

1. Fork 第 04 周打卡 仓库， Clone 到本地计算机

点进仓库——fork——创建 fork 项目——clone——复制克隆到本地的网址，到终端

克隆到本地

```
git clone https://gitcode.com/luluchris/week04.git
```

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo
$ git clone https://gitcode.com/luluchris/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 | 261.00 KiB/s, done.
```

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo
$ cd week04/

(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week04 (main)
$ pwd
/c/Users/17437/repo/week04

(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week04 (main)
$ git remote show origin
* remote origin
  Fetch URL: https://gitcode.com/luluchris/week04.git
  Push URL: https://gitcode.com/luluchris/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

从 code 手动选择打开文件夹

快捷键: ctrl e 跳到最后; ctrl a 跳到最开始

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week04 (main)
$ cp ../myproject/environment.yml ./

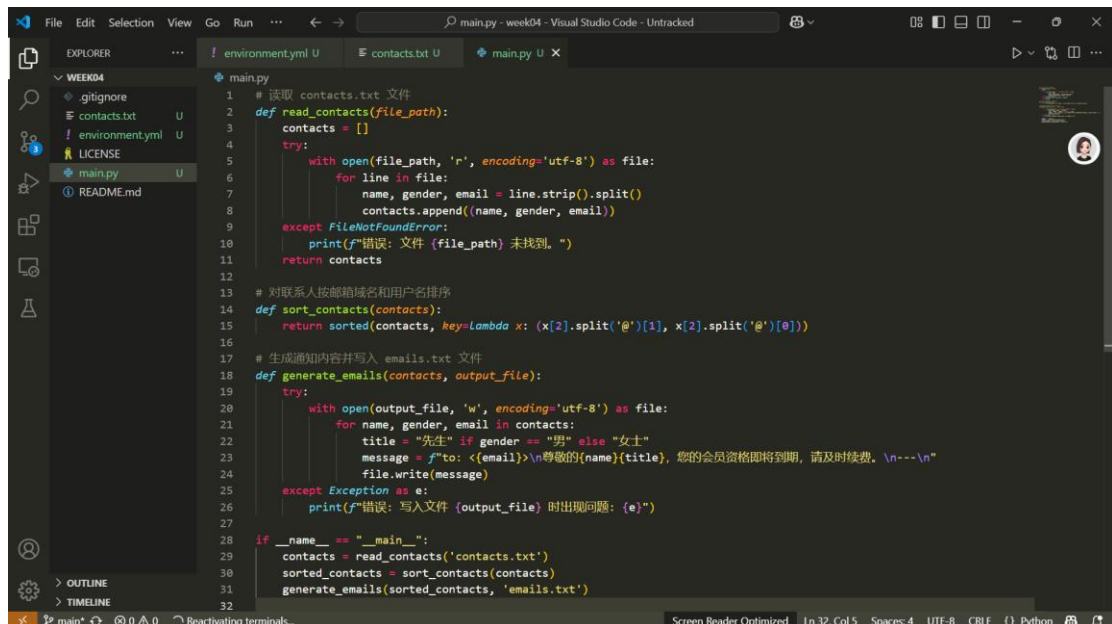
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week04 (main)
$ ls -l
total 25
-rw-r--r-- 1 17437 197609 87 4月 7 19:25 environment.yml
-rw-r--r-- 1 17437 197609 18805 4月 7 19:15 LICENSE
-rw-r--r-- 1 17437 197609 2239 4月 7 19:15 README.md
```

3. 新建一个 contacts.txt 文件

```
(base) 17437@DESKTOP-34KCETK 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
```

