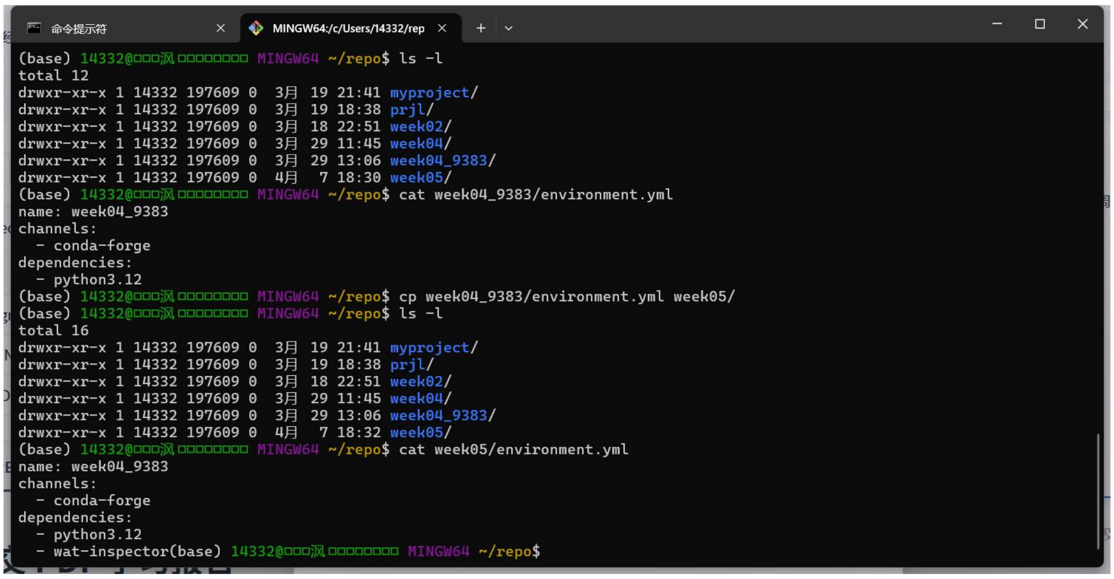
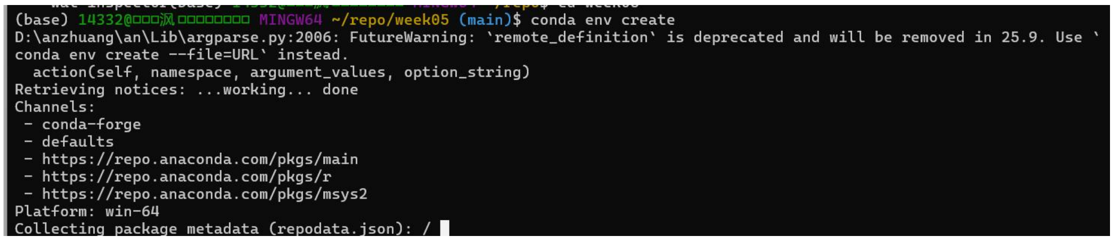
part1-常用的对象检视函数和语句

1.用 VS Code 打开项目目录，新建一个 environment.yml 文件，指定安装 Python 3.12，然后运行 conda env create 命令创建 Conda 环境

