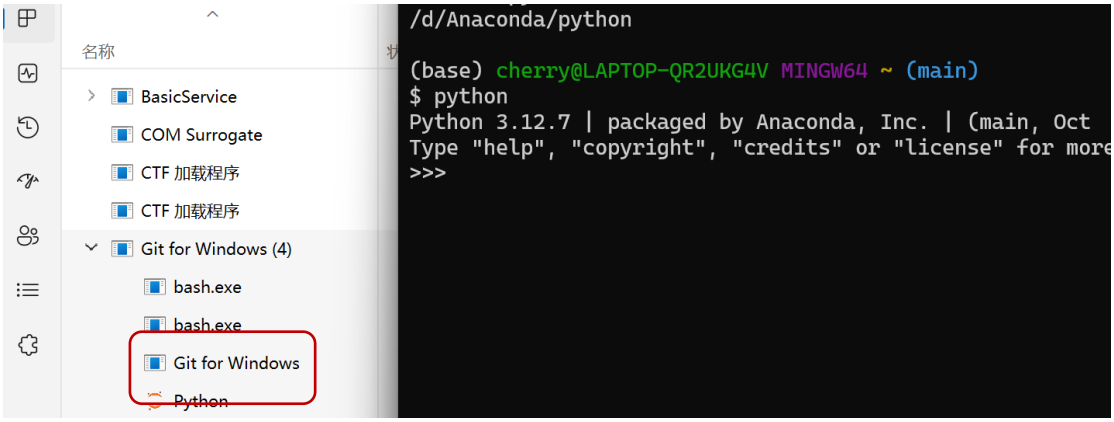
第 4 周 Python 数据类型 (初级)

1.进程



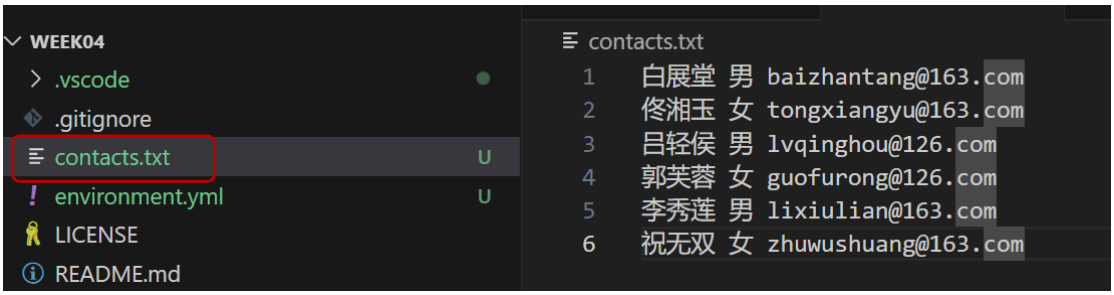

2.fork 第 04 周打卡仓库至我名下，并 clone 至本地

```
(base) cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo (main)
$ git clone https://gitcode.com/Wxy496417/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 | 663.00 KiB/s, done.
```

3.创建 conda 环境

```
(base) cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week04 (main)
$ cat environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
(base) cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week04 (main)
$ conda env create
```

4.新建一个 contacts.txt 文件




在终端中显示

```
(base) cherry@LAPTOP-QR2UKG4V 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
```

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，并按要求排序。

- 交给大模型生成静态代码

```python
main.py > ...
def process_contacts():
    try:
        # 读取 contacts.txt 文件
        with open('contacts.txt', 'r', encoding='utf-8') as file:
            contacts = file.readlines()

        # 解析联系人信息
        parsed_contacts = []
        for contact in contacts:
            name, gender, email = contact.strip().split()
            parsed_contacts.append((name, gender, email))

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

        # 生成邮件内容
```

- 在终端中运行

```
if __name__ == "__main__":
    process_contacts()
(week04)
cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week04 (main)
$ python main.py
邮件通知文件生成成功！
```

- 查看运行结果

```
cherry@LAPTOP-QR2UKG4V 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。
```

- 对代码进行调试

```
cherry@LAPTOP-QR2UKG4V MINGW64 ~/repo/week04 (main)
$ python -m pdb main.py
> c:\users\cherry\repo\week04\main.py(1)<module>()
-> def process_contacts():
(Pdb)
```

