

第四周金融编程与计算作业

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

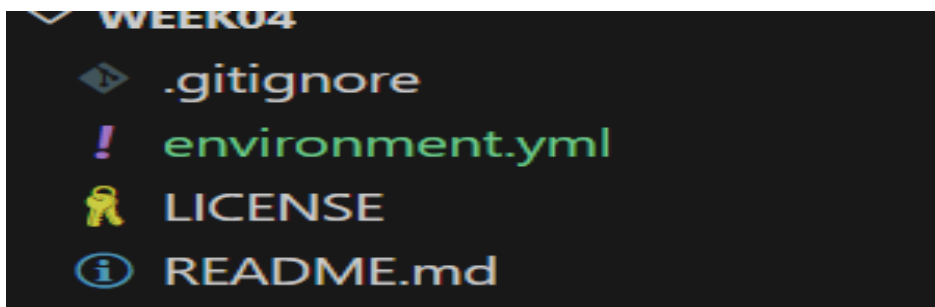
(1) 克隆仓库，查看远程仓库地址：

```
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~  
$ cd repo  
  
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo  
$ git clone https://gitcode.com/Typing_lqqq/week04.git  
Cloning into 'week04'...  
remote: Enumerating objects: 5, done.  
remote: Counting objects: 100% (5/5), done.  
remote: Compressing objects: 100% (5/5), done.  
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)  
Unpacking objects: 100% (5/5), 8.43 KiB | 454.00 KiB/s, done.  
  
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo  
$ cd week04/  
  
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)  
$ pwd  
/c/Users/86157/repo/week04  
  
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)  
$ git remote show origin  
* remote origin  
Fetch URL: https://gitcode.com/Typing_lqqq/week04.git  
Push URL: https://gitcode.com/Typing_lqqq/week04.git  
HEAD branch: main  
Remote branch:  
main tracked  
Local branch configured for 'git pull':  
main merges with remote main  
Local ref configured for 'git push':  
main pushes to main (up to date)
```

2 用 VSCode 打开项目目录，新建一个 environment.yml 文件，指定安装 Python3.12，然后运行 conda env create 命令创建 Conda 环境

(1) 复制之前的文件：

```
node.js v20.10.2  
  
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)  
$ cp D:/86157/rrpp/myproject/environment.yml ./
```



```
WEEK04  
├── .gitignore  
├── environment.yml  
├── LICENSE  
└── README.md
```

(2) 查看一下程序：

```
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
$ ls -l
total 25
-rw-r--r-- 1 86157 197609 76 3月 28 20:57 environment.yml
-rw-r--r-- 1 86157 197609 18805 3月 28 20:49 LICENSE
-rw-r--r-- 1 86157 197609 2239 3月 28 20:49 README.md

(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
$ cat environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
```

(3) 创建环境:

```
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
$ conda env create
D:\86157\anaconda3\Lib\argparse.py:2006: FutureWarning: 'remote_definition' is deprecated and
se 'conda env create --file=URL' instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
Channels:
  - conda-forge
  - https://repo.anaconda.com/pkgs/main
  - https://repo.anaconda.com/pkgs/r
  - https://repo.anaconda.com/pkgs/msys2
Platform: win-64
Collecting package metadata (repodata.json): -
```

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

(1) 新建一个 contacts.txt 文件

! environment.yml	U	1 白展堂 男 baizhantang@163.com
× contacts.txt	U	2 佟湘玉 女 tongxiangyu@163.com
main.py	U	3 吕轻侯 男 lvqinghou@126.com
WEEK04		4 郭芙蓉 女 guofurong@126.com
.gitignore		5 李秀莲 男 lixiulian@163.com
contacts.txt	U	6 祝无双 女 zhuwushuang@163.com
environment.yml	U	7
LICENSE		
main.py	U	

(2) cat: 读取文本

```
(base) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
$ cat contacts.txt

(base) 86157@LAPTOP-GMTRB58B 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) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
```

可以叠加读取:

```
(base) 86157@LAPTOP-GMTRB58B 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) 86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
$ cat contacts.txt environment.yml
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
name: week04
channels:
- conda-forge
dependencies:
- python=3.12
```

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

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）

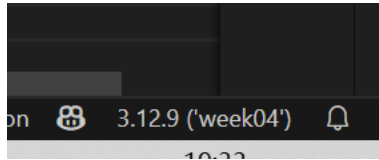
5 可以将以上“任务要求”的文本，复制粘贴到大模型（比如豆包、DeepSeek）里，请 AI 来帮助编写程序初稿

