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 main.py 命令 (作用是启动 Python 解释器，执行 main.py 里的代码直至结束 (EOF) 或报错 (Exception))，检查运行结果是否符合预期

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) vs. 全局变量 (global variable)

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)

```
(base) Administrator@PC-20241018DCUL MINGW64 ~/repo
$ git clone https://gitcode.com/mutecamel/week04.git
Cloning into 'week04'...
remote: Enumerating objects: 5, done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 5 (from 1)
Unpacking objects: 100% (5/5), 8.43 KiB | 221.00 KiB/s, done.
```

```
(base) Administrator@PC-20241018DCUL MINGW64 ~/repo/week04 (main)
$ cat contacts.txt
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
(base) Administrator@PC-20241018DCUL MINGW64 ~/repo/week04 (main)
```

environment.yml U ● | contacts.txt U | main.py 5, U ×

main.py > ...

```python
def read_contacts(file_path):
    contacts = []
    try:
        with open(file_path, "r", encoding="utf-8") as file:
            for line in file:
                name, gender, email = line.strip().split()
                contacts.append((name, gender, email))
    except FileNotFoundError:
        print(f"错误: 文件 {file_path} 未找到。")
    return contacts


def generate_emails(contacts):
    emails = []
    for name, gender, email in sorted(
        contacts, key=lambda x: (x[2].split("@")[1], x[2].split("@")[0])
    ):
        title = "先生" if gender == "男" else "女士"
        email_text = f"to: <{email}>\n尊敬的{name}{title}，您的会员资格即将到期，请及时续费。\
        emails.append(email_text)
    return emails


def write_emails(emails, output_file):
    try:
        with open(output_file, "w", encoding="utf-8") as file:
            for email in emails:
                file.write(email + "\n")
    except Exception as e:
        print(f"写入文件时出错: {e}")


if __name__ == "__main__":
    contacts = read_contacts("contacts.txt")
    emails = generate_emails(contacts)
    write_emails(emails, "emails.txt")
```

```python
def generate_emails(contacts):
    emails = []
    for name, gender, email in sorted(
        contacts, key=lambda x: (x[2].split("@")[1], x[2].split("@")[0])
    ):
        title = "先生" if gender == "男" else "女士"
        email_text = f"to: <{email}>\n尊敬的{name}{title}，您的会员资格即将到期，
        emails.append(email_text)
    return emails


def write_emails(emails, output_file):
    try:
        with open(output_file, "w", encoding="utf-8") as file:
            for email in emails:
                file.write(email + "\n")
    except Exception as e:
        print(f"写入文件时出错：{e}")


if __name__ == "__main__":
    contacts = read_contacts("contacts.txt")
    emails = generate_emails(contacts)
    write_emails(emails, "emails.txt")
    print("邮件已成功生成到 emails.txt 文件中。")
(week04)
Administrator@PC-20241018DCUL MINGW64 ~/repo/week04 (main)
$ python main.py
邮件已成功生成到 emails.txt 文件中。
(week04)
```

```
Administrator@PC-20241018DCUL 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: <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>
尊敬的祝无双女士，您的会员资格即将到期，请及时续费。
---
(week04)
```

```
Administrator@PC-20241018DCUL MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\administrator\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
(Pdb) l
  1  -> def read_contacts(file_path):
  2         contacts = []
  3         try:
  4             with open(file_path, "r", encoding="utf-8") as file:
  5                 for line in file:
  6                     name, gender, email = line.strip().split()
  7                     contacts.append((name, gender, email))
  8         except FileNotFoundError:
  9             print(f"错误：文件 {file_path} 未找到。")
 10         return contacts
 11
(Pdb) n
> c:\users\administrator\repo\week04\main.py(13)<module>()
-> def generate_emails(contacts):
(Pdb) l
  8         except FileNotFoundError:
  9             print(f"错误：文件 {file_path} 未找到。")
 10         return contacts
 11
 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[2].split("@")[0])
 17         ):
 18             title = "先生" if gender == "男" else "女士"
(Pdb)
```

```
-> if __name__ == "__main__":
(Pdb) p sort_contacts
*** NameError: name 'sort_contacts' is not defined
(Pdb) p write_emails
<function write_emails at 0x000000000163F700>
(Pdb) l
 28                     file.write(email + "\n")
 29         except Exception as e:
 30             print(f"写入文件时出错：{e}")
 31
 32
 33  -> if __name__ == "__main__":
 34         contacts = read_contacts("contacts.txt")
 35         emails = generate_emails(contacts)
 36         write_emails(emails, "emails.txt")
 37         print("邮件已成功生成到 emails.txt 文件中。")
[EOF]
(Pdb) l .
 28                     file.write(email + "\n")
 29         except Exception as e:
 30             print(f"写入文件时出错：{e}")
 31
 32
 33  -> if __name__ == "__main__":
 34         contacts = read_contacts("contacts.txt")
 35         emails = generate_emails(contacts)
 36         write_emails(emails, "emails.txt")
 37         print("邮件已成功生成到 emails.txt 文件中。")
[EOF]
(Pdb) p __name__
'__main__'
(Pdb)
```

```
(Pdb) p len(contacts)
*** NameError: name 'contacts' is not defined
(Pdb) p __name__
'__main__'
(Pdb) p len
<built-in function len>
(Pdb) p type
<class 'type'>
(Pdb) n
--Return--
> <string>(1)<module>()->None
(Pdb)
Traceback (most recent call last):
  File "C:\Users\Administrator\anaconda3\envs\week04\lib\pdb.py", line 1726, in main
    pdb._runscript(mainpyfile)
  File "C:\Users\Administrator\anaconda3\envs\week04\lib\pdb.py", line 1586, in _run
    self.run(statement)
  File "C:\Users\Administrator\anaconda3\envs\week04\lib\bdb.py", line 580, in run
    exec(cmd, globals, locals)
  File "<string>", line 1, in <module>
  File "c:\users\administrator\repo\week04\main.py", line 34, in <module>
    contacts = read_contacts("contacts.txt")
  File "c:\users\administrator\repo\week04\main.py", line 6, in read_contacts
    name, gender, email = line.strip().split()
ValueError: not enough values to unpack (expected 3, got 0)
Uncaught exception. Entering post mortem debugging
Running 'cont' or 'step' will restart the program
> c:\users\administrator\repo\week04\main.py(6)read_contacts()
-> name, gender, email = line.strip().split()
```

```
(Pdb) l
  4             with open(file_path, "r", encoding="utf-8") as file:
  5                 for line in file:
  6  >>                 name, gender, email = line.strip().split()
  7                     contacts.append((name, gender, email))
  8         except FileNotFoundError:
  9  ->         print(f"错误：文件 {file_path} 未找到。")
 10         return contacts
 11
 12
 13     def generate_emails(contacts):
 14         emails = []
(Pdb) n
Post mortem debugger finished. The C:\Users\Administrator\repo\week04\main.py
> c:\users\administrator\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
(Pdb) l .
  1  -> def read_contacts(file_path):
  2         contacts = []
  3         try:
  4             with open(file_path, "r", encoding="utf-8") as file:
  5                 for line in file:
  6                     name, gender, email = line.strip().split()
  7                     contacts.append((name, gender, email))
  8         except FileNotFoundError:
  9             print(f"错误：文件 {file_path} 未找到。")
 10         return contacts
 11
(Pdb) p with
*** SyntaxError: unexpected EOF while parsing
(Pdb) p open
<built-in function open>
```

