【week04 克隆到本地 repo 里】

```
(base) 86139@LAPTOP-J15OR7EU MINGW64 ~
$ cd repo

(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo
$ git clone https://gitcode.com/Vicky_1057/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 | 136.00 KiB/s, done.
```

【CTRL E 光标跳到最后
CTRL A 跳到最开始】

【VS Code 创建 environment.yml 文件】

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

【从 myproject 复制到 week04】

```
(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week04 (main)
$ cp ../myproject/environment.yml ./
```

【新建 contact.txt 文件】

```
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
```

```
(base) 86139@LAPTOP-J15OR7EU 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
```

【激活 week04】

```
(base) 86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week04 (main)
$ conda activate week04
```

【豆包 AI 生成代码】

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：未找到文件 {file_path}")
    return contacts


def generate_emails(contacts):
    emails = []
    for name, gender, email in contacts:
        title = "先生" if gender == "男" else "女士"
        email_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。\n---"
        emails.append(email_content)
    return emails


def sort_contacts(contacts):
    return sorted(contacts, key=lambda x: (x[2].split('@')[1], x[2].split('@')[0]))


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')
    sorted_contacts = sort_contacts(contacts)
    emails = generate_emails(sorted_contacts)
    write_emails(emails, 'emails.txt')
```

【ruff 设定自动保存和格式】

C: > Users > 86139 > AppData > Roaming > Code > User > {} settings.json > {} [python] > {} editor.codeActionsOnSave

```jsonc
{
    "workbench.colorTheme": "Monokai",
    "workbench.startupEditor": "none",
    "[python]": {
        "editor.formatOnSave": true,
        "editor.codeActionsOnSave": {
            "source.fixAll": "explicit",
            "source.organizeImports": "explicit"
        },
        "editor.defaultFormatter": "charliermarsh.ruff",
    },
    "notebook.formatOnSave.enabled": true,
    "notebook.codeActionsOnSave": {
        "notebook.source.fixAll": "explicit",
        "notebook.source.organizeImports": "explicit"
    },
    "ruff.nativeServer": "on",
    "editor.largeFileOptimizations": false,
    "python.condaPath": "C:\\Users\\86139\\anaconda3\\python.exe",
    "files.associations":{
        "*.py":"python"
    },
    "terminal.integrated.defaultProfile.windows": "Command Prompt",
    "python.defaultInterpreterPath": "C:\\Users\\86139\\anaconda3\\python.exe",
    "python.interpreter.infoVisibility": "always",
    "editor.accessibilitySupport": "off",
    "explorer.confirmDelete": false,
}
```

【python 运行 main.py】

```
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week04 (main)
$ python main.py
```

【生成 email.txt】

```
(week04)
86139@LAPTOP-J15OR7EU 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。
---
```

【先删除之前生成的 emails.txt】

```
(week04)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week04 (main)
$ rm emails.txt
```

【运用 pdb 检查程序的内部运行】python -m pdb main.py

```
(week04)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\86139\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
```

l (显示代码)

```
(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：未找到文件 {file_path}")
 10         return contacts
```

```
(Pdb) h l
      Usage: l(ist) [first[, last] | .]

      List source code for the current file.  Without arguments,
      list 11 lines around the current line or continue the previous
      listing.  With . as argument, list 11 lines around the current
      line.  With one argument, list 11 lines starting at that line.
      With two arguments, list the given range; if the second
      argument is less than the first, it is a count.

      The current line in the current frame is indicated by "->".
      If an exception is being debugged, the line where the
      exception was originally raised or propagated is indicated by
      ">>", if it differs from the current line.
```

【help】

```
(Pdb) l 1,5
  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:
```

【1，5 行】

```
(Pdb) l .
 17             email_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。\n---"
 18             emails.append(email_content)
 19         return emails
 20
 21
 22  -> def sort_contacts(contacts):
 23         return sorted(contacts, key=lambda x: (x[2].split("@")[1], x[2].split("@")[0]))
 24
 25
 26     def write_emails(emails, output_file):
 27         try:
```

【上下五行】

n (执行当前行)

```
(Pdb) n
> c:\users\86139\repo\week04\main.py(13)<module>()
-> def generate_emails(contacts):
```

p (打印表达式)

```
(Pdb) p read_contacts
<function read_contacts at 0x0000029123DD7A60>
```

s (步入调用)

```
(Pdb) s
> c:\users\86139\repo\week04\main.py(22)<module>()
-> def sort_contacts(contacts):
```

pp (美观打印)