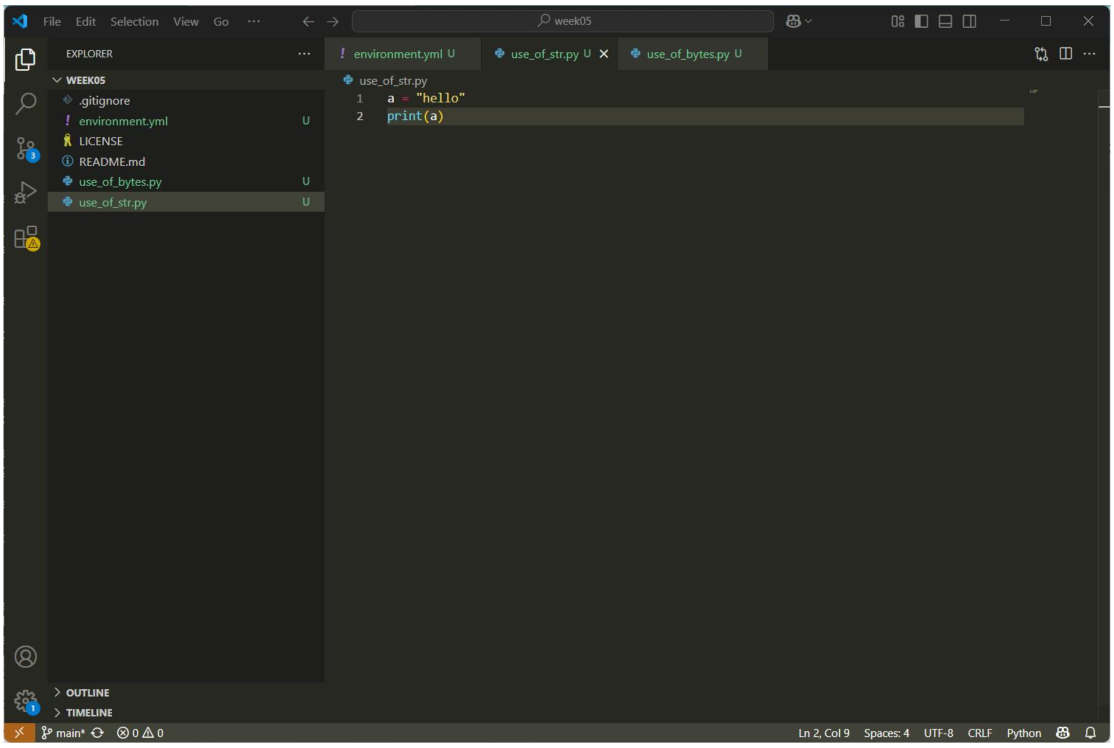


```
(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo$ ls -l
total 12
drwxr-xr-x 1 14332 197609 0  3月  19 21:41 myproject/
drwxr-xr-x 1 14332 197609 0  3月  19 18:38 prjl/
drwxr-xr-x 1 14332 197609 0  3月  18 22:51 week02/
drwxr-xr-x 1 14332 197609 0  3月  29 11:45 week04/
drwxr-xr-x 1 14332 197609 0  3月  29 13:06 week04_9383/
drwxr-xr-x 1 14332 197609 0  4月   7 18:30 week05/
(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo$ cat week04_9383/environment.yml
name: week04_9383
channels:
  - conda-forge
dependencies:
  - python3.12
(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo$ cp week04_9383/environment.yml week05/
(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo$ ls -l
total 16
drwxr-xr-x 1 14332 197609 0  3月  19 21:41 myproject/
drwxr-xr-x 1 14332 197609 0  3月  19 18:38 prjl/
drwxr-xr-x 1 14332 197609 0  3月  18 22:51 week02/
drwxr-xr-x 1 14332 197609 0  3月  29 11:45 week04/
drwxr-xr-x 1 14332 197609 0  3月  29 13:06 week04_9383/
drwxr-xr-x 1 14332 197609 0  4月   7 18:32 week05/
(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo$ cat week05/environment.yml
name: week04_9383
channels:
  - conda-forge
dependencies:
  - python3.12
  - wat-inspector(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo$
```

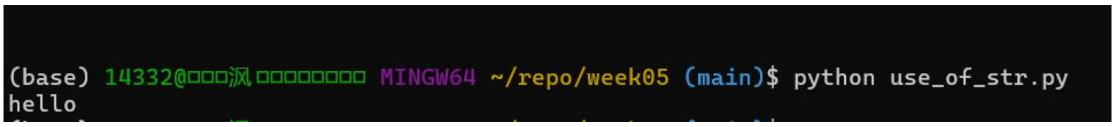


```
(base) 14332@□□□沨□□□□□□□□ MINGW64 ~/repo/week05 (main)$ conda env create
D:\anzhuang\an\Lib\argparse.py:2006: FutureWarning: `remote_definition` is deprecated and will be removed in 25.9. Use `
conda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - defaults
 - 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): /
```

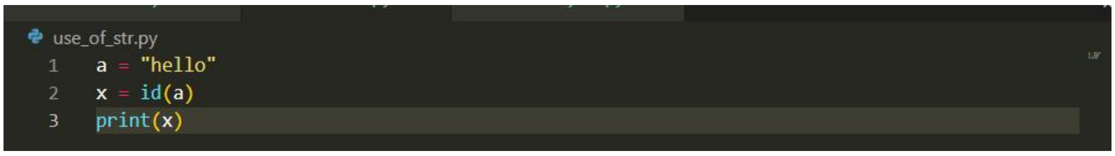
2. 逐个 创建 use_of_{name}.py 文件，
3. id() -- 返回对象在虚拟内存中的地址 (正整数)，如果 id(a) == id(b)，那么 a is b (is 是个运算符)