先运行 pdb 即调试器，再由调试器家在后面的代码

显示路径，第 1 行

```
(Pdb) l
  1  -> def process_contacts():
  2         try:
  3             # 读取 contacts.txt 文件
  4             with open("contacts.txt", "r", encoding="utf-8") as file:
  5                 contacts = file.readlines()
  6
  7             # 解析联系人信息
  8             parsed_contacts = []
  9             for contact in contacts:
 10                 name, gender, email = contact.strip().split()
 11                 parsed_contacts.append((name, gender, email))
(Pdb) Z
```

箭头指向的是即将运行但还没有运行的代码（代码的第一行）

- l 命令：查看代码运行的位置；n 命令：执行当前行；p 命令：打印/显示表达式

```
(Pdb) p process_contacts
<function process_contacts at 0x0000021FF6ACF920>
```

定义了一个函数

- 定义函数括号中的内容是形参，只有在这个函数调用以后在这个函数内部才会有

```
(Pdb) l 2,10
  2         try:
  3             # 读取 contacts.txt 文件
  4             with open("contacts.txt", "r", encoding="utf-8") as file:
  5                 contacts = file.readlines()
  6
  7             # 解析联系人信息
  8             parsed_contacts = []
  9             for contact in contacts:
 10                 name, gender, email = contact.strip().split()
```

查看第 2 到 10 行代码

```
 41     if __name__ == "__main__":
 42  ->     process_contacts()
[EOF]
(Pdb) p __name__
'__main__'
```

变量名

是这个字符串

```
(Pdb) s
> c:\users\cherry\repo\week04\main.py(42)<module>()
-> process_contacts()
(Pdb)
--Call--
> c:\users\cherry\repo\week04\main.py(1)process_contacts()
-> def process_contacts():
(Pdb) n
> c:\users\cherry\repo\week04\main.py(2)process_contacts()
-> try:
```

- 调用这个函数，此时使用 n 命令会跳到第二行

```
(Pdb) n
> c:\users\cherry\repo\week04\main.py(4)process_contacts()
-> with open("contacts.txt", "r", encoding="utf-8") as file:
(Pdb) p contacts
['白展堂 男 baizhantang@163.com\n', '佟湘玉 女 tongxiangyu@163.com\n', '吕轻侯 男 lvqinghou@126.com\n', '郭芙蓉 女 guofu
rong@126.com\n', '李秀莲 男 lixiulian@163.com\n', '祝无双 女 zhuwushuang@163.com\n']
(Pdb) pp contacts
['白展堂 男 baizhantang@163.com\n',
 '佟湘玉 女 tongxiangyu@163.com\n',
 '吕轻侯 男 lvqinghou@126.com\n',
 '郭芙蓉 女 guofurong@126.com\n',
 '李秀莲 男 lixiulian@163.com\n',
 '祝无双 女 zhuwushuang@163.com\n']
```

美观排列

```
  8             parsed_contacts = []
  9             for contact in contacts:
 10  ->              name, gender, email = contact.strip().split()
 11                  parsed_contacts.append((name, gender, email))
 12
 13             # 按邮箱域名和用户名排序
 14             sorted_contacts = sorted(
 15                 parsed_contacts, key=lambda x: (x[2].split("@")[1], x[
(Pdb) p contact
'白展堂 男 baizhantang@163.com\n'
```

循环语句

```
(Pdb) n
> c:\users\cherry\repo\week04\main.py(9)process_contacts()
-> for contact in contacts:
(Pdb) p contact
'白展堂 男 baizhantang@163.com\n'
(Pdb) n
> c:\users\cherry\repo\week04\main.py(10)process_contacts()
-> name, gender, email = contact.strip().split()
(Pdb) p contact
'佟湘玉 女 tongxiangyu@163.com\n'
```

- 循环语句，从第一个人开始循环

```
(Pdb) p name
'佟湘玉'
(Pdb) p gendar
*** NameError: name 'gendar' is not defined
(Pdb) p gender
'女'
(Pdb) p email
'tongxiangyu@163.com'
(Pdb)
```