```
(Pdb) p contacts
[('白展堂', '男', 'baizhantang@163.com'), ('佟湘玉', '女', 'tongxia
ngyu@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')]
(Pdb) pp contacts
[('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')]
```

```
(Pdb) p type(contacts)
<class 'list'>
(Pdb) p len(contacts)
6
(Pdb) q
(week04)
86139@LAPTOP-J15OR7EU MINGW64 ~/repo/week04 (main)
$
```

【q 退出】

c (继续执行)

```
(Pdb) c
The program finished and will be restarted
> c:\users\86139\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
```

Python 语法核心概念整理

1. 保留字 (Reserved Keywords)

定义：Python 内置的特殊单词，具有固定语法含义，不可用作变量名。

常见保留字： if, else, for, return, try, except, def, with, in, lambda, None

特征：在代码中通常显示为红色（VS Code 默认配色）。

2. 语句 (Statement) 与表达式 (Expression)

语句： 逻辑完整的代码行，执行特定操作（如赋值、循环）。

```
a = 10  # 赋值语句
for i in range(5): print(i)  # 循环语句
if condition: action()  # 条件语句
```

表达式： - 可计算为一个值的代码片段，是语句的组成部分。

```
email.split('@')  # 方法调用表达式
f"尊敬的{name}"  # F-string 表达式
a + b * c  # 算术表达式
```

3. 缩进 (Indentation)

作用：替代 `{}` 界定代码块（如函数体、循环体）。

规则：同一层级的代码必须严格对齐（如 4 个空格或 1 个制表符）。缩进错误会导致语法错误

```
def greet():
    print("Hello")  # 缩进表示属于函数体
    if True:
        print("World")  # 更深缩进表示嵌套语句
```

4. 局部变量：在函数内部定义，仅在函数调用期间有效。

```
def func():
    localVar = 10  # 局部变量
```

全局变量：在模块顶层定义，可在任意位置访问。

```
globalVar = 20  # 全局变量
def func():
    print(globalVar)  # 可访问全局变量
```

LEGB 规则（变量查找顺序）：

1. Local：当前函数内的局部作用域。
2. Enclosing：嵌套函数的外层作用域。

3. Global：模块顶层的全局作用域。
4. Built-in：Python 内置函数/变量（如 `print`, `sum`）。

5. 函数：使用 `def` 关键字，参数为形参（抽象占位符）。

```
def add(a, b):  # 形参 a, b
    return a + b
```

调用：传递实参（具体值），返回值可赋值给变量。

```
result = add(3, 5)  # 实参 3, 5
print(result)  # 输出 8
```

6. 字面值 ：直接表示数据的形式，无需计算。

常见类型：

字符串 (str)：`"Hello"`
整数 (int)：`42`
列表 (list)：`[1, 2, 3]`
字典 (dict)：`{"a": 1, "b": 2}`
元组 (tuple)：`(1, "apple")`

7. 运算符 (Operator)

赋值运算符：`=`
比较运算符：`==`, `>`, `<`
三目运算符：`expr1 if condition else expr2`
名称访问运算符：`.`（如 `file.write`）
调用运算符：`()`（如 `split('@')`）

```
a = 10  # 赋值
is_positive = a > 0  # 比较
result = a if a > 0 else -a  # 三目运算符
```

8. 形参：函数定义时的占位符。

```
def greet(name):  # 形参 name
    print(f"Hello, {name}!")
```

实参：函数调用时传递的具体值。

```
greet("Alice")  # 实参 "Alice"
```

返回值：函数执行后返回的结果，无返回值时默认返回 `None`。

```
def add(a, b):
    return a + b  # 返回值为 a + b
```

9. 对象：Python 中一切皆为对象（如整数、字符串、函数）。

类型：对象的类别，可通过 `type()` 查看。

```
num = 10
print(type(num))  # 输出 <class 'int'>
```

属性：对象的数据特征（如文件编码）。

```
file = open("test.txt", "r")
print(file.encoding)  # 输出 'utf-8'
```

方法：对象的行为（如列表的 `append`）。

```
fruits = ["apple"]
fruits.append("banana")  # 调用方法
```

总结

保留字：Python 内置的关键字，不可用作变量名。
语句与表达式：语句是完整操作，表达式是语句的组成部分。
缩进：Python 强制使用缩进界定代码块。
变量作用域：遵循 LEGB 规则，从局部到全局查找变量。
函数：通过形参接收输入，通过返回值输出结果。
对象模型：Python 的核心，所有数据均为对象，通过属性和方法交互。