

Python 数据类型 & Debug

一、计算机组成原理与操作系统（极简）

数据在不通电的情况下可以长期持久地 (persistently) 存储在 **磁盘** (如固态硬盘 SSD、机械硬盘 HDD) 或磁带 (常用于数据备份、长期归档) 里。但在需要呈现 (print、render、show、display、play)、计算加工 (compute、transform、analyze、machine learning、deep learning) 或编解码 (encode、decode) 时, 就需要通电的 **CPU** 和 **内存** (硬件), 在操作系统 (软件) 里以 **进程** (process) 为单元 (相互隔离) 进行处理。例如, Microsoft Word 启动后就是一个进程, 我们在 Word 进程里打开某个 `.docx` 文档, 将其从磁盘加载 (**读取**) 到内存, 然后在图形界面 (GUI) 里查看和编辑 (**计算**) 内存中的文档, 最后将内存数据保存 (**写入**) 到磁盘。同理, Python 解释器 (interpreter) 启动后也是一个进程, 她按照流程 (flow) 执行我们准备好的 Python 代码, 根据我们代码的要求, 转告 (即 **调用**, call) 操作系统或其他软件 (即 **依赖项**, dependency), 委托她们替我们执行各种 “读取——计算——写入” 等工作。我们并不需要完全理解依赖项内部的工作细节 (黑箱), 只需要清楚每个调用的主体 (即 **对象**, object) 是什么 **类型** (type), 每个调用的输入 (即 **参数**, parameter/argument)、输出 (即 **返回值**, return value) 是什么类型, 以及调用会对内存数据、磁盘文件做什么修改, 就足以支持我们自动批量地完成工作了。

二、如何面对 AI 编程

1. 创建一个新的项目

- ① Fork 第 04 周打卡 仓库至个人名下, 然后将名下此仓库 Clone 到本地计算机 (路径: repo)
- ② 用 VS Code 打开项目目录, 新建一个 environment.yml 文件, 修改内容, 安装 Python 3.12

```
! environment.yml
1  name: week04
2  channels:
3    - conda-forge
4  dependencies:
5    - python=3.12
```

- ③ 终端运行 `conda env create` 命令创建 Conda 环境 (路径: repo/week04)
- ④ 新建一个 `contacts.txt` 文件, 内容如下:

```
≡ contacts.txt
1  白展堂 男 baizhantang@163.com
2  佟湘玉 女 tongxiangyu@163.com
3  吕轻侯 男 lvqinghou@126.com
4  郭芙蓉 女 guofurong@126.com
5  李秀莲 男 lixiulian@163.com
6  祝无双 女 zhuwushuang@163.com
7
```

2. **要求**：新建一个 `main.py` 文件，里面写 Python 代码，读取 `contacts.txt` 文件的内容，进行数据处理后，输出一个 `emails.txt` 文件，要求输出是先按邮箱域名排序 (126.com 排在 163.com 之前)，然后再按邮箱用户名排序 (guofurong 排在 lvqinghou 之前)

```
to: <guofurong@126.com>
尊敬的郭芙蓉女士，您的会员资格即将到期，请及时续费。
---
to: <lvqinghou@126.com>
尊敬的吕轻侯先生，您的会员资格即将到期，请及时续费。
---
to: <baizhantang@163.com>
尊敬的白展堂先生，您的会员资格即将到期，请及时续费。
```