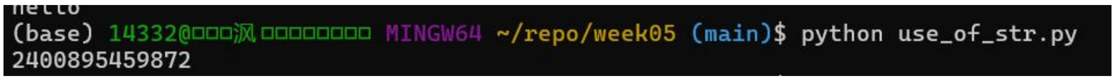
```python
a = "hello"
print(a)
```



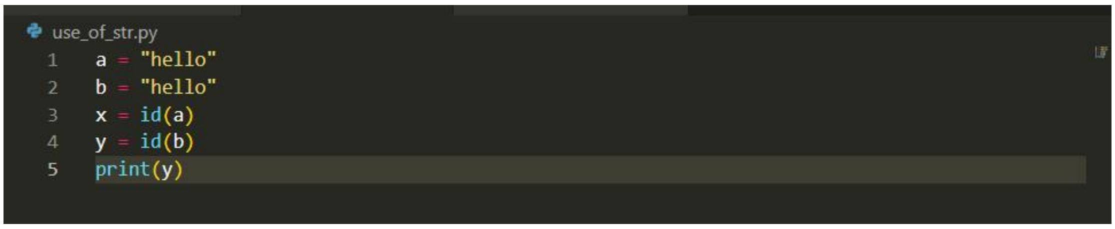
```
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.py
hello
```



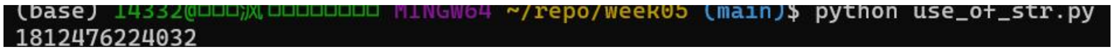
```python
a = "hello"
x = id(a)
print(x)
```



```
hello
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.py
2400895459872
```

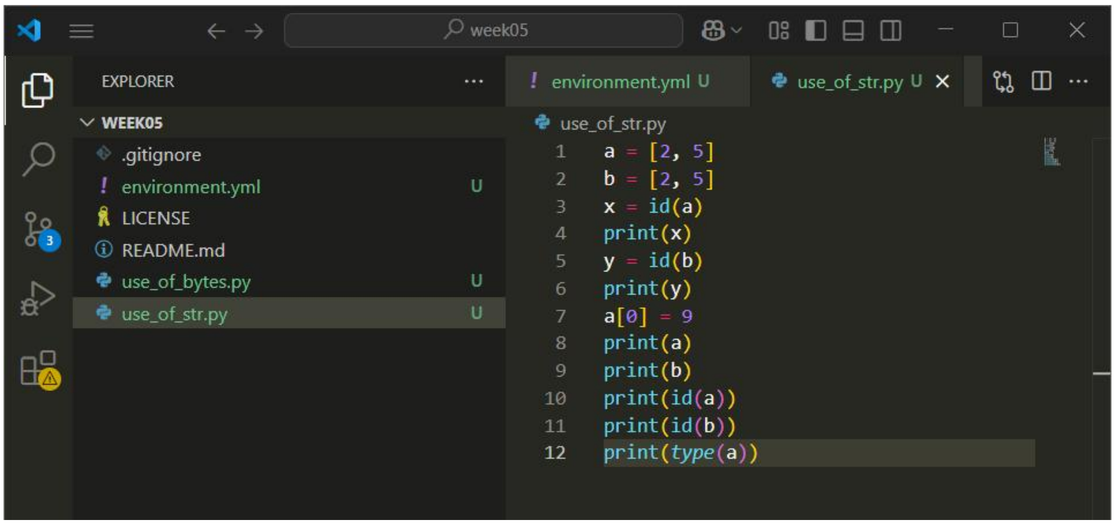


```python
a = "hello"
b = "hello"
x = id(a)
y = id(b)
print(y)
```

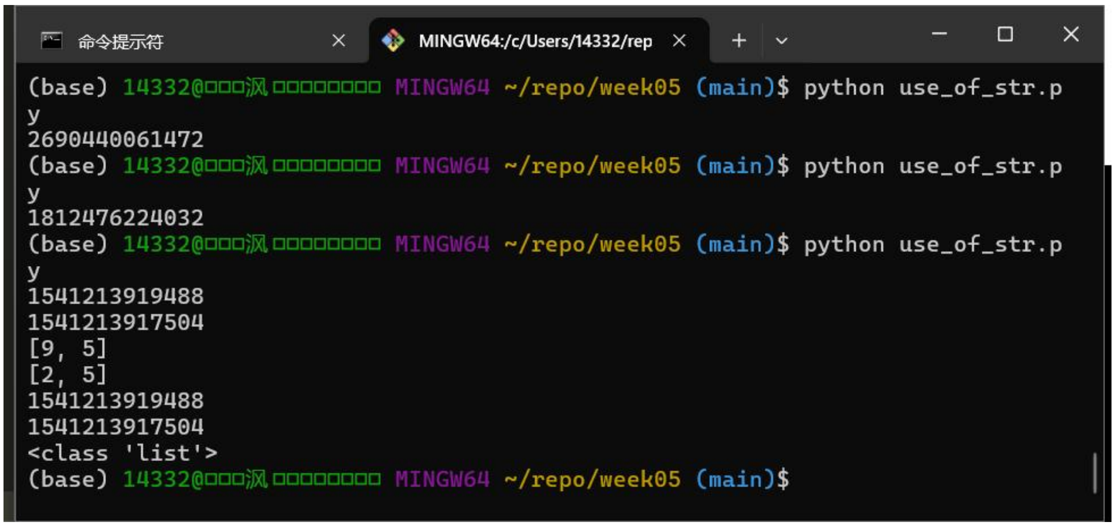


```
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.py
1812476224032
```

