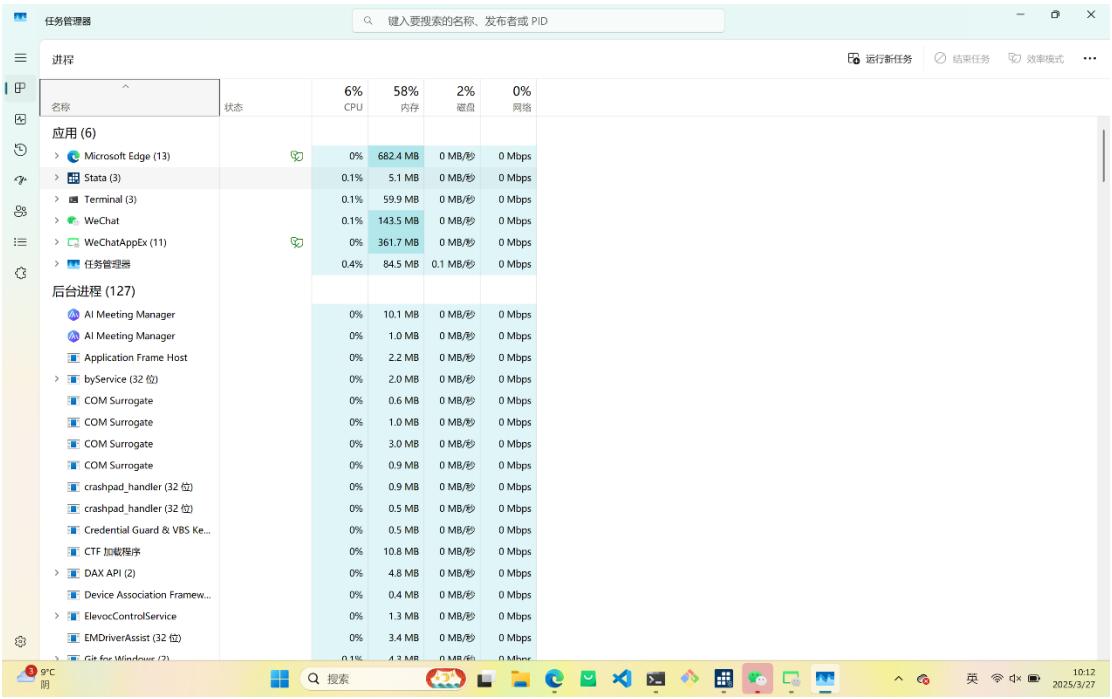
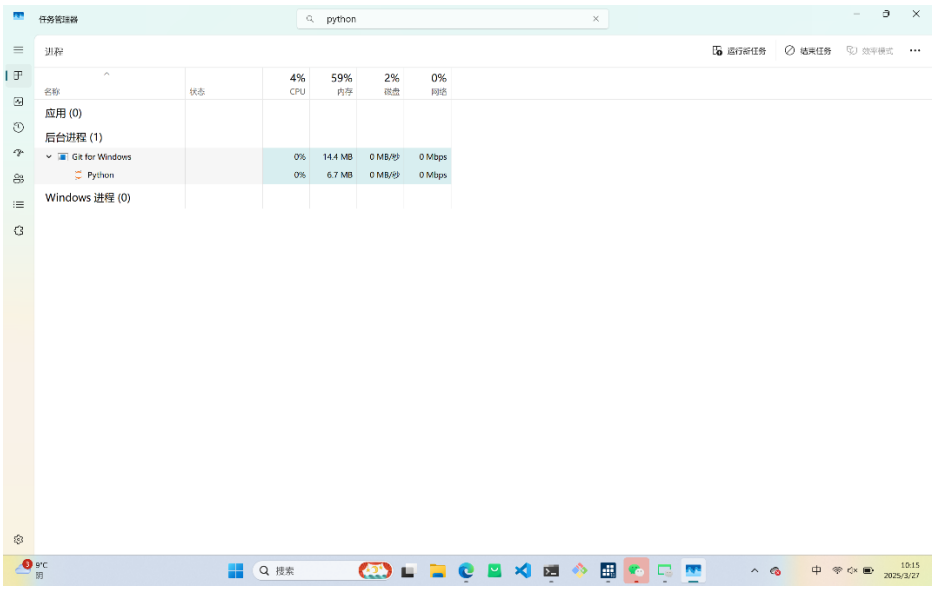
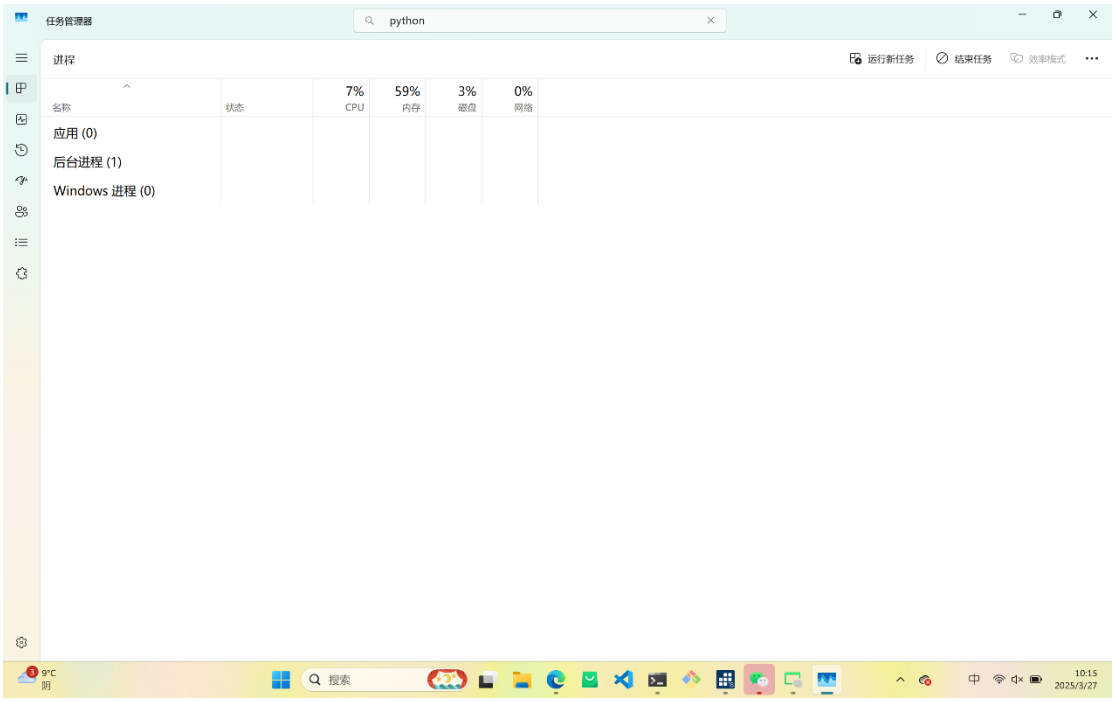
数据在不通电的情况下可以长期持久地存储在磁盘或磁带里。但在需要呈现、计算加工或编解码时，就需要通电的 CPU 和内存(硬件)，在操作系统（软件）里以进程(process）为单元（相互隔离）进行处理。