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）

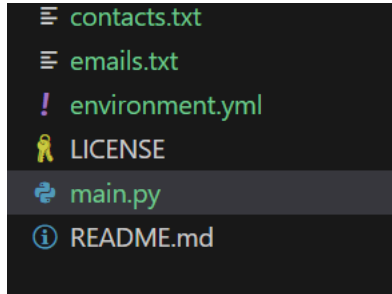
（1）AI 按 4 中的要求生成的代码：

```
main.py > ...
1  try:
2      # 读取 contacts.txt 文件
3      with open("contacts.txt", "r", encoding="utf-8") as file:
4          contacts = []
5          for line in file:
6              name, gender, email = line.strip().split()
7              contacts.append((name, gender, email))
8
9      # 按邮箱域名和用户名排序
10     contacts.sort(key=lambda x: (x[2].split("@")[1], x[2].split("@")[0]))
11
12     # 生成邮件内容
13     email_content = []
14     for name, gender, email in contacts:
15         title = "先生" if gender == "男" else "女士"
16         message = f"to: <{email}>\n尊敬的{name}{title}，您的会员资格即将到期，请及时续费。\\n-
17         email_content.append(message)
18
19     # 去除最后一个多余的分隔符
20     if email_content:
21         email_content[-1] = email_content[-1].rstrip("\\n-")
22
23     # 写入 emails.txt 文件
24     with open("emails.txt", "w", encoding="utf-8") as out_file:
25         out_file.write("\\n".join(email_content))
26
27 except FileNotFoundError:
28     print("未找到 contacts.txt 文件，请检查文件是否存在。")
29 except ValueError:
30     print("contacts.txt 文件格式有误，请确保每行包含姓名、性别和邮箱，用空格分隔。")
```

（2）配置版本



(3) 生成 emails.txt



(4) 运行 emails.txt

```
86157@LAPTOP-GMTRB58B 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: <lixiumian@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)
```

(5) 调试器

```
86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\86157\repo\week04\main.py(1)<module>()
-> try:
(Pdb) █
```

常用命令：

l：显示代码，查看代码运行到哪
箭头指的是即将运行还没运行的代码

```

(Pdb) l
1  -> try:
2      # 读取 contacts.txt 文件
3      with open("contacts.txt", "r", encoding="utf-8") as file:
4          contacts = []
5          for line in file:
6              name, gender, email = line.strip().split()
7              contacts.append((name, gender, email))
8
9      # 按邮箱域名和用户名排序
10     contacts.sort(key=lambda x: (x[2].split("@")[1], x[2].split("@")[0]))
11

```

n: 执行当前行

```

(Pdb) n
> c:\users\86157\repo\week04\main.py(3)<module>()
-> with open("contacts.txt", "r", encoding="utf-8") as file:
(Pdb) l
1      try:

```

运行完全部

```

(Pdb) n
--Return--
> c:\users\86157\repo\week04\main.py(24)<module>()->None
-> with open("emails.txt", "w", encoding="utf-8") as out_file:
(Pdb) n
Return

```

p: 打印表达式

```

(Pdb) p contacts
[]

```

```

(Pdb) p title
'女士'

```

```

(Pdb) p email_content
['to: <guofurong@126.com>\n尊敬的郭芙蓉女士，您的会员资格即将到期，请及时续费。\\n---']
(Pdb) p email_content

```

s: 步入调用

```

(Pdb) s
--Call--
> <frozen codecs>(319).decode()

```

l.: 空格上下加五行

```

-> email_content.append(message)
(Pdb) l
12     # 生成邮件内容
13     email_content = []
14     for name, gender, email in contacts:
15         title = "先生" if gender == "男" else "女士"
16         message = f"to: <{email}>\n尊敬的{name}{title}, 您的会员资格即将到期, 请及时
续费。 \n---"
17     -> email_content.append(message)
18
19     # 去除最后一个多余的分隔符
20     if email_content:
21         email_content[-1] = email_content[-1].rstrip("---")
22

```

pp : 美观打印

```

(Pdb) pp message
'to: <guofurong@126.com>\n尊敬的郭芙蓉女士, 您的会员资格即将到期, 请及时续费。 \n---'
(Pdb)