4. type() -- 返回对象的类型

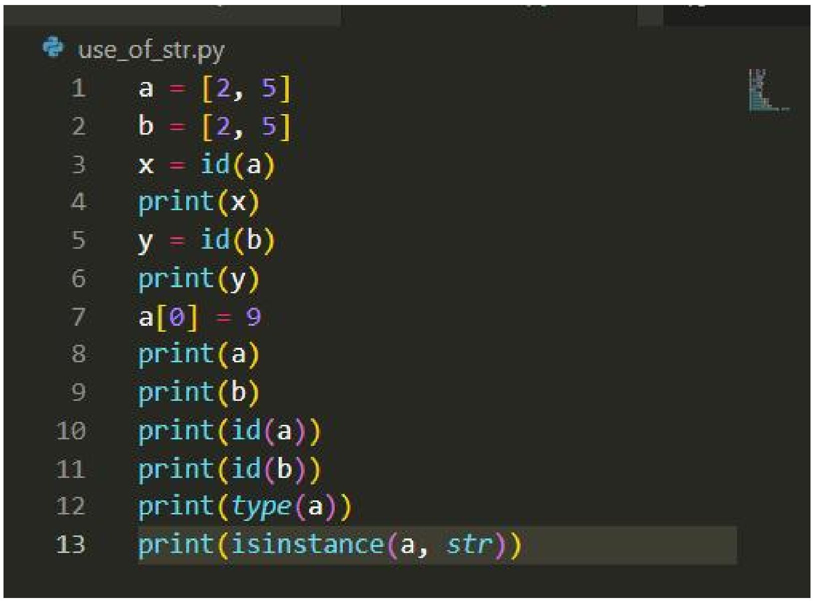


```python
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
```

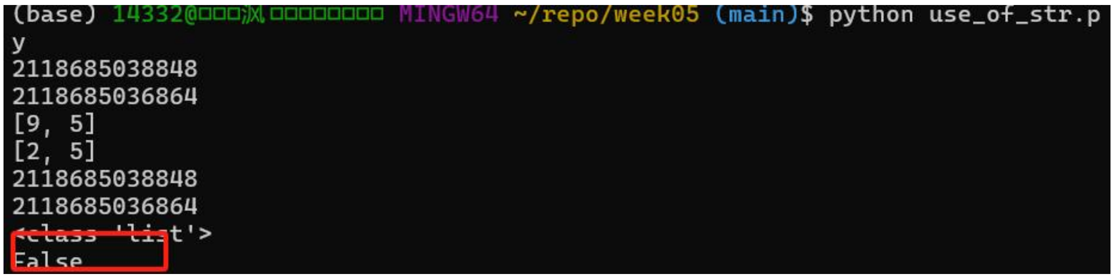


```
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
2690440061472
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
1812476224032
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
1541213919488
1541213917504
[9, 5]
[2, 5]
1541213919488
1541213917504
<class 'list'>
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$
```

5.

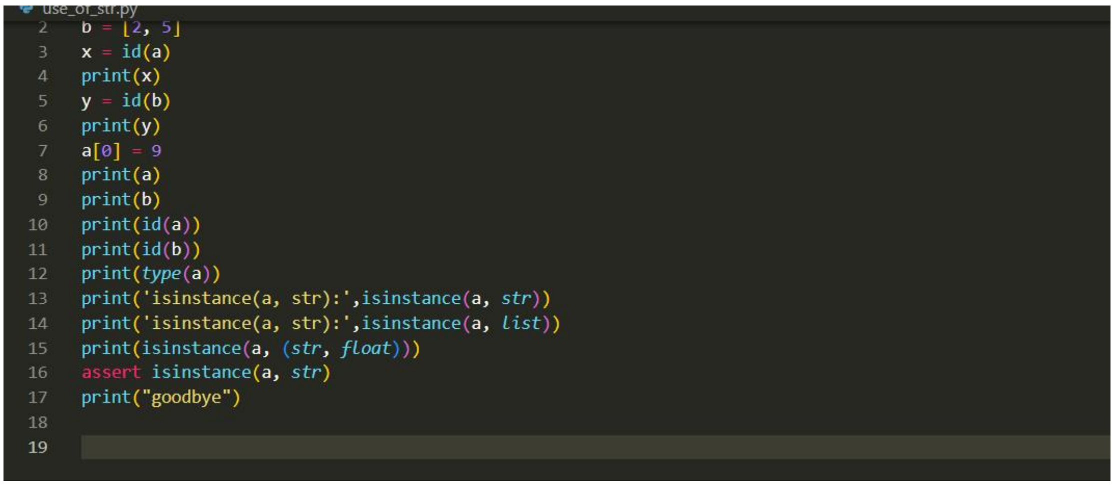


```
use_of_str.py
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print(isinstance(a, str))
```

