第四周任务

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。

本周我们的目标是初步理解 Python 语言里 变量 (variable)、函数 (function)、对象 (object)、类型 (type)、属性 (attribute)、方法 (method)、调用 (call)、形参 (parameter)、实参 (argument)、返回值 (return value) 等等基本概念。我们要明白，*Python 编程本质上是拼接操纵各种对象* (胶水语言)。我们将运用 `pdb` **调试器** (debugger) 这个关键工具，结合案例来讲解这些概念。

1. Fork 第04周打卡 仓库至你的名下，然后将你名下的这个仓库 Clone 到你的本地计算机
2. 用 VS Code 打开项目目录，新建一个 `environment.yml` 文件，指定安装 Python 3.12，然后运行 `conda env create` 命令创建 Conda 环境
3. 新建一个 `contacts.txt` 文件，每行写一个联系人，每个联系人都包含姓名、性别、邮箱三个字段，用空格分隔，例如

   ```
   白展堂 男 baizhantang@163.com
   佟湘玉 女 tongxiangyu@163.com
   吕轻侯 男 lvqinghou@126.com
   郭芙蓉 女 guofurong@126.com
   李秀莲 男 lixiulian@163.com
   祝无双 女 zhuwushuang@163.com
   ```

   建议活学活用，改换其他例子 (甚至是一些极端情况) 测试程序的稳健性

4. 新建一个 `main.py` 文件，里面写 Python 代码，要求读取 `contacts.txt` 文件的内容，进行数据处理后，输出一个 `emails.txt` 文件，例如

   ```
   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: <lixiulian@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>
   ```

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）

5. 可以将以上 “任务要求” 的文本，复制粘贴到大模型 (比如豆包、DeepSeek) 里，请 AI 来帮助编写程序初稿
6. AI 回复的只是静态代码，而且可能含有错误，所以我们必须在 Conda 环境里运行代码，逐行调试，检查每一行代码的运行都符合我们的期望 (越是初学者越应该慢慢调试、检查、试验，借此学习)
    - 将大模型提供的代码复制粘贴进 `main.py` 文件，记得保存
    - 在 VS Code 扩展商店里安装 `Python` 扩展，使得在编写 `.py` 文件时能够显示和选择 Python 解释器 (需要绕过防火墙)
    - 在 VS Code 扩展商店里安装 `Ruff` 扩展，按照文档配置 `Ruff`，实现在保存 `.py` 文件时能够自动规范化 Python 代码

- 运行 `python -m pdb main.py` 命令 (作用是以调试模式 (debug mode) 启动 Python 解释器，准备执行 `main.py` 里的代码)
- 在 `(pdb)` 提示符下练习使用 `l` (显示代码)、`n` (执行当前行)、`p` (打印表达式)、`s` (步入调用)、`pp` (美观打印)、`c` (继续执行) 等命令 (参考文档)
- 在调试过程中，利用 `wat-inspector` (第三方软件包，需要安装) 检查 (inspect) 各种对象 (参考文档)
- 在调试过程中，观察代码逐步运行的效果，学习理解以下 Python **基本概念** (建议观看下面的录播讲解)
    - Python 语法保留字 (reserved key words)
    - 语句 (statement) 和表达式 (expression)
    - 缩进 (indent)
    - 局部变量 (local variable)、全局变量 (global variable)、LEGB 规则
    - 函数 (function) 的定义 (define) 和调用 (call)
    - 字面值 (literal) (字符串 (str)、整数 (int)、列表 (list)、字典 (dict)、元组 (tuple))
    - 运算符 (operator)
    - 形参 (parameter)、实参 (argument)、返回值 (return value)
    - 对象 (object)、类型 (type)、属性 (attribute)、方法 (method)

```
Think@LAPTOP-IVVORB8S MINGW64 ~/chao/week04 (main)
$ python -m pdb main.py
> c:\users\think\chao\week04\main.py(1)<module>()
-> def read_contacts(file_path):
(Pdb)
```

调试代码：加载这个 PDB 的模块，先要运行 PDB 这个模块里面编好了程序，这个程序就是所谓的调试器

```
 12
 13  -> def generate_emails(contacts):
 14         emails = []
 15         for name, gender, email in sorted(
 16             contacts, key=lambda x: (x[2].split("@")[1], x
```

箭头所指是即将运行但是还没运行的代码

命令解释：

1. l（显示代码）

含义：l 是 list 的缩写，用于显示当前执行位置附近的源代码。

作用：帮助你查看当前正在执行的代码上下文，了解程序的执行流程。默认情况下，它会显示当前行及其前后几行的代码。例如，当你使用 pdb 调试程序暂停在某一行时，输入 l 可以看到当前行以及周围的代码，方便你定位问题。

2. n（执行当前行）

含义：n 是 next 的缩写，用于执行当前行并前进到下一行。

作用：单步执行代码，每次执行一行。当你遇到一个函数调用时，使用 n 会直接执行完这个函数调用，而不会进入函数内部，继续执行下一行代码。这对于快速浏览代码执行流程很有用。

3. p（打印表达式）

含义：p 是 print 的缩写，用于打印指定表达式的值。

作用：在调试过程中，你可以使用 p 命令来查看变量的值、表达式的计算结果等。例如，如果你想知道某个变量 x 的值，在 pdb 提示符下输入 p x 即可打印出 x 的值。

4. s（步入调用）

含义：s 是 step 的缩写，用于执行当前行并步入函数调用内部。

作用：与 n 不同，当遇到函数调用时，s 会进入函数内部，允许你逐行调试函数内部的代码。这对于深入了解函数的执行逻辑、查找函数内部的问题非常有帮助。

5. pp（美观打印）

含义：pp 是 pretty print 的缩写，用于以更美观的格式打印复杂的数据结构，如列表、字典等。

作用：当你需要查看复杂的数据结构时，普通的 p 命令可能会输出混乱的结果。使用 pp 命令可以将数据结构以更易读的格式打印出来，方便你查看数据的内容。

6. c（继续执行）

含义：c 是 continue 的缩写，用于继续执行程序，直到下一个断点或程序结束。

作用：当你已经完成了对某个代码段的调试，想要让程序继续执行时，可以使用 c 命令。它会跳过 pdb 的单步调试模式，让程序正常运行，直到遇到下一个断点或程序结束。

```
(Pdb) l .
 19              email_text = f"to: <{email}>\n尊敬的 {name}{ti
tle}，您的会员资格即将到期，请及时续费。\n---"
 20              emails.append(email_text)
 21          return emails
 22
 23
 24  -> def write_emails(emails, output_file):
 25          try:
 26              with open(output_file, "w", encoding="utf-8")
as file:
 27                  file.write("\n".join(emails))
 28              print(f"邮件内容已成功写入 {output_file} 文件
。")
 29          except Exception as e:
(Pdb)
```

L .:这是箭头上下 5 行的意思

```
(Pdb) l 30,39
 30              print(f"错误：写入 {output_file} 文件时出错 -
{e}")
 31
 32
 33      if __name__ == "__main__":
 34          contacts = read_contacts("contacts.txt")
 35          if contacts:
 36              emails = generate_emails(contacts)
 37              write_emails(emails, "emails.txt")
[EOF]
(Pdb)
```

指定行数

自己练习一下这几个命令

python 保留字

粉红色字体的都是保留字：在 python 的语法上有特殊的含义，不能在别的地方乱用

Python 的 LEGB 规则