3. 将上述“要求”复制粘贴到大模型，请 AI 帮助编写程序初稿（本文选择 deepseek）



以下是实现你需求的 Python 程序 `main.py`，它会读取 `contacts.txt` 并按指定规则生成

`emails.txt`：

```
python
复制

def process_contacts():
    # 读取联系人数据
    with open('contacts.txt', 'r', encoding='utf-8') as f:
        contacts = [line.strip().split() for line in f if line.strip()]

    # 处理排序规则：先按域名（126.com优先），再按用户名
    def sort_key(contact):
        email = contact[2]
        username, domain = email.split('@')
        return (domain, username) # 元组排序会先比较domain，再比较username

    contacts_sorted = sorted(contacts, key=sort_key)

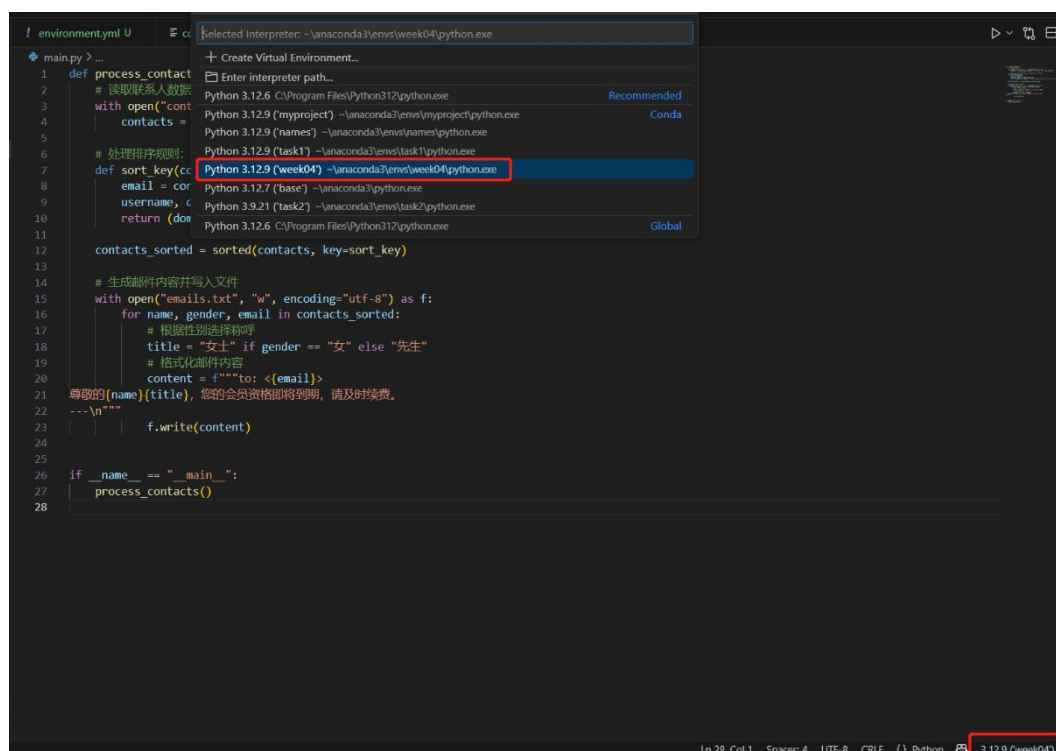
    # 生成邮件内容并写入文件
    with open('emails.txt', 'w', encoding='utf-8') as f:
        for name, gender, email in contacts_sorted:
            # 根据性别选择称呼
            title = "女士" if gender == "女" else "先生"
            # 格式化邮件内容
            content = f"to: <{email}>\n尊敬的{name}{title}，您的会员资格即将到期，请及时续费。"
            f.write(content)
            f.write("\n")

if __name__ == "__main__":
    process_contacts()
```

