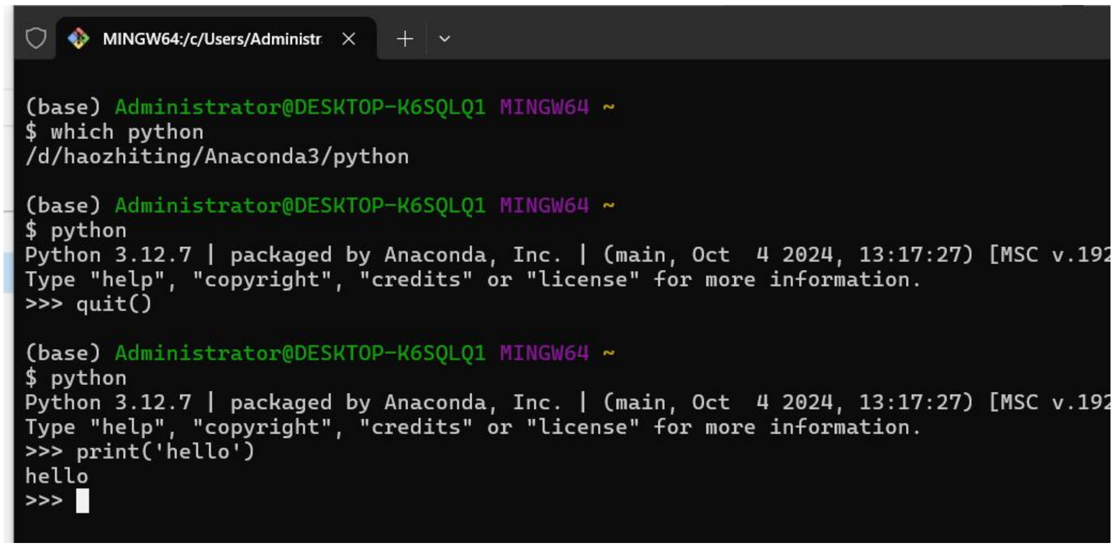
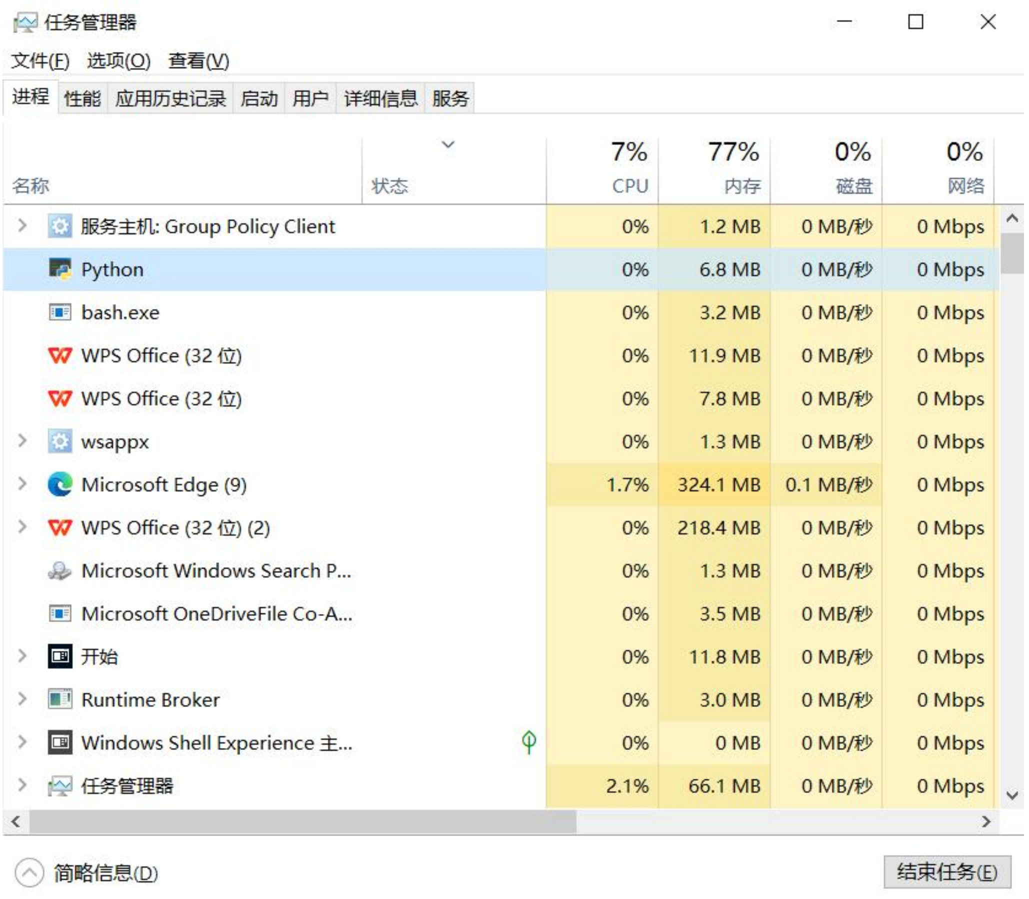
第四周学习笔记

1.数据在不通电的情况下可以长期持久地（persistently）存储在 磁盘（如固态硬盘 SSD、机械硬盘 HDD）或磁带（常用于数据备份、长期归档）里。但在需要呈现（print、render、show、display、play)、计算加工（compute、transform、analyze、machine learning、deep learning）或编解码（encode、decode）时，就需要通电的 CPU 和 内存（硬件），在操作系统（软件）里以 进程（process）为单元（相互隔离）进行处理。例如，Microsoft Word 启动后就是一个进程，我们在 Word 进程里打开某个 .docx 文档，将其从磁盘加载（读取）到内存，然后在图形界面（GUI）里查看和编辑（计算）内存中的文档，最后将内存数据保存（写入）到磁盘。同理，Python 解释器（interpreter）启动后也是一个进程，她按照流程（flow）执行我们准备好的 Python 代码，根据我们代码的要求，转告（即 调用, call）操作系统或其他软件（即 依赖项, dependency），委托她们替我们执行各种 “读取——计算——写入” 等工作。我们并不需要完全理解依赖项内部的工作细节（黑箱），只需要清楚每个调用的主体（即 对象, object）是什么 类型（type），每个调用的输入（即 参数, parameter/argument）、输出（即 返回值, return value）是什么类型，以及调用会对内存数据、磁盘文件做什么修改，就足以支持我们自动批量地完成工作了。