- 字符串

```
(Pdb) p contact
'佟湘玉 女 tongxiangyu@163.com\n'
(Pdb) p contact.strip()
'佟湘玉 女 tongxiangyu@163.com'
(Pdb) p contact.strip().split()
['佟湘玉', '女', 'tongxiangyu@163.com']
```

拆成了一个列表，分成了三个字符串构成的列表

```
(Pdb) pp parsed_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', 'lixiulian@163.com'),
 ('祝无双', '女', 'zhuwushuang@163.com')]
```

- for 循环的最终运行结果

```
13          # 按邮箱域名和用户名排序
14          sorted_contacts = sorted(
15  ->          parsed_contacts, key=lambda x: (x[2].split("@")[1], x[2].split("@")[0])
16          )
17
```

排序

- 排序结果

```
(Pdb) pp sorted_contacts
[('郭芙蓉', '女', 'guofurong@126.com'),
 ('吕轻侯', '男', 'lvqinghou@126.com'),
 ('白展堂', '男', 'baizhantang@163.com'),
 ('李秀莲', '男', 'lixiulian@163.com'),
 ('佟湘玉', '女', 'tongxiangyu@163.com'),
 ('祝无双', '女', 'zhuwushuang@163.com')]
```

```
(Pdb) c
邮件通知文件生成成功!
The program finished and will be restarted
> c:\users\cherry\repo\week04\main.py(1)<module>()
-> def process_contacts():
```

一口气运行完，代码没有报错，调试器又要从第一行开始运行，按 q 退出调试器

- 安装 wat-inspector 第三方软件包

environment.yml U × | contacts.txt U | m

environment.yml

```
1   name: week04
2   channels:
3     - conda-forge
4   dependencies:
5     - python=3.12
6     - wat-inspector
```

- 学习理解基本的 python 概念

1) Python 语法保留字（def、代码中的紫色部分，在 python 语法中有特殊含义）：

```
>>> name = 'cherry'
>>> print(name)
cherry
>>> def = 'cherry'
  File "<stdin>", line 1
    def = 'cherry'
        ^
SyntaxError: invalid syntax
>>> which = 'cherry'
>>> print(which)
cherry
```

Define 是保留字，不能作为变量名，which 可以作为变量名

2) 语句和表达式：

语句：逻辑上完整的一句话，赋值语句、for 循环语句等

表达式：构成语句的一个元素，语句中的某些词。语句可以嵌套子语句，语句中有表达式，表达式也可以嵌套

```
for contact in contacts:
    name, gender, email = contact.strip().split()
    parsed_contacts.append((name, gender, email))
```

For 循环语句

表达式

3) 缩进：在 python 中有严格的对齐，def 语句和 def 语句对齐，for 循环语句和 for 循环语句对齐

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

# 按邮箱域名和用户名排序
sorted_contacts = sorted(
    parsed_contacts, key=lambda x: (x[2].split("@")[1], x[2].split("@")[0])
)
```

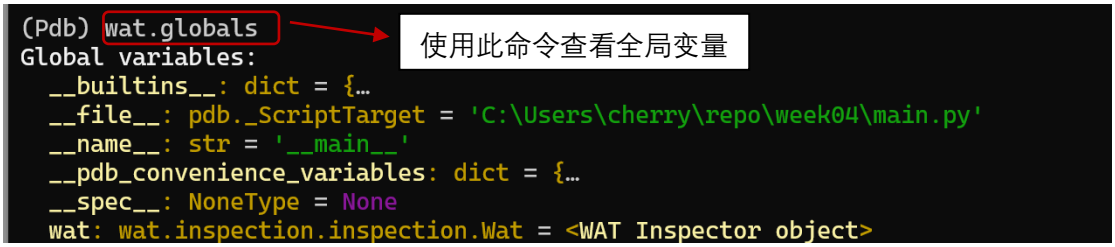
For 循环语句的子语句，要对齐

4) 局部变量、全局变量、LEGB 规则

