

金融计算与编程第四周学习笔记

1. 把名下的仓库 Clone 到我的电脑。

```
MINGW64/c/Users/ASUS/repo/week04
drwxr-xr-x 1 ASUS 197121 0 Mar 19 20:27 week03/

(base) ASUS@%C:~$ MINGW64 ~/repo
$
$ git clone https://gitcode.com/wybfhm/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 | 287.00 KiB/s, done.

(base) ASUS@%C:~$ MINGW64 ~/repo
$ cd week04

(base) ASUS@%C:~/repo/week04 (main)
$ pwd
/c/Users/ASUS/repo/week04

(base) ASUS@%C:~/repo/week04 (main)
$ git remote show origin
* remote origin
```

2. 创建 conda 环境。Ctrl+E 可以直接切换到代码的最后，ctrl+A 切换到最前面。

```
MINGW64/c/Users/ASUS/repo/week04
channels:
- conda-forge
dependencies:
- python=3.12
- pandas
(base) ASUS@%C:~/repo/week04 (main)
$ cp ../myproject/environment.yml ./

(base) ASUS@%C:~/repo/week04 (main)
$ ls -l
total 25
-rw-r--r-- 1 ASUS 197121 18805 Mar 26 20:30 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Mar 26 20:30 README.md
-rw-r--r-- 1 ASUS 197121 88 Mar 26 20:46 environment.yml

(base) ASUS@%C:~/repo/week04 (main)
$ cat environment.yml
name: week04
channels:
- conda-forge
dependencies:
- python=3.12
- pandas
done
#
# To activate this environment, use
#
#     $ conda activate week04
#
# To deactivate an active environment, use
#
#     $ conda deactivate
(base) ASUS@%C:~/repo/week04 (main)
```

3. 文本文件的最后最好加一个空行。

```
MINGW64:/c/Users/ASUS/repo/week04
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.comname: week04
channels:
- conda-forge
dependencies:
- python=3.12
(base) ASUS@%C 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) ASUS@%C MINGW64 ~/repo/week04 (main)
$
```

4. 生成 txt 文件。

```
MINGW64:/c/Users/ASUS/repo/week04
print(f"写入文件时出错: {e}")

if __name__ == "__main__":
    contacts = read_contacts("contacts.txt")
    sorted_contacts = sort_contacts(contacts)
    emails = generate_emails(sorted_contacts)
    write_emails(emails, "emails.txt")
(week04)
ASUS@%C MINGW64 ~/repo/week04 (main)
$ python main.py
(week04)
ASUS@%C MINGW64 ~/repo/week04 (main)
$ ls -l
total 31
-rw-r--r-- 1 ASUS 197121 18805 Mar 26 20:30 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Mar 26 20:30 README.md
-rw-r--r-- 1 ASUS 197121 204 Mar 26 21:02 contacts.txt
-rw-r--r-- 1 ASUS 197121 666 Mar 27 15:12 emails.txt
-rw-r--r-- 1 ASUS 197121 72 Mar 26 20:47 environment.yml
-rw-r--r-- 1 ASUS 197121 1431 Mar 27 15:01 main.py
(week04)
ASUS@%C MINGW64 ~/repo/week04 (main)
$
(week04)
ASUS@%C 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。
---
(week04)
ASUS@%C MINGW64 ~/repo/week04 (main)
$
```

5. python 基础概念学习：

```
MINGW64/c/Users/ASUS/repo/week04
-rw-r--r-- 1 ASUS 197121 18805 Mar 26 20:30 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Mar 26 20:30 README.md
-rw-r--r-- 1 ASUS 197121 204 Mar 26 21:02 contacts.txt
-rw-r--r-- 1 ASUS 197121 666 Mar 27 15:38 emails.txt
-rw-r--r-- 1 ASUS 197121 72 Mar 26 20:47 environment.yml
-rw-r--r-- 1 ASUS 197121 1431 Mar 27 15:01 main.py

(base) ASUS@%C:~$ rm emails.txt

(base) ASUS@%C:~$ ls -l
total 30
-rw-r--r-- 1 ASUS 197121 18805 Mar 26 20:30 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Mar 26 20:30 README.md
-rw-r--r-- 1 ASUS 197121 204 Mar 26 21:02 contacts.txt
-rw-r--r-- 1 ASUS 197121 72 Mar 26 20:47 environment.yml
-rw-r--r-- 1 ASUS 197121 1431 Mar 27 15:01 main.py

(base) ASUS@%C:~$ python -m pdb main.py
> c:\users\asus\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
(Pdb) |
```

