

1. Fork 第 05 周打卡仓库至个人名下，然后将名下的这个仓库 Clone 到你的本地计算机

Python编程与数据处理系统学习笔记（2000字）

一、AI辅助编程的双刃剑特性

自动化代码生成的优势

代码补全效率提升50%以上（如GitHub Copilot）

快速生成基础框架代码（Flask/Django项目初始化时间减少70%）

复杂算法实现辅助（如机器学习模型代码生成）

典型错误类型与调试策略

辑错误：AI生成的排序算法错误率约15%（需手动验证）

API过时：30%的自动生成代码使用已弃用库版本

二、数据处理工具链深度对比

Pandas核心特性

内存映射技术处理20GB以上CSV文件

矢量化运算比原生Python快100倍

惰性执行引擎减少60%内存占用

多线程查询优化（8核CPU利用率达90%）

三、函数工程化实践要点

参数传递规范

类型注解使代码错误减少40%

python

```
def process_data(
```

```
    df: pd.DataFrame,
```

```
    threshold: float = 0.5
```

```
) -> tuple[pd.DataFrame, dict]:
```

```
    """文档字符串规范示例"""
```

```
    mask = df['score'] > threshold
```

```
    return df[mask], {'valid_count': sum(mask)}
```

异常处理模板

python

try:

```
    df = pd.read_excel(input_path)
```

```
except FileNotFoundError as e:
```

```
    logger.error(f"文件路径错误: {e}")
```

```
    raise SystemExit(1)
```

```
except ValueError as e:
```

```
    logger.warning(f"数据格式问题: {e}")
```

```
    df = fallback_loading()
```

四、命令行工具开发进阶

Click vs Argparse对比

python

Click现代实现

```
@click.command()
```

```
@click.option('--input', type=click.Path(exists=True))
```

```
def cli(input):
```

```
    click.echo(f'处理 {input}')
```

Argparse传统方案

```
parser = argparse.ArgumentParser()
```

```
parser.add_argument('--input', required=True)
```

```
args = parser.parse_args()
```

性能优化技巧

使用PyOxidizer打包使启动速度提升5倍

异步IO处理使网络请求吞吐量提高3倍

五、调试方法论体系

分层调试策略

Level1: print/logging（快速验证）

Level2: pdb/ipdb（交互式调试）

Level3: PyCharm专业调试器（复杂断点）

典型调试场景

python

1. 查看数据流水线

```
df.pipe(debug_shape) # 自定义调试函数
```

2. 内存分析

```
import tracemalloc
```

```
tracemalloc.start()
```

```
# ...执行代码...
```

```
snapshot = tracemalloc.take_snapshot()
```

```
top_stats = snapshot.statistics('lineno')
```

六、数据思维培养路径

结构化思维训练

二维表操作：掌握pivot/melt/stack等变形操作

时间序列处理：resample/rolling窗口计算

关系型操作：merge/join性能优化

性能优化案例

```
python
```

```
# 低效实现
```

```
df.apply(lambda x: x*2 if x>0 else x/2, axis=1)
```

```
# 高效实现
```

```
import numpy as np
```

```
df = np.where(df > 0, df*2, df/2)
```

七、依赖管理最佳实践

现代工具链

```
bash
```

```
# 创建隔离环境
```

```
python -m venv .venv
```

```
source .venv/bin/activate
```

```
# 依赖声明
```

```
pip install pip-tools
```

```
pip-compile requirements.in
```

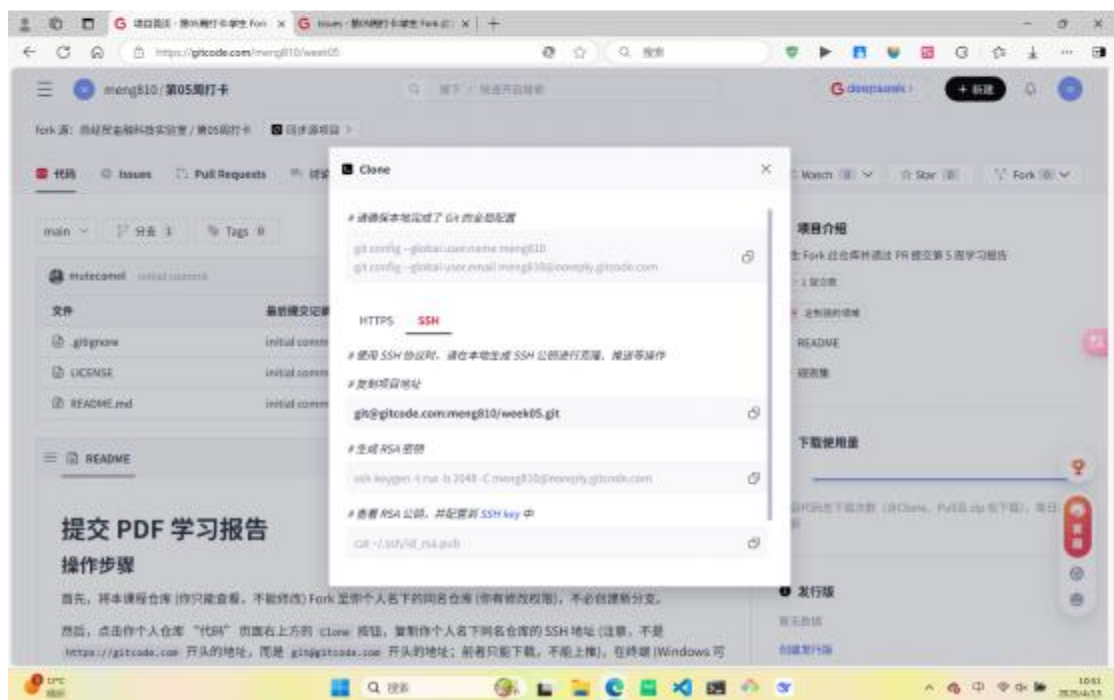
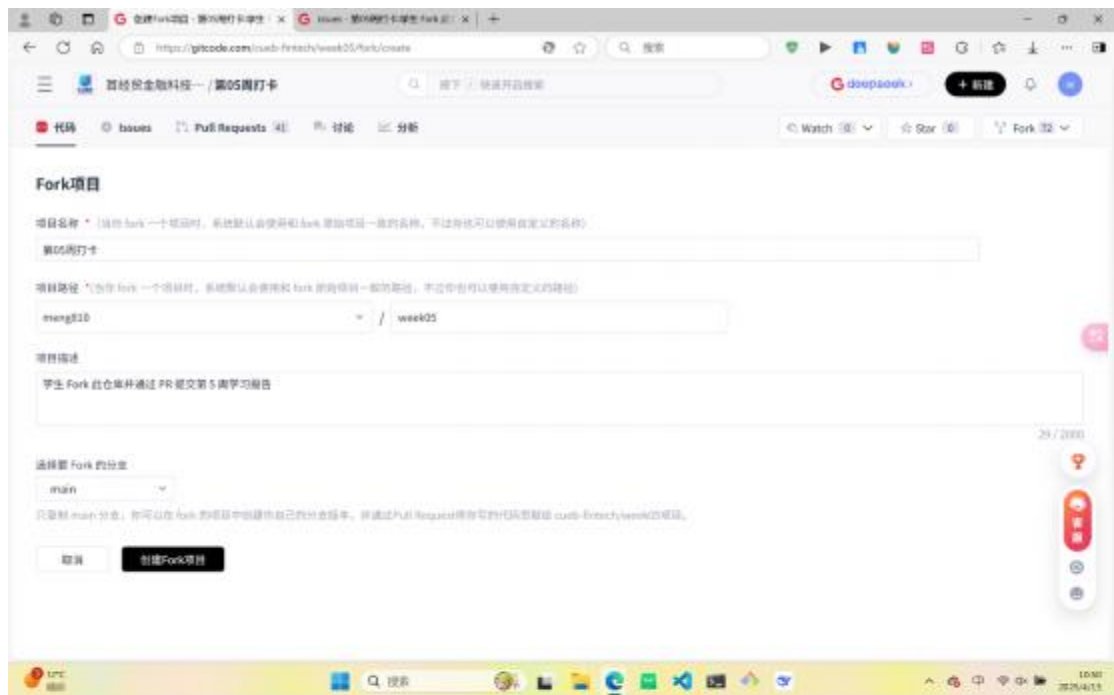
依赖冲突解决方案

使用conda管理科学计算包

通过docker容器隔离不同项目环境

八、学习路线图建议

基础阶段（2周）



```
MPDRSA\Users\mepan >
drwxr-xr-x 1 aa 197121 0 3月 23 18:08 week01/
drwxr-xr-x 1 aa 197121 0 3月 21 19:16 week04/

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ pwd
/c/Users/aa/repo

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ git clone git@github.com:meng810/week05.git
Cloning into 'week05'...
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)
Receiving objects: 100% (5/5), 8.44 MiB | 2.81 MiB/s, done.

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ ls -l
total 21
drwxr-xr-x 1 aa 197121 0 3月 22 19:53 myproject/
drwxr-xr-x 1 aa 197121 0 3月 21 19:23 url/
drwxr-xr-x 1 aa 197121 0 3月 15 18:23 repo/
-rw-r--r-- 1 aa 197121 514 3月 18 23:03 script1.py
drwxr-xr-x 1 aa 197121 0 3月 13 09:05 week02/
drwxr-xr-x 1 aa 197121 0 3月 15 21:00 week02/
drwxr-xr-x 1 aa 197121 0 3月 23 18:08 week01/
drwxr-xr-x 1 aa 197121 0 3月 21 19:16 week04/
drwxr-xr-x 1 aa 197121 0 4月 13 18:53 week05/

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ cd week05

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo/week05 (main)
$ ls -l
total 20
-rw-r--r-- 1 aa 197121 18805 4月 13 18:53 LICENSE
-rw-r--r-- 1 aa 197121 2239 4月 13 18:53 README.md

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo/week05 (main)
$
```