```
Call wat.locals or wat() to inspect local variables.
Call wat.globals to inspect global variables.
```

用 wat.locals 或者 wat（）检查局部变量

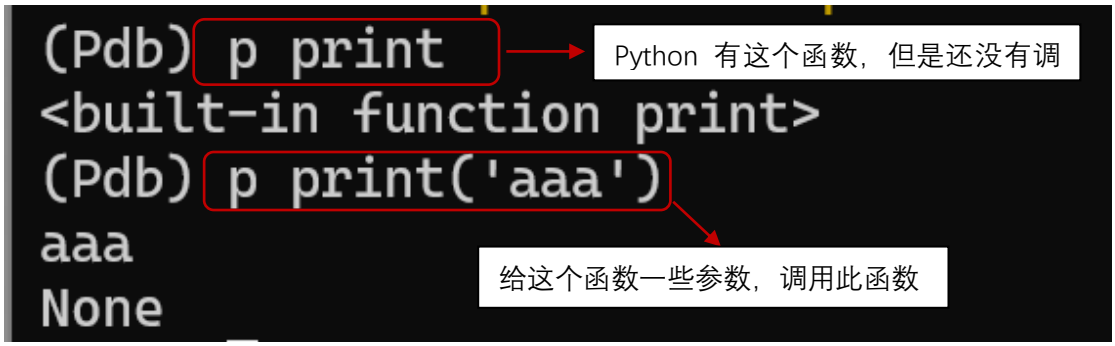

- 如果使用 s 命令调用函数，此时 wat（）显示此调用函数中的一些变量（局部变量）




- 通过 define 定义的变量，需要调用这个函数运行了之后才能查看的一些变量是局部变量，局部变量取决于代码运行到了哪里；而全局变量是一直可以查看的，与代码运行到哪里无关
- LEGB 规则：python 查找变量的顺序。L：局部作用域；E：闭包（嵌套）作用域，外层函数的局部变量；G：全局作用域；B：内置作用域

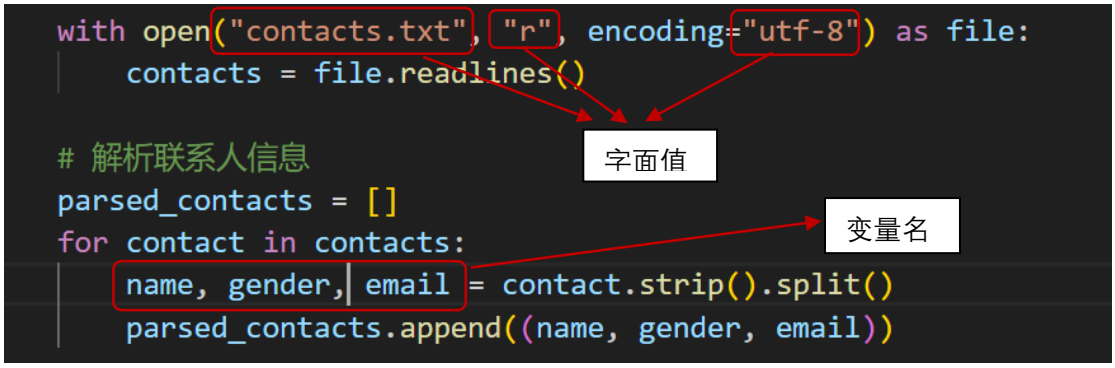
5) 函数 (function) 的定义 (define) 和调用 (call)

- 定义函数：def;
- 调用：

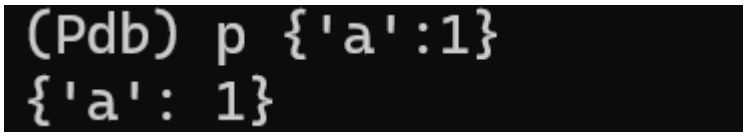



6) 字面值 (literal) (字符串 (str)、整数 (int)、列表 (list)、字典 (dict)、元组 (tuple))

- 字面值：



- 字典：大括号




7) 运算符

“=”是赋值语句，“==”才是运算符；if…else…也是运算符；“.”是名称访问运算符

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

- 形参是指在调用的时候会有值传进来，那个传进来的具体的值就是实参；return 后面的东西就是返回值，函数没有返回值就是 none

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

Python 内存管理的东西都是对象。Value:值;Type：类型（list 表示列表）；len：长度；

public attribute：公开的属性，方法是属性的一种。属性就是指一些特点，而方法是指这个类型的能做什么