

第四周操作笔记

Contacts.txt

```
! environment.yml  contacts.txt X  main.py
contacts.txt
1 曾小贤 男 zengxiaoxian@163.com
2 胡一菲 女 huyifei@163.com
3 吕子乔 男 lvziquiao@126.com
4 陈美嘉 女 chenmeijia@126.com
5 张伟 男 zhangwei@163.com
6 唐悠悠 女 tangyouyou@163.com
```

```
(base) Administrator@DESKTOP-2HD7O7S MINGW64 ~/Desktop/首
作业/金融编程与计算/week04 (main)
$ cat contacts.txt
曾小贤 男 zengxiaoxian@163.com
胡一菲 女 huyifei@163.com
吕子乔 男 lvziquiao@126.com
陈美嘉 女 chenmeijia@126.com
张伟 男 zhangwei@163.com
唐悠悠 女 tangyouyou@163.com
```

main.py 代码

```
def process_contacts():
    # 读取 contacts.txt 文件
    with open('contacts.txt', 'r', encoding='utf-8') as f:
        lines = f.readlines()

    contacts = []
    for line in lines:
        # 处理每行数据
        line = line.strip()
        if not line:
            continue
        parts = line.split()
        if len(parts) < 3:
            continue # 跳过格式错误行
        name, gender, email = parts[0], parts[1], parts[2]

        # 拆分邮箱的用户名和域名
        if '@' not in email:
            continue # 跳过无效邮箱
        username, domain = email.split('@', 1)
        contacts.append({
            'name': name,
            'gender': gender,
            'email': email,
            'username': username,
            'domain': domain
        })

# 自定义排序规则
```

```
def sort_key(contact):
    # 优先按域名排序（126.com 在前）
    domain_priority = 0 if contact['domain'] == '126.com' else 1
    # 其次按用户名排序（字典序）
    return (domain_priority, contact['username'])
```

```
sorted_contacts = sorted(contacts, key=sort_key)
```

```
# 生成输出内容
output = []
for contact in sorted_contacts:
    # 生成性别称呼
    title = '女士' if contact['gender'] == '女' else '先生'
    # 生成邮件内容
    content = f"to: <{contact['email']}>\n" \
             f"尊敬的{contact['name']} {title}：您的会员资格即将到期，\n" \
             f"请及时续费。"
    output.append(content)
```

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

```
if __name__ == '__main__':
    process_contacts()
```

emails.txt

```
$ python main.py
(myproject)
Administrator@DESKTOP-2HD707S MINGW64 ~/Desktop/首经贸/2024-2025第二学期作业/金融编程与计算
$ cat emails.txt
to: <chenmeijia@126.com>
尊敬的陈美嘉女士：您的会员资格即将到期，请及时续费。
---
to: <lvziqiao@126.com>
尊敬的吕子乔先生：您的会员资格即将到期，请及时续费。
---
to: <huyifei@163.com>
尊敬的胡一菲女士：您的会员资格即将到期，请及时续费。
---
to: <tangyouyou@163.com>
尊敬的唐悠悠女士：您的会员资格即将到期，请及时续费。
---
to: <zengxiaoxian@163.com>
尊敬的曾小贤先生：您的会员资格即将到期，请及时续费。
---
to: <zhangwei@163.com>
尊敬的张伟先生：您的会员资格即将到期，请及时续费。
```

验证概念 (proof of concept, PoC)

`l` 命令应该列出当前执行的代码位置周围的代码，

`n` 是执行下一行代码，但不进入函数内部，

`s` 则是进入函数内部进行逐行调试。

`p` 用于打印变量的值，`pp` 则是以更美观的格式打印，比如对于字典或列表。

`c`则是继续执行直到遇到下一个断点或程序结束。

q : 退出

```
Administrator@DESKTOP-2HD707S MINGW64 ~/Desktop/曾经贤/2024-2025第二学期作业/金融编程与计算
$ python -m pdb main.py
> c:\users\lululu~yaoyao\desktop\曾经贤\2024-2025第二学期作业\金融编程与计算\week04\main.py
(<module>)>
-> def process_contacts():
(Pdb) 1
1 -> def process_contacts():
2     # 读取 contacts.txt 文件
3     with open('contacts.txt', 'r', encoding='utf-8') as f:
4         lines = f.readlines()
5
6     contacts = []
7     for line in lines:
8         # 处理每行数据
9         line = line.strip()
10        if not line:
11            continue
```