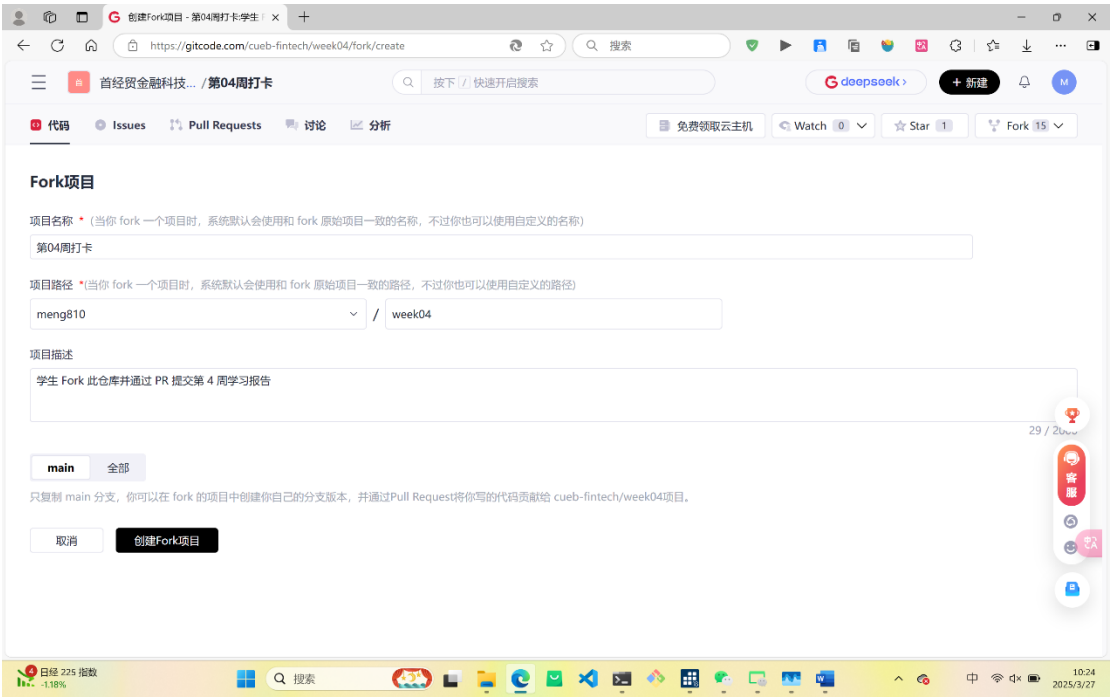
Python 解释器启动后也是一个进程

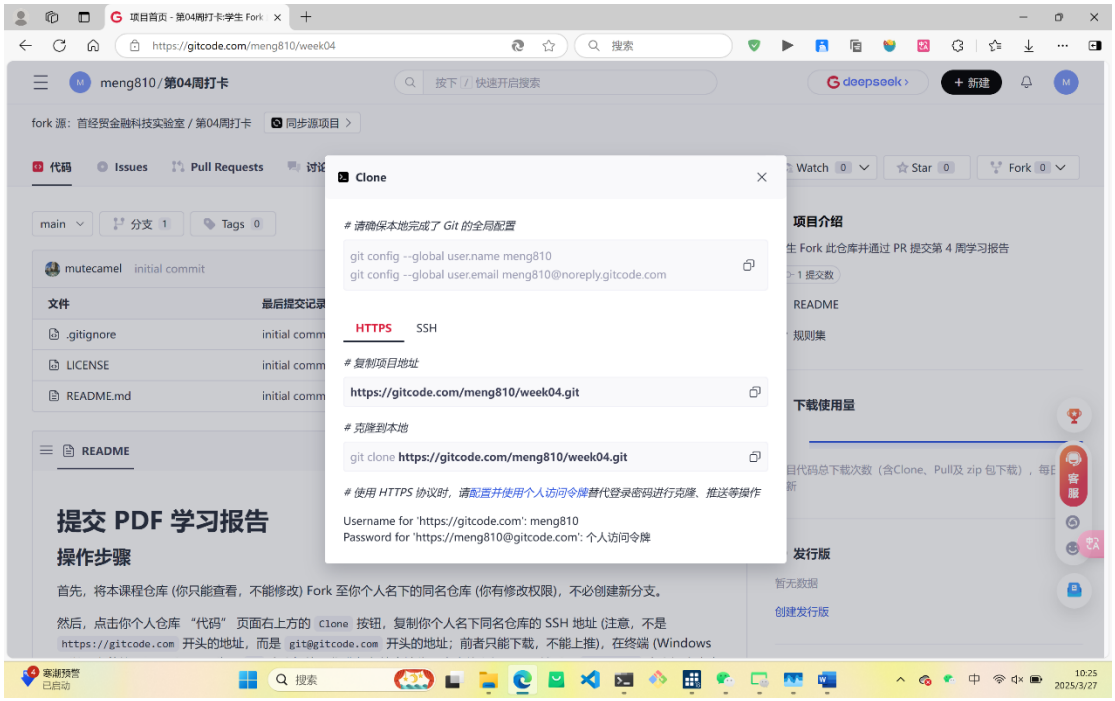
```
(base) aa@LAPTOP-GU1RAG6H MINGW64 ~
$ which python
/c/Users/aa/anaconda3/python

(base) aa@LAPTOP-GU1RAG6H MINGW64 ~
$ python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> quit()

(base) aa@LAPTOP-GU1RAG6H MINGW64 ~
$
```