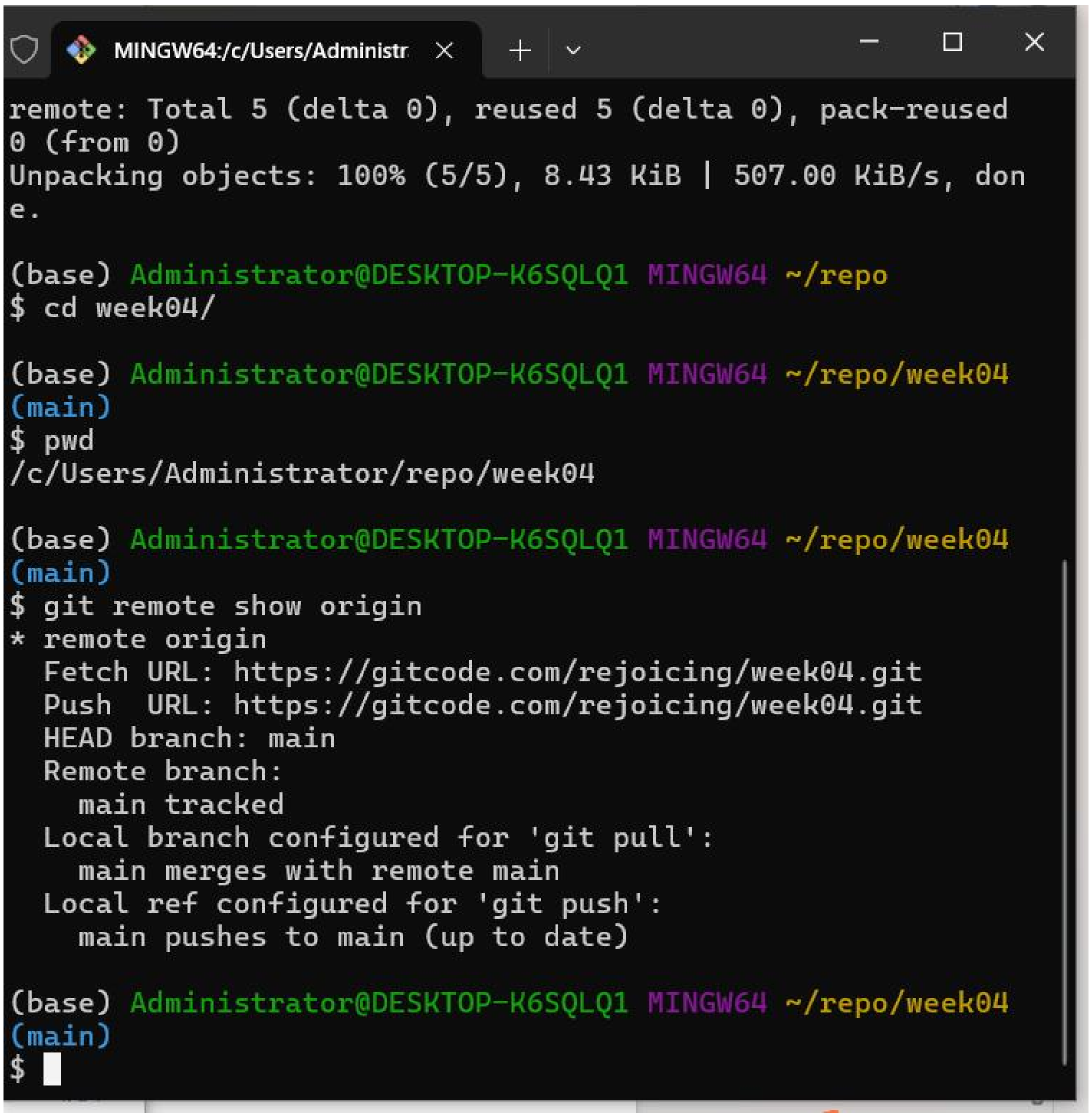


```
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~
$ which python
/d/haozhiting/Anaconda3/python

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~
$ python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.192
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~
$ python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.192
Type "help", "copyright", "credits" or "license" for more information.
>>> print('hello')
hello
>>> 
```



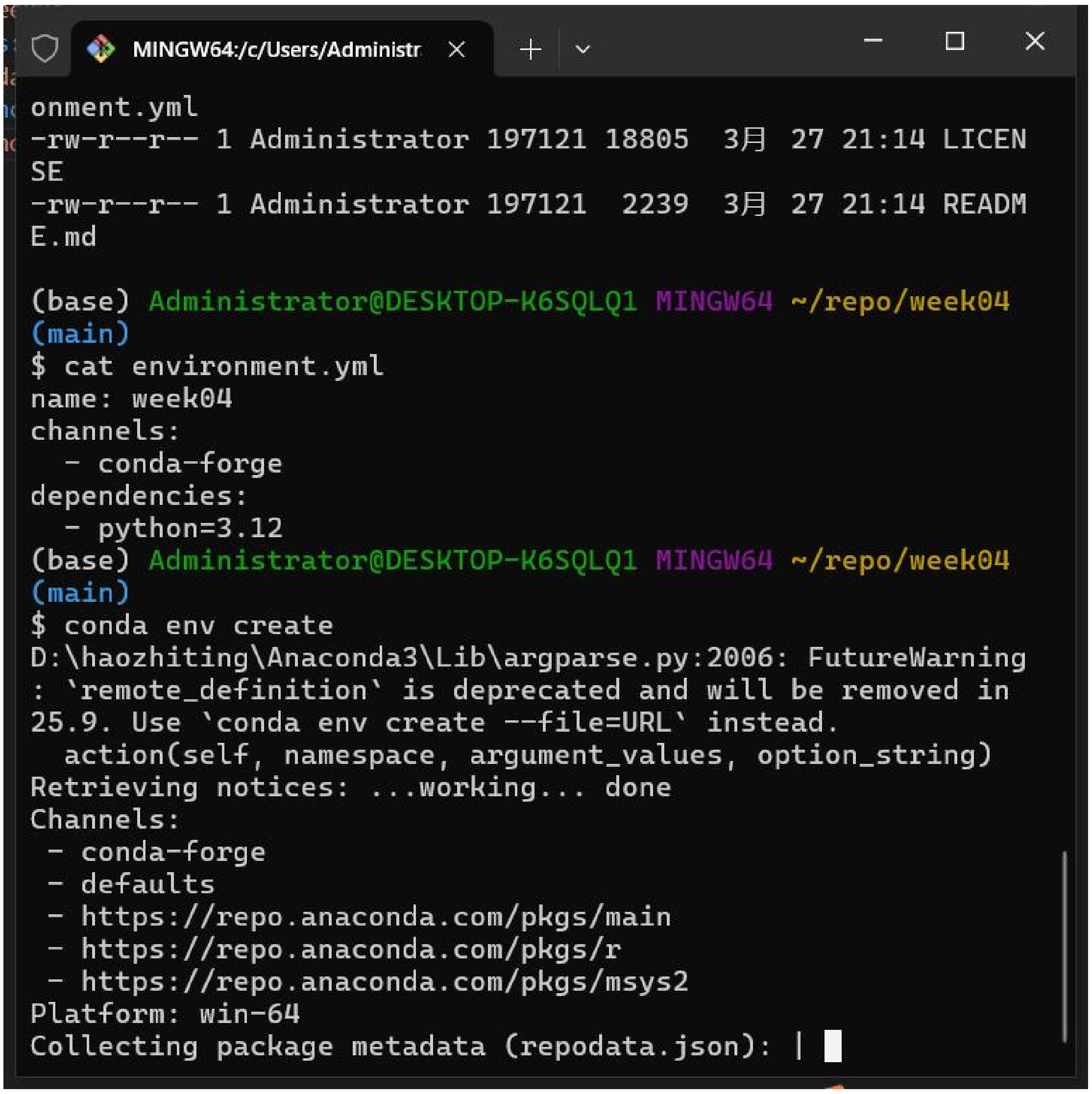
名称	状态	7% CPU	77% 内存	0% 磁盘	0% 网络
服务主机: Group Policy Client		0%	1.2 MB	0 MB/秒	0 Mbps
Python		0%	6.8 MB	0 MB/秒	0 Mbps
bash.exe		0%	3.2 MB	0 MB/秒	0 Mbps
WPS Office (32 位)		0%	11.9 MB	0 MB/秒	0 Mbps
WPS Office (32 位)		0%	7.8 MB	0 MB/秒	0 Mbps
wsappx		0%	1.3 MB	0 MB/秒	0 Mbps
Microsoft Edge (9)		1.7%	324.1 MB	0.1 MB/秒	0 Mbps
WPS Office (32 位) (2)		0%	218.4 MB	0 MB/秒	0 Mbps
Microsoft Windows Search P...		0%	1.3 MB	0 MB/秒	0 Mbps
Microsoft OneDriveFile Co-A...		0%	3.5 MB	0 MB/秒	0 Mbps
开始		0%	11.8 MB	0 MB/秒	0 Mbps
Runtime Broker		0%	3.0 MB	0 MB/秒	0 Mbps
Windows Shell Experience 主...		0%	0 MB	0 MB/秒	0 Mbps
任务管理器		2.1%	66.1 MB	0 MB/秒	0 Mbps

