

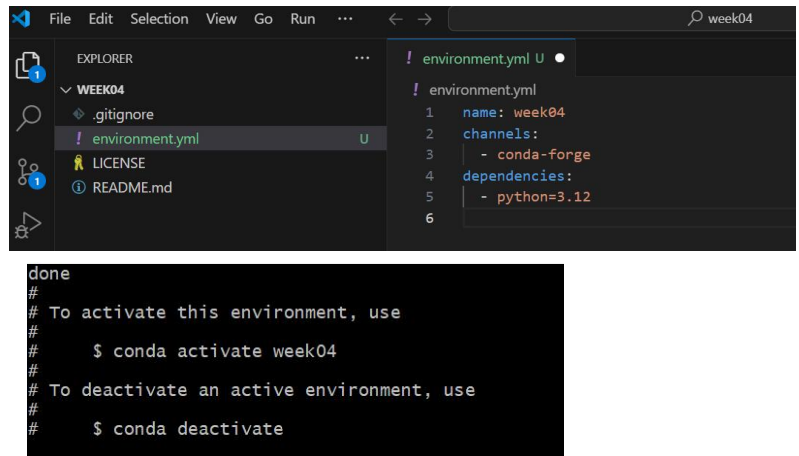
1. Fork 第 04 周打卡 仓库至你的名下, 然后将你名下的这个仓库 Clone 到你的本地计算机

```
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~
$ cd repo
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo
$ ls -l
ls: cannot access '-l': No such file or directory
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo
$ ls -l
total 17
drwxr-xr-x 1 LENOVO 197121 0 Mar 20 22:48 myproject/
drwxr-xr-x 1 LENOVO 197121 0 Mar 16 22:52 mywork/
drwxr-xr-x 1 LENOVO 197121 0 Mar 19 23:41 prj1/
-rw-r--r-- 1 LENOVO 197121 316 Mar 11 15:47 script1.py
drwxr-xr-x 1 LENOVO 197121 0 Mar 16 23:22 week01/
drwxr-xr-x 1 LENOVO 197121 0 Mar 16 23:22 week02/
drwxr-xr-x 1 LENOVO 197121 0 Mar 24 15:10 week03/
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo
$ git clone https://gitcode.com/damchun/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 | 253.00 KiB/s, done.
```

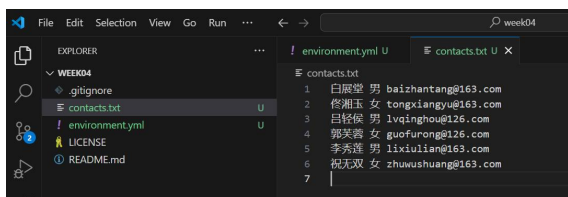
```
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo
$ git clone https://gitcode.com/damchun/week04.git
Cloning into 'week04'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (5/5), 8.43 KiB | 253.00 KiB/s, done.
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo
$ cd week04/
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ pwd
c:/Users/LENOVO/repo/week04
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ git remote show origin
remote origin
Fetch URL: https://gitcode.com/damchun/week04.git
Push URL: https://gitcode.com/damchun/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 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

```
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ ls -l ./myproject
bash: ls-l: command not found
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ ls -l ./myproject
total 5
-rw-r--r-- 1 LENOVO 197121 87 Mar 20 22:17 environment.yml
-rw-r--r-- 1 LENOVO 197121 853 Mar 20 23:19 main.py
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ cat ./myproject/environment.yml
name: myproject
channels:
  - conda-forge
dependencies:
  - python=3.12
  - pandas
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ cp ./myproject/environment.yml .
(base) LENOVOBLAPTOP-STLH7AK0 MINGW64 ~/repo/week04 (main)
$ ls -l
total 25
-rw-r--r-- 1 LENOVO 197121 18805 Mar 28 14:16 LICENSE
-rw-r--r-- 1 LENOVO 197121 2239 Mar 28 14:16 README.md
-rw-r--r-- 1 LENOVO 197121 87 Mar 28 14:26 environment.yml
```



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