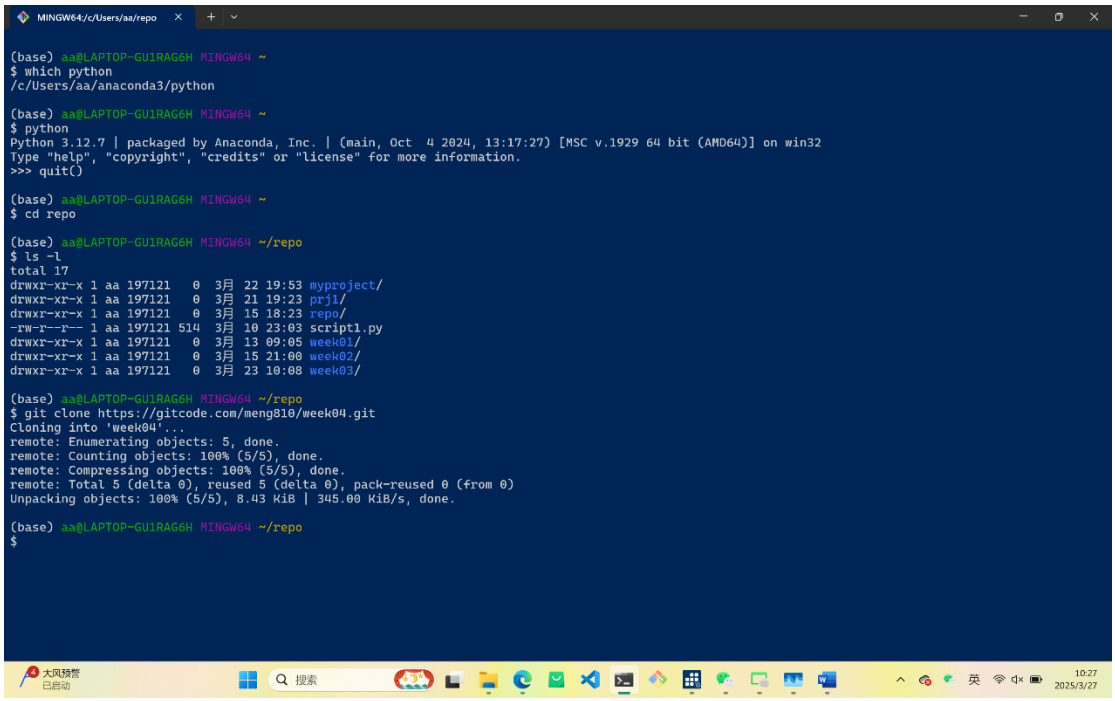




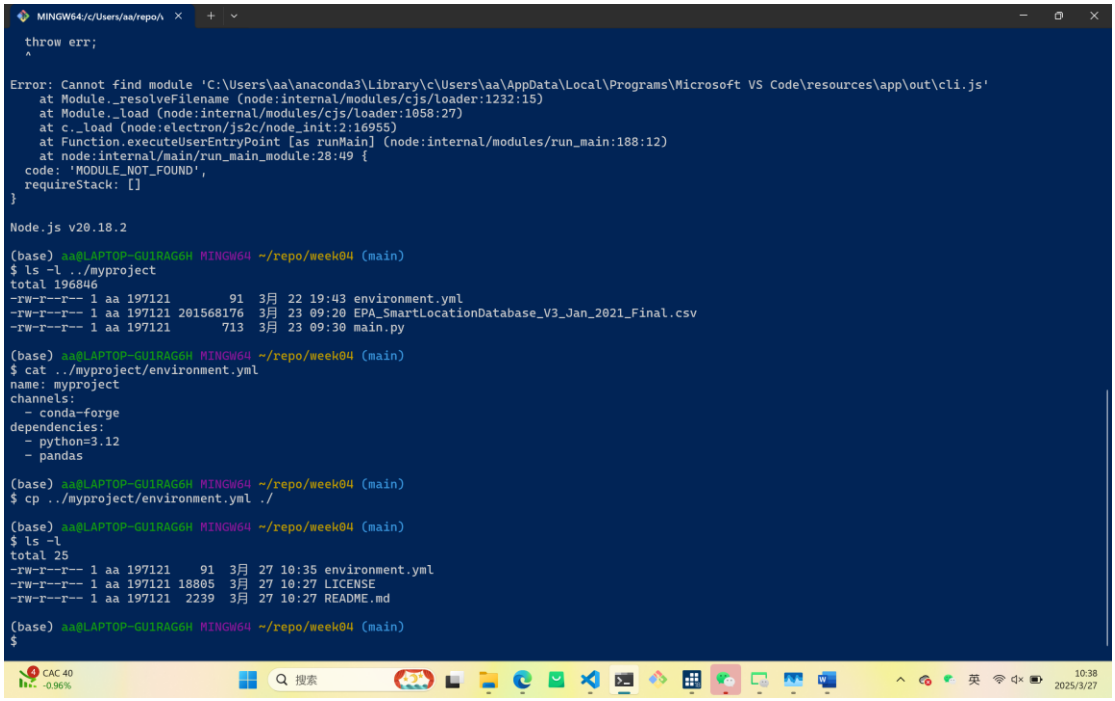
1. Fork 第 04 周打卡仓库至你的名下，然后将你名下的这个仓库 Clone 到你的本地计算机