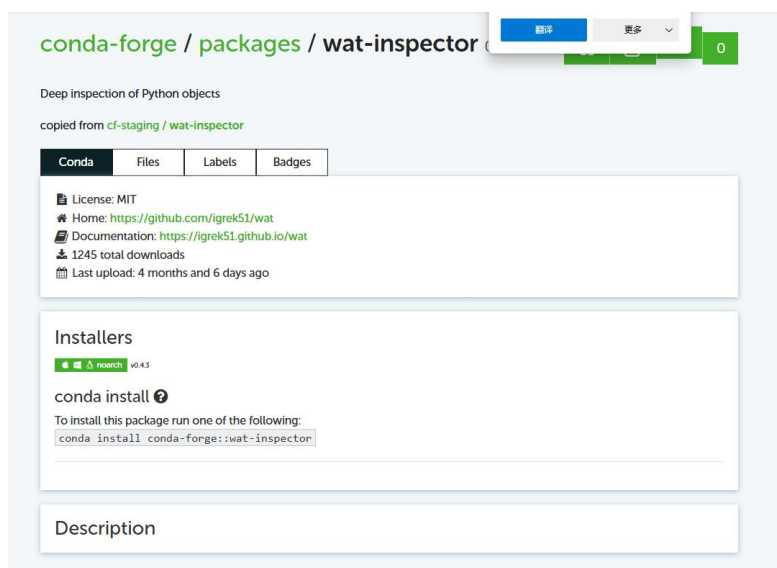
```
12    process_contacts()
(Pdb) p contacts
*** NameError: name 'contacts' is not defined
(Pdb) p read contacts
*** SyntaxError: invalid syntax
(Pdb) p read_contacts
*** NameError: name 'read_contacts' is not defined
(Pdb) p process_contacts
<function process_contacts at 0x000001A5AFD6F560>
(Pdb) q
```

与老师代码不一致，没有显示出来

```
(Pdb) p contacts
*** NameError: name 'contacts' is not defined
(Pdb) p contacts_file
*** NameError: name 'contacts_file' is not defined
(Pdb) p contacts.txt
*** NameError: name 'contacts' is not defined
(Pdb) p emails_file
*** NameError: name 'emails_file' is not defined
(Pdb) p emails.txt
*** NameError: name 'emails' is not defined
(Pdb) p "emails.txt"
'emails.txt'
(Pdb) pp emails.txt
*** NameError: name 'emails' is not defined
(Pdb) pp "emails.txt"
'emails.txt'
(Pdb) q
```

wat-inspector 软件包

conda install conda-forge::wat-inspector



conda-forge / packages / wat-inspector

Deep inspection of Python objects

copied from cf-staging / wat-inspector

Conda Files Labels Badges

License: MIT
Home: <https://github.com/jgrek51/wat>
Documentation: <https://jgrek51.github.io/wat>
1245 total downloads
Last upload: 4 months and 6 days ago

Installers

conda install

To install this package run one of the following:

```
conda install conda-forge::wat-inspector
```

Description

保留字：# 常见保留字

if, else, for, while, def, class, return, import, True, False, None, async, await

import keyword

print(keyword.kwlist) # 输出当前 Python 版本的所有保留字

```
>>> import keyword
>>> print(keyword.kwlist) # 输出当前 Python 版本的所有保留字
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'cla
ss', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from
', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pas
s', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

语句里会包含一些表达式

大模型给出的区别

2. 语句 (Statement) vs 表达式 (Expression)

特征	语句 (Statement)	表达式 (Expression)
定义	执行操作的代码单元，不返回值。	计算并返回值的代码片段。
示例	x = 5、if x > 0:、def func():	3 + 4、x * y、len("hello")
特点	以换行符或分号结束，可能包含表达式。	可以嵌套在语句中使用。
返回值	无（但可能修改状态，如赋值）。	始终生成一个值。

跟老师颜色不一致：梅红色的是保留字，蓝色是变量名，黄色：含义明确的一些函数。

```
main.py > process_contacts
1 def process_contacts():
2     # 读取 contacts.txt 文件
3     with open('contacts.txt', 'r', encoding='utf-8') as f:
4         lines = f.readlines()
5
6     contacts = []
7     for line in lines:
8         # 处理每行数据
9         line = line.strip()
10        if not line:
11            continue
12        parts = line.split()
13        if len(parts) < 3:
14            continue # 跳过格式错误行
15        name, gender, email = parts[0], parts[1], parts[2]
16
17        # 拆分邮箱的用户名和域名
18        if '@' not in email:
19            continue # 跳过无效邮箱
20        username, domain = email.split('@', 1)
21        contacts.append({
22            'name': name,
23            'gender': gender,
24            'email': email,
25            'username': username,
26            'domain': domain
27        })
28
29    # 自定义排序规则
30    def sort_key(contact):
```

局部变量和全局变量