```
(base) LENOVOBLAPTOP-STLH7AK0 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) LENOVOBLAPTOP-STLH7AK0 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
(base) LENOVOBLAPTOP-STLH7AK0 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

```
1 # main.py
2 # 读取 contacts.txt 文件内容，并处理
3
4 # 打开文件
5 with open('contacts.txt', 'r', encoding='utf-8') as f:
6     contacts = [line.strip().split() for line in f.readlines()]
7
8 # 处理数据
9 processed = []
10 for name, gender, email in contacts:
11     username, domain = email.split('@')
12     processed.append({
13         "name": name,
14         "gender": gender,
15         "email": email,
16         "domain": domain,
17         "username": username,
18     })
19
20 # 排序：按照 domain 排序
21 processed.sort(key=lambda x: (x["domain"], x["username"]))
22
23 # 输出结果
24 emails = []
25 for contact in processed:
26     title = "发件人: %s" % contact["name"]
27     line = f"<{contact['email']}> {contact['name']} {contact['domain']} {contact['username']}"
28     emails.append(line)
29
30 # 写入文件
31 with open('emails.txt', 'w', encoding='utf-8') as f:
32     f.write('\n'.join(emails))
33
34 if __name__ == '__main__':
35     main()
```

```
LENOVO@LAPTOP-STLH7AKO 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
(week04)
LENOVO@LAPTOP-STLH7AKO MINGW64 ~/repo/week04 (main)
$ python main.py
(week04)
LENOVO@LAPTOP-STLH7AKO MINGW64 ~/repo/week04 (main)
$ ls -l
total 31
-rw-r--r-- 1 LENOVO 197121 18805 Mar 28 14:16 LICENSE
-rw-r--r-- 1 LENOVO 197121 2239 Mar 28 14:16 README.md
-rw-r--r-- 1 LENOVO 197121 204 Mar 28 14:37 contacts.txt
-rw-r--r-- 1 LENOVO 197121 664 Mar 28 15:30 emails.txt
-rw-r--r-- 1 LENOVO 197121 76 Mar 28 14:29 environment.yml
-rw-r--r-- 1 LENOVO 197121 1194 Mar 28 15:51 main.py
(week04)
LENOVO@LAPTOP-STLH7AKO 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)
```

```
LENOVO@LAPTOP-STLH7AKO MINGW64 ~/repo/week04 (main)
$ rm emails.txt
(week04)
LENOVO@LAPTOP-STLH7AKO MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 LENOVO 197121 18805 Mar 28 14:16 LICENSE
-rw-r--r-- 1 LENOVO 197121 2239 Mar 28 14:16 README.md
-rw-r--r-- 1 LENOVO 197121 204 Mar 28 14:37 contacts.txt
-rw-r--r-- 1 LENOVO 197121 76 Mar 28 14:29 environment.yml
-rw-r--r-- 1 LENOVO 197121 1194 Mar 28 15:51 main.py
(week04)
LENOVO@LAPTOP-STLH7AKO MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\lenovo\repo\week04\main.py(1) <module>()
-> def process_contacts():
(Pdb) |
```

在 Python 调试器(pdb)中，`n`和`s`是两个最常用的单步执行命令，它们的区别可以用一个简单的比喻来解释：

`n` (next) - 下一步

- ****通俗解释****：像"快进"一样执行当前行代码，****不进入****函数内部
- ****场景****：当你只想看当前函数的执行过程，不想深入被调用的函数细节时
- ****示例****：

```
python
def greet(name):
    print(f'Hello, {name}') # 用 n 会直接执行完这行，不会进入 print 函数内部

greet("Alice") # 用 n 会直接执行完整个函数调用
'''
```

`s` (step) - 步入

- ****通俗解释****: 像"显微镜"一样**进入**当前行调用的函数内部
- ****场景****: 当你想深入查看某个函数的具体实现时
- ****示例****:

```
```python
def greet(name):
 print(f'Hello, {name}') # 用 s 会进入 print 函数的实现

greet("Alice") # 用 s 会进入 greet 函数内部
```
```

实际调试演示

假设调试以下代码:

```
```python
def cook():
 print("加热食材") # 断点停在这里
 add_seasoning()

def add_seasoning():
 print("加盐")
 print("加胡椒")
```
```

cook()

```

在 pdb 中:

1. 使用 `n` 时:

- 会直接执行完 `print("加热食材")` 和 `add\_seasoning()` 两行
- 看不到 `add\_seasoning` 函数内部的执行过程

2. 使用 `s` 时:

- 执行 `print("加热食材")` 时会进入 print 函数的底层实现
- 执行到 `add\_seasoning()` 时会进入该函数内部, 看到 `加盐` 和 `加胡椒` 的执行

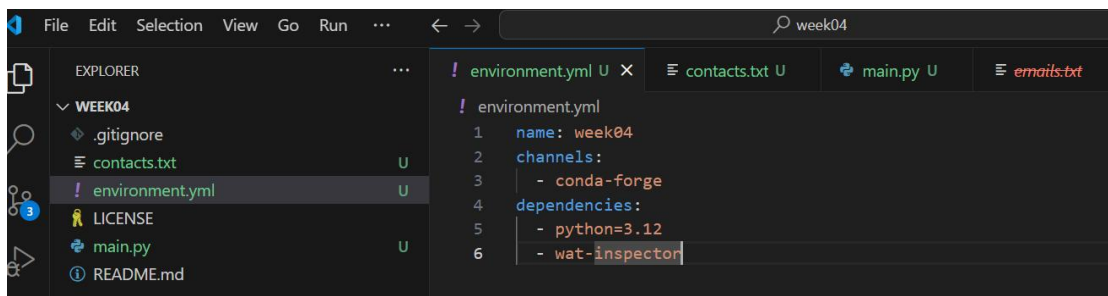
过程

### ### 记忆技巧

- `n` 像"鸟瞰"(只看表面)
- `s` 像"潜水"(深入内部)
- 当不确定用哪个时, 先用`s`进入函数, 如果发现不需要深入, 再用`c`跳出继续执行

这两个命令配合使用, 可以像控制电影播放一样精确控制调试过程: 用`s`进入细节镜头, 用`n`快速跳过已知正常的片段。

### q 退掉



```
(Pdb) wat
*** NameError: name 'wat' is not defined
(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
.nodocs 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\LENOVO\repo\week04\main.py'
 __name__: str = '__main__'
 __pdb_convenience_variables__: dict = {...
 __spec__: NoneType = None
 wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb) |
```

### Python 变量寻找时的 LEGB 规则