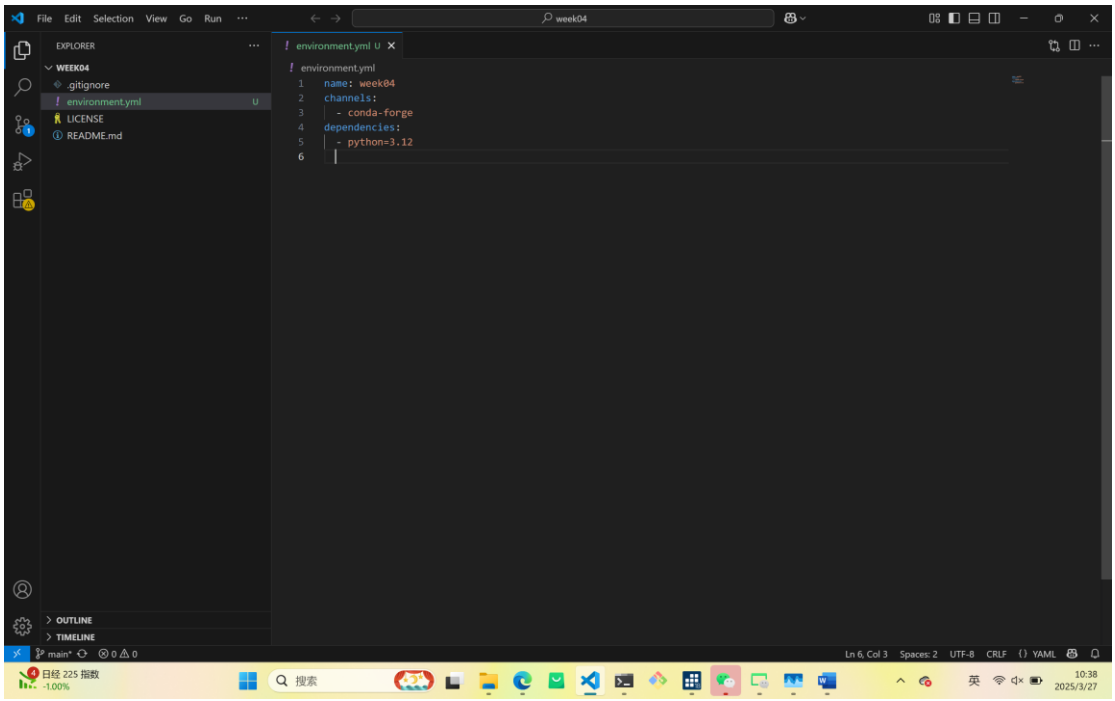


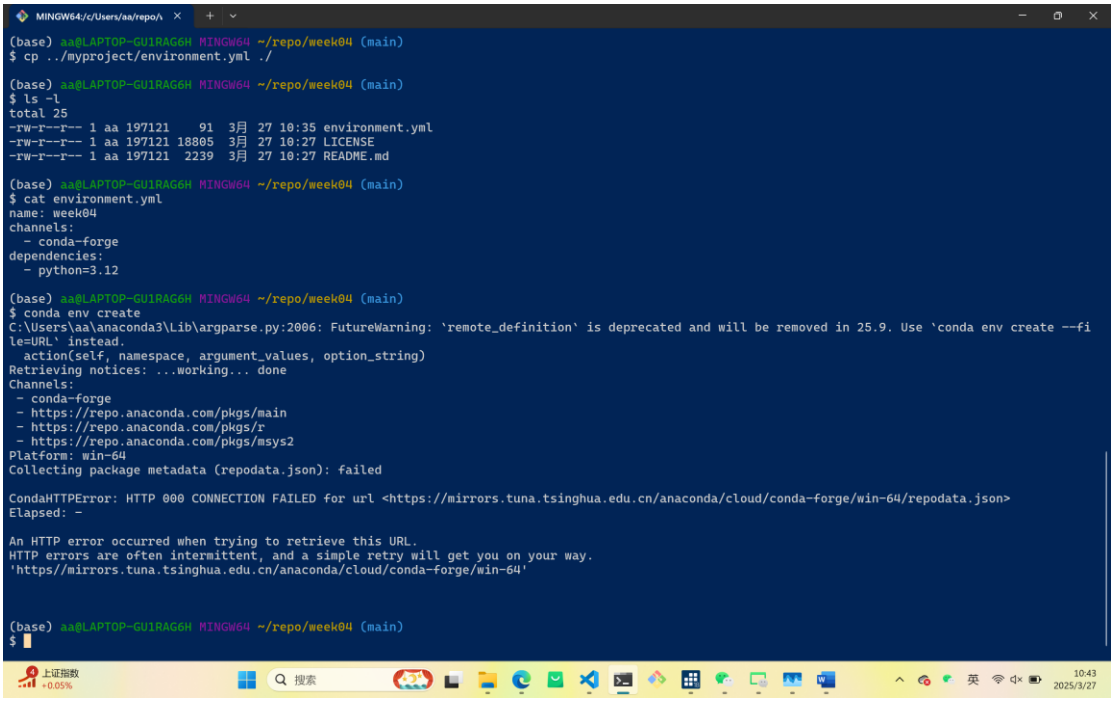


2. 用 VS Code 打开项目目录，新建一个 environment.yml 文件

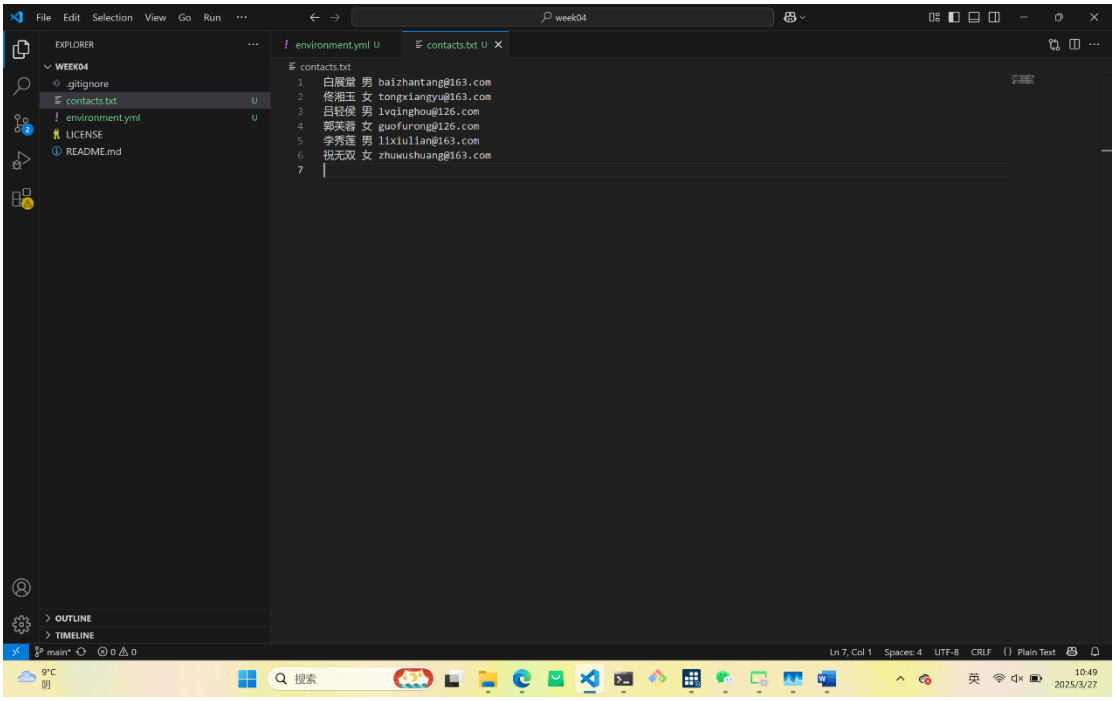


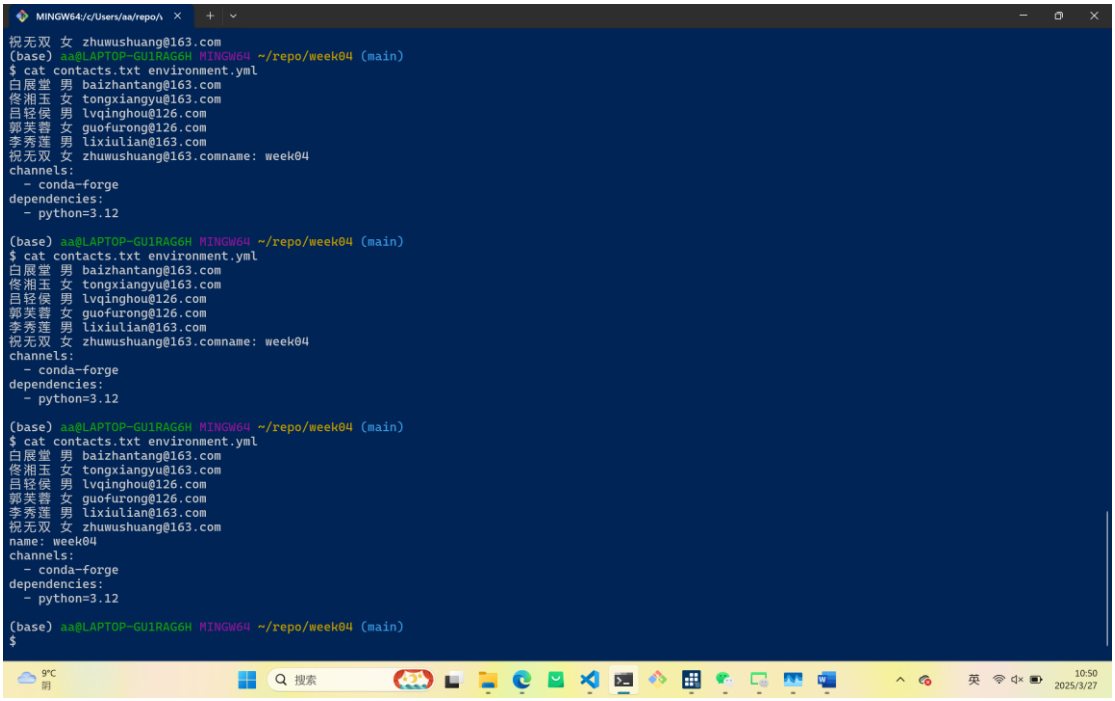
指定安装 Python 3.12



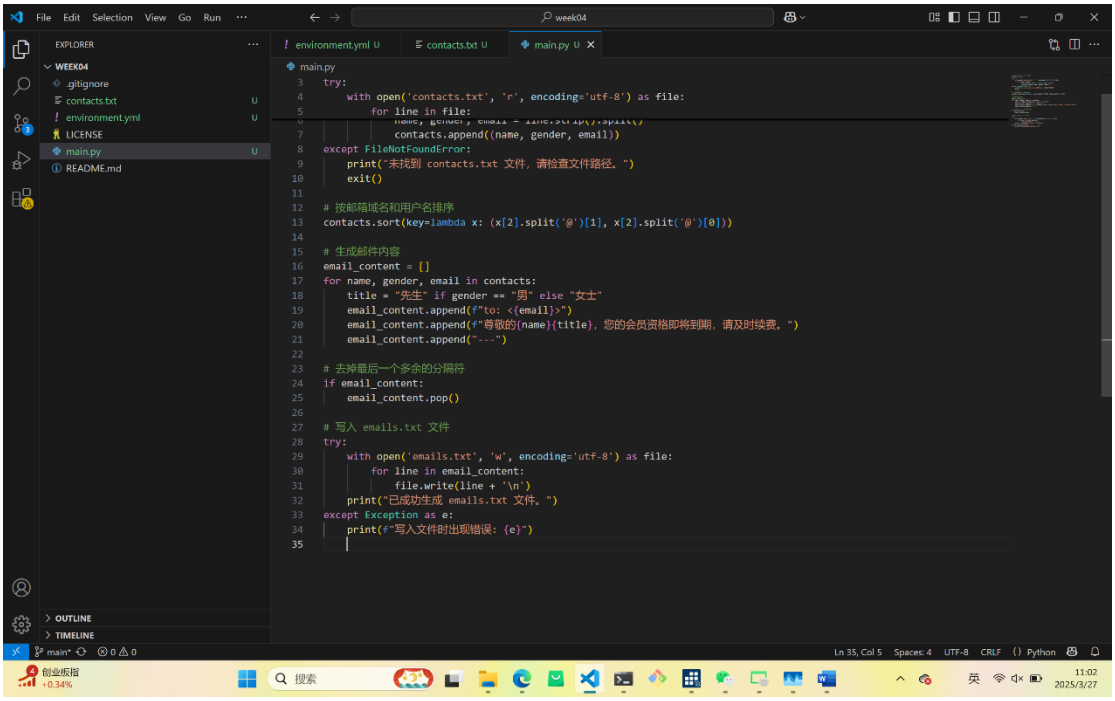


3.新建一个 contacts.txt 文件，每行写一个联系人，每个联系人都包含姓名、性别、邮箱三个字段，用空格分隔



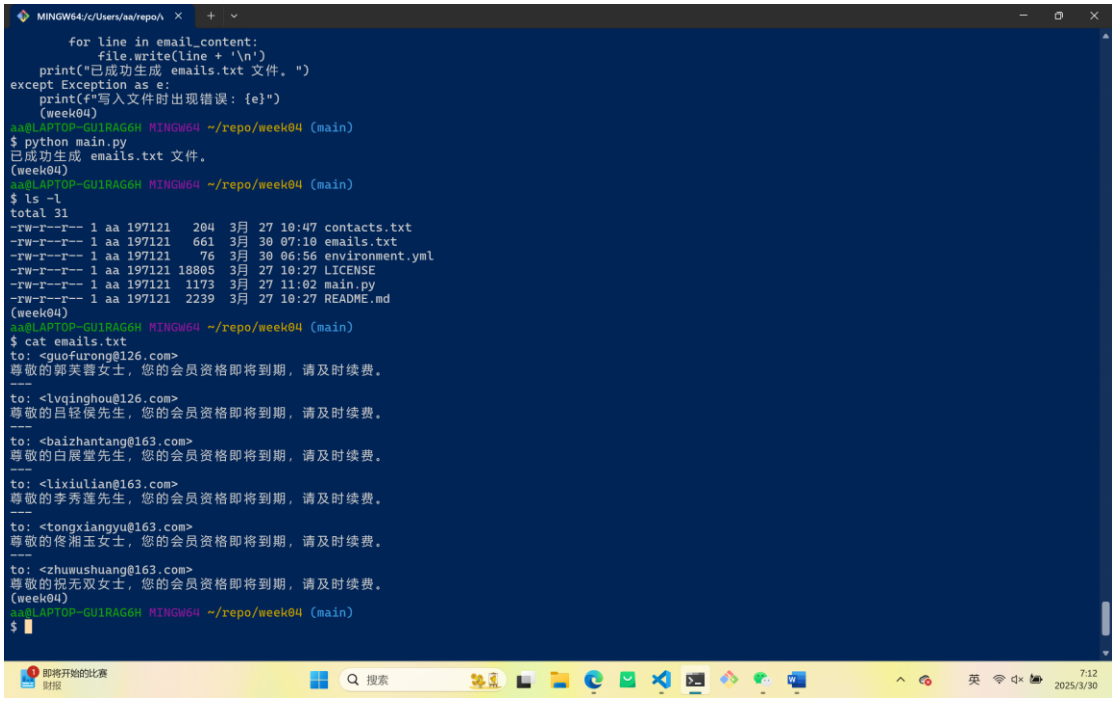


新建一个 main.py 文件，里面写 Python 代码，要求读取 contacts.txt 文件的内容，进行数据处理后，输出一个 emails.txt 文件



6. 在 Conda 环境里运行代码将大模型提供的代码复制粘贴进 main.py 文件，保存

运行 python main.py 命令



先删除之前生成的 emails

```
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week04 (main)