4. 新建一个 main.py 文件

5. 请 AI 来帮助编写程序初稿



```
1 # 读取 contacts.txt 文件
2 def read_contacts(file_path):
3     contacts = []
4     try:
5         with open(file_path, 'r', encoding='utf-8') as file:
6             for line in file:
7                 name, gender, email = line.strip().split()
8                 contacts.append((name, gender, email))
9     except FileNotFoundError:
10        print(f"错误: 文件 {file_path} 未找到。")
11    return contacts
12
13 # 对联系人按邮箱域名和用户名排序
14 def sort_contacts(contacts):
15    return sorted(contacts, key=lambda x: (x[2].split('@')[1], x[2].split('@')[0]))
16
17 # 生成通知内容并写入 emails.txt 文件
18 def generate_emails(contacts, output_file):
19    try:
20        with open(output_file, 'w', encoding='utf-8') as file:
21            for name, gender, email in contacts:
22                title = "先生" if gender == "男" else "女士"
23                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。"
24                file.write(message)
25    except Exception as e:
26        print(f"错误: 写入文件 {output_file} 时出现问题: {e}")
27
28 if __name__ == "__main__":
29    contacts = read_contacts('contacts.txt')
30    sorted_contacts = sort_contacts(contacts)
31    generate_emails(sorted_contacts, 'emails.txt')
```

```
17437@DESKTOP-34KCETK 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。
---
```

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

python -m pdb main.py 用调试器加载代码

pdb 提示符:

l (显示代码): 显示箭头指向即将运行但还没有运行的代码

```
(Pdb) l
1      # 读取 contacts.txt 文件
2      -> def read_contacts(file_path):
3          contacts = []
```

L 进阶: l. 显示箭头上下 5 行 (l. 1, 5 第一行到第五行) ([h / 查看攻略](#))

n (执行当前行): next one

```
(Pdb) n
> c:\users\17437\repo\week04\main.py(14)<module>()
-> def sort_contacts(contacts):
```

p (打印表达式): 只能有已经运行过的部分, 可以查看运行附近的变量

s (步入调用): 与 n 效果相似, 看看里面在干什么

```
32
(Pdb) s
> c:\users\17437\repo\week04\main.py(18)<module>()
-> def generate_emails(contacts, output_file):
```

pp (美观打印)

```
(Pdb) p contacts_file
*** NameError: name 'contacts_file' is not defined
(Pdb) p
*** SyntaxError: invalid syntax
(Pdb) p contacts.txt
*** AttributeError: 'list' object has no attribute 'txt'
(Pdb) p 'contacts.txt'
'contacts.txt'
```

找了好半天 ai 给我弄的名字

```
(Pdb) p 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', 'lixliulian@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', 'lixliulian@163.com'),
 ('祝无双', '女', 'zhuwushuang@163.com')]
```

c (继续执行): 一条路走到底

```
(Pdb) c
The program finished and will be restarted
> c:\users\17437\repo\week04\main.py(2)<module>()
-> def read_contacts(file_path):
```

q (退出解释器)

7. python 基本概念

```
main.py
1  # 读取 contacts.txt 文件
2  def read_contacts(file_path):
3      contacts = []
4      try:
5          with open(file_path, 'r', encoding='utf-8') as file:
6              for line in file:
7                  name, gender, email = line.strip().split()
8                  contacts.append((name, gender, email))
9      except FileNotFoundError:
10         print(f"错误: 文件 {file_path} 未找到。")
11         return contacts
12
13 # 对联系人按邮箱域名和用户名排序
14 def sort_contacts(contacts):
15     return sorted(contacts, key=lambda x: (x[2].split('@')[1], x[2].split('@')[0]))
16
17 # 生成通知内容并写入 emails.txt 文件
18 def generate_emails(contacts, output_file):
19     try:
20         with open(output_file, 'w', encoding='utf-8') as file:
21             for name, gender, email in contacts:
22                 title = "先生" if gender == "男" else "女士"
23                 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---\\n"
24                 file.write(message)
25     except Exception as e:
26         print(f"错误: 写入文件 {output_file} 时出现问题: {e}")
27
28 if __name__ == "__main__":
29     contacts = read_contacts('contacts.txt')
30     sorted_contacts = sort_contacts(contacts)
31     generate_emails(sorted_contacts, 'emails.txt')
```

- Python 语法保留字 (reserved key words)

红色部分, 在 python 语法中有特殊含义 (*def*, *while*)

- 语句 (statement) 和表达式 (expression)

语句是逻辑上完整的一句话, 语句里面有子语句 (可以折叠的, *for* 循环语句)

表达式是构成语句的元素, 例如: `open(file_path, 'r', encoding='utf-8')`

语句里面有子语句, 有嵌套像 *define* 语句、*try* 语句、*with* 语句还有赋值语句,

for 循环语句。等号右边是表达式, 左边是赋值:

```
name, gender, email = line.strip().split()
```

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

 函数调用表达式~, 表达式可嵌套好多

自动上色之后白色是变量名, 黄色是字符串, 绿色是函数

- 缩进 (indent)