2.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



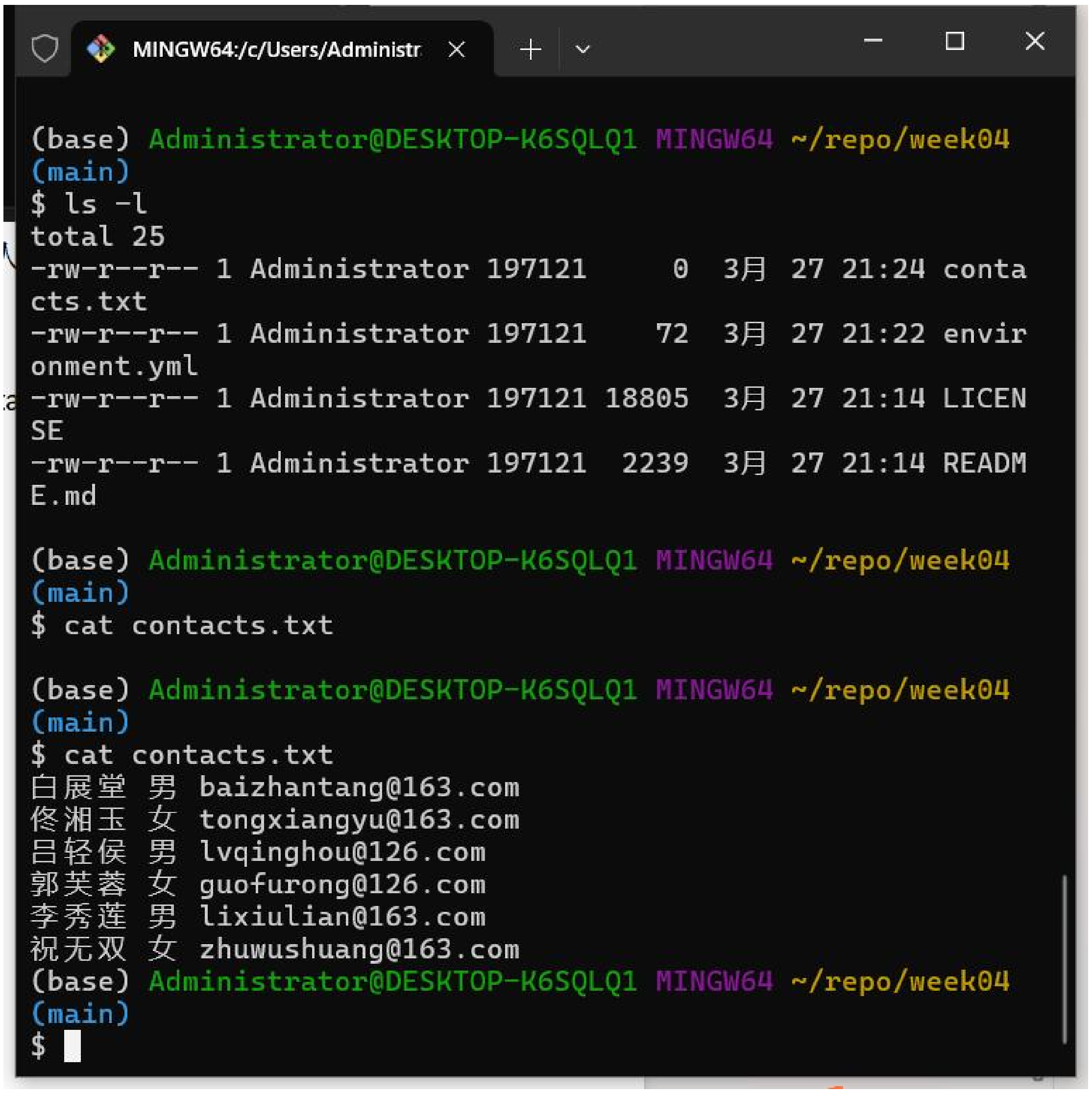
```
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused
0 (from 0)
Unpacking objects: 100% (5/5), 8.43 KiB | 507.00 KiB/s, don
e.

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo
$ cd week04/

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ pwd
/c/Users/Administrator/repo/week04

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ git remote show origin
* remote origin
  Fetch URL: https://gitcode.com/rejoicing/week04.git
  Push  URL: https://gitcode.com/rejoicing/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)

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$
```

3.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



```
onment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICEN
SE
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 READM
E.md

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ cat environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ conda env create
D:\haozhiting\Anaconda3\Lib\argparse.py:2006: FutureWarning
: `remote_definition` is deprecated and will be removed in
25.9. Use `conda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - defaults
 - 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): |
```

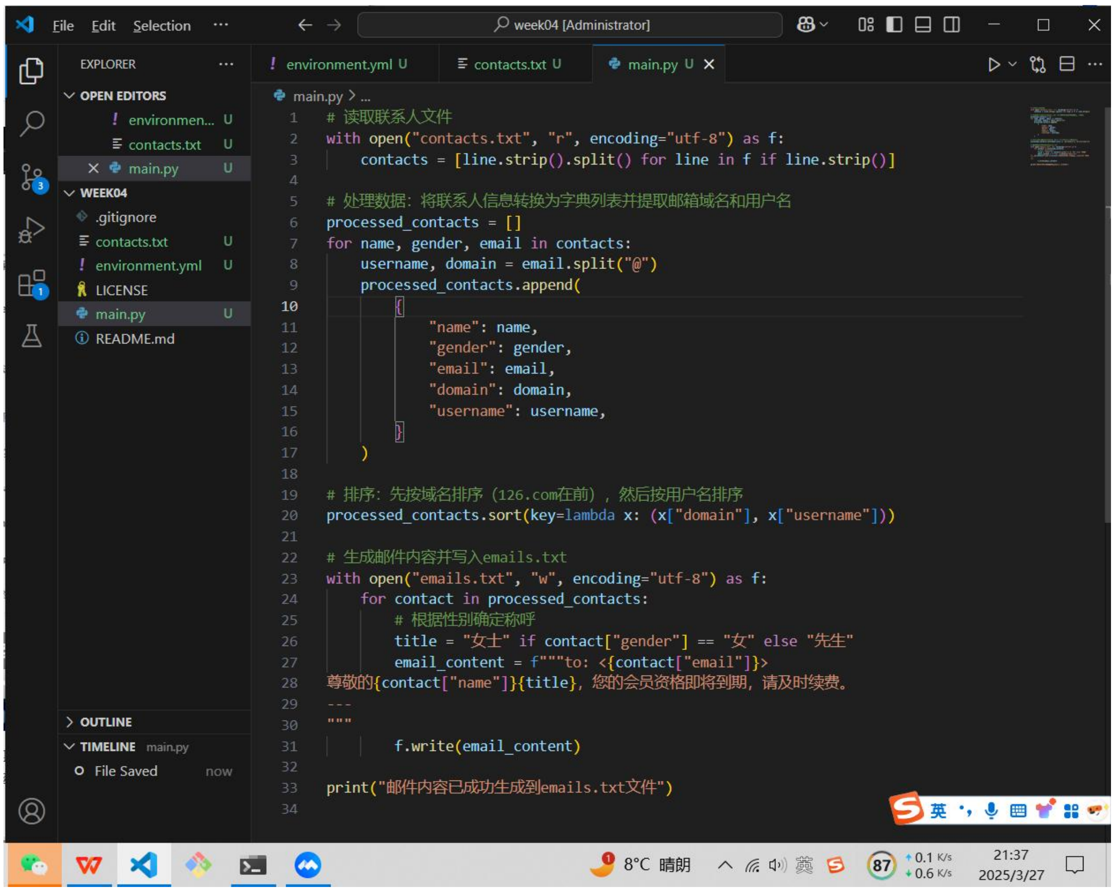
4.　新建一个 contacts.txt 文件，每行写一个联系人，每个联系人都包含姓名、性别、邮箱三个字段，用空格分隔



```
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ ls -l
total 25
-rw-r--r-- 1 Administrator 197121     0  3月  27 21:24 conta
cts.txt
-rw-r--r-- 1 Administrator 197121    72  3月  27 21:22 envir
onment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICEN
SE
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 READM
E.md

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ cat contacts.txt

(base) Administrator@DESKTOP-K6SQLQ1 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@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04
(main)
$ 
```

5. 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）

6. 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



```python
# 读取联系人文件
with open("contacts.txt", "r", encoding="utf-8") as f:
    contacts = [line.strip().split() for line in f if line.strip()]

# 处理数据：将联系人信息转换为字典列表并提取邮箱域名和用户名
processed_contacts = []
for name, gender, email in contacts:
    username, domain = email.split("@")
    processed_contacts.append(
        {
            "name": name,
            "gender": gender,
            "email": email,
            "domain": domain,
            "username": username,
        }
    )

# 排序：先按域名排序（126.com在前），然后按用户名排序
processed_contacts.sort(key=lambda x: (x["domain"], x["username"]))

# 生成邮件内容并写入emails.txt
with open("emails.txt", "w", encoding="utf-8") as f:
    for contact in processed_contacts:
        # 根据性别确定称呼
        title = "女士" if contact["gender"] == "女" else "先生"
        email_content = f"""to: <{contact["email"]}>
尊敬的{contact["name"]}{title}，您的会员资格即将到期，请及时续费。
---
"""
        f.write(email_content)

print("邮件内容已成功生成到emails.txt文件")
```