$ rm emails.txt
(week04)
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week04 (main)
$ ls -l
total 30
-rw-r--r-- 1 aa 197121   204  3月 27 10:47 contacts.txt
-rw-r--r-- 1 aa 197121    76  3月 30 06:56 environment.yml
-rw-r--r-- 1 aa 197121 18805  3月 27 10:27 LICENSE
-rw-r--r-- 1 aa 197121  1173  3月 27 11:02 main.py
-rw-r--r-- 1 aa 197121  2239  3月 27 10:27 README.md
(week04)
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week04 (main)
$
```

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）

```
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\aa\repo\week04\main.py(2)<module>()
-> contacts = []
(Pdb)
```

l (显示代码)

```
(Pdb) l
  1     # 读取 contacts.txt 文件
  2     contacts = []
  3  -> try:
  4         with open('contacts.txt', '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("未找到 contacts.txt 文件，请检查文件路径。")
 10         exit()
 11
```

n (执行当前行)

```
(Pdb) n
> c:\users\aa\repo\week04\main.py(5)<module>()
-> for line in file:
```

p (打印表达式)，p 后面跟的是任意一个表达式

c (继续执行)

```
(Pdb) c
已成功生成 emails.txt 文件。
The program finished and will be restarted
> c:\users\aa\repo\week04\main.py(2)<module>()
-> contacts = []
(Pdb)
```