```
(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.
```

```
Local variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\lululu~yaoyao\Desktop\首经贸\2024-2025第二学期作业\金融编程与计算\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __spec__: NoneType = None
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
(Pdb) n
> c:\Users\lululu~yaoyao\desktop\首经贸\2024-2025第二学期作业\金融编程与计算\week04\main.py(54)<module>
le>O
-> if __name__ == '__main__':
(Pdb) n
> c:\Users\lululu~yaoyao\desktop\首经贸\2024-2025第二学期作业\金融编程与计算\week04\main.py(55)<module>
le>O
-> process_contacts()
(Pdb) n
--Return--
> c:\Users\lululu~yaoyao\desktop\首经贸\2024-2025第二学期作业\金融编程与计算\week04\main.py(55)<module>
le>O->None
-> process_contacts()
(Pdb) wat()
Local variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\lululu~yaoyao\Desktop\首经贸\2024-2025第二学期作业\金融编程与计算\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __return__: NoneType = None
  __spec__: NoneType = None
  process_contacts: function = <function process_contacts at 0x0000012DC74DF560>
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
```

全局变量：总是能够访问到；局部变量：只有运行到那里才能访问到

```
(Pdb) wat.globals
Global variables:
  __builtins__: dict = {...}
  __file__: pdb._ScriptTarget = 'C:\Users\lululu~yaoyao\Desktop\首经贸\2024-2025第二学期作业\金融编程与计算\week04\main.py'
  __name__: str = '__main__'
  __pdb_convenience_variables__: dict = {...}
  __spec__: NoneType = None
  wat: wat.inspection.inspection.Wat = <WAT Inspector object>
```

LEGB 规则

Local（局部） → Enclosing（闭包） → Global（全局） → Built-in（内置）

形参：函数定义时声明的参数名称

实参：调用函数时传递的具体值

对象：Python 中所有数据

查看对象所有属性和方法：

```
>>> dir("hello")
['_add_', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
'__ge__', '__getattr__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__',
'__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
'__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode',
'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii',
'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle',
'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix',
'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

1. 特殊方法（双下划线方法，Magic Methods）

这些方法定义了字符串对象的底层行为，通常由 Python 解释器隐式调用：

```
__add__(self, other): 字符串拼接（如 "a" + "b" → "ab"）。
__eq__(self, other): 判断字符串是否相等（== 操作符）。
__len__(self): 返回字符串长度（len("abc") → 3）。
__getitem__(self, index): 通过索引访问字符（"abc"[0] → "a"）。
__str__(self): 返回字符串的友好表示（print(str) 时调用）。
__repr__(self): 返回字符串的正式表示（调试时使用）。
```

2. 字符串操作与转换

大小写转换：

```
capitalize(): 首字母大写，其余小写（"hello" → "Hello"）。
upper(): 全大写（"hello" → "HELLO"）。
lower(): 全小写（"HELLO" → "hello"）。
swapcase(): 大小写互换（"HeLlO" → "hElLo"）。
casefold(): 更严格的格式转换（用于不区分大小写的比较，如 "ß" → "ss"）。
```

空白处理：

```
strip(): 去除两端空白符。
lstrip() /rstrip(): 去除左/右端空白符。
```

4. 字符串查找与替换

```
find(sub): 返回子串首次出现的索引（未找到返回 -1）。
index(sub): 类似 find，但未找到时抛出 ValueError。
count(sub): 统计子串出现次数。
replace(old, new): 替换所有匹配的子串。
format(): 格式化字符串（如 "{ } world".format("Hello"））。
translate(table): 通过映射表替换字符（需配合 maketrans 使用）。
```

5. 编码与格式化

```
encode(encoding="utf-8"): 将字符串编码为字节（如 "中文".encode() → b'\xe4\xb8\xad\xe6\x96\x87'）。
format_map(mapping): 类似 format，但直接传入字典（"{name}".format_map({"name": "Alice"})）。
```

分割与拼接：

```
split(sep): 按分隔符分割字符串（"a,b,c".split(",") → ["a", "b", "c"]）。
join(iterable): 将可迭代对象拼接为字符串（",".join(["a", "b"]) → "a,b"）。
partition(sep): 按分隔符分割为三部分（"a.b.c".partition(".") → ("a", ".", "b.c"））。
```

3. 字符串检查与判断

内容检查：

```
startswith(prefix): 是否以指定字符串开头。
endswith(suffix): 是否以指定字符串结尾。
isalnum(): 是否仅由字母或数字组成。
isalpha(): 是否仅由字母组成。
isdigit(): 是否仅由数字组成（支持 Unicode 数字）。
isnumeric(): 是否仅由数字字符（包括中文数字等）。
isidentifier(): 是否是合法的 Python 标识符（如变量名）。
```

格式检查：

```
islower(): 是否全小写。
isupper(): 是否全大写。
isspace(): 是否仅由空白符组成。
istitle(): 是否每个单词首字母大写。
```