冒号后下一行会向内缩进 4 个字符, 表示为上一行的子语句

python 语法中通过缩进对齐来界定子语句的边界

- 局部变量 (local variable)、全局变量 (global variable)、LEGB 规则

```
(Pdb) wat()
Local variables:
  __builtins__: dict = {...
  __file__: pdb._ScriptTarget = 'C:\Users\17437\repo\week04\main
.py'
  __name__: str = '__main__'
  __pdb_convenience_variables: dict = {...
  __spec__: NoneType = None
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
```

当前视野内能够找到这些变量, 随着 n, 变量的数量会增加

```
11         return contacts
(Pdb) wat
*** NameError: name 'wat' is not defined
(Pdb) wat ()
*** SyntaxError: invalid character ' (' (U+FF08)
(Pdb) wat()
*** NameError: name 'wat' is not defined
(Pdb) import wat
(Pdb) wat ()
*** SyntaxError: invalid character ' (' (U+FF08)
(Pdb) wat()
Local variables:
  file_path: str = 'contacts.txt'
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
```

一些愚蠢的错误, 要先 import wat, 然后还要注意 () 是英文版本

我们在调用的时候进到一个房间, 那么这个房间里的一些变量就叫做局部变量

```
(Pdb) wat.globals
Global variables:
  __builtins__: dict = {...
  __file__: pdb._ScriptTarget = 'C:\Users\17437\repo\week04\main
.py'
  __name__: str = '__main__'
  __pdb_convenience_variables: dict = {...
  __spec__: NoneType = None
  generate_emails: function = <function generate_emails at 0x000
00296430539C0>
  read_contacts: function = <function read_contacts at 0x0000029
643053B00>
  sort_contacts: function = <function sort_contacts at 0x0000029
643053CE0>
```

查看全局变量 wat.globals

总结：在 `define` 里边才定义的变量，比如说这里面 `contacts` 或者什么 `name` `gender` `email` 这些东西，他们就是只有在调用这个函数走进来了以后运行了以后，他们才有这些，就是局部变量是取决于运行到哪里的，你进到哪个房间，你才能看到那个局部变量。但全局变量总是能够访问到的，在这个代码里面，任何一个地方都能反映到全局变量。

LEGB 查找顺序

当 Python 解释器遇到一个变量时，会按照 LEGB 的顺序依次查找该变量的定义：

1. 首先在局部作用域 (Local) 中查找。
 2. 如果局部作用域中没有找到，就到闭包作用域 (Enclosing) 中查找。
 3. 若闭包作用域中也没有，接着到全局作用域 (Global) 中查找。
 4. 要是全局作用域中还是没有，最后会在内置作用域 (Built-in) 中查找。
 5. 如果在所有作用域中都没有找到该变量，就会抛出 `NameError` 异常。
- 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))

“男” 字面值，多为引号字符串；或者是 `[]`、`-1`

```
(Pdb) p {'a': 1}
{'a': 1}
(Pdb) p {'a': 1, 'b': 2}
{'a': 1, 'b': 2}
```

字典

- 运算符 (operator)

1. 一个等号 `=` 是赋值语句；两个等号 `==` 是运算符

```
title = "先生" if gender == "男" else "女士"
```

三目运算符，由三个表达式构成，先判断中间的表达式是否成立，成立取前者，不成立取后者

2. 点 `.` 也是运算符：名称访问运算符就是这个 `file.write`，是在这个 `file` 对象的这个名称空间里边，我要访问它的名称空间里边的叫 `write` 的东西。

3. `()`：调用运算符

4. `def generate_emails(contacts, output_file):` () 不是运算符, 语法要求~

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

`def sort_contacts(contacts):` 橙色形参, 抽象

`contacts = read_contacts('contacts.txt')` 黄色实参, 能找到东西, 具体

`return contacts` 红色后面返回值

- 对象 (object)、类型 (type)、属性 (attribute)、方法 (method) **非常重要**

内存所管理东西都是对象, 且都有 reference, 通过变量名能够找到它在内存里面。一旦那个对象没有 reference, 比如说没有变量指向它了, 那个对象的内存就会被回收清理, 就释放变成自由内存了, 然后操作系统就可以重新分配内存, 因为内存是紧缺有限的。

```
(Pdb) wat / emails
*** NameError: name 'emails' is not defined
(Pdb) wat / emails.txt
*** NameError: name 'emails' is not defined
(Pdb) wat / contacts

value: [
  ('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', 'lixiumian@163.com'),
  ('祝无双', '女', 'zhuwushuang@163.com'),
]
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...
```