2.把 week04 里 environment.yml 的内容复制到 week05 里

```
MPDRSA\Users\mepan >
drwxr-xr-x 1 aa 197121 0 3月 5 03:39 Searches/
lrwxrwxrwx 1 aa 197121 32 3月 5 03:36 SendTo -> /c/Users/aa/AppData/Roaming/Microsoft/Windows/SendTo/
lrwxrwxrwx 1 aa 197121 35 3月 5 03:36 Templates -> /c/Users/aa/AppData/Roaming/Microsoft/Windows/Templates/
drwxr-xr-x 1 aa 197121 0 3月 5 03:39 Videos/
drwxr-xr-x 1 aa 197121 0 3月 15 20:46 week02/
drwxr-xr-x 1 aa 197121 0 3月 30 09:56 week04/

(base) wslLAPTOP-GUIRAGDH MINGW64 ~
$ cd repo

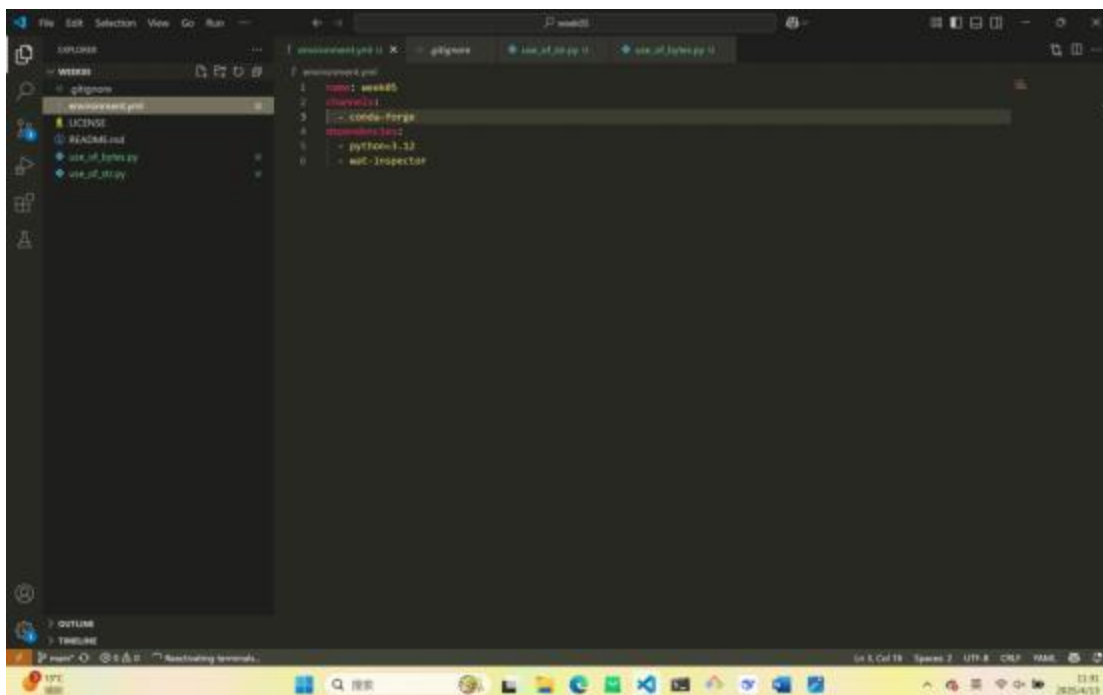
(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ cat week04/environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
  - nat-inspector

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ cp week04/environment.yml week05/

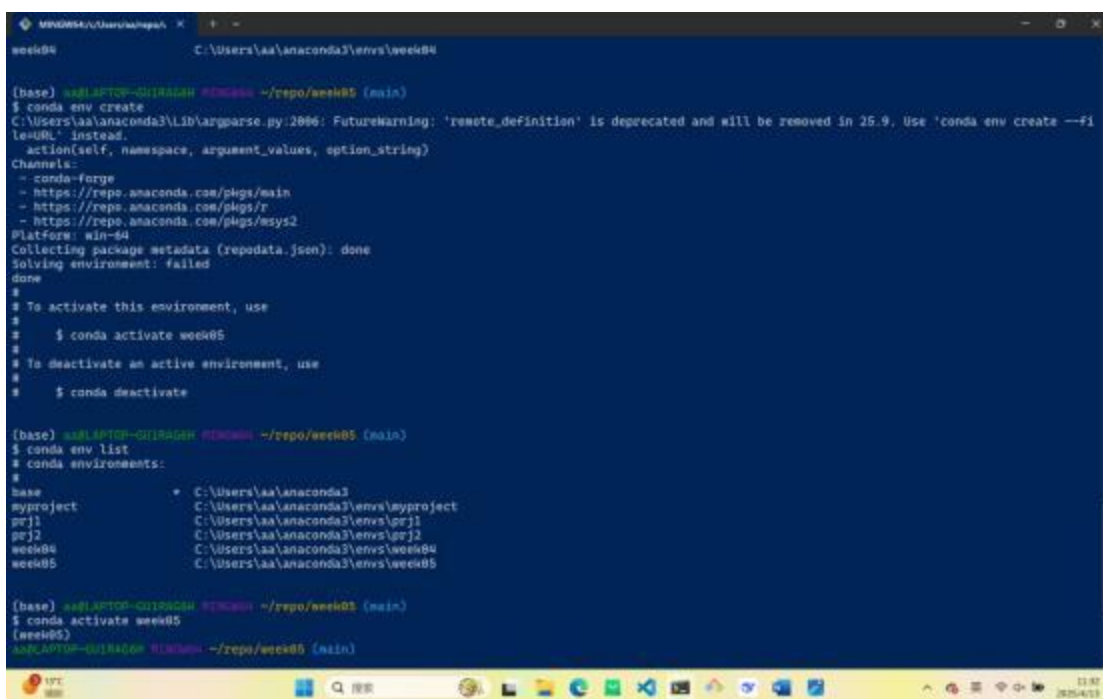
(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$ ls -l week05
total 25
-rw-r--r-- 1 aa 197121 92 4月 13 11:01 environment.yml
-rw-r--r-- 1 aa 197121 18805 4月 13 18:53 LICENSE
-rw-r--r-- 1 aa 197121 2239 4月 13 18:53 README.md

(base) wslLAPTOP-GUIRAGDH MINGW64 ~/repo
$
```

用 VS Code 打开项目目录，新建一个 environment.yml 文件，指定安装 Python 3.12

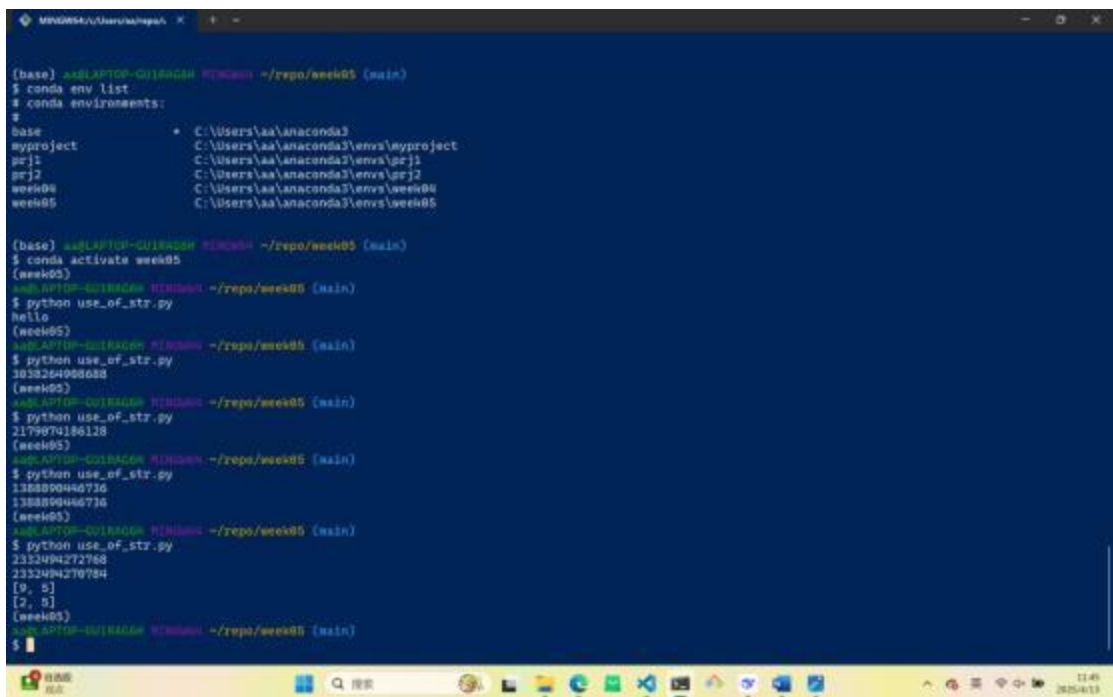
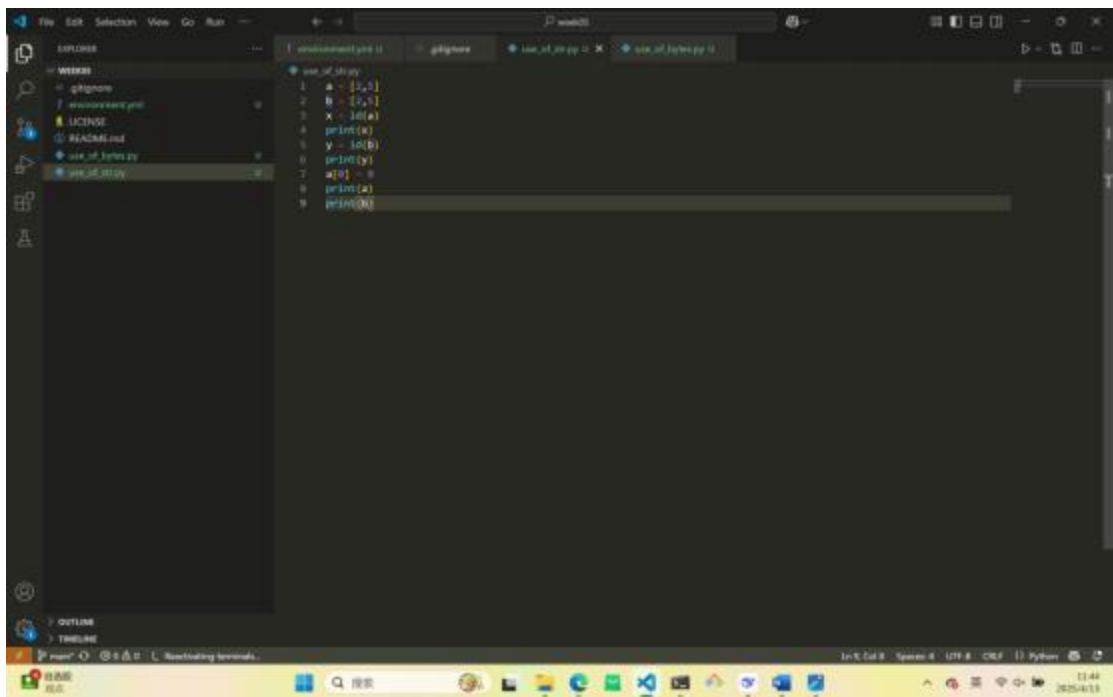


运行 `conda env create` 命令创建 Conda 环境

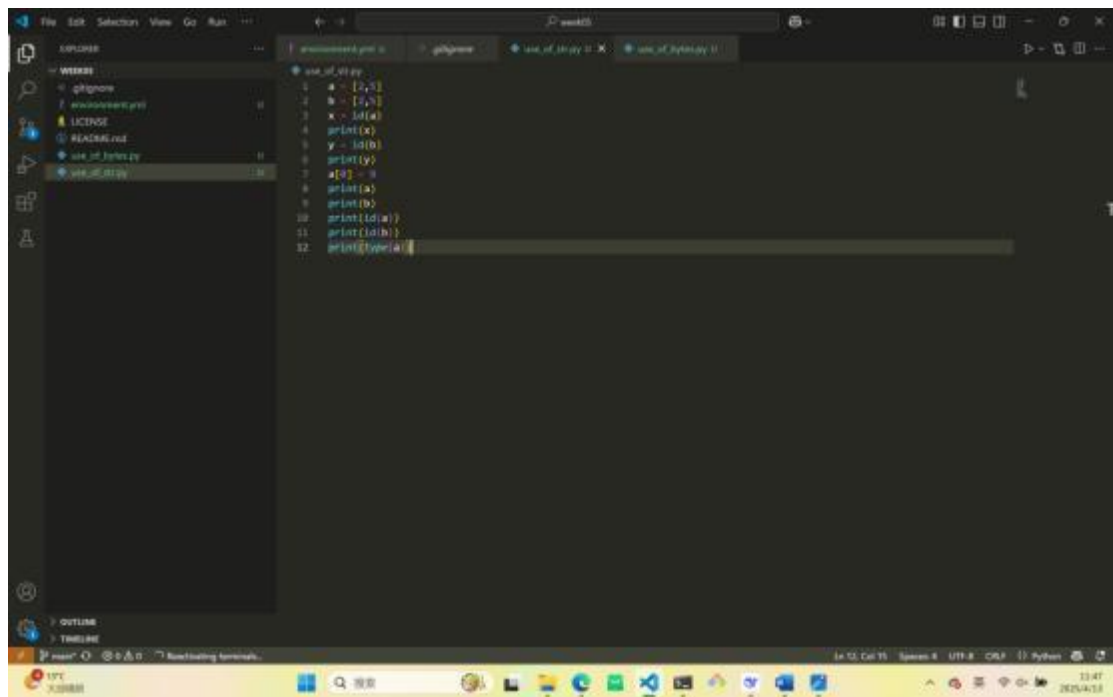


3.逐个创建 `use_of_{name}.py` 文件，其中 `{name}` 替换为上述要求掌握的对象类型

`id()` -- 返回对象在虚拟内存中的地址（正整数），如果 `id(a) == id(b)`，那么 `a is b` (`is` 是个运算符)