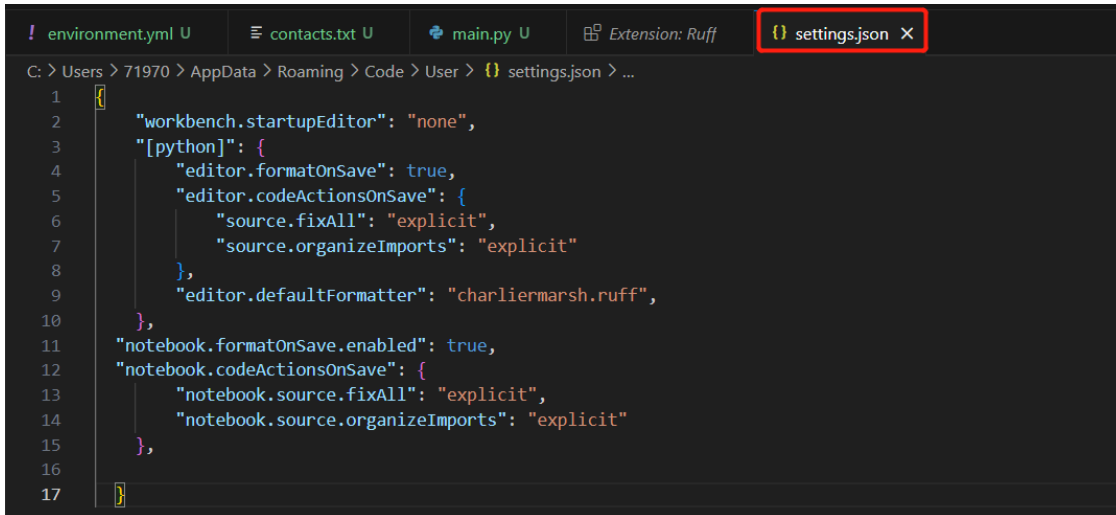
4. 将上述代码复制粘贴到 `main.py` 并保存

5. 安装拓展

- ① Python 拓展：编写 `.py` 文件时能够显示和选择 Python 解释器（可以点右下角选择或者 `ctrl shift p` 打开菜单输入 `select interpreter` 再选择相应的解释器）