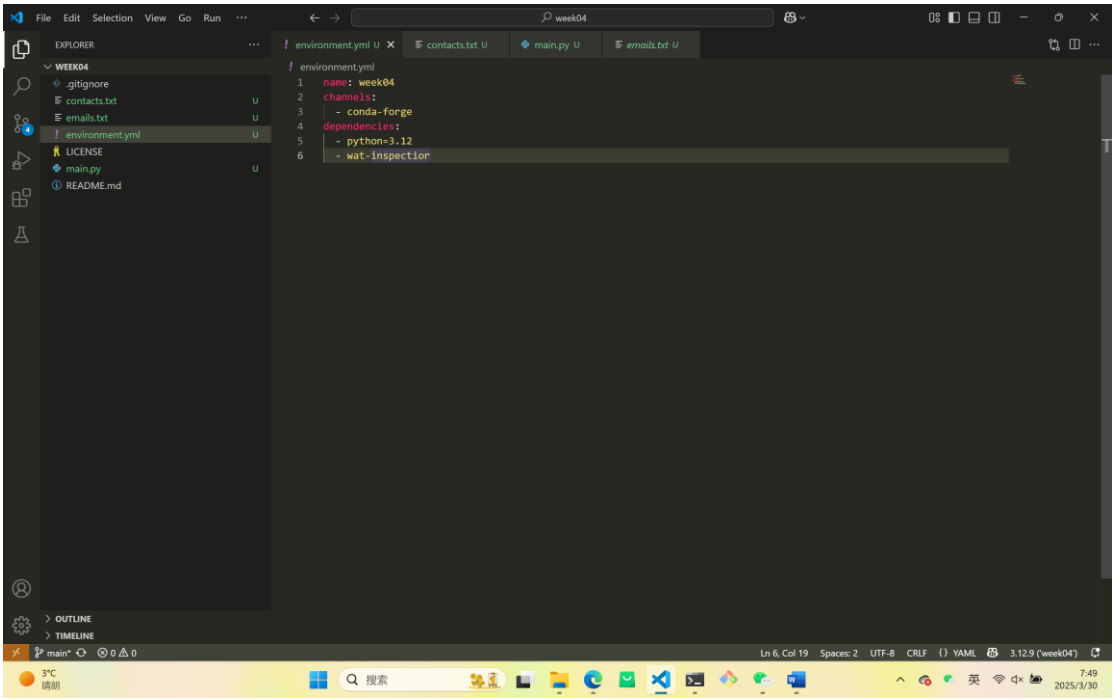
L .显示上下五行

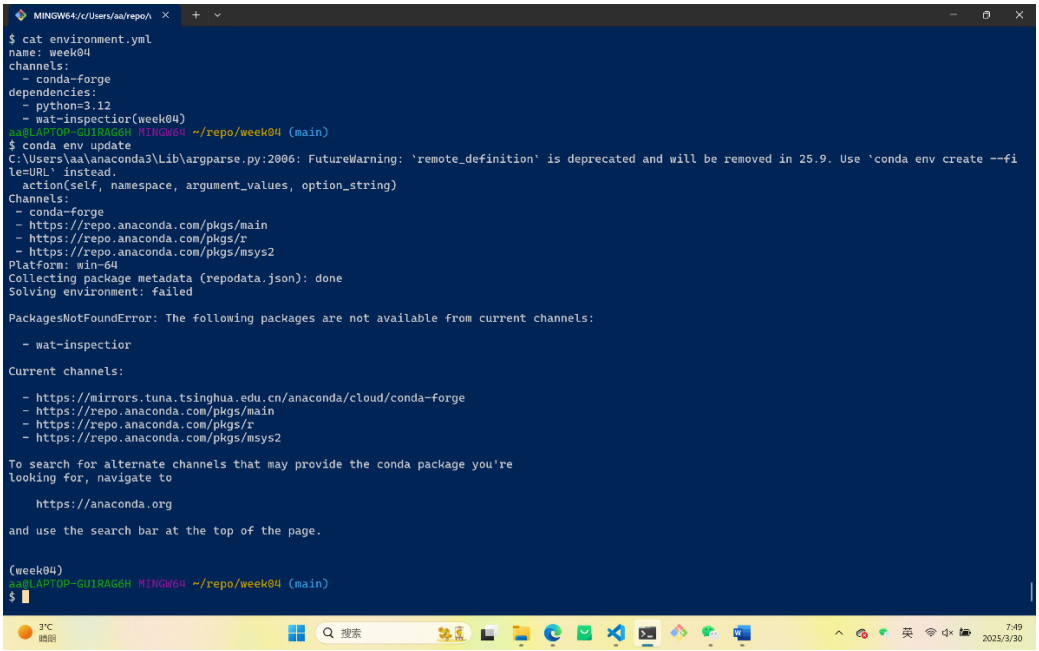
```
(Pdb) l .
  2     contacts = []
  3     try:
  4         with open('contacts.txt', 'r', encoding='utf-8') as file:
  5             for line in file:
  6                 name, gender, email = line.strip().split()
  7  ->             contacts.append((name, gender, email))
  8     except FileNotFoundError:
  9         print("未找到 contacts.txt 文件，请检查文件路径。")
 10         exit()
 11
 12     # 按邮箱域名和用户名排序
```

L 1,5 显示 1-5 行

```
(Pdb) l 1,5
  1     # 读取 contacts.txt 文件
  2     contacts = []
  3     try:
  4         with open('contacts.txt', 'r', encoding='utf-8') as file:
  5             for line in file:
(Pdb)
```

安装 wat-inspector





理解以下 Python 基本概念

（1）Python 语法保留字 (reserved key words)