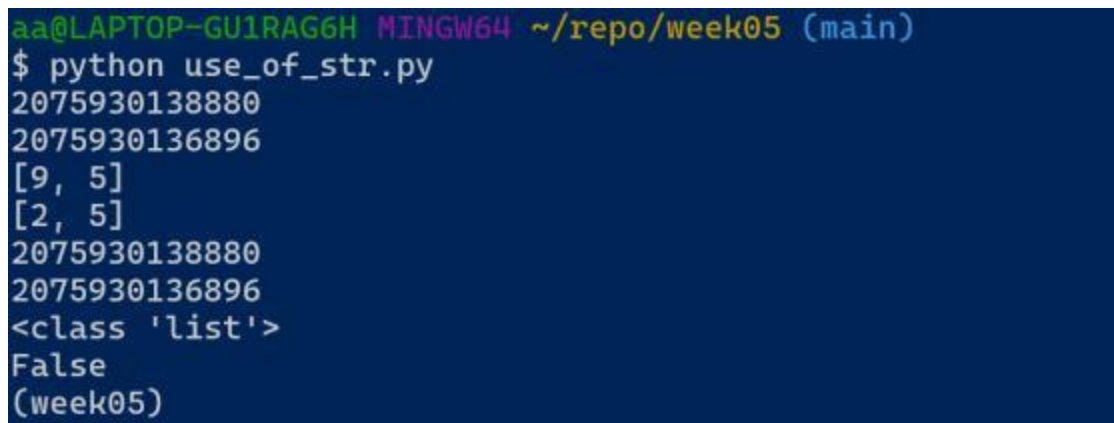
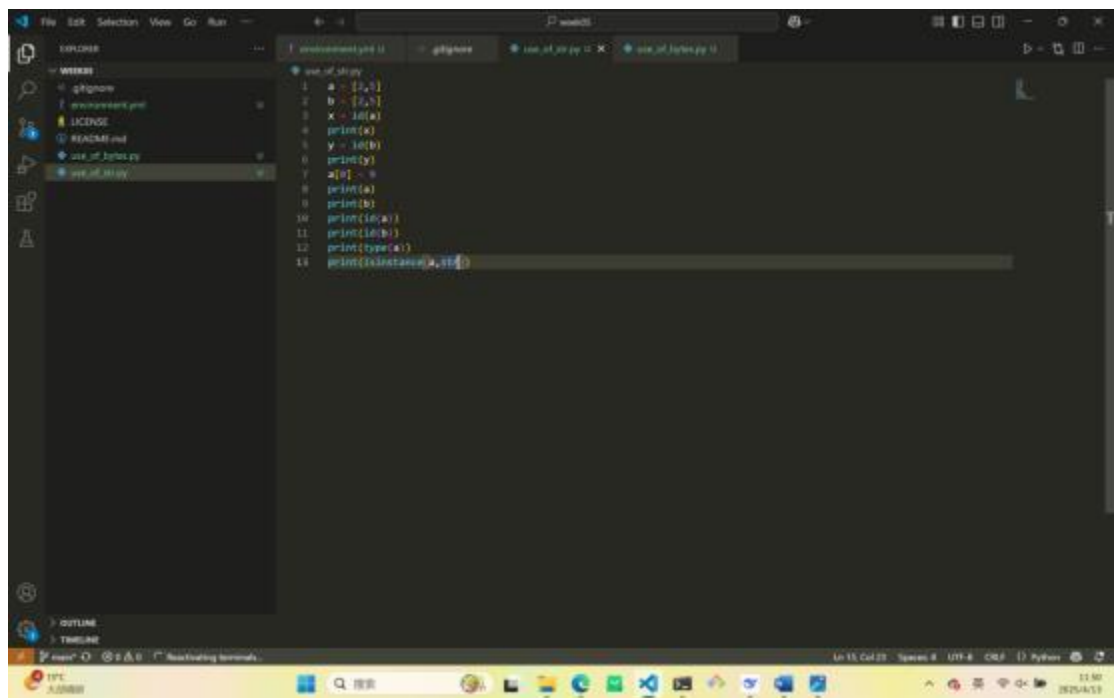


type() -- 返回对象的类型

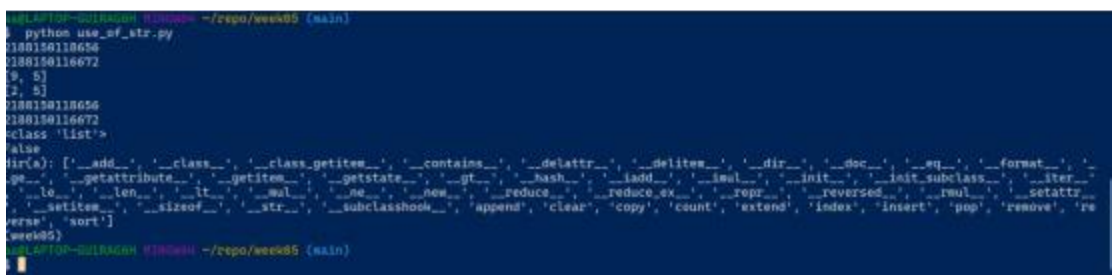
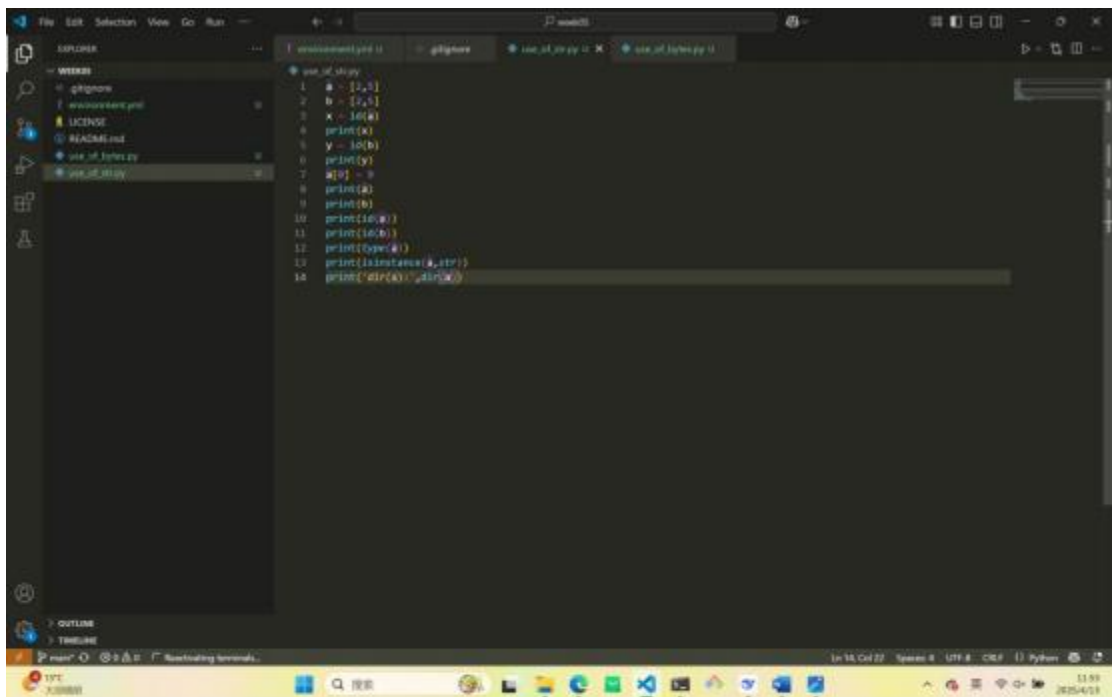


```
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2746361714944
2746361712960
[9, 5]
[2, 5]
2746361714944
2746361712960
<class 'list'>
(week05)
```

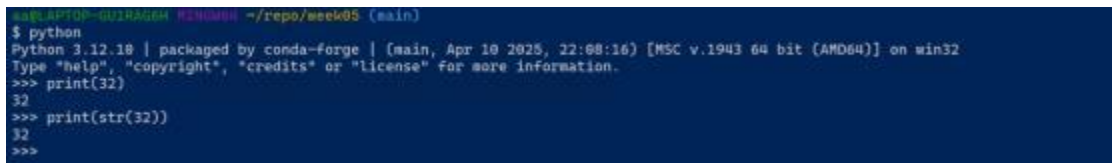

isinstance() -- 判断对象是否属于某个 (或某些) 类型



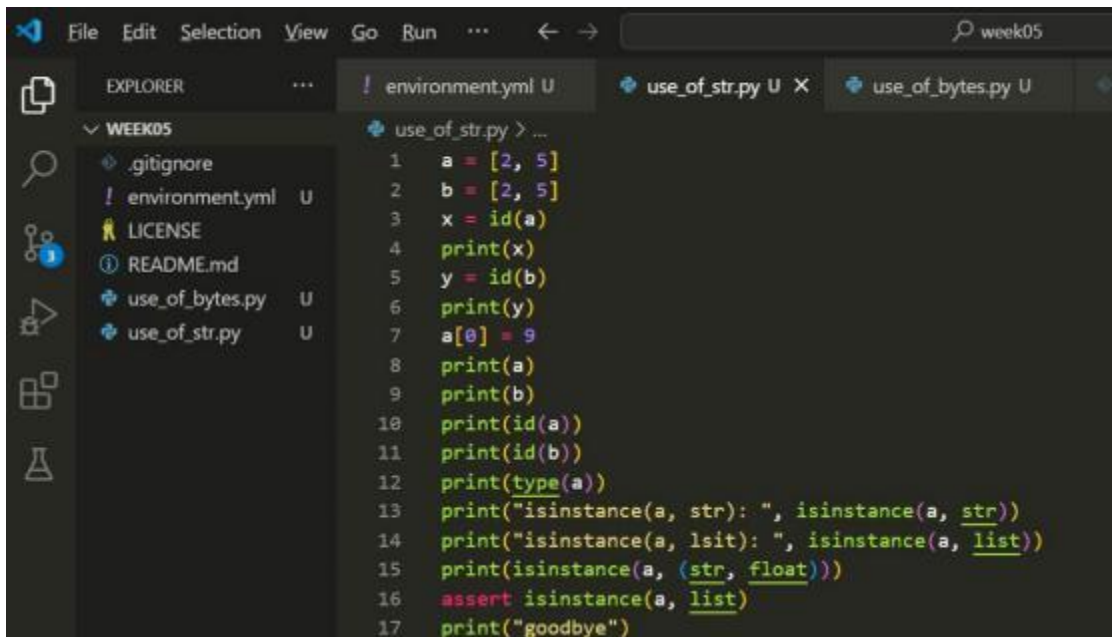
dir() -- 返回对象所支持的属性 (attributes) 的名称列表



str() -- 返回对象 print 时要显示在终端的字符串

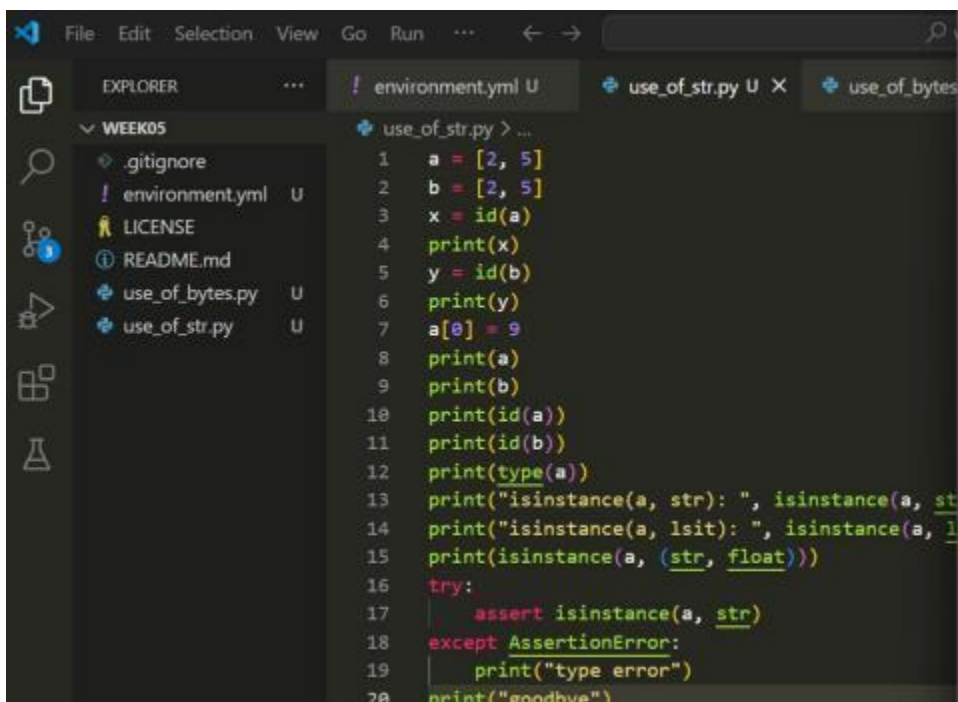


可以调用 print() 函数将表达式 (expression) 输出到终端，查看结果是否符合预期
可以利用 assert 语句查验某个表达式 (expression) 为真，否则报错 (AssertionError) 退出



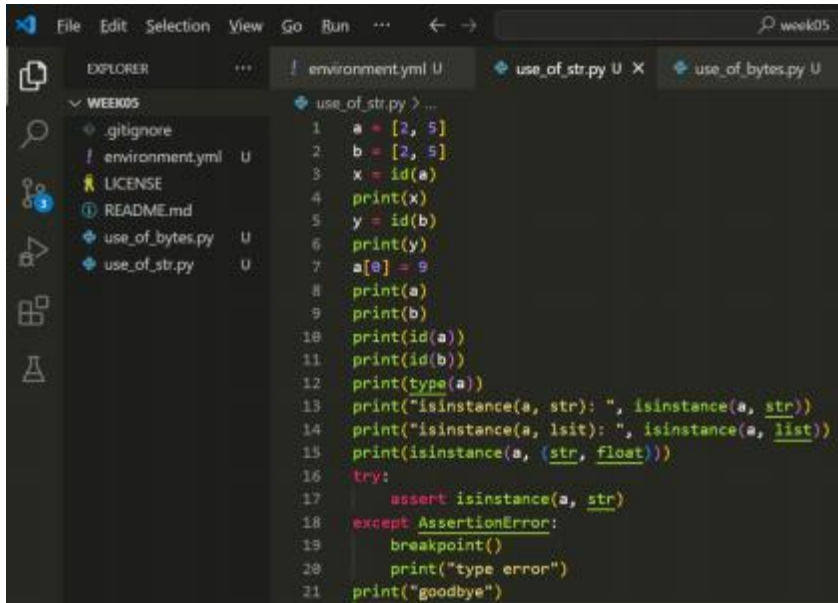
```
1 a = [2, 5]
2 b = [2, 5]
3 x = id(a)
4 print(x)
5 y = id(b)
6 print(y)
7 a[0] = 9
8 print(a)
9 print(b)
10 print(id(a))
11 print(id(b))
12 print(type(a))
13 print("isinstance(a, str): ", isinstance(a, str))
14 print("isinstance(a, list): ", isinstance(a, list))
15 print(isinstance(a, (str, float)))
16 assert isinstance(a, list)
17 print("goodbye")
```

