables "func1", "n", "a", and "b". The "Objects" panel shows the "function" object "func1(a, b)".

At the bottom, there is a message box with a robot icon that says: "Greetings, human! 🤖 I'm a new experimental AI Tutor ready to help you. Your code and visualization will be automatically sent to me, so **do not copy-paste them** into your question. Ask your question below. Or choose a template, edit it, and click 'Send':"

## 推荐资源

以下是一些有用的资源:

- ▶ 在线的 Python 代码可视化工具: [Python Tutor](#)
- ▶ Python 课程: UC Berkely CS61A <https://cs61a.org>
- ▶ Textbook: Composing Programs  
<https://www.composingprograms.com>, 社区中文版
- ▶ Markdown 官方教程: <https://markdown.com.cn>
- ▶ 打字练习: <https://www.keybr.com>

个人建议: 多与 Python 交互, 独立思考, 练习与交流.

- ▶ **Google and Bing are your friends.** (Baidu)
- ▶ 线上提问时请描述清楚问题, 已经做了哪些尝试.
- ▶ **截图 + 代码 + 报错信息.**

# 题外话: 一个有趣的视频

火柴人 VS 编程 (Coding)

解析视频



Welcome Your Feedback!

That's all for today!

My next tutorial will be on **March 27th**, and will cover:

程序调试, 大语言模型辅助程序设计, 函数与枚举遍历算法.