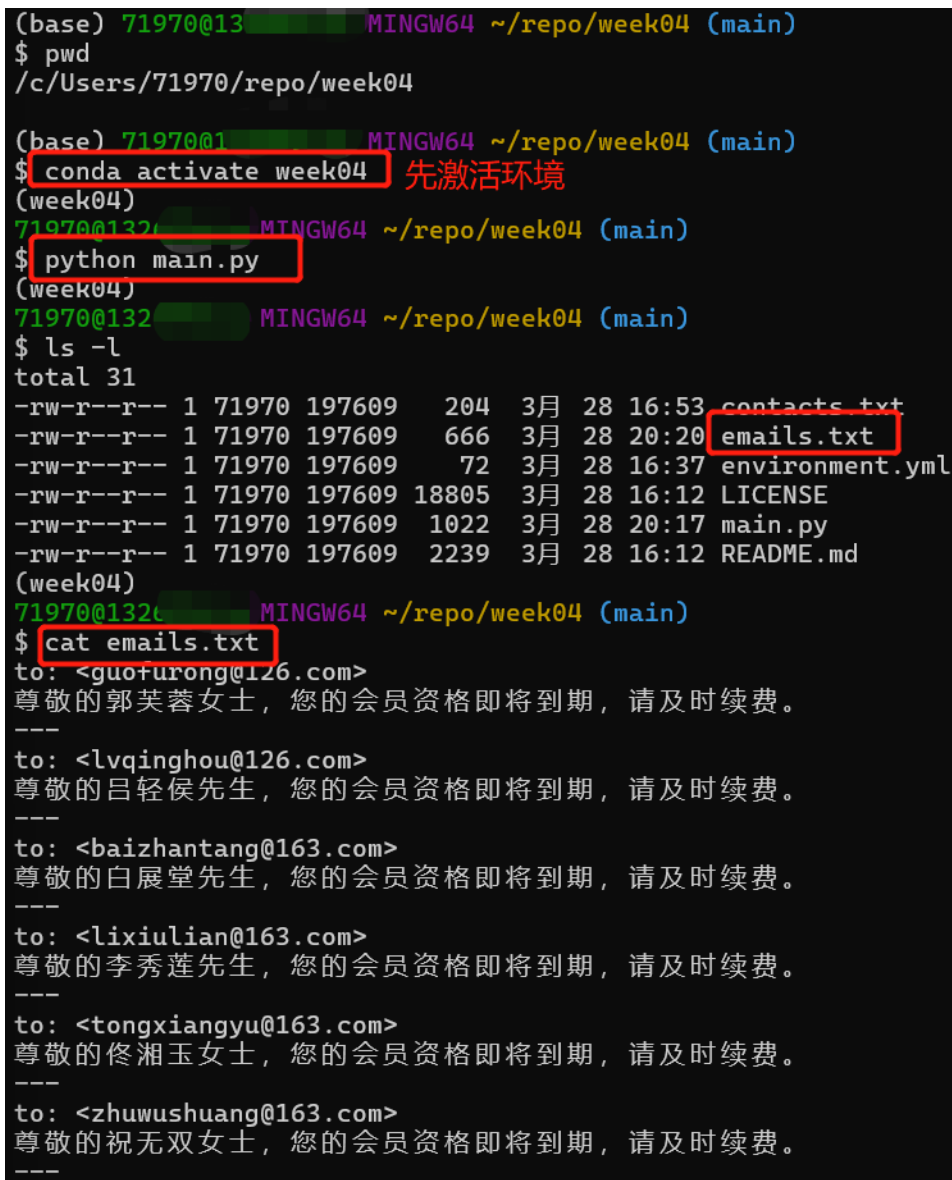


②Ruff 拓展：按照 Ruff 文档配置（打开菜单搜索 settings），实现在保存 .py 文件时能够自动规范化 Python 代码



```
1 {
2   "workbench.startupEditor": "none",
3   "[python]": {
4     "editor.formatOnSave": true,
5     "editor.codeActionsOnSave": {
6       "source.fixAll": "explicit",
7       "source.organizeImports": "explicit"
8     },
9     "editor.defaultFormatter": "charliermarsh.ruff",
10  },
11  "notebook.formatOnSave.enabled": true,
12  "notebook.codeActionsOnSave": {
13    "notebook.source.fixAll": "explicit",
14    "notebook.source.organizeImports": "explicit"
15  },
16 }
17 }
```

6. 接下来在终端运行代码（激活环境、运行代码、查看输出）



```
(base) 71970@132 MINGW64 ~/repo/week04 (main)
$ pwd
/c/Users/71970/repo/week04

(base) 71970@132 MINGW64 ~/repo/week04 (main)
$ conda activate week04 先激活环境
(week04)
71970@132 MINGW64 ~/repo/week04 (main)
$ python main.py
(week04)
71970@132 MINGW64 ~/repo/week04 (main)
$ ls -l
total 31
-rw-r--r-- 1 71970 197609 204 3月 28 16:53 contacts.txt
-rw-r--r-- 1 71970 197609 666 3月 28 20:20 emails.txt
-rw-r--r-- 1 71970 197609 72 3月 28 16:37 environment.yml
-rw-r--r-- 1 71970 197609 18805 3月 28 16:12 LICENSE
-rw-r--r-- 1 71970 197609 1022 3月 28 20:17 main.py
-rw-r--r-- 1 71970 197609 2239 3月 28 16:12 README.md
(week04)
71970@132 MINGW64 ~/repo/week04 (main)
$ cat emails.txt
to: <guofurong@126.com>
尊敬的郭芙蓉女士，您的会员资格即将到期，请及时续费。
---
to: <lvqinghou@126.com>
尊敬的吕轻侯先生，您的会员资格即将到期，请及时续费。
---
to: <baizhantang@163.com>
尊敬的白展堂先生，您的会员资格即将到期，请及时续费。
---
to: <lixiaulian@163.com>
尊敬的李秀莲先生，您的会员资格即将到期，请及时续费。
---
to: <tongxiangyu@163.com>
尊敬的佟湘玉女士，您的会员资格即将到期，请及时续费。
---
to: <zhuwushuang@163.com>
尊敬的祝无双女士，您的会员资格即将到期，请及时续费。
---
```