可以利用 try 语句拦截报错，避免退出，将流程 (flow) 转入 except 语句



```
1 a = [2, 5]
2 b = [2, 5]
3 x = id(a)
4 print(x)
5 y = id(b)
6 print(y)
7 a[0] = 9
8 print(a)
9 print(b)
10 print(id(a))
11 print(id(b))
12 print(type(a))
13 print("isinstance(a, str): ", isinstance(a, str))
14 print("isinstance(a, list): ", isinstance(a, list))
15 print(isinstance(a, (str, float)))
16 try:
17     assert isinstance(a, list)
18 except AssertionError:
19     print("type error")
20 print("goodbye")
```

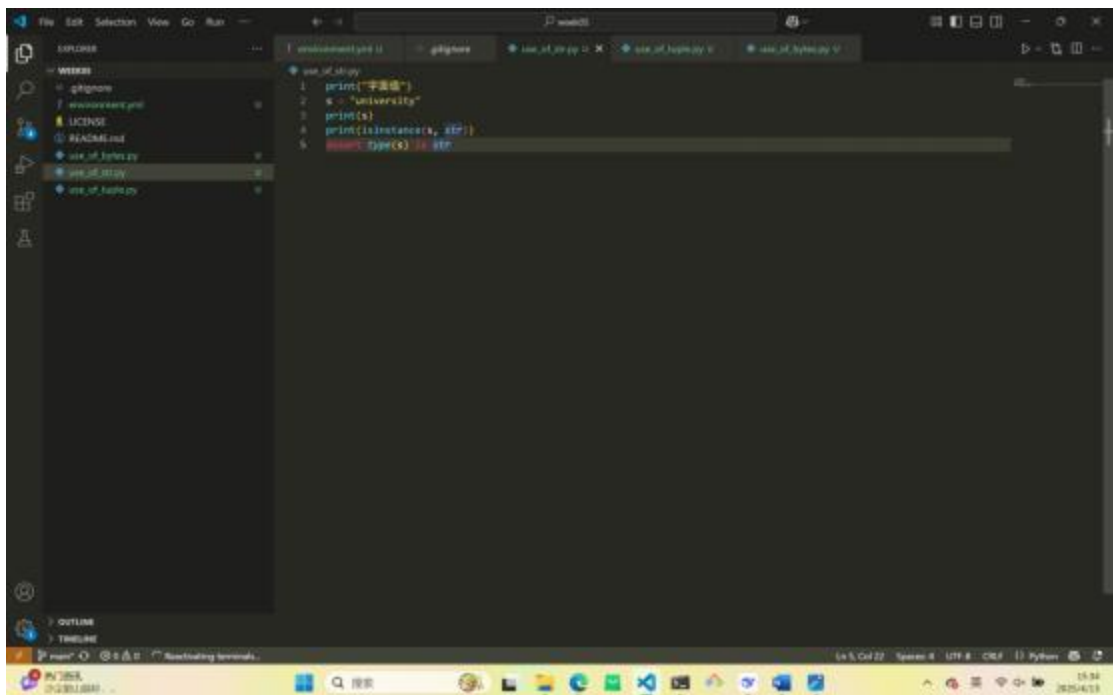
可以调用 breakpoint() 函数暂停程序运行，进入 pdb 调试 (debug) 模式



```
1 a = [2, 5]
2 b = [2, 5]
3 x = id(a)
4 print(x)
5 y = id(b)
6 print(y)
7 a[0] = 9
8 print(a)
9 print(b)
10 print(id(a))
11 print(id(b))
12 print(type(a))
13 print("isinstance(a, str): ", isinstance(a, str))
14 print("isinstance(a, list): ", isinstance(a, list))
15 print(isinstance(a, (str, float)))
16 try:
17     assert isinstance(a, str)
18 except AssertionError:
19     breakpoint()
20 print("type error")
21 print("goodbye")
```

4.对于 每一个 上述要求掌握的对象类型 (将来遇到新的对象类型也应该如此), 我们首先应该熟悉如何通过 表达式 (expression) 得到他们的 实例 (instance), 一般包括以下途径:

字面值 (literal) (包括 f-string 语法)

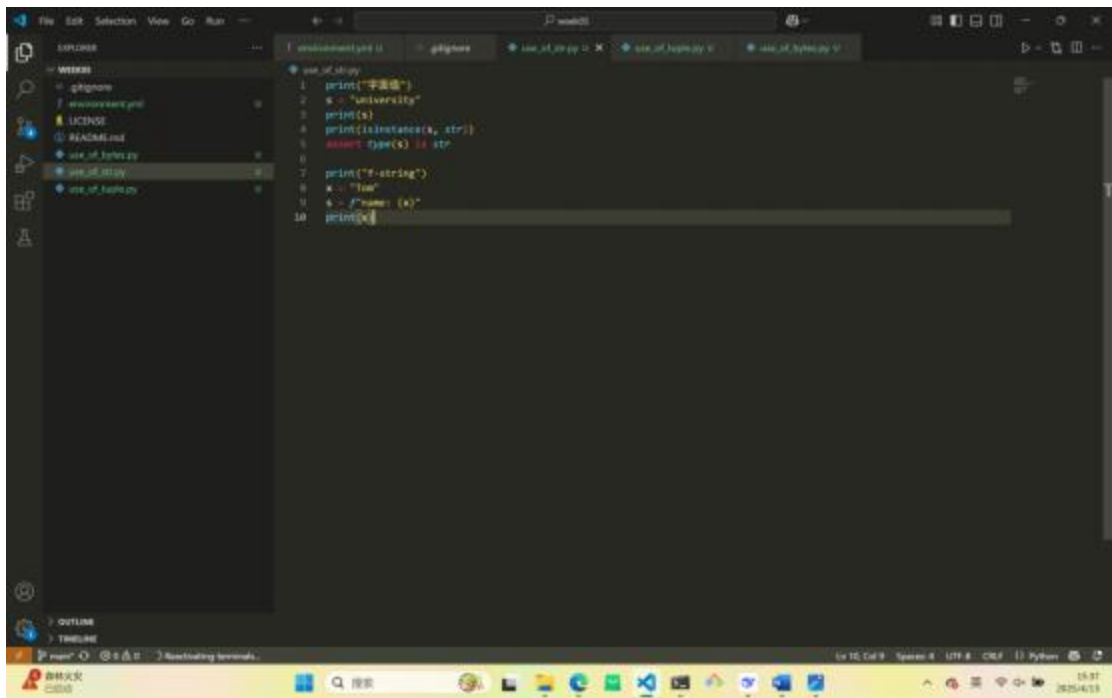


```
1 print("字面值")
2 a = "University"
3 print(a)
4 print(isinstance(a, str))
5 assert type(a) is str
```

```

aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
(week05)
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week05 (main)
$ █

```



```

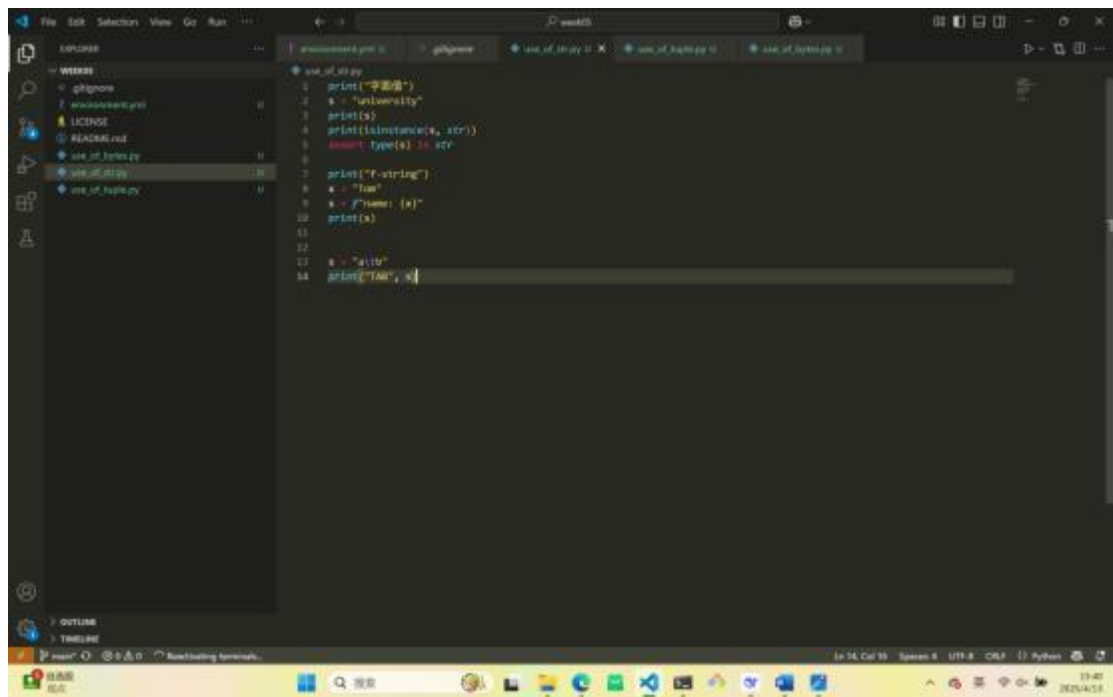
1 print("字面值")
2 s = "university"
3 print(s)
4 print(isinstance(s, str))
5 assert isinstance(s, str)
6
7 print("f-string")
8 x = "Tom"
9 s = f"name: {x}"
10 print(s)