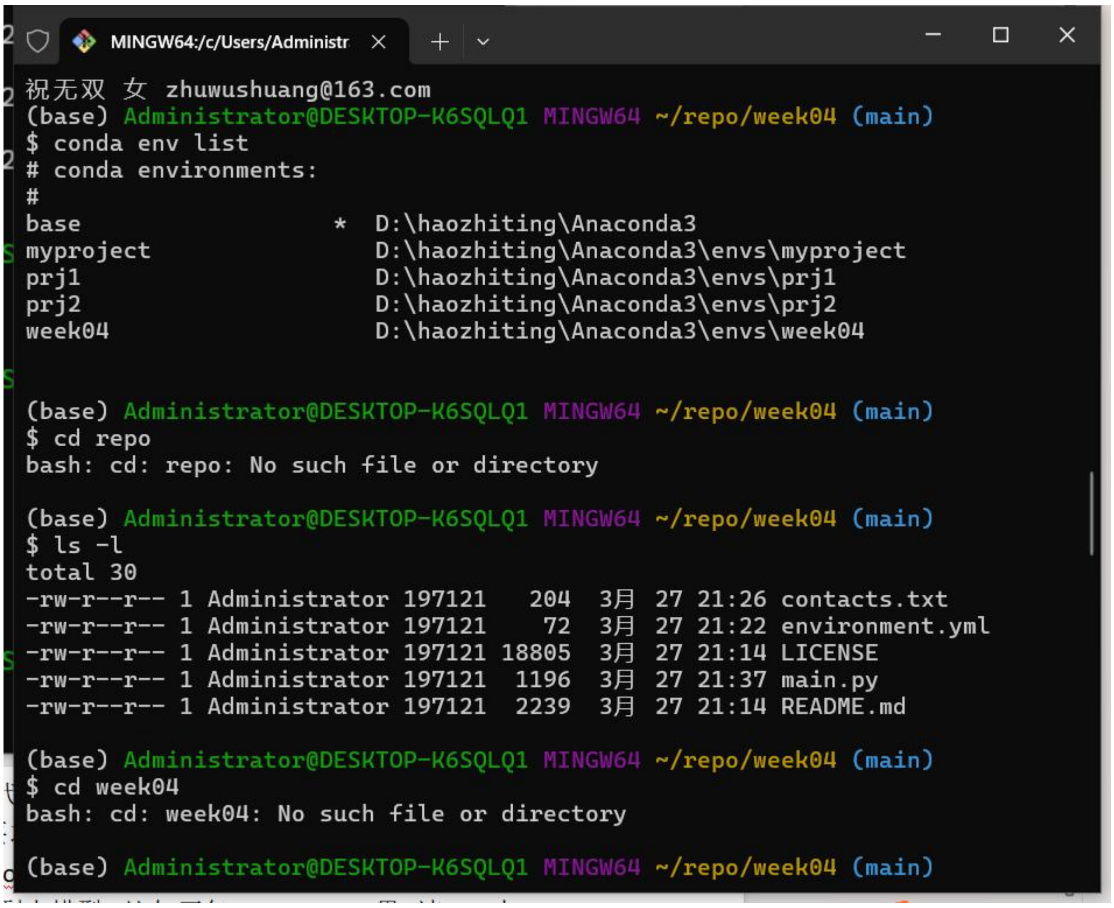
7. AI 回复的只是静态代码，而且可能含有错误，所以我们必须在 Conda 环境里运行代码，逐行调试，检查每一行代码的运行都符合我们的期望 (越是初学者越应该慢慢调试、检查、试验，借此学习)

将大模型提供的代码复制粘贴进 main.py 文件，记得保存

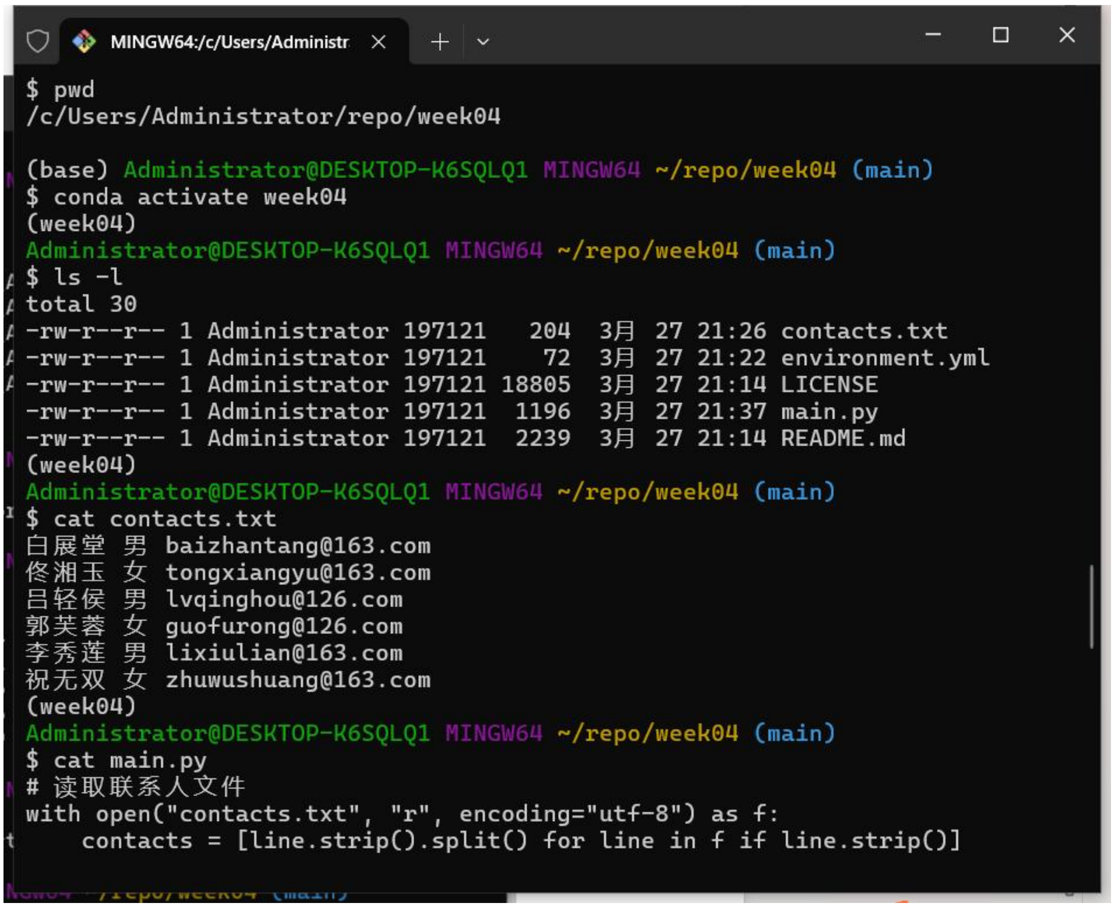
在 VS Code 扩展商店里安装 Python 扩展，使得在编写 .py 文件时能够显示和选择 Python 解释器 (需要绕过防火墙)