```

c : 继续执行

```

(Pdb) c
The program finished and will be restarted
> c:\users\86157\repo\week04\main.py(1)<module>()
-> try:

```

q:退出

```

(Pdb) q
(week04)
86157@LAPTOP-GMTRB58B MINGW64 ~/repo/week04 (main)

```

经过一系列调试, emails.txt 出现:

<pre> WEEK04 ├── .gitignore ├── contacts.txt ├── emails.txt ├── environment.yml ├── LICENSE ├── main.py └── README.md </pre>	<pre> 1 to: <guofurong@126.com> 2 尊敬的郭芙蓉女士, 您的会员资格即将到期, 请及时续费。 3 --- 4 to: <lvqinghou@126.com> 5 尊敬的吕轻侯先生, 您的会员资格即将到期, 请及时续费。 6 --- 7 to: <baizhantang@163.com> 8 尊敬的白展堂先生, 您的会员资格即将到期, 请及时续费。 9 --- 10 to: <lixjulian@163.com> 11 尊敬的李秀莲先生, 您的会员资格即将到期, 请及时续费。 12 --- 13 to: <tongxiangyu@163.com> 14 尊敬的佟湘玉女士, 您的会员资格即将到期, 请及时续费。 15 --- 16 to: <zhuwushuang@163.com> 17 尊敬的祝无双女士, 您的会员资格即将到期, 请及时续费。 18 </pre>
--	---

(6) 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)

```

1  name: week04
2  ∨ channels:
3      - conda-forge
4  ∨ dependencies:
5      - python=3.12
6      - wat-inspector
7

```

(7) 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

①Python 语法保留字(reserved key words): 被 Python 语言本身占用，有特殊含义的单词，不能用作普通标识符（如变量名、函数名等）

举例：if 用于条件判断、for 用于循环、def 用于定义函数等。

```

try:
    # 读取 c
    with open(
        cont
        for

```

```

# 写入 e
with open
    out

except File
    print("
except Value
    print("

```

②语句(statement)和表达式(expression): 语句可以包含表达式，表达式不可以包含依据

语句：是 Python 中执行某个操作的指令，能改变程序状态或执行特定任务。如 print("Hello") 是输出语句，if x > 5: 是条件判断语句。

表达式：由操作数和运算符组成，能计算出一个值。如 3 + 5 是算术表达式，计算结果为 8；"abc" + "def" 是字符串拼接表达式，结果为"abcdef"。

```

for line in file:
    name, gender, email = line.strip().split()
    contacts.append((name, gender, email))

```

表达式：

```

l = line.strip().split()

contacts.append((name, gender, email))

```

③缩进(indent): Python 通过缩进表示代码块，相同缩进的代码属于同一个代码块。缩进是强制的，是 Python 语法风格的重要部分。

严格的对齐



④局部变量 (local variable)、全局变量 (global variable)、LEGB 规则
局部变量：在函数内部定义的变量，只在函数内部有效，函数执行结束后，局部变量占用的内存空间会被释放。

全局变量：在函数外部（模块顶层）定义的变量，在整个模块中都可访问。

```
(Pdb) wat()
Local variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\86157\repo\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __spec__: NoneType = None
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>

(Pdb) p __file__
'C:\\Users\\86157\\repo\\week04\\main.py'

Global variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\86157\repo\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __spec__: NoneType = None
  contacts: list = [...]
  email: str = 'guofurong@126.com'
  email_content: list = []
  file: _io.TextIOWrapper = <_io.TextIOWrapper name='contacts.txt' mode='r' encoding='utf-8'>
  gender: str = '女'
  line: str = '祝无双 女 zhuwushuang@163.com...'
  name: str = '郭芙蓉'
  title: str = '女士'
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
```

LEGB 规则：LEGB 规则规定了 Python 在查找变量时的顺序，即先在局部作用域查找，接着是闭包作用域、全局作用域，最后是内置作用域。这个规则有助于你理解变量在不同作用域中的可见性与访问权限。

Local（局部作用域）：这是指在函数或者类方法内部所定义的变量。当你在函数内部使用一个变量时，Python 首先会在该函数的局部作用域里查找这个变量。

Enclosing（闭包作用域）：闭包作用域是在嵌套函数里，外层函数的作用域。当内部函数引用了外层函数的变量时，就会形成闭包。Python 在查找变量时，若在局部作用域中未找到，就会去闭包作用域里查找。