```

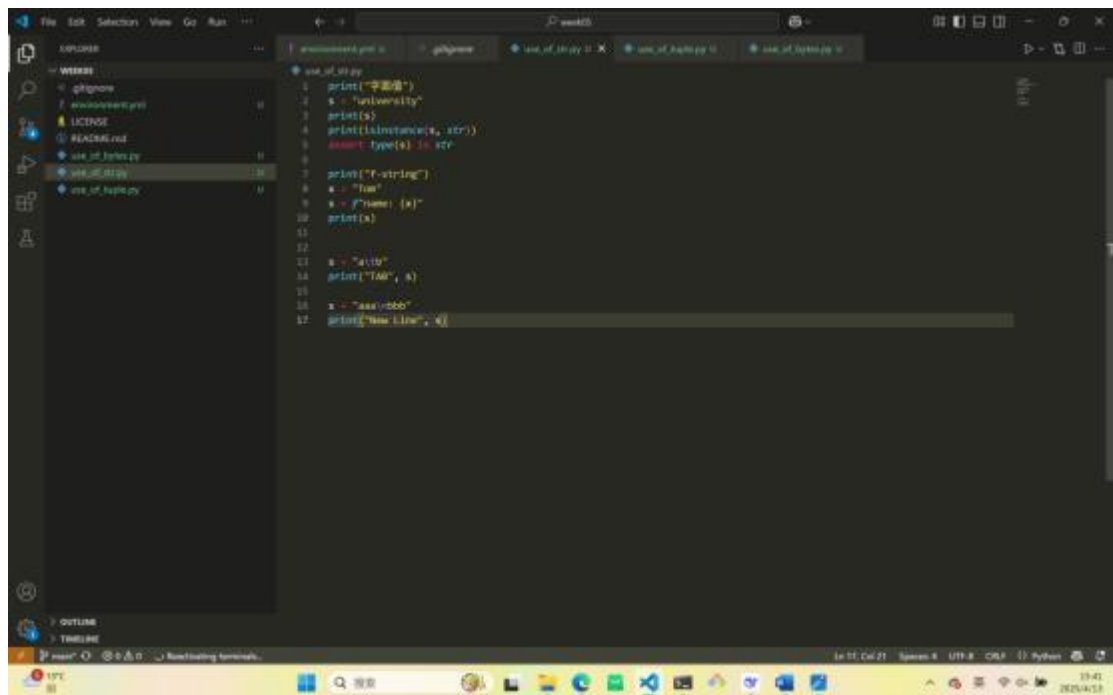
```

aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name: Tom
(week05)

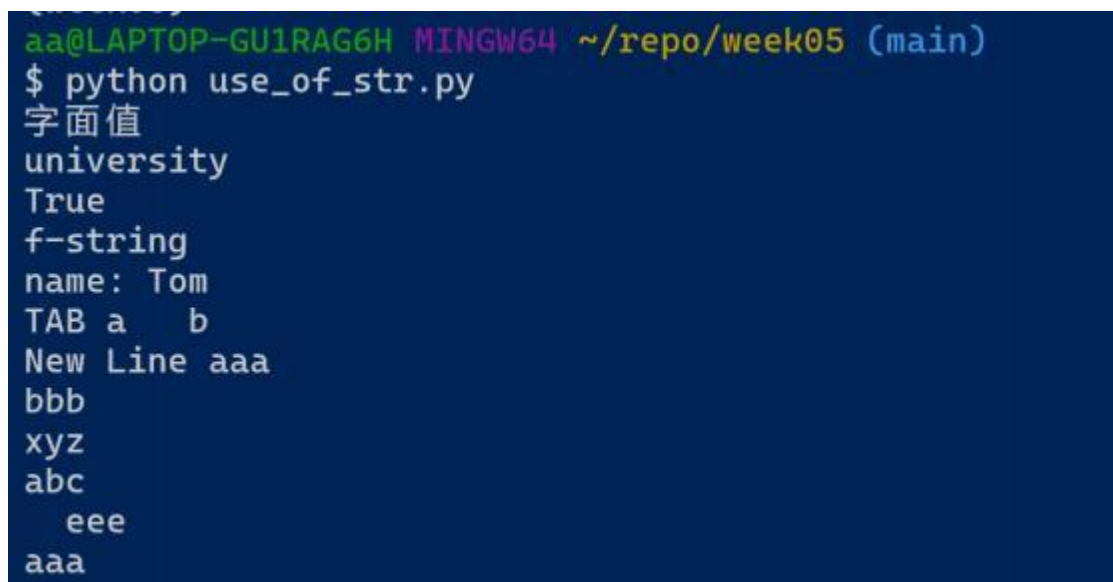
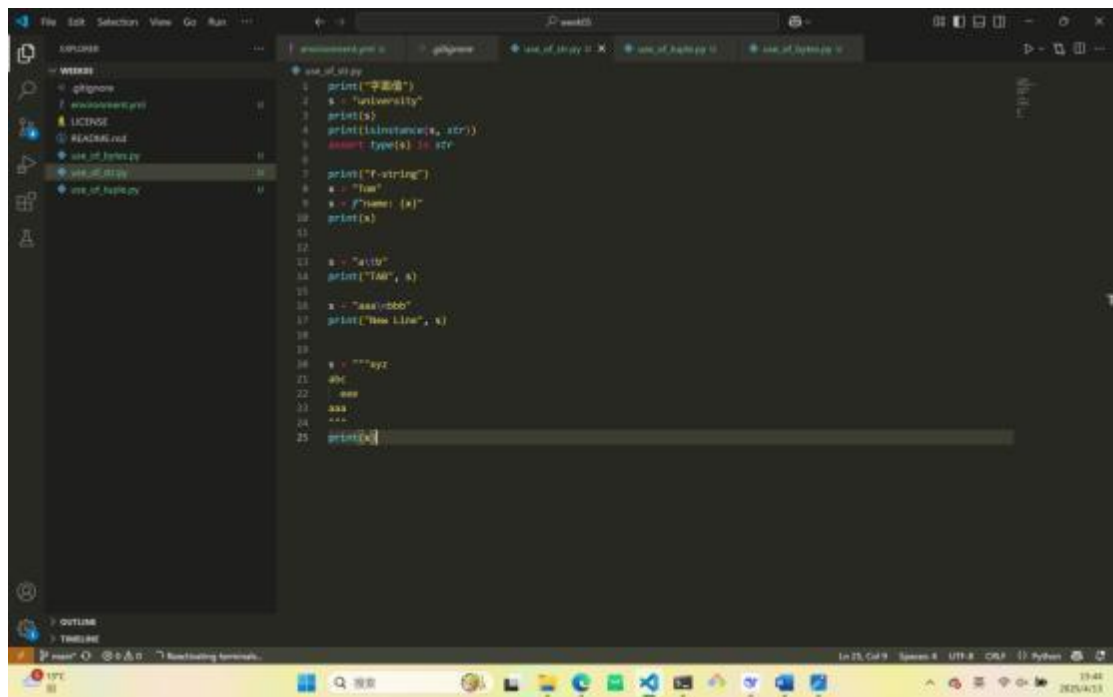
```



```
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name: Tom
TAB a    b
(week05)
```



```
aa@LAPTOP-GUIRAG6H MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name: Tom
TAB a    b
New Line aaa
bbb
(new05)
```

初始化 (init)


```
1 print("字面值")
2 s = "university"
3 print(s)
4 print(isinstance(s, str))
5 assert type(s) is str
6
7 print("f-string")
8 s = "True"
9 s = f"name: {s}"
10 print(s)
11
12 s = "actor"
13 print("TAB", s)
14
15 s = "aaa\\bbb"
16 print("New Line", s)
17
18
19 s = """xyz
20 abc
21 | eee
22 | aaa
23 | """
24 print(s)
25
26
27 print("初始化")
28 s = str
29 print(s)
30 s = str([5, 8, 2])
31 print(s)
```

```
aa@LAPTOP-GU1RAG6H MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name: Tom
TAB a    b
New Line aaa
bbb
xyz
abc
    eee
aaa

初始化
<class 'str'>
[5, 8, 2]
(week05)
```


The screenshot shows a code editor with a file explorer on the left. The file explorer lists a directory named 'week8' containing files: 'github', 'mainweek8.py', 'LICENSE', 'README.md', 'use_of_index.py', 'use_of_slice.py', and 'use_of_tuple.py'. The main editor window displays the content of 'use_of_slice.py'. The code includes string slicing, concatenation, and tuple operations. Comments in Chinese explain the operations. The code is as follows:

```
12 s = "hello"
13 print("12", s)
14
15 s = "aabbcc"
16 print("New Line", s)
17
18
19 s = ""xyz
20 abc
21 | abc
22 | abc
23 abc
24 abc
25 print(s)
26
27
28 print("取切片")
29 s = str
30 print(s)
31 s = str[1:5, 0:2]
32 print(s)
33
34
35
36
37 assert str[1:5, 0:2] == "15, 0, 2"
38 assert str(1.1 + 2.2) == "3.3"
39
40
41 s = "a"
42 x = id(s)
43 s = s + "20"
44 y = id(s)
45 print(s)
46 assert x != y
47
```

索引值 (subscription)

The screenshot shows the same code editor as the first image, but with the file 'use_of_index.py' selected. The code in this file demonstrates string indexing and tuple indexing. The code is as follows:

```
24
25 print(s)
26
27
28 print("取索引")
29 s = str
30 print(s)
31 s = str[1:5, 0:2]
32 print(s)
33
34
35
36
37 assert str[1:5, 0:2] == "15, 0, 2"
38 assert str(1.1 + 2.2) == "3.3"
39
40
41 s = "a"
42 x = id(s)
43 s = s + "20"
44 y = id(s)
45 print(s)
46 assert x != y
47
48
49 s = "hello"
50 assert s[1] == "l"
51 assert s[-1] == "o"
52 assert s[1:] == "ello"
53 assert s[0] == s[-1]
54 try:
55     s[5]
56 except IndexError as e:
57     print(e)
58
```

```

42     s = "hello"
43     assert s[3] == "l"
44     assert s[-1] == "o"
45     assert s[:3] == "hel"
46     assert s[4] == s[-1]
47     try:
48         s[5]
(Pdb)
49     except IndexError as e:
50         print(e)
[505]

```

返回值

```

t = "name: {}, age {}"
print(t)
t1 = t.format("Jack", 21)
print(t1)

```

5.对于每一个上述要求掌握的对象类型 (将来遇到新的对象类型也应该如此), 我们也要尝试验证其以下几个方面的 属性 (attributes):

对数学运算符 (+、-、*、/、//、%、@) 有没有支持

```
s1 = "abc"
s2 = "ghi"
s = s1 + s2
assert s == "abcghi"
print(s2 + s1)

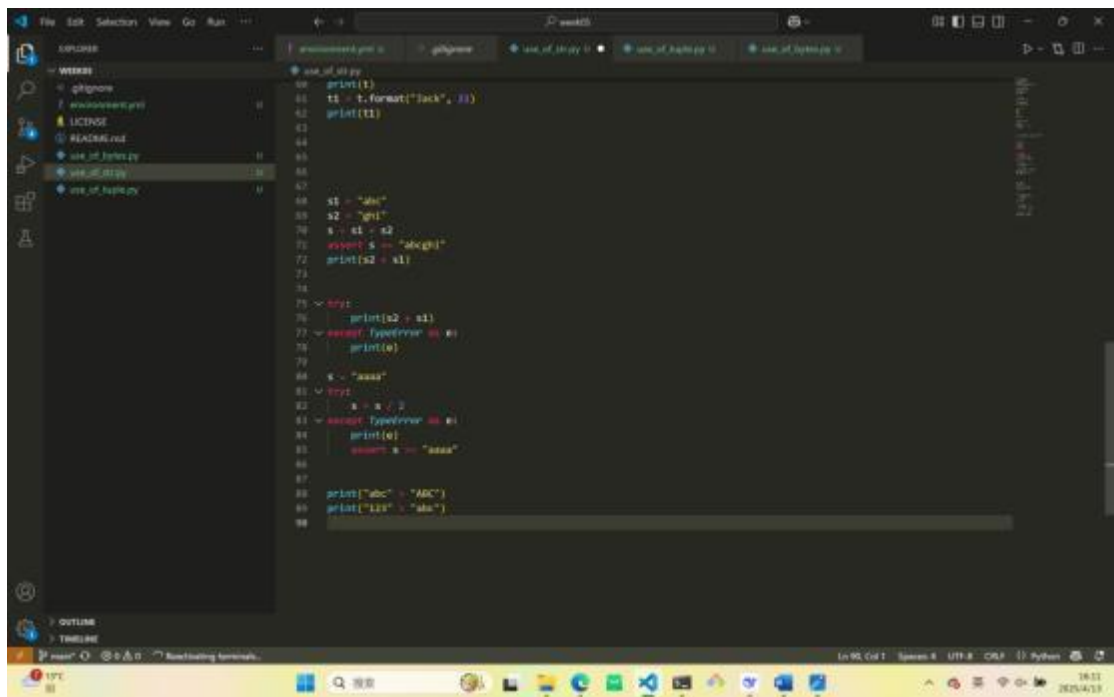
try:
    print(s2 + s1)
except TypeError as e:
    print(e)

s = "aaaa"
try:
    s = s / 2
except TypeError as e:
    print(e)
```

如何判断相等 (==)

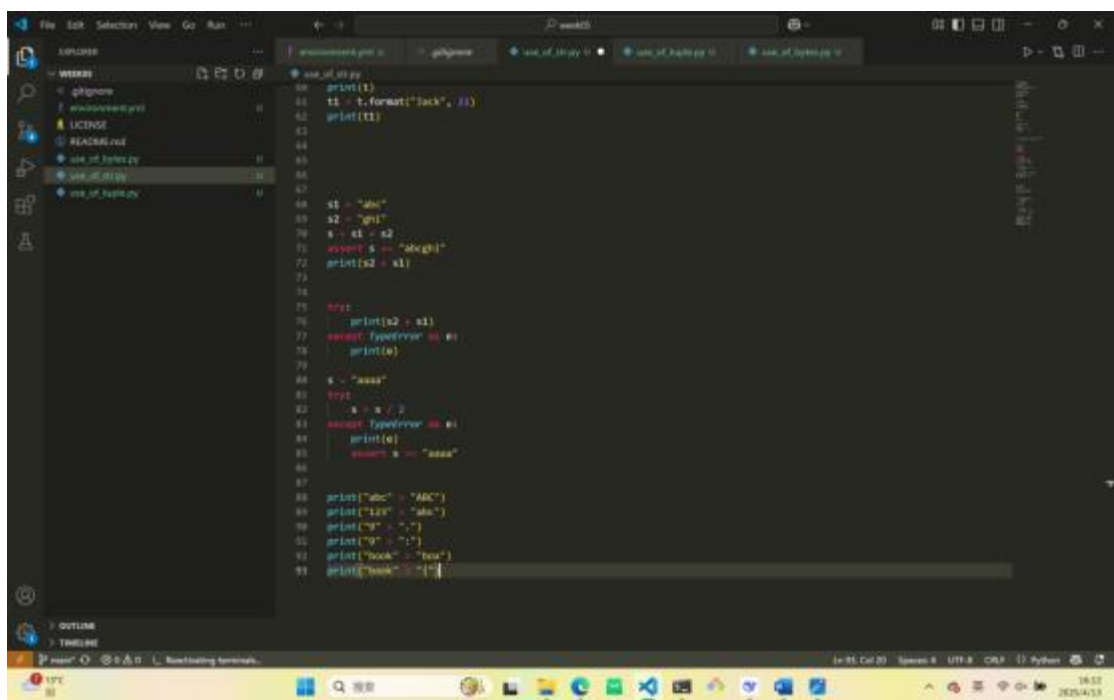
[illegible]