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

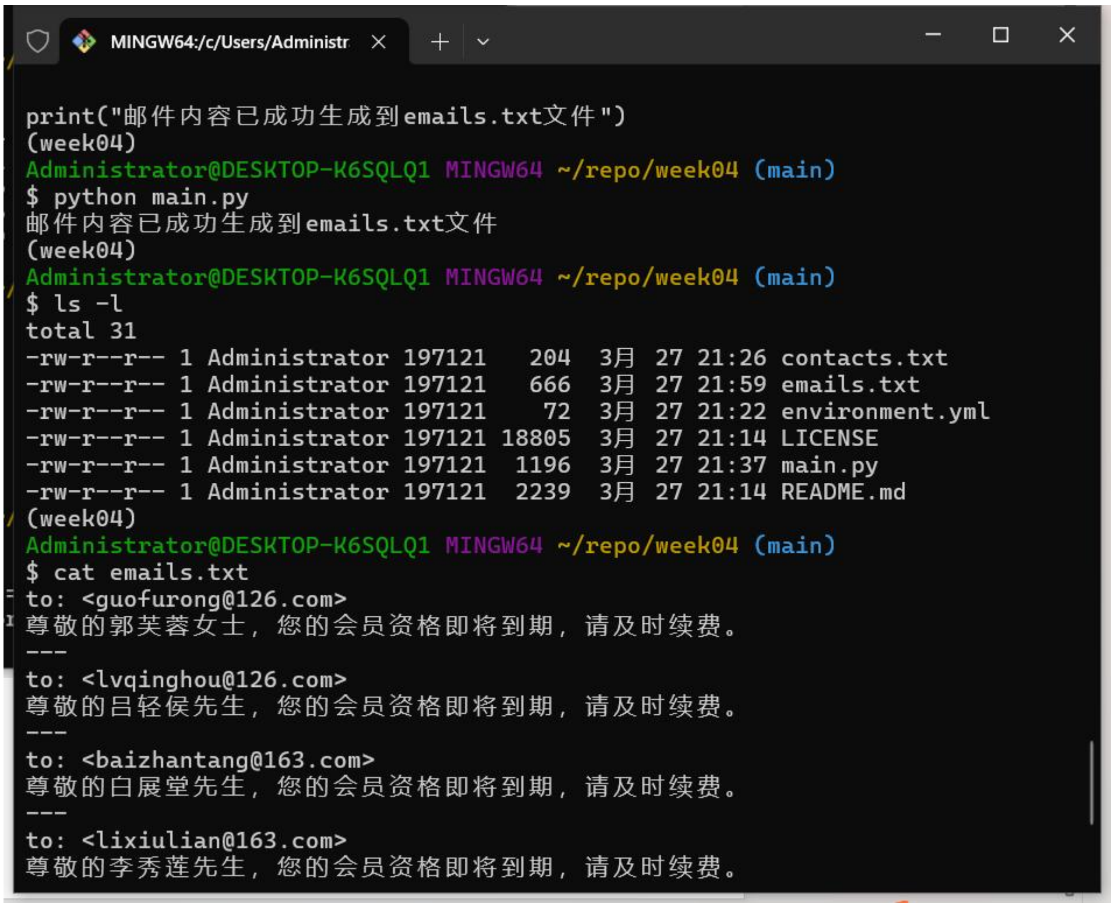
运行 python main.py 命令 (作用是启动 Python 解释器，执行 main.py 里的代码直至结束 (EOF) 或报错 (Exception))，检查运行结果是否符合预期



```
祝无双 女 zhuwushuang@163.com
(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ conda env list
# conda environments:
#
base                  *  D:\haozhiting\Anaconda3
myproject                D:\haozhiting\Anaconda3\envs\myproject
prj1                     D:\haozhiting\Anaconda3\envs\prj1
prj2                     D:\haozhiting\Anaconda3\envs\prj2
week04                   D:\haozhiting\Anaconda3\envs\week04


(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ cd repo
bash: cd: repo: No such file or directory

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 Administrator 197121   204  3月  27 21:26 contacts.txt
-rw-r--r-- 1 Administrator 197121    72  3月  27 21:22 environment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICENSE
-rw-r--r-- 1 Administrator 197121  1196  3月  27 21:37 main.py
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 README.md

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ cd week04
bash: cd: week04: No such file or directory

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
```

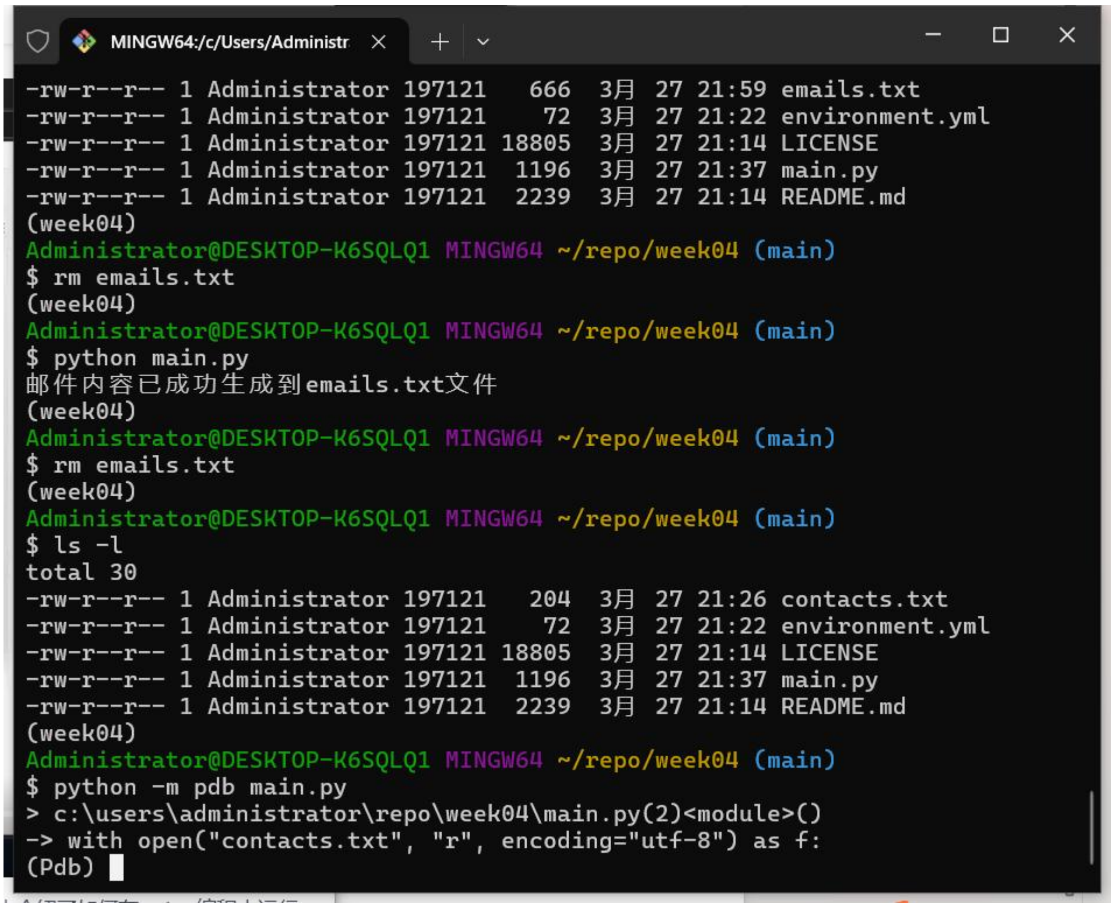


```
$ pwd
/c/Users/Administrator/repo/week04

(base) Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ conda activate week04
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 Administrator 197121   204  3月  27 21:26 contacts.txt
-rw-r--r-- 1 Administrator 197121    72  3月  27 21:22 environment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICENSE
-rw-r--r-- 1 Administrator 197121  1196  3月  27 21:37 main.py
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 README.md
(week04)
Administrator@DESKTOP-K6SQLQ1 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)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ cat main.py
# 读取联系人文件
with open("contacts.txt", "r", encoding="utf-8") as f:
    contacts = [line.strip().split() for line in f if line.strip()]
```