三、运用 pdb 检查程序内部运行

1. 运行 `python -m pdb main.py` (以调试模式 debug mode 启动 Python 解释器, 准备执行 `main.py` 代码)

```
(week04)
71970@13269 MINGW64 ~/repo/week04 (main)
$ rm emails.txt
(week04)
71970@13269 MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 71970 197609 204 3月 28 16:53 contacts.txt
-rw-r--r-- 1 71970 197609 72 3月 28 16:37 environment.yml
-rw-r--r-- 1 71970 197609 18805 3月 28 16:12 LICENSE
-rw-r--r-- 1 71970 197609 1022 3月 28 20:17 main.py
-rw-r--r-- 1 71970 197609 2239 3月 28 16:12 README.md
(week04)
71970@13269 MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\71970\repo\week04\main.py(1)<module>()
-> def process_contacts():
(Pdb)
```

2. pdb 提示符下的一些命令

① `→` 箭头: 即将运行但还未运行的代码

② 直接回车重复上一个命令; `q` 退出 pdb

③ `l (list)`: 列出当前行附近的代码 (默认 11 行)。

1. `(list .)`: 列出当前执行点周围的代码, 上下 5 行 (表示当前位置)。

11 `(longlist)`: 列出当前函数或块的完整代码。

130,50 `(list 30,50)`: 显示文件中第 30 行到第 50 行的代码片段。

④ `n (next)`: 执行当前行, 不进入函数调用 (直接跳过函数, 停在下一行)。

`s (step in)`: 执行当前行, 进入函数调用内部 (会跳转到被调用函数的首行)。如果语句是在定义函数而不是运行, 则效果和 `n` 一致。

⑤ `p (print)`: 打印变量或表达式的值 (适合简单输出)。不能查看未运行的

PS: `len`、`type` 这类是可以直接 `p` 打印而不需要先定义的 (内置)

`pp (pretty print)`: 格式化打印复杂数据结构 (如字典、列表等, 更易读)

```
(Pdb) pp contacts
[{'email': 'baizhantang@163.com', 'gender': '男', 'name': '白展堂'},
 {'email': 'tongxiangyu@163.com', 'gender': '女', 'name': '佟湘玉'},
 {'email': 'lvqinghou@126.com', 'gender': '男', 'name': '吕轻侯'},
 {'email': 'guofurong@126.com', 'gender': '女', 'name': '郭芙蓉'},
 {'email': 'lixiaolian@163.com', 'gender': '男', 'name': '李秀莲'},
 {'email': 'zhuwushuang@163.com', 'gender': '女', 'name': '祝无双'}]
```

四、Python 基础概念与对象检视

1. 安装 wat-inspector 软件包（修改 environment.yml 文件并在终端 conda env update）

```
(week04)
71970@132699zjq MINGW64 ~/repo/week04 (main)
$ conda list
# packages in environment at C:\Users\71970\anaconda3\envs\week04:
#
# Name                        Version      Build                Channel
bzip2                         1.0.8        h2466b09_7          conda-forge
ca-certificates               2025.1.31    h56e8100_0          conda-forge
libexpat                      2.6.4        he0c23c2_0          conda-forge
libffi                        3.4.6        h537db12_0          conda-forge
liblzma                       5.6.4        h2466b09_0          conda-forge
libsqlite                     3.49.1       h67fdade_2          conda-forge
libzlib                       1.3.1        h2466b09_2          conda-forge
openssl                       3.4.1        ha4e3fda_0          conda-forge
pip                           25.0.1       pyh8b19718_0        conda-forge
python                        3.12.9       h3f84c4b_1_cpython  conda-forge
setuptools                    75.8.2       pyhff2d567_0        conda-forge
tk                             8.6.13       h5226925_1          conda-forge
tzdata                        2025b        h78e105d_0          conda-forge
ucrt                          10.0.22621.0 h57928b3_1          conda-forge
vc                             14.3         hb6f10ac_24         conda-forge
vc14_runtime                  14.42.34438 hfd919c2_24         conda-forge
wat-inspector                  0.4.3        pyhff2d567_0        conda-forge
wheel                         0.45.1       pyhd8ed1ab_1        conda-forge
```