对于比较运算符 (>、=、<=) 有没有支持



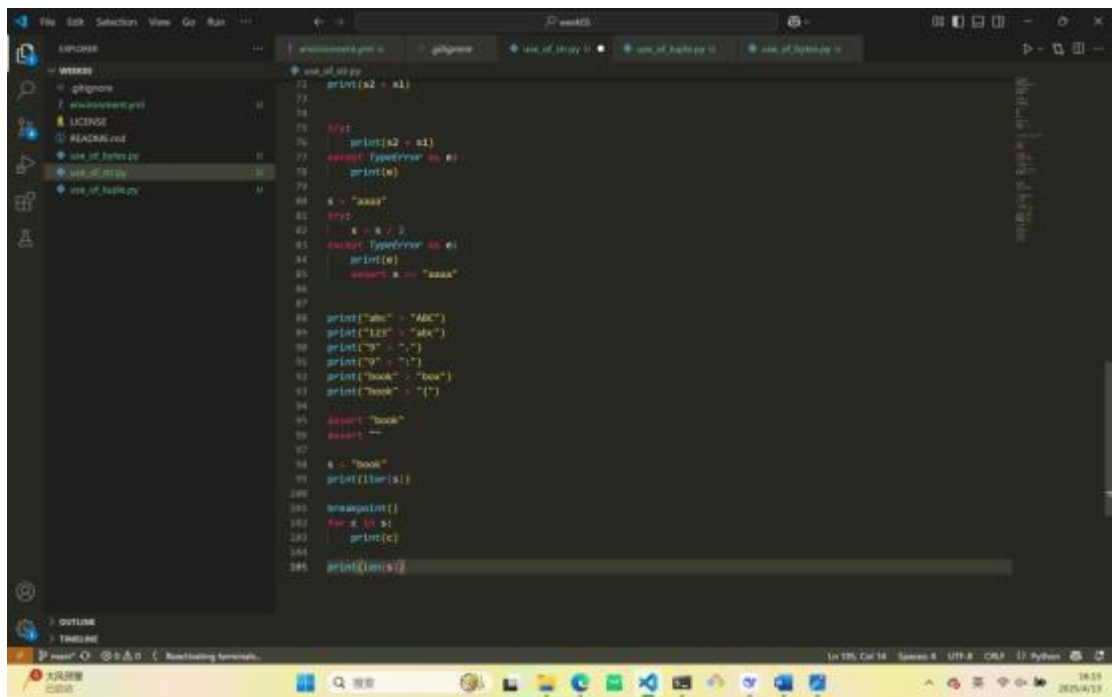
```
10 print(t)
11 t1 = t.format("jack", 33)
12 print(t1)
13
14
15
16 s1 = "abc"
17 s2 = "ghi"
18 s = s1 + s2
19 except: s == "abcghi"
20 print(s2 + s1)
21
22
23
24
25 try:
26     print(s2 + s1)
27 except TypeError as e:
28     print(e)
29
30 s = "aaaa"
31 try:
32     s = s + 2
33 except TypeError as e:
34     print(e)
35     s += "aaaa"
36
37
38 print("abc" + "ABC")
39 print("123" + "abc")
40
```

什么值被当作 True，什么值被当作 False

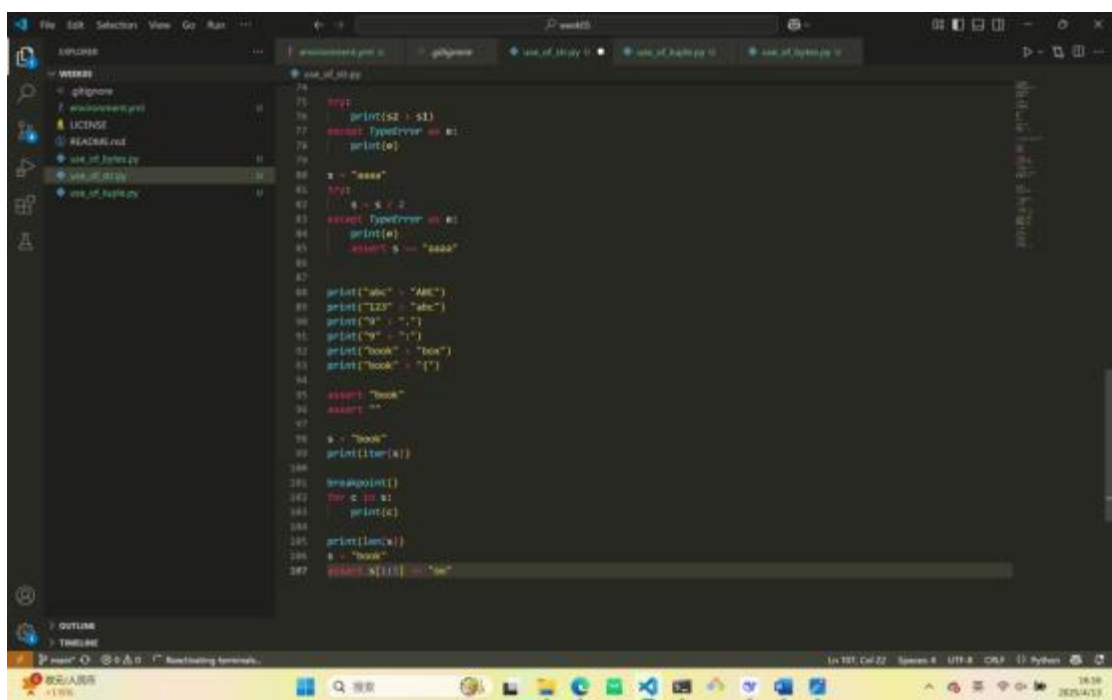


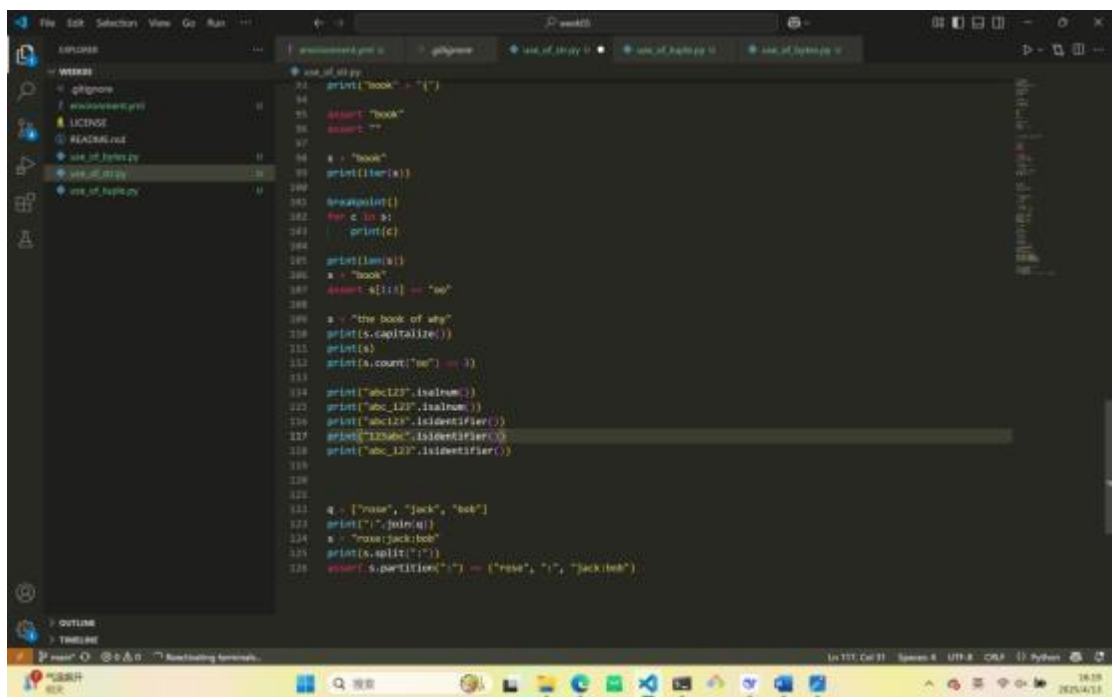
```
10 print(t)
11 t1 = t.format("jack", 33)
12 print(t1)
13
14
15
16 s1 = "abc"
17 s2 = "ghi"
18 s = s1 + s2
19 except: s == "abcghi"
20 print(s2 + s1)
21
22
23
24
25 try:
26     print(s2 + s1)
27 except TypeError as e:
28     print(e)
29
30 s = "aaaa"
31 try:
32     s = s + 2
33 except TypeError as e:
34     print(e)
35     s += "aaaa"
36
37
38 print("abc" + "ABC")
39 print("123" + "abc")
40 print("9" + ".")
41 print("9" + "1")
42 print("9abc" + "1abc")
43 print("9abc" + "1")
```

是否支持返回长度 (len)

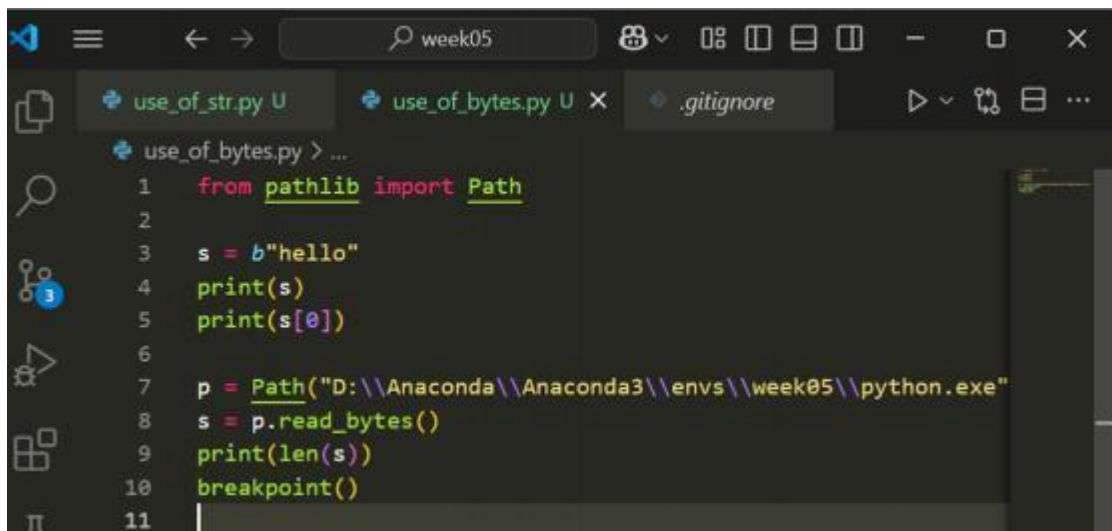


是否 (如何) 支持索引操作 (subscription) (`[]` 运算符)





6.1 字节和返回长度 (len)