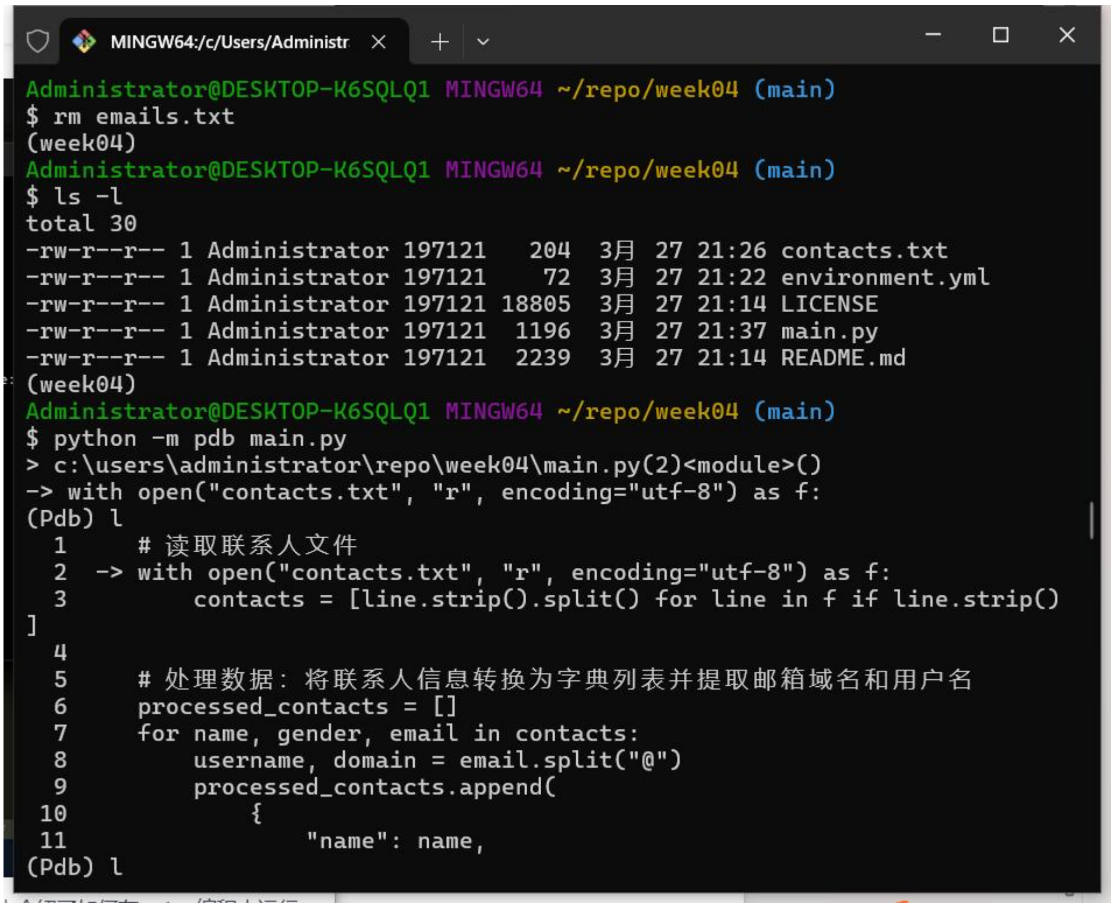
```
print("邮件内容已成功生成到emails.txt文件")
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ python main.py
邮件内容已成功生成到emails.txt文件
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ ls -l
total 31
-rw-r--r-- 1 Administrator 197121   204  3月 27 21:26 contacts.txt
-rw-r--r-- 1 Administrator 197121   666  3月 27 21:59 emails.txt
-rw-r--r-- 1 Administrator 197121    72  3月 27 21:22 environment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月 27 21:14 LICENSE
-rw-r--r-- 1 Administrator 197121  1196  3月 27 21:37 main.py
-rw-r--r-- 1 Administrator 197121  2239  3月 27 21:14 README.md
(week04)
Administrator@DESKTOP-K6SQLQ1 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。
```

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)



```
-rw-r--r-- 1 Administrator 197121   666  3月  27 21:59 emails.txt
-rw-r--r-- 1 Administrator 197121    72  3月  27 21:22 environment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICENSE
-rw-r--r-- 1 Administrator 197121  1196  3月  27 21:37 main.py
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 README.md
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ rm emails.txt
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ python main.py
邮件内容已成功生成到emails.txt文件
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ rm emails.txt
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 Administrator 197121   204  3月  27 21:26 contacts.txt
-rw-r--r-- 1 Administrator 197121    72  3月  27 21:22 environment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICENSE
-rw-r--r-- 1 Administrator 197121  1196  3月  27 21:37 main.py
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 README.md
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\administrator\repo\week04\main.py(2)<module>()
-> with open("contacts.txt", "r", encoding="utf-8") as f:
(Pdb) 
```

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)



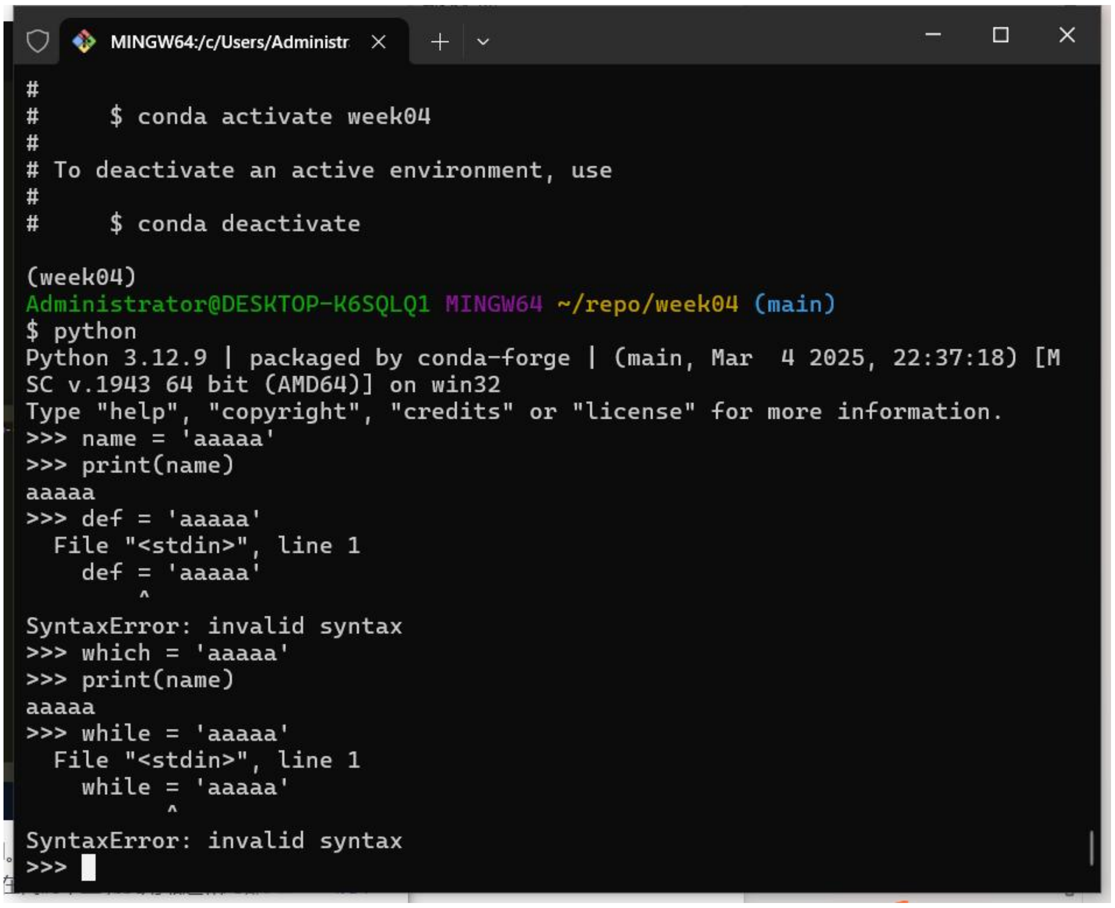
```
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ rm emails.txt
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 Administrator 197121   204  3月  27 21:26 contacts.txt
-rw-r--r-- 1 Administrator 197121    72  3月  27 21:22 environment.yml
-rw-r--r-- 1 Administrator 197121 18805  3月  27 21:14 LICENSE
-rw-r--r-- 1 Administrator 197121  1196  3月  27 21:37 main.py
-rw-r--r-- 1 Administrator 197121  2239  3月  27 21:14 README.md
(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\administrator\repo\week04\main.py(2)<module>()
-> with open("contacts.txt", "r", encoding="utf-8") as f:
(Pdb) l
  1     # 读取联系人文件
  2  -> with open("contacts.txt", "r", encoding="utf-8") as f:
  3         contacts = [line.strip().split() for line in f if line.strip()
]
  4
  5     # 处理数据：将联系人信息转换为字典列表并提取邮箱域名和用户名
  6     processed_contacts = []
  7     for name, gender, email in contacts:
  8         username, domain = email.split("@")
  9         processed_contacts.append(
 10             {
 11                 "name": name,
(Pdb) l
```

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)