如何使用 wat-inspector（深度检查 python 对象）:

```
(week04)
71970@132699zjq MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\71970\repo\week04\main.py(1)<module>()
-> def read_contacts(filename):
(Pdb) import wat
(Pdb) wat
Try wat / object or wat.modifiers / object to inspect an object. Modifiers are:
  .short or .s to hide attributes (variables and methods)
  .dunder to print dunder attributes
  .code to print source code of a function, method or class
  .long to print non-abbreviated values and documentation
  .nodoc to hide documentation for functions and classes
  .caller to show how and where the inspection was called
  .all to include all information
  .ret to return the inspected object
  .str to return the output string instead of printing
  .gray to disable colorful output in the console
  .color to enforce colorful outputs in the console
Call wat.locals or wat() to inspect local variables.
Call wat.globals to inspect global variables.
(Pdb) wat()
Local variables:
  __builtins__: dict = {...
  __file__: pdb._ScriptTarget = 'C:\Users\71970\repo\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...
  __spec__: NoneType = None
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
```

2. Python 保留字 (reserved key words): 有特殊语法含义, 不能用作变量名、函数名或其他标识符

```
71970@132699zjq MINGW64 ~/repo/week04 (main)
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar 4 2025, 22:37:18) [MSC
v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not',
'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

3. 语句 (statement): 可以嵌套 (子语句)

表达式 (expression): 语句的组成部分, 也可以嵌套

4. 缩进 (indent): 被缩进 (四格) 的语句是上一级别的子语句

5.局部变量 (local variable)、全局变量 (global variable)、LEGB 规则:

1. 局部变量 (Local Variable)

- **定义位置:** 在 **函数内部** 定义 (包括参数)。
- **作用域:** 仅在 **定义它的函数内部** 有效, 函数外无法访问。
- **生命周期:** 函数调用时创建, 函数结束时销毁。
- **示例:**

```
python 复制  
  
def my_func():  
    x = 10 # 局部变量, 仅在 my_func 内有效  
    print(x) # 输出 10  
  
my_func()  
print(x) # 报错: NameError, x 未定义
```

2. 全局变量 (Global Variable)

- **定义位置:** 在 **函数外部** 定义 (通常是模块级)。
- **作用域:** 整个 **当前模块 (文件)** 均可访问。
- **生命周期:** 从定义开始, 直到程序结束或模块被卸载。
- **示例:**

```
python 复制  
  
y = 20 # 全局变量, 整个文件有效  
  
def my_func():  
    print(y) # 可以访问全局变量 y  
  
my_func() # 输出 20  
print(y) # 输出 20
```

LEGB 规则: Python 变量作用域解析顺序

LEGB 是 Python 查找变量名的优先级规则, 按以下顺序逐层搜索:

1. **L**ocal (局部) : 当前函数或方法内部的变量。
2. **E**nclosing (闭包) : 嵌套函数中, 外层函数的变量 (非全局) 。
3. **G**lobal (全局) : 当前模块 (文件) 顶层的变量。
4. **B**uilt-in (内置) : Python 内置关键字和函数 (如 `print`、`len`) 。

6.函数 (function) 的定义: `def`; 调用 (call): 通过 函数名 + 括号 执行函数, 传递参数

7.字面值 (literal) (字符串 (str)、整数 (int)、列表 (list)、字典 (dict)、元组 (tuple)):

直接写在代码中的常量值, 如"hello" (str)、42 (int)、[1,2] (list)、{"a":1} (dict)、(1,2) (tuple)

8.运算符 (operator): `==`, `()`, `if else` 等

9.形参 (parameter)、实参 (argument)、返回值 (return value):

函数定义时的变量 (形参), 调用时传入的值 (实参), 函数通过 `return` 返回的结果 (返回值)

10.对象 (object)、类型 (type)、属性 (attribute)、方法 (method): 可以用 `wat` 检查