Global（全局作用域）：全局作用域是指在模块层面所定义的变量。若在局部作用域和闭包作用域中都没有找到变量，Python 就会去全局作用域里查找。

Built-in（内置作用域）：内置作用域是 Python 内置的一些函数和变量所在的作用域，像 print、len 等。若在前面三个作用域中都没有找到变量，Python 就会去内置作用域里查找。

⑤函数 (function) 的定义 (define) 和调用 (call)

定义：使用 def 关键字来定义函数，指定函数名、参数列表和函数体。如 def add(a, b):return a + b 定义了一个名为 add 的函数，接收两个参数并返回它们的和。

调用：使用函数名和参数列表来执行函数，如 result = add(3, 5) 调用 add

函数并将结果赋值给 result。

⑥字面值 (literal) (字符串 (str)、整数 (int)、列表 (list)、字典 (dict)、元组 (tuple))

定义：在代码中直接表示值的符号。

字符串 (str)：用单引号或双引号括起来的文本，如 "hello" 。

整数 (int)：表示整数数值，如 5 。

列表 (list)：用方括号括起来的有序元素集合，如 [1, 2, 3] 。

字典 (dict)：用花括号括起来的键值对集合，如 {"name": "Alice", "age": 25} 。

元组 (tuple)：用圆括号括起来的有序不可变元素集合，如 (1, "a") 。

```
="utf-8")
```

```
"先生"
```

⑦运算符 (operator)

定义：用于执行特定运算的符号，对操作数进行操作并返回结果。

算术运算符：+ (加法)、- (减法)、* (乘法) 等。

比较运算符：> (大于)、< (小于)、== (等于) 等。

逻辑运算符：and (与)、or (或)、not (非)。

```
== "
```

```
(x[2].split("@")[1], x[2].split("@")[0]))
```

名称访问运算符：

```
e.wri
```

⑧形参 (parameter)、实参 (argument)、返回值 (return value)

形参：在函数定义时括号内列出的变量名，用于接收调用函数时传入的值。如 def greet(name): 中的 name 是形参。

实参：在函数调用时实际传递给函数的值。如 greet("Bob") 中 "Bob" 是实参。

返回值：函数执行完成后返回给调用者的值，使用 return 语句指定。如 def square(x): return x * x 中，函数返回 x 的平方值。

⑨对象 (object)、类型 (type)、属性 (attribute)、方法 (method)

对象：Python 中一切皆是对象，是类的实例，具有数据和行为。如 my_list = [1, 2, 3]，my_list 就是列表类型的对象。

类型：用于标识对象所属类别，如 int 类型、str 类型等。可以用 type() 函数查看对象类型，如 type(5) 返回 <class 'int'> 。

属性：对象所拥有的数据成员。如自定义类的实例可以有各种属性，像 class Person: def __init__(self, name): self.name = name，name 就是 Person 类实例的属性。

方法：属于对象的函数，用于定义对象的行为。如列表对象的 append 方法，my_list.append(4) 可以向列表中添加元素 。

```
(Pdb) wat / email_content
```

值:

```
value: [  
  '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: <lixjulian@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。  
  ',  
>
```

```
type: list  
len: 6  
  
Public attributes:  
def append(object, /) # Append object to the end of the list.  
def clear() # Remove all items from list.  
def copy() # Return a shallow copy of the list.  
def count(value, /) # Return number of occurrences of value.  
def extend(iterable, /) # Extend list by appending elements from the iterable.  
def index(value, start=0, stop=9223372036854775807, /) # Return first index of value....  
def insert(index, object, /) # Insert object before index.  
def pop(index=-1, /) # Remove and return item at index (default last)....  
def remove(value, /) # Remove first occurrence of value....  
def reverse() # Reverse *IN PLACE*.  
def sort(*, key=None, reverse=False) # Sort the list in ascending order and return None....
```

```
(Pdb) p contacts.append  
<built-in method append of list object at 0x000001CA60951200>  
(Pdb) p contacts.copy  
<built-in method copy of list object at 0x000001CA60951200>  
(Pdb) █
```

```
(Pdb) wat / contacts.append  
  
value: <built-in method append of list object at 0x000001CA60951200>  
type: builtin_function_or_method  
signature: def append(object, /)  
"""Append object to the end of the list."""
```