红色的代码

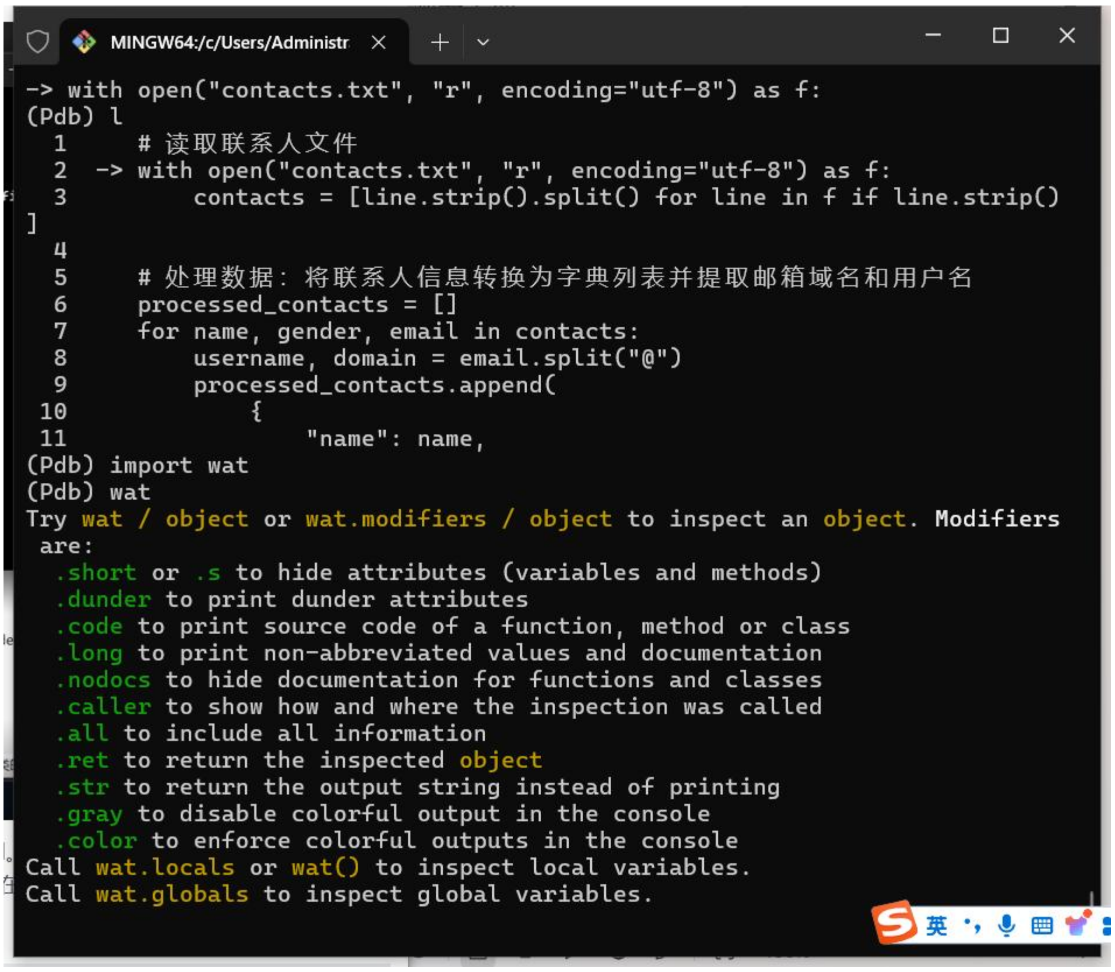


```
#
#     $ conda activate week04
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(week04)
Administrator@DESKTOP-K6SQLQ1 MINGW64 ~/repo/week04 (main)
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [M
SC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> name = 'aaaaa'
>>> print(name)
aaaaa
>>> def = 'aaaaa'
  File "<stdin>", line 1
    def = 'aaaaa'
        ^
SyntaxError: invalid syntax
>>> which = 'aaaaa'
>>> print(name)
aaaaa
>>> while = 'aaaaa'
  File "<stdin>", line 1
    while = 'aaaaa'
          ^
SyntaxError: invalid syntax
>>>
```

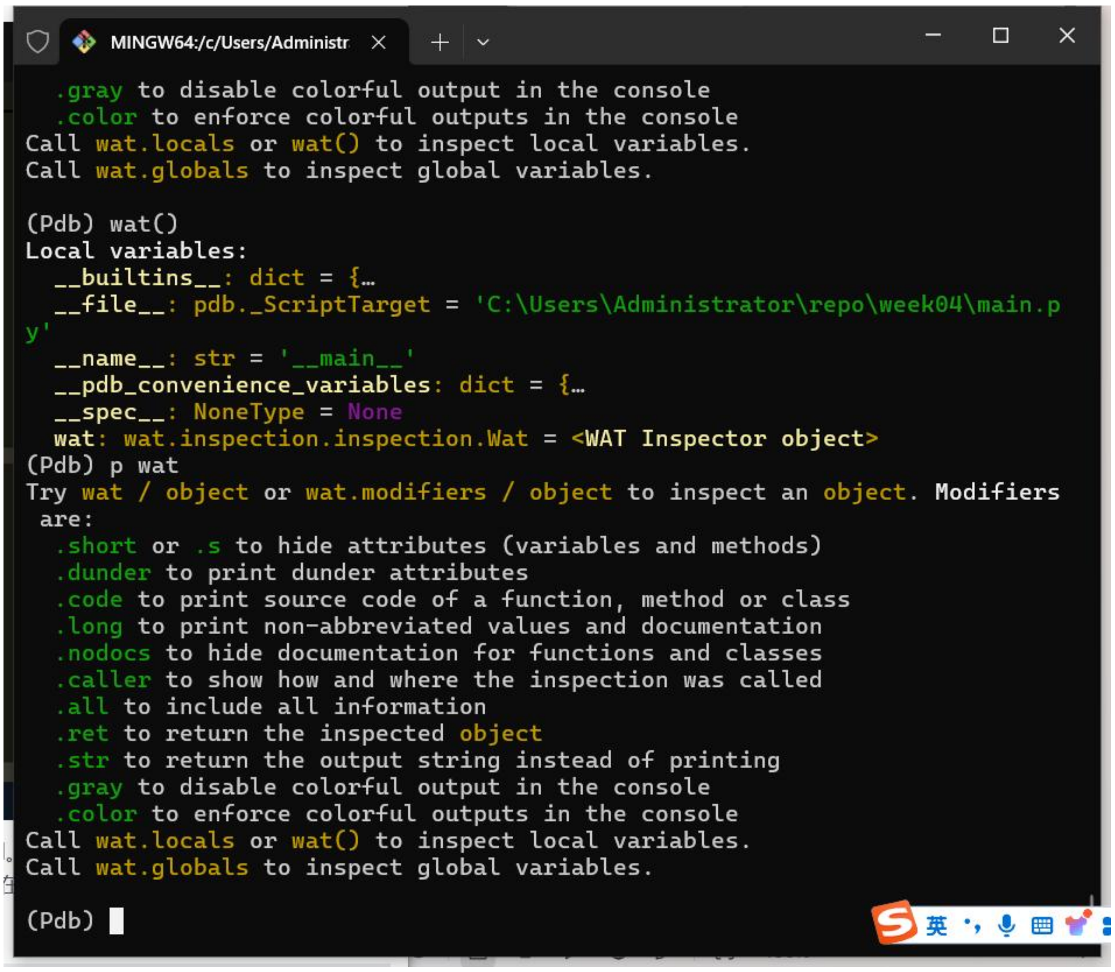
语句 (statement) 和表达式 (expression)

缩进 (indent)