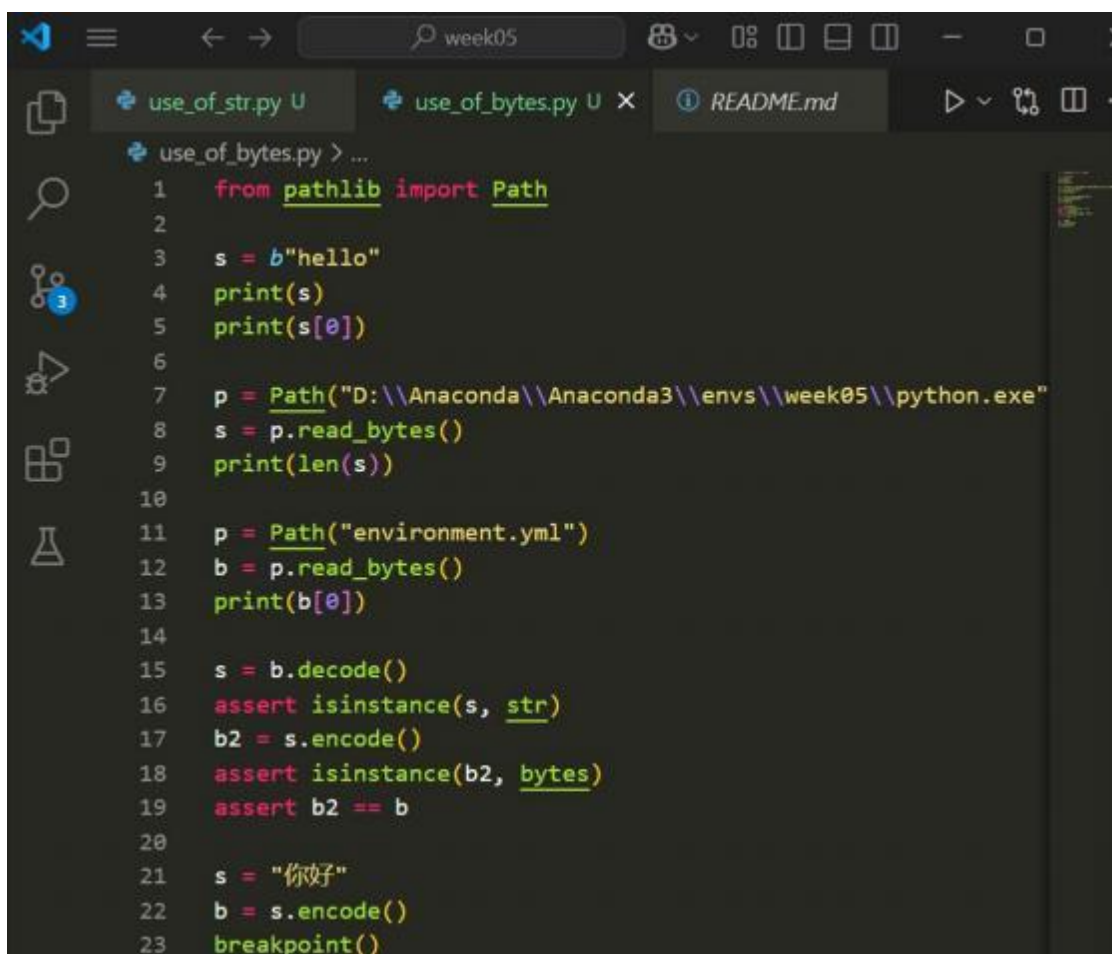


```
1 from pathlib import Path
2
3 s = b"hello"
4 print(s)
5 print(s[0])
6
7 p = Path("D:\\Anaconda\\Anaconda3\\envs\\week05\\python.exe")
8 s = p.read_bytes()
9 print(len(s))
10 breakpoint()
11
```

6.2 字符串编解码

字符串编码得到字节，字节解码得到字符串，编解码方案有很多，

<https://wwwh.ascii-code.com>，为其中一种。



```
1 from pathlib import Path
2
3 s = b"hello"
4 print(s)
5 print(s[0])
6
7 p = Path("D:\\Anaconda\\Anaconda3\\envs\\week05\\python.exe")
8 s = p.read_bytes()
9 print(len(s))
10
11 p = Path("environment.yml")
12 b = p.read_bytes()
13 print(b[0])
14
15 s = b.decode()
16 assert isinstance(s, str)
17 b2 = s.encode()
18 assert isinstance(b2, bytes)
19 assert b2 == b
20
21 s = "你好"
22 b = s.encode()
23 breakpoint()
```

```

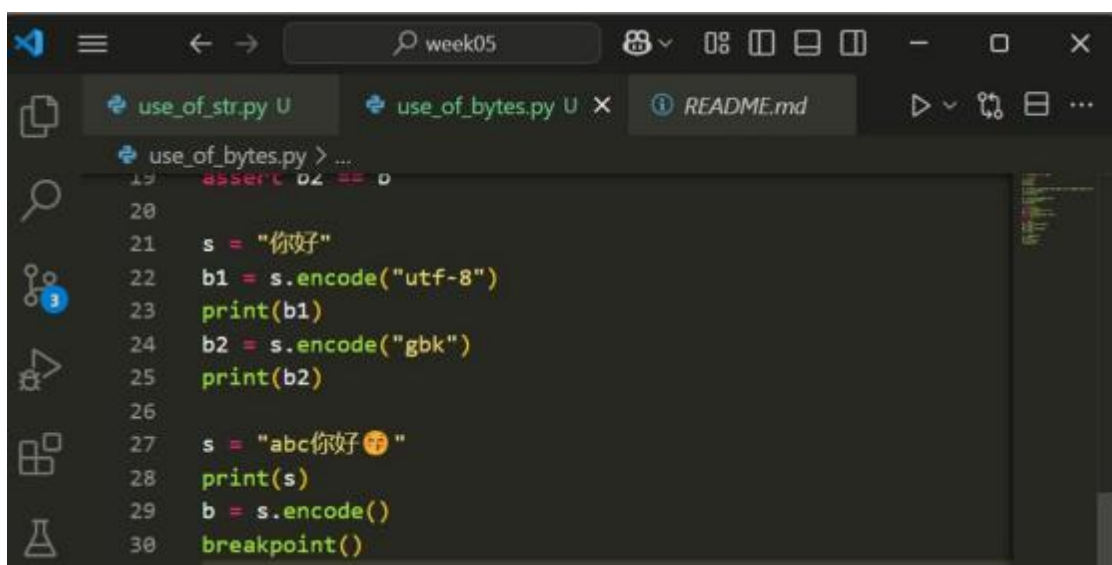
$ python use_of_bytes.py
b'hello'
104
93184
110
--Return--
> e:\研究生上课资料\研一下 上课资料\金融编程与计算\week05\use_of
_bytes.py(23)<module>()->None
-> breakpoint()
(Pdb) l
18     assert isinstance(b2, bytes)
19     assert b2 == b
20
21     s = "你好"
22     b = s.encode()

```

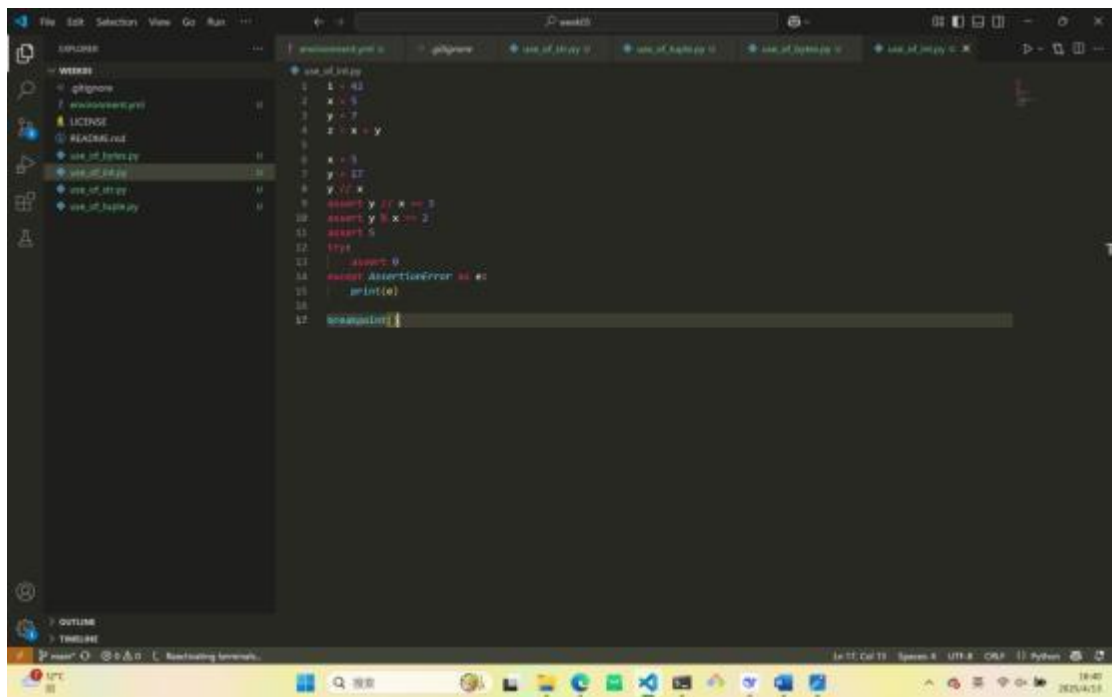
```

20
21     s = "你好"
22     b1 = s.encode("utf-8")
23     print(b1)
24     b2 = s.encode("gbk")
25     print(b2)
26     breakpoint()
27

```

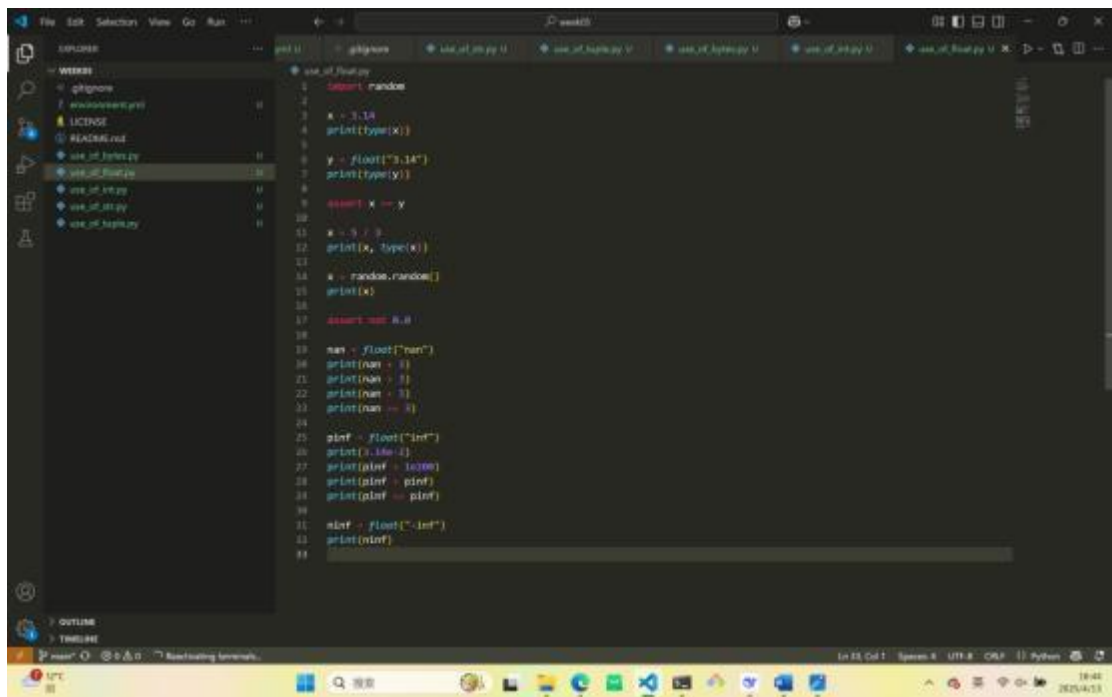


7、创建 use_of_int.py:



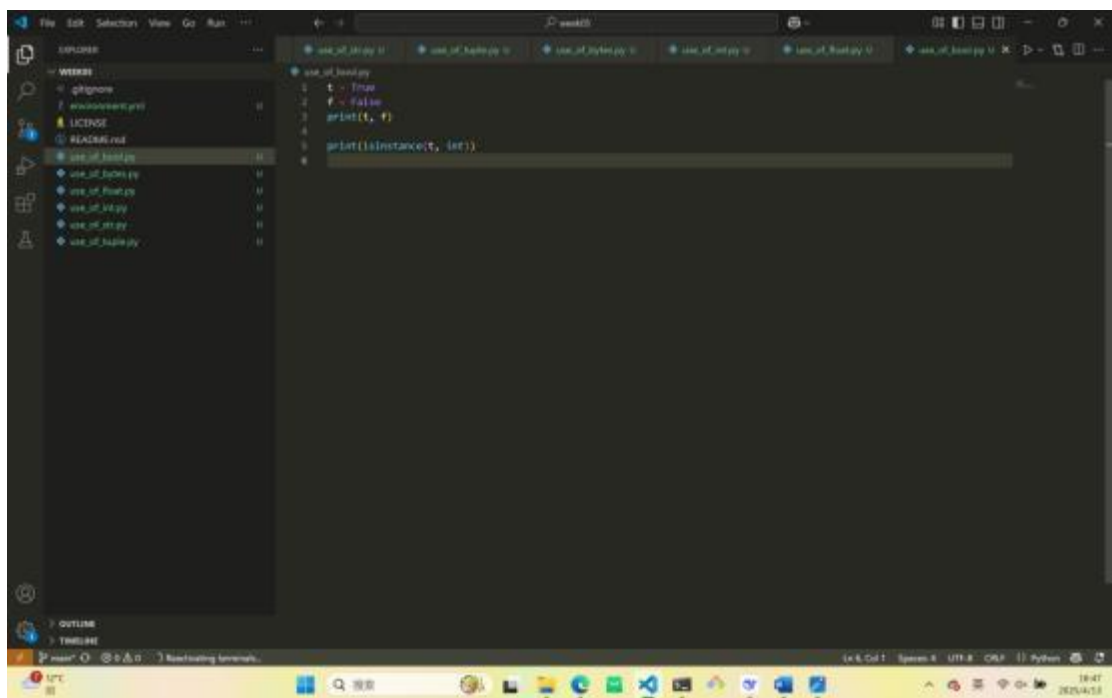
```
(Pdb) for i in x:  
*** IndentationError: expected an indented block after 'for' sta  
tement on line 1  
(Pdb) for i in x: print(i)  
*** TypeError: 'int' object is not iterable  
(Pdb) p iter(x)  
*** TypeError: 'int' object is not iterable  
(Pdb) p len(x)  
*** TypeError: object of type 'int' has no len()  
(Pdb) p x[0]  
*** TypeError: 'int' object is not subscriptable  
(Pdb) █
```

