```
(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo
$ cd week04

(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ pwd
/c/Users/ZMX/repo/week04

(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ git remote show origin
* remote origin
  Fetch URL: https://gitcode.com/Zzunkn/week04.git
  Push  URL: https://gitcode.com/Zzunkn/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) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ ls -l
total 25
-rw-r--r-- 1 ZMX 197121    87  3月 29 10:10 environment.yml
-rw-r--r-- 1 ZMX 197121 18805  3月 29 10:05 LICENSE
-rw-r--r-- 1 ZMX 197121  2239  3月 29 10:05 README.md
```

```
(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ ls -l
total 25
-rw-r--r-- 1 ZMX 197121    72  3月 29 10:11 environment.yml
-rw-r--r-- 1 ZMX 197121 18805  3月 29 10:05 LICENSE
-rw-r--r-- 1 ZMX 197121  2239  3月 29 10:05 README.md

(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ cat environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ conda env create
D:\anaconda3\Lib\argparse.py:2006: FutureWarning: `remote_definition` is depr
nda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - https://repo.anaconda.com/pkgs/main
```

```
  - python=3.12
(base) ZMX@LAPTOP-QCK3FO52 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) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo
$ cd week04

(base) ZMX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
$ conda activate week04
(week04)
```

environment.yml U | contacts.txt U | main.py U | emails.txt U

emails.txt

```
1  to: <guofurong@126.com>
2  尊敬的郭芙蓉女士，您的会员资格即将到期，请及时续费。
3  ---
4  to: <lvqinghou@126.com>
```

MINGW64:/c/Users/ZMX/repc

```
  contacts = []
  for line in lines:
      name, gender, email = line.strip().split()
      contacts.append((name, gender, email))

  # 按邮箱域名和用户名排序
  sorted_contacts = sorted(contacts, key=lambda x: (x[2].split('@')[1], x[2].spli

  # 生成邮件内容
  email_content = []
  for name, gender, email in sorted_contacts:
      title = "先生" if gender == "男" else "女士"
      email_text = f"to: <{email}>\n尊敬的{name}{title}，您的会员资格即将到期，请
      email_content.append(email_text)

  # 写入 emails.txt 文件
  with open('emails.txt', 'w', encoding='utf-8') as output_file:
      output_file.write('\n'.join(email_content).rstrip('---'))

xcept FileNotFoundError:
  print("错误：contacts.txt 文件未找到！")
xcept Exception as e:
  print(f"错误：发生了一个未知错误：{e}")
  (week04)
IX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
 python main.py
eek04)
IX@LAPTOP-QCK3FO52 MINGW64 ~/repo/week04 (main)
```

Python 解释器

以下是对这些 Python 基本概念的详细解释：

1. Python 语法保留字 (reserved key words) 保留字是 Python 语言中具有特殊意义的单词，你不能将它们用作变量名、函数名或其他标识符。例如 `if`、`else`、`for`、`while` 等。以下是一个简单的例子，展示使用保留字作为变量名会引发错误： ```python # 错误示例，if 是保留字，不能作为变量名 # if = 1 ```

2. 语句 (statement) 和表达式 (expression) - **语句**：是执行某种操作的代码单元，它可以改变程序的状态。例如赋值语句、`if` 语句、`for` 语句等。 ```python # 赋值语句 x = 5 # if 语句 if x > 3: print("x 大于 3") ``` - **表达式**：是产生值的代码片段，它可以由变量、字面

值、运算符和函数调用组成。 ```python # 表达式 result = 2 + 3 # 2 + 3 是表达式 ```

3. 缩进 (indent) 在 Python 中，缩进用于表示代码块。Python 不像其他语言使用大括号 `{}` 来界定代码块，而是使用缩进来区分不同的代码块。通常使用 4 个空格作为一个缩进层级。 ```python if True: # 这个代码块缩进了 4 个空格 print("条件为真") ```

4. 局部变量 (local variable)、全局变量 (global variable)、LEGB 规则 - **局部变量**：在函数内部定义的变量，只能在函数内部访问。 ```python def my_function(): # 局部变量 local_var = 10 print(local_var) my_function() # 下面这行代码会报错，因为 local_var 是局部变量，在函数外部无法访问 # print(local_var) ``` - **全局变量**：在函数外部定义的变量，可以在整个程序中访问，但在函数内部修改全局变量需要使用 `global` 关键字。 ```python # 全局变量 global_var = 20 def modify_global(): global global_var global_var = 30 modify_global() print(global_var) # 输出 30 ``` - **LEGB 规则**：是 Python 查找变量的顺序，即 Local（局部作用域）、Enclosing（闭包作用域）、Global（全局作用域）、Built-in（内置作用域）。Python 会按照这个顺序依次查找变量。

5. 函数 (function) 的定义 (define) 和调用 (call) - **定义函数**：使用 `def` 关键字来定义函数，函数可以有参数和返回值。 ```python def add_numbers(a, b): return a + b ``` - **调用函数**：通过函数名和传递参数来调用函数。 ```python result = add_numbers(3, 5)

print(result) # 输出 8 ``` ###

6. 字面值 (literal) 字面值是直接在代码中表示数据的常量。常见的字面值类型有： - **字符串 (str)**：用单引号或双引号括起来的文本。 ```python my_string = "Hello, World!" ``` - **整数 (int)**：表示整数的数值。 ```python my_int = 10 ``` - **列表 (list)**：用方括号括起来的有序元素集合。 ```python my_list = [1, 2, 3] ``` - **字典 (dict)**：用花括号括起来的键值对集合。 ```python my_dict = {'name': 'John', 'age': 25} ``` - **元组 (tuple)**：用圆括号括起来的有序元素集合，不可变。```python my_tuple = (1, 2, 3) ```