让调试器运行，l 有显示代码的作用

```
MINGW64/c/Users/ASUS/repo/week04
total 30
-rw-r--r-- 1 ASUS 197121 18805 Mar 26 20:30 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Mar 26 20:30 README.md
-rw-r--r-- 1 ASUS 197121 204 Mar 26 21:02 contacts.txt
-rw-r--r-- 1 ASUS 197121 72 Mar 26 20:47 environment.yml
-rw-r--r-- 1 ASUS 197121 1431 Mar 27 15:01 main.py

(base) ASUS@%C:~$ python -m pdb main.py
> c:\users\asus\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
(Pdb) l
1  -> def read_contacts(file_path):
2      contacts = []
3      try:
4          with open(file_path, "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(f"错误: 文件 {file_path} 未找到。")
10     return contacts
11
(Pdb) |
```

箭头指向的是即将要运行的代码，n 是执行代码的意思，ll 显示全部代码，p 打印表达式，s 是步入调用，定义是定义，运行是运行，l 是显示运行的上面五行加下面五行的代码，pp 是美观打印。

6. 安装 wat。

```
MINGW64/c/Users/ASUS/repo/week04
> c:\users\asus\repo\week04\main.py(1)<module>()
-> def read_contacts(file_path):
(Pdb) q

(base) ASUS@%C:~$ ls -l
total 31
-rw-r--r-- 1 ASUS 197121 18805 Mar 26 20:30 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Mar 26 20:30 README.md
-rw-r--r-- 1 ASUS 197121 204 Mar 26 21:02 contacts.txt
-rw-r--r-- 1 ASUS 197121 666 Mar 27 18:06 emails.txt
-rw-r--r-- 1 ASUS 197121 91 Mar 27 21:23 environment.yml
-rw-r--r-- 1 ASUS 197121 1431 Mar 27 15:01 main.py

(base) ASUS@%C:~$ cat environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector

(base) ASUS@%C:~$
```

发现没有切换到 week04 环境，用 conda activate week04 切换。

```

- python=3.12
- wat-inspector(week04)
ASUS@%C:~$ conda env update
Channels:
- conda-forge
- 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): done
Solving environment: done
#
# To activate this environment, use
#
#     $ conda activate week04
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#
(week04)
ASUS@%C:~$

```

7. def, while 为保留字，不能作为变量名，python 代码里的红色字体是保留字，不能作为变量名。

```

#
# To activate this environment, use
#
#     $ conda activate week04
#
# To deactivate an active environment, use
#
#     $ conda deactivate
#
(week04)
ASUS@%C:~$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [MSC v.1
943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> name = 'Yun Wang'
>>> print(name)
Yun Wang
>>> def = 'Yun Wang'
      File "<stdin>", line 1
        def = 'Yun Wang'
            ^
SyntaxError: invalid syntax
>>>

```

语句：逻辑上完整的一句话，for 循环语句，赋值语句，if 语句，return 语句，with 语句，表达式是构成语句的元素，相当于词语。语句可以嵌套，语句里可以有表达式，file 是表达式，if 语句里的条件判断也是表达式。一个表达式自己也构成语句，表达式是也可以嵌套，如 `__name__ == "__main__"`，name 和 main 两个语句构成了新语句。在 vs code 里，白色代码是变量名，黄色是字符串，类型比较明确的是绿色，橙色的是参数。

缩进在 python 里很重要，缩进必须严格对齐，缩进代表层级。

局部变量：


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

随着代码的运行，wat()里的局部变量会随之更新。

```
19     return emails
20
21
22 -> def sort_contacts(contacts):
23     def custom_sort_key(contact):
24         email = contact[2]
25         domain, username = email.split("@")[:-1]
26         return (domain, username)
27
(Pdb) n
> c:\users\asus\repo\week04\main.py(31)<module>()
-> def write_emails(emails, output_file):
(Pdb) wat()
Local variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\ASUS\repo\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __spec__: NoneType = None
  generate_emails: function = <function generate_emails at 0x0000023B96A477E0>
  read_contacts: function = <function read_contacts at 0x0000023B96A476A0>
  sort_contacts: function = <function sort_contacts at 0x0000023B96A5E200>
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb)
```

全局变量：全局变量总能访问到，是顶级的。

```
__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.globals
Global variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\ASUS\repo\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __spec__: NoneType = None
  contacts: list = [...]
  generate_emails: function = <function generate_emails at 0x0000023B96A477E0>
  read_contacts: function = <function read_contacts at 0x0000023B96A476A0>
  sort_contacts: function = <function sort_contacts at 0x0000023B96A5E200>
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
  write_emails: function = <function write_emails at 0x0000023B96A5F920>
(Pdb)
```

一旦 return，中途里面的东西就会被清理。相当于的房间里的东西就是局部变量，一旦运行完毕就会被清理。

函数里的函数是可以访问外层函数的变量的，legb 概念。

调用函数必须满足参数。

两个等号是运算符，“.”是名称访问运算符。Python 所管理的内存都是对

象。对象都有自己的类型。

```
MINGW64:/c/Users/ASUS/repo/week04
('吕轻侯', '男', 'lvqinghou@126.com'),
('郭芙蓉', '女', 'guofurong@126.com'),
('李秀莲', '男', 'lixulian@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 re
turn None...

(Pdb)
```