Type 类型; Len 长度是 6

Public attributes 公开属性, 绿色的 () 是方法

所有的对象都有类型, 只有知道它是什么类型, 才能知道它能干啥不能干啥。属性就是这个类型, 它有一些什么值方面的一些特点 (螺丝有多长直径是多少? 这些固定的值) 这个

方法是这个类型，它能干些什么事，这个对象调用方法，往里面传一些参数，它就能做一些事。对象为什么能做这些事？因为这些对象已经预先写好了一些程序。

8. add、commit、push 到 GitCode 平台你名下的仓库里，最后提交 PR

- 点进仓库——fork——创建 fork 项目——clone——SSH#克隆到本地，到终端

克隆到本地

```
git clone git@gitcode.com:luluchris/week02.git
```

- cd repo 后复制上面的地址

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~  
$ cd repo  
  
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo  
$ pwd  
/c/Users/17437/repo  
  
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo  
$ git clone git@gitcode.com:luluchris/week02.git
```

用 code 打开可以看到 week 文件夹和里面的原始三个文件就算成功了

- 将笔记文档另存为 pdf 的形式保存在对应的 week0x 文件夹中，在 code 显示即可
- Rename 成学习报告 x

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo  
$ cd week02  
  
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)  
$ ll  
total 464  
-rw-r--r-- 1 17437 197609 18805 4月 8 23:27 LICENSE  
-rw-r--r-- 1 17437 197609 2239 4月 8 23:27 README.md  
-rw-r--r-- 1 17437 197609 449104 4月 8 23:34 学习报告2.pdf
```

- 回到终端一些操作：cd week0x 看看 ll

- git add . :将所有改动添加到购物车中

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)  
$ git add 学习报告2.pdf
```

使用 git status 查看全都是绿色的

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)  
$ git status  
On branch main  
Your branch is up to date with 'origin/main'.  
  
Changes to be committed:  
  (use "git restore --staged <file>..." to unstage)  
    new file:   学习报告2.pdf
```


(git log 可以看提交日志)

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)
$ git log
commit 1eda0e772f79b63f00e277bec29041b10ad24cfe (HEAD -> main, origin/main, origin/HEAD)
Author: mutecamel <mutecamel@gmail.com>
Date:   Wed Mar 5 11:07:04 2025 +0800

    initial commit
```

- git commit -m "添加了学习报告 02", 交上去之后 status 就干净了~

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)
$ git commit -m "添加了学习报告 02"
[main c03eaf3] 添加了学习报告 02
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 学习报告2.pdf

(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit, working tree clean
```

(git log 也有相应记录)

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)
$ git log
commit c03eaf313d88f4557538f06142077556d4f4c31a (HEAD -> main)
Author: luluchris <luluchris@noreply.gitcode.com>
Date:   Wed Apr 9 00:39:10 2025 +0800

    添加了学习报告 02

commit 1eda0e772f79b63f00e277bec29041b10ad24cfe (origin/main, origin/HEAD)
Author: mutecamel <mutecamel@gmail.com>
Date:   Wed Mar 5 11:07:04 2025 +0800

    initial commit
```

- git push origin main, (git log 之后又有新的变化)

```
(base) 17437@DESKTOP-34KCETK MINGW64 ~/repo/week02 (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 402.90 KiB | 8.95 MiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Start Git Hooks Checking
To gitcode.com:luluchris/week02.git
    1eda0e7..c03eaf3  main -> main
```

刷新网页后就有内容啦~



文件	最后提交记录	最后更新时间
.gitignore	initial commit	1 个月前
LICENSE	initial commit	1 个月前
README.md	initial commit	1 个月前
学习报告2.pdf	添加了学习报告02	17 分钟前

- 点击网页中的 pull requests 新建一个，直接合入，更改标题为：第 x 周交作业---！
- 直接上交即可~