安装 what inspector：在 vs code 的 python3.12 week04 环境中添加依赖 what inspector，并通过 conda environment update 更新环境。该工具可深度检视 Python 对象。

Python 基础概念

语法保留字：在 Python 语法中有特殊含义的单词，如 try、with、as、for、return、def 等，不能用作变量名。可通过 import keyword 及 print(keyword.kwlist)列出所有保留字。

语句和表达式：语句类似语言中的一句话，是逻辑上完整的单元，如函数定义、try 语句、with 语句、赋值语句、for 循环语句、if 语句等，可嵌套子语句。表达式是构成语句的元素，比语句小，类似语言中的词，不能包含语句。例如 with 后的部分、变量名、函数调用、条件判断部分等都是表达式，表达式可嵌套，最终能计算出一个值。

缩进：在 Python 中，缩进用于界定子语句的边界，如函数定义、try 语句、with 语句等内部的子语句都需缩进。缩进严格要求对齐，与其他语言用大括号作为分隔符不同，缩进代表层级，对代码逻辑结构至关重要。

局部变量和全局变量：局部变量在函数内部定义，作用域仅限于函数内部，函数调用结束后，局部变量会被清理（内存回收）。可通过 what.locals 或 what()检查局部变量。全局变量在模块顶层定义（不缩进），在代码任何位置都可访问。变量查找遵循 LEGB 规则，即优先在局部作用域（L - local）查找，然后是闭包作用域（E - enclosing，函数嵌套时外层函数的局部变量），接着是全局作用域（G - global），最后是内置作用域（B - built - in），若都找不到则报错 Name error。

函数的定义和调用：函数定义使用 def 关键字，定义内部可嵌套定义。函数调用通过在函数名后加括号，并传入所需参数实现。若参数提供不足，函数调用会报错。

字面值：如字符串（包括普通字符串、f - string）、整数、空列表、字典（如{a: 1, b: 2}）、元组等都是字面值，是创建对象的一种方式。

运算符：包括比较运算符（如==）、赋值运算符（单个=用于赋值语句）、三目运算符（if else 形式，根据中间表达式的真假取左右表达式的值）、名称访问运算符（.，用于访问对象名称空间中的成员，如 file.write、contacts.append）、调用运算符（()，用于调用可调用对象，如函数）等。

行参和实参：函数定义时括号内的参数为形参（parameter），是占位符，在调用时传入具体值的参数为实参（argument），实参的值传递给形参，使函数能运行。

返回值：函数中 return 语句后面的值或表达式的计算结果为返回值。若函数没有 return 语句，则返回 None，None 是一个特殊对象，表示没有值。

对象类型属性方法：Python 中内存管理的所有东西都是对象，通过变量名可引用对象，对象无引用时内存会被回收。使用 what 工具可深度检查对象，如 what/emails 可查看列表对象 emails 的值、类型、长度、公开属性和方法等。对象都有类型，类型决定对象能做什么，属性是对象值方面的特点，方法是对象能执行的操作，通过调用对象的方法并传入参数可让对象执行相应操作。