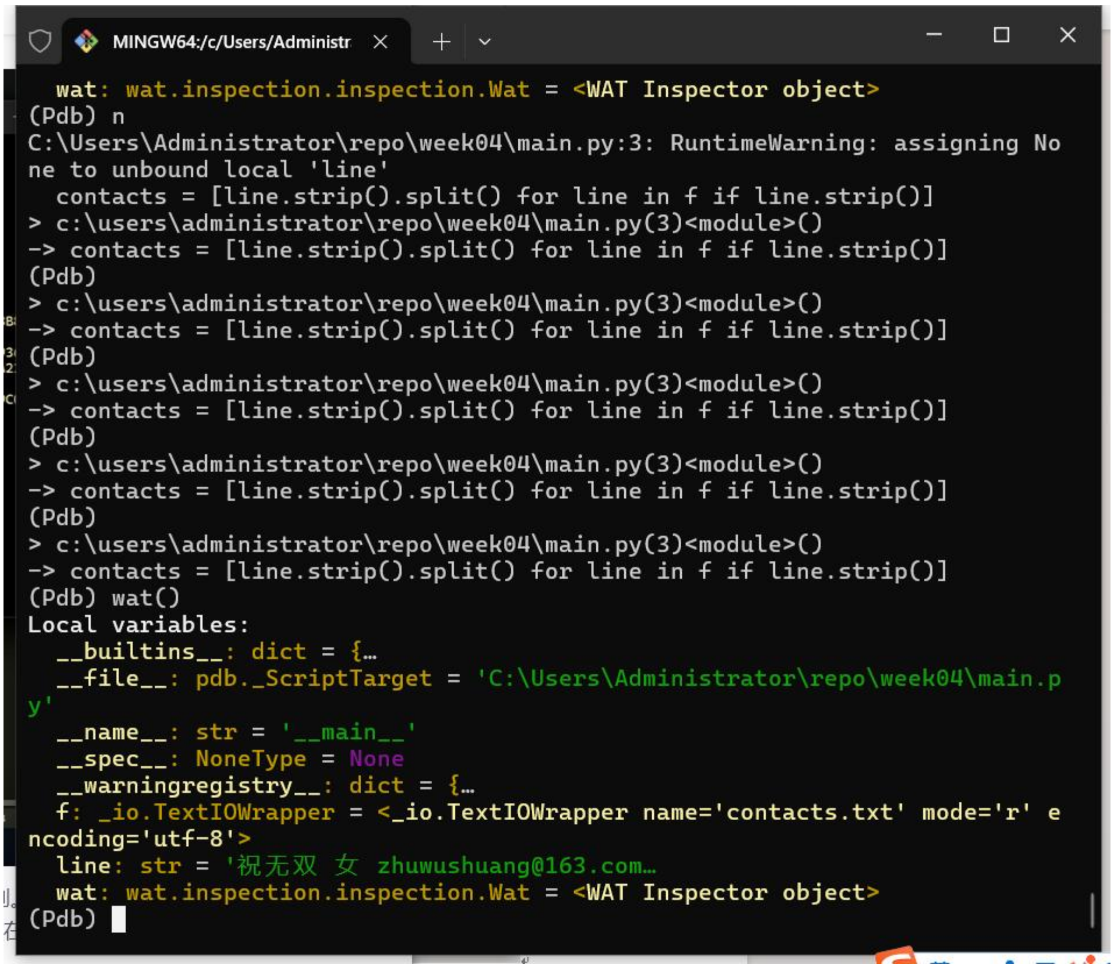
局部变量 (local variable) vs. 全局变量 (global variable)



```
-> with open("contacts.txt", "r", encoding="utf-8") as f:
(Pdb) l
        # 读取联系人文件
     -> with open("contacts.txt", "r", encoding="utf-8") as f:
            contacts = [line.strip().split() for line in f if line.strip()
]

        # 处理数据：将联系人信息转换为字典列表并提取邮箱域名和用户名
        processed_contacts = []
        for name, gender, email in contacts:
            username, domain = email.split("@")
            processed_contacts.append(
                {
                    "name": name,
(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.
```



```
  .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\Administrator\repo\week04\main.p
y'
  __name__: str = '__main__'
  __pdb_convenience_variables: dict = {…
  __spec__: NoneType = None
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb) p 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) 
```

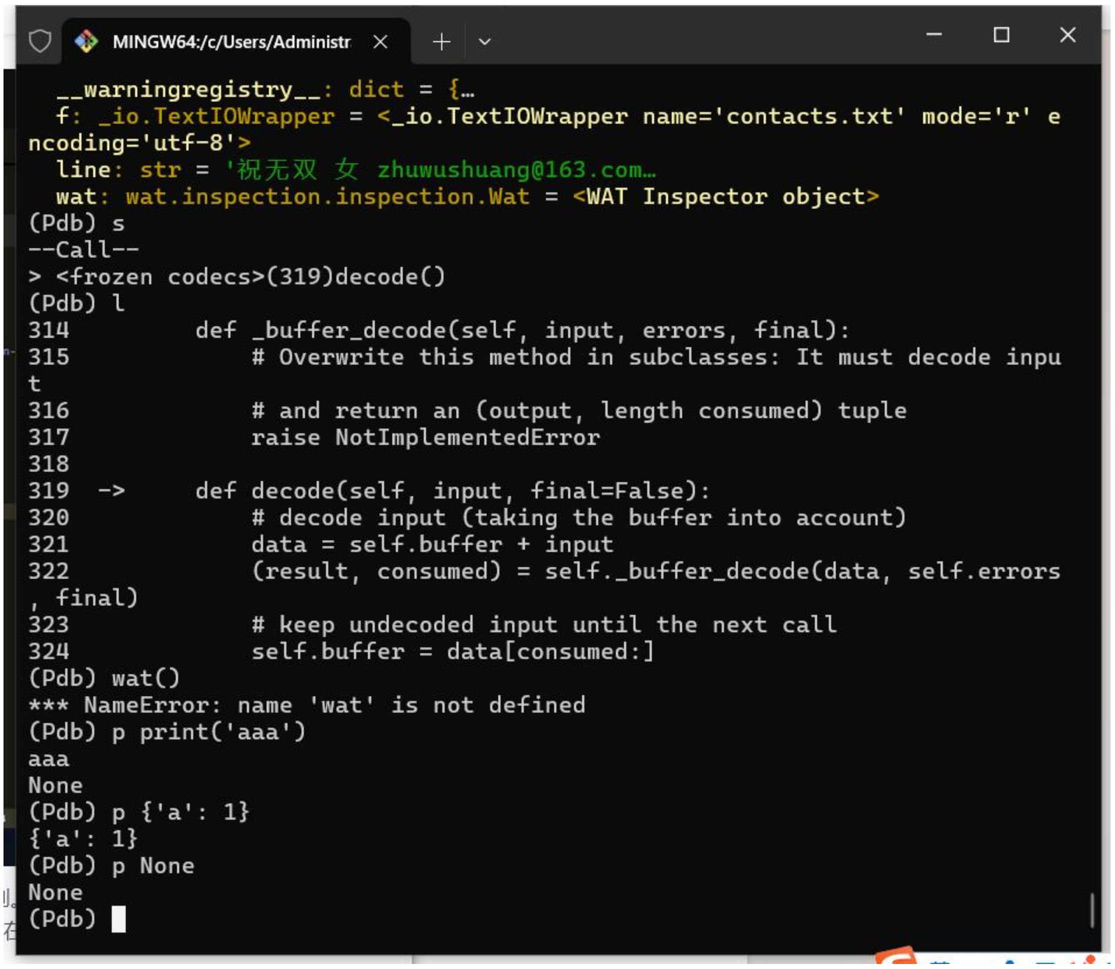


```
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb) n
C:\Users\Administrator\repo\week04\main.py:3: RuntimeWarning: assigning No
ne to unbound local 'line'
  contacts = [line.strip().split() for line in f if line.strip()]
> c:\users\administrator\repo\week04\main.py(3)<module>()
-> contacts = [line.strip().split() for line in f if line.strip()]
(Pdb)
> c:\users\administrator\repo\week04\main.py(3)<module>()
-> contacts = [line.strip().split() for line in f if line.strip()]
(Pdb)
> c:\users\administrator\repo\week04\main.py(3)<module>()
-> contacts = [line.strip().split() for line in f if line.strip()]
(Pdb)
> c:\users\administrator\repo\week04\main.py(3)<module>()
-> contacts = [line.strip().split() for line in f if line.strip()]
(Pdb)
> c:\users\administrator\repo\week04\main.py(3)<module>()
-> contacts = [line.strip().split() for line in f if line.strip()]
(Pdb) wat()
Local variables:
  __builtins__: dict = {…
  __file__: pdb._ScriptTarget = 'C:\Users\Administrator\repo\week04\main.p
y'
  __name__: str = '__main__'
  __spec__: NoneType = None
  __warningregistry__: dict = {…
  f: _io.TextIOWrapper = <_io.TextIOWrapper name='contacts.txt' mode='r' e
ncoding='utf-8'>
  line: str = '祝无双 女 zhuwushuang@163.com…
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb) 
```

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)



```
  __warningregistry__: dict = {…
  f: _io.TextIOWrapper = <_io.TextIOWrapper name='contacts.txt' mode='r' e
ncoding='utf-8'>
  line: str = '祝无双 女 zhuwushuang@163.com…
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb) s
--Call--
> <frozen codecs>(319)decode()
(Pdb) l
314         def _buffer_decode(self, input, errors, final):
315             # Overwrite this method in subclasses: It must decode inpu
t
316             # and return an (output, length consumed) tuple
317             raise NotImplementedError
318
319  ->     def decode(self, input, final=False):
320             # decode input (taking the buffer into account)
321             data = self.buffer + input
322             (result, consumed) = self._buffer_decode(data, self.errors
, final)
323             # keep undecoded input until the next call
324             self.buffer = data[consumed:]
(Pdb) wat()
*** NameError: name 'wat' is not defined
(Pdb) p print('aaa')
aaa
None
(Pdb) p {'a': 1}
{'a': 1}
(Pdb) p None
None
(Pdb) █
```

形参 (parameter)、实参 (argument)、返回值 (return value)
对象 (object)、类型 (type)、属性 (attribute)、方法 (method)