Python 语法保留字是被 Python 语言赋予了特定含义和用途的单词，它们是 Python 语法的重要组成部分。在编写代码时，不能将保留字用作变量名、函数名或其他标识符，否则会引发语法错误。例如，if 用于条件判断语句，while 用于循环语句，def 用于定义函数等。

（2）语句 (statement) 和表达式 (expression)

语句 (statement)：是 Python 程序中执行操作的基本单元，它可以改变程序的状态、控制程序的流程或者执行特定的任务。语句通常以换行符或分号（较少使用）结束。常见的语句包括赋值语句、条件语句、循环语句、函数定义语句等。例如，赋值语句用于给变量赋值，条件语句根据条件的真假执行不同的代码块，循环语句用于重复执行一段代码。

表达式 (expression)：是由值、变量、运算符和函数调用等组成的组合，它会计算出一个结果。表达式本身不会改变程序的状态，只是用于计算值。例如，算术表达式 2 + 3 会计算出结果 5，函数调用表达式 len('hello') 会计算出字符串 'hello' 的长度 5。

（3）缩进 (indent)

在 Python 中，缩进是用来表示代码块的层次结构的，而不像其他一些编程语言使用大括号 {} 来界定代码块。Python 强制要求使用一致的缩进方式来表示代码的逻辑结构，通常建议使用 4 个空格作为一个缩进级别。缩进相同的代码行属于同一个代码块，例如在函数定义、条件语句和循环语句中，缩进的代码行构成了相应语句的执行体。

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

局部变量 (local variable)：是在函数内部定义的变量，其作用域仅限于定义它的函数内部。当函数执行结束后，局部变量所占用的内存会被释放，在函数外部无法访问该局部变量。

全局变量 (global variable)：是在函数外部定义的变量，其作用域是整个程序。全局变量可以在程序的任何地方被访问，但如果要在函数内部修改全局变量的值，需要使用 global 关键字进行声明。

LEGB 规则：是 Python 查找变量的顺序规则。当在代码中引用一个变量时，Python 会按照以下顺序依次查找该变量：

局部作用域（Local）：当前函数内部的作用域。

封闭作用域（Enclosing）：如果当前函数是嵌套函数，那么其外层函数的作用域。

全局作用域（Global）：整个模块的作用域，即文件级别的作用域。

内置作用域（Built-in）：Python 内置的函数和变量所在的作用域。

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

函数定义 (define)：是创建函数的过程，使用 def 关键字来定义一个函数。函数定义包括函数名、参数列表、冒号和函数体。函数体是一组缩进的代码行，用于实现函数的具体功能。函数可以接受输入参数，并可以返回一个或多个值。

函数调用 (call)：是使用已定义的函数的过程。通过函数名和传递相应的参数来调用函数，函数会执行其函数体中的代码，并可以返回一个结果。调用函数时，传递的参数会被赋值给函数定义中的参数，函数执行完毕后，可能会返回一个值供调用者使用。

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

字面值 (literal)：是在代码中直接表示的数据值。不同类型的数据有不同的字面值表示方式。

字符串 (str)：是由零个或多个字符组成的序列，通常用单引号 ' 或双引号 " 括起来，例如 'hello' 或 "world"。

整数 (int)：表示整数值，如 1、100、-5 等。

列表 (list)：是一种可变的有序序列，用方括号 [] 表示，列表中的元素可以是不同类型的数据，例如 [1, 'a', True]。

字典 (dict)：是一种无序的键值对集合，用花括号 {} 表示，每个键值对由键和值组成，键和值之间用冒号 : 分隔，键必须是唯一的，例如 {'name': 'John', 'age': 30}。

元组 (tuple)：是一种不可变的有序序列，用圆括号 () 表示，元组中的元素可以是不同类型的数据，例如 (1, 'b', False)。

（6）运算符 (operator)

运算符是用于对操作数进行运算的符号。Python 提供了多种类型的运算符，包括算术运算符（如 +、-、*、/ 等）、比较运算符（如 ==、>、< 等）、逻辑运算符（如 and、or、not 等）、赋值运算符（如 =、+=、-= 等）、位运算符（如 &、|、^ 等）等。运算符可以结合操作数构成表达式，用于进行各种计算和逻辑判断。

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

形参 (parameter)：是在函数定义时声明的参数，它是函数接收输入值的占位符。

形参定义了函数期望接收的参数的名称和类型（在 Python 中，参数类型通常是动态的）。例如，在函数定义 def add(a, b) 中，a 和 b 就是形参。

实参 (argument)：是在函数调用时传递给函数的实际值。实参的值会被赋值给函数定义中的形参，从而使函数可以使用这些值进行计算。例如，在函数调用 add(3, 5) 中，3 和 5 就是实参。

返回值 (return value)：是函数执行完毕后返回给调用者的结果。函数可以使用 return 语句来指定返回值，如果函数没有显式使用 return 语句，则默认返回 None。返回值可以是任何类型的数据，调用者可以使用变量来接收返回值，以便在后续代码中使用。

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

对象 (object)：在 Python 中，一切皆对象。对象是数据和操作这些数据的方法的集合。每个对象都有自己的身份（可以通过 id() 函数查看）、类型和值。例如，整数、字符串、列表等都是对象。

类型 (type)：每个对象都属于一个特定的类型，类型定义了对象的行为和操作。例如，整数对象属于 int 类型，字符串对象属于 str 类型。可以使用 type() 函数来查看对象的类型。

属性 (attribute)：是对象所拥有的变量，用于存储对象的状态信息。属性可以是数据属性（存储数据值）或方法属性（存储函数）。可以通过点号 . 来访问对象的属性，例如 obj.attr。

方法 (method)：是对象所拥有的函数，用于执行特定的操作。方法与对象的类型相关，不同类型的对象可能有不同的方法。可以通过点号 . 来调用对象的方法，例如 obj.method()。方法通常会对对象本身进行操作或返回一个结果。