7. 运算符 (operator) 运算符用于对数据进行操作。常见的运算符有算术运算符（`+`、`-`、`*`、`/` 等）、比较运算符（`==`、`!=`、`>`、`<` 等）、逻辑运算符（`and`、`or`、`not`）等。 ```python # 算术运算符 a = 5 b = 3 result = a + b # 加法运算 # 比较运算符 is_equal = a == b # 比较是否相等 # 逻辑运算符 is_true = (a > b) and (b > 0) # 逻辑与运算 ```

8. 形参 (parameter)、实参 (argument)、返回值 (return value) - **形参**：在函数定义时指定的参数，用于接收调用函数时传递的值。 ```python def multiply(a, b): # a 和 b 是形参 return a * b ``` - **实参**：在调用函数时传递给函数的具体值。 ```python result = multiply(2, 3) # 2 和 3 是实参 ``` - **返回值**：函数执行完毕后返回的结果，使用 `return` 语句返回。 ```python def square(x): return x * x result = square(4) # 函数返回值为 16 ```

9. 对象 (object)、类型 (type)、属性 (attribute)、方法 (method) - **对象**：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。 - **类型**：对象的类型决定了对象可以执行的操作和具有的属性。可以使用 `type()` 函数查看对象的类型。 ```python my_list = [1, 2, 3] print(type(my_list)) # 输出 <class 'list'> ``` - **属性**：对象的属性是与对象关联的数据。例如，列表对象有 `length` 属性表示列表的长度。 ```python my_list = [1, 2, 3] length = len(my_list) # len() 函数获取列表的长度属性 ``` - **方法**：对象的方法是与对象关联的函数，用于执行特定的操作。例如，列表对象有 `append()` 方法用于在列表末尾添加元素。 ```python my_list = [1, 2, 3] my_list.append(4) # 调用 append() 方法添加元素 print(my_list) # 输出 [1, 2, 3, 4] ``` 这些基本概念是 Python 编程的基础，理解它们对于学习和使用 Python 非常重要。

下面将逐行解释你提供的代码，说明其每一步的作用。 ```python try: # 读取 contacts.txt 文件 with open('contacts.txt', 'r', encoding='utf-8') as file: lines = file.readlines() ``` - 这里使用了`try`语句块，目的是捕获可能出现的异常。在这个代码块里执行的操作一旦出现异常，程序就会跳转到对应的`except`语句块进行处理。 - `with open('contacts.txt', 'r', encoding='utf-8') as file`：这行代码以只读模式（`'r'`）打开名为`contacts.txt`的文件，并且指定文件编码为`utf-8`。`with`语句会自动管理文件的打开和关闭，保证文件使用完毕后能正确关闭。 - `lines =

file.readlines()`：调用`readlines()`方法把文件中的所有行读取出来，存于列表`lines`中，列表里的每个元素对应文件里的一行。 ```python contacts = [] for line in lines: name, gender, email = line.strip().split() contacts.append((name, gender, email)) ``` - `contacts = []`：初始化一个空列表`contacts`，用于存储从文件中读取到的联系人信息。- `for line in lines:`：对`lines`列表进行遍历，`line`代表文件中的每一行。 - `name, gender, email = line.strip().split()`：`strip()`方法用于去除行首尾的空白字符，`split()`方法以空白字符为分隔符对行进行分割。分割后的结果会分别赋值给`name`、`gender`和`email`这三个变量。 - `contacts.append((name, gender, email))`：把解析得到的联系人信息以元组的形式添加到`contacts`列表中。 ```python # 按邮箱域名和用户名排序 sorted_contacts = sorted(contacts, key=lambda x: (x[2].split('@')[1], x[2].split('@')[0])) ``` - `sorted(contacts, key=lambda x: (x[2].split('@')[1], x[2].split('@')[0]))`：使用`sorted()`函数对`contacts`列表进行排序。`key`参数是一个匿名函数`lambda x: (x[2].split('@')[1], x[2].split('@')[0])`，其作用是提取联系人邮箱的域名和用户名，先按照域名排序，若域名相同则按照用户名排序。排序后的结果存储在`sorted_contacts`列表中。 ```python # 生成邮件内容 email_content = [] for name, gender, email in sorted_contacts: title = "先生" if gender == "男" else "女士" email_text = 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---" email_content.append(email_text) ``` - `email_content = []`：初始化一

个空列表`email_content`，用于存储生成的邮件内容。 - `for name, gender, email in sorted_contacts:`：对排序后的联系人列表进行遍历。 - `title = "先生" if gender == "男" else "女士"`：运用条件表达式来确定称呼，若性别为“男”，称呼就是“先生”，否则为“女士”。 - `email_text = 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---"`：使用 f-string 格式化字符串，生成邮件文本内容。 - `email_content.append(email_text)`：把生成的邮件内容添加到`email_content`列表中。 ```python # 写入 emails.txt 文件 with open('emails.txt', 'w', encoding='utf-8') as output_file: output_file.write('\n'.join(email_content).rstrip('---')) except FileNotFoundError: print("错误：contacts.txt 文件未找到!") except Exception as e: print(f"错误：发生了一个未知错误：{e}") ``` - `with open('emails.txt', 'w', encoding='utf-8') as output_file`：以写入模式（`'w'`）打开`emails.txt`文件，指定编码为`utf-8`。 - `output_file.write('\n'.join(email_content).rstrip('---'))`：将`email_content`列表中的元素用换行符连接成一个字符串，然后使用`rstrip('---')`方法去除字符串末尾的`---`，最后把处理后的字符串写入文件。 - `except FileNotFoundError:`：捕获`FileNotFoundError`异常，若`contacts.txt`文件不存在，就输出错误信息。 - `except Exception as e:`：捕获其他所有异常，输出具体的错误信息。