8.创建 use_of_float.py:



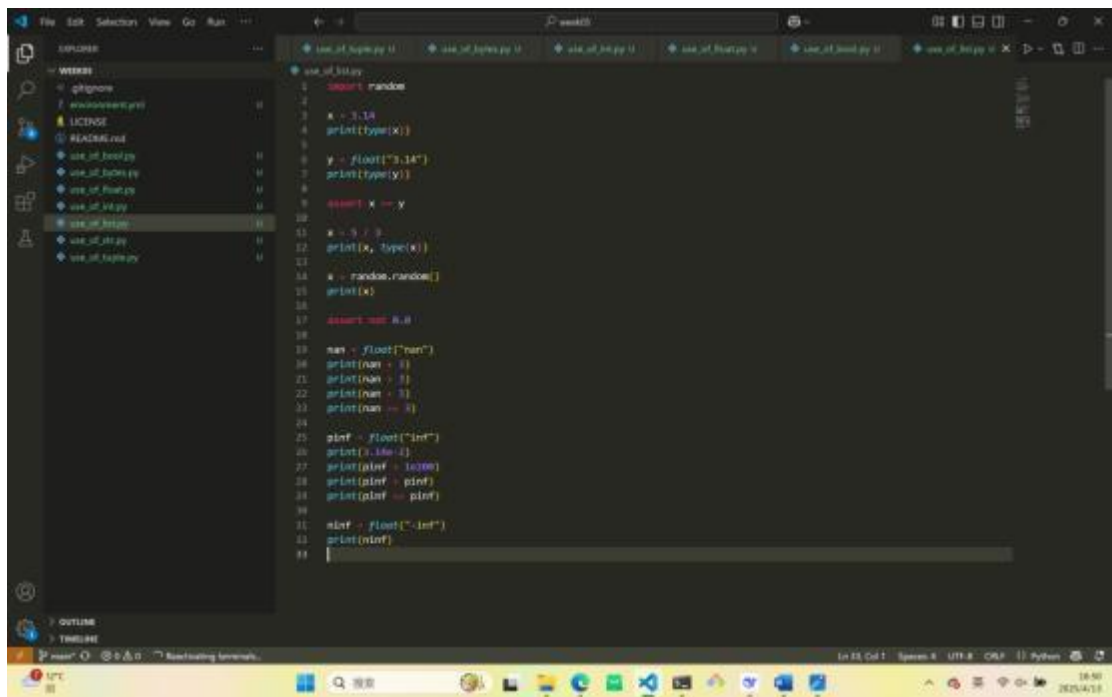
```
1 import random
2
3 x = 3.14
4 print(type(x))
5
6 y = float("3.14")
7 print(type(y))
8
9 assert x == y
10
11 x = 5 / 2
12 print(x, type(x))
13
14 x = random.random()
15 print(x)
16
17 assert not x > 1
18
19 nan = float("nan")
20 print(nan == 1)
21 print(nan == 1)
22 print(nan == 1)
23 print(nan == 1)
24
25 pinf = float("inf")
26 print(1 <= 1)
27 print(pinf < 1000)
28 print(pinf > pinf)
29 print(pinf == pinf)
30
31 ninf = float("-inf")
32 print(ninf)
```

9.创建 use_of_bool.py:

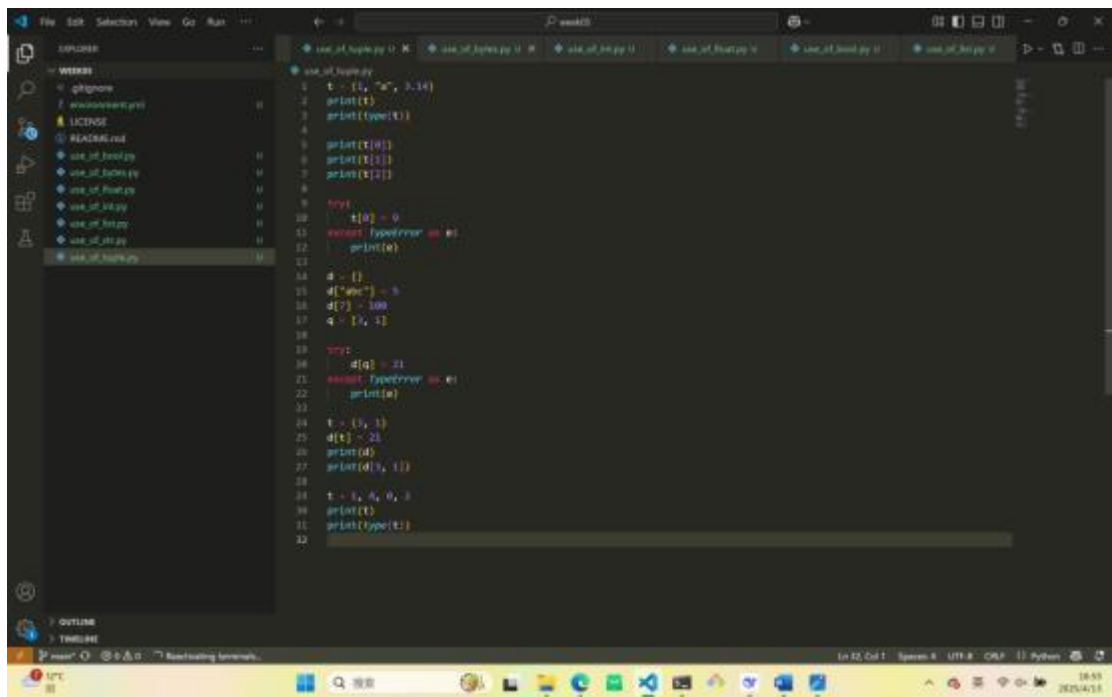


```
1 t = True
2 f = False
3 print(t, f)
4
5 print(isinstance(t, int))
```

10.创建 use_of_list.py:



11.创建 use_of_tuple.py:

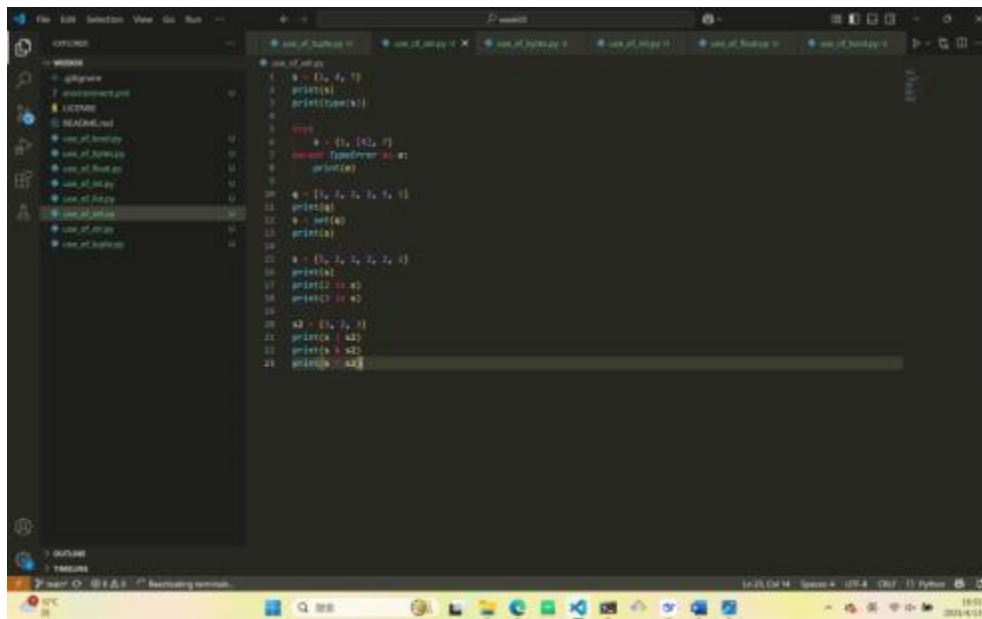


```

$ python use_of_tuple.py
(1, 'a', 3.14)
<class 'tuple'>
1
a
3.14
'tuple' object does not support item assignment
unhashable type: 'list'
{'abc': 5, 7: 100, (3, 1): 21}
21
(1, 4, 0, 2)
<class 'tuple'>
(week05)

```

12.创建 use_of_set.py:

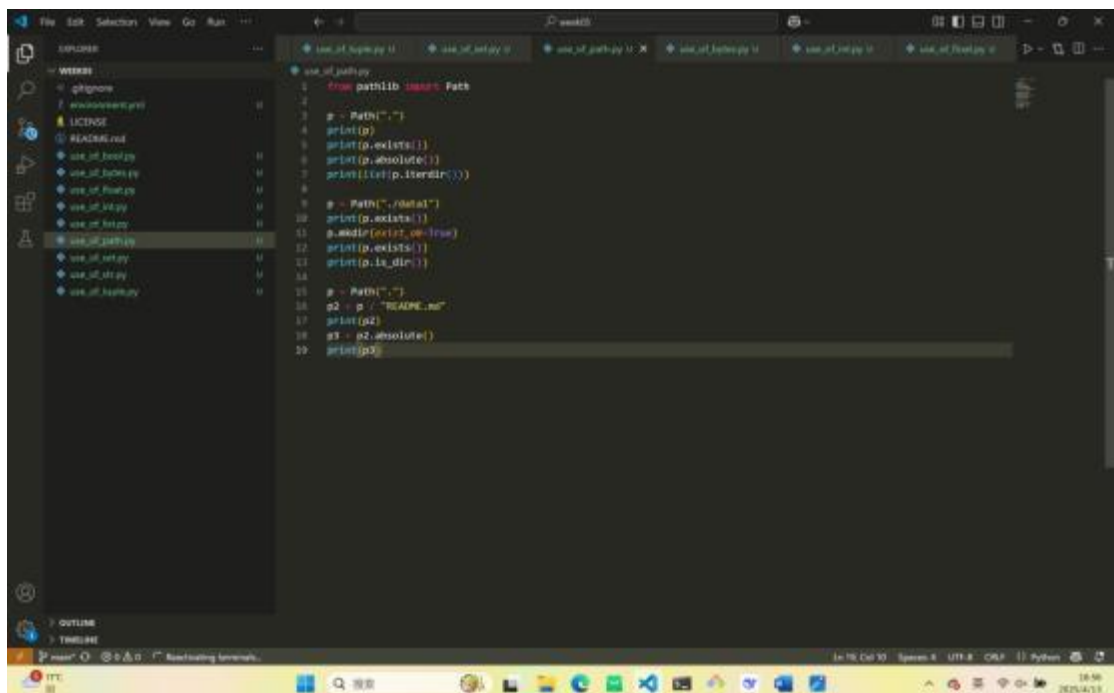


```

$ python use_of_set.py
{1, 4, 7}
<class 'set'>
unhashable type: 'list'
[1, 2, 1, 2, 5, 1]
{1, 2, 5}
{1, 2, 5}
True
False
(week05)

```

13.创建 use_of_path.py:




```

[EOF]
(Pdb) import wat
(Pdb) wat / p

str: .
repr: WindowsPath('.')
type: pathlib.WindowsPath
parents: pathlib.Path, pathlib.PureWindowsPath, pathlib.PurePath

Public attributes:
  anchor: str = ''
  drive: str = ''
  name: str = ''
  parent: pathlib.WindowsPath = .
  parents: pathlib._PathParents = <WindowsPath.parents>
  parts: tuple = ()
  root: str = ''
  stem: str = ''
  suffix: str = ''
  suffixes: list = []

  def absolute() # Return an absolute version of this path by pr
    pending the current...
  def as_posix() # Return the string representation of the path
    with forward (/)...
  def as_uri() # Return the path as a 'file' URI.
  def chmod(mode, *, follow_symlinks=True) # Change the permissi
    ons of the path, like os.chmod().
  def cwd() # Return a new path pointing to the current working
    directory.
  def exists(*, follow_symlinks=True) # Whether this path exists
  def expanduser() # Return a new path with expanded ~ and ~user

```

14.创建 use_of_datetime.py