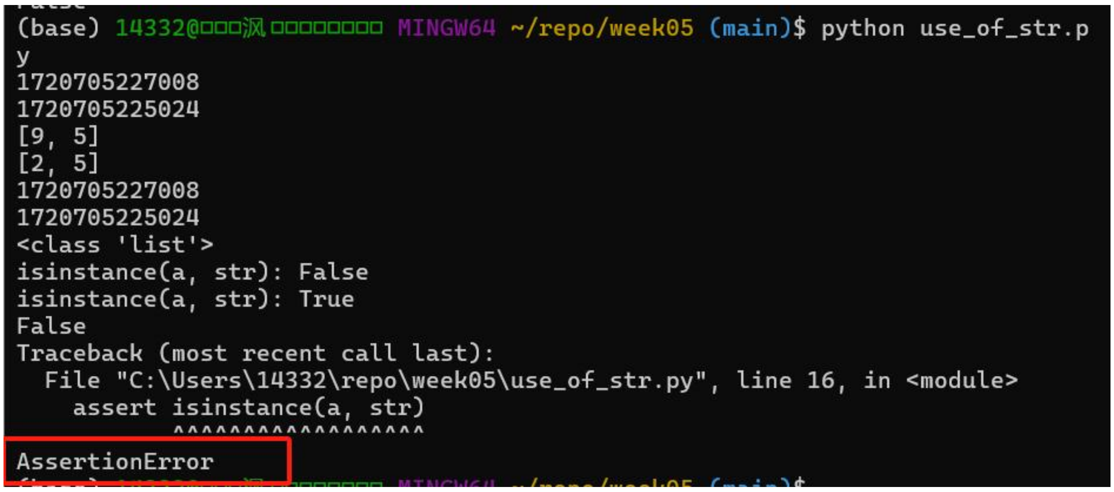


```
(base) 14332@□□□泅 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
2118685038848
2118685036864
[9, 5]
[2, 5]
2118685038848
2118685036864
<class 'list'>
False
```

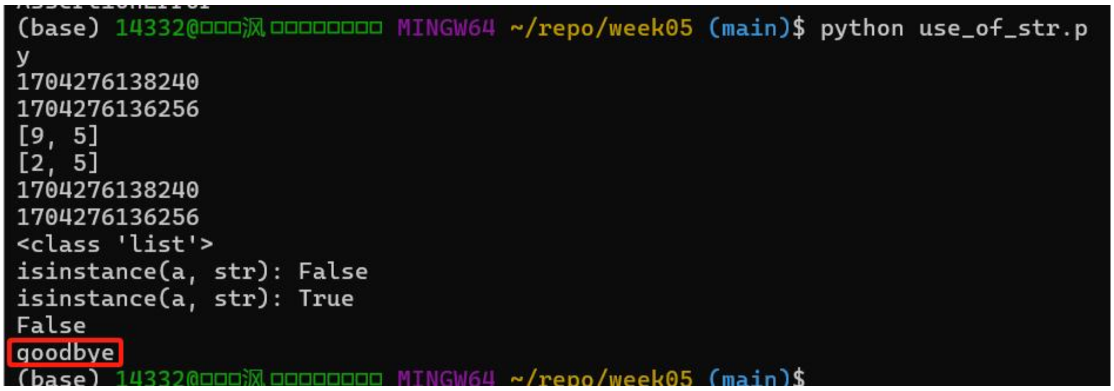
6. 利用 assert 语句查验某个表达式 (expression) 为真，否则报错 (AssertionError) 退出



```
use_of_str.py
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print('isinstance(a, str):',isinstance(a, str))
print('isinstance(a, str):',isinstance(a, list))
print(isinstance(a, (str, float)))
assert isinstance(a, str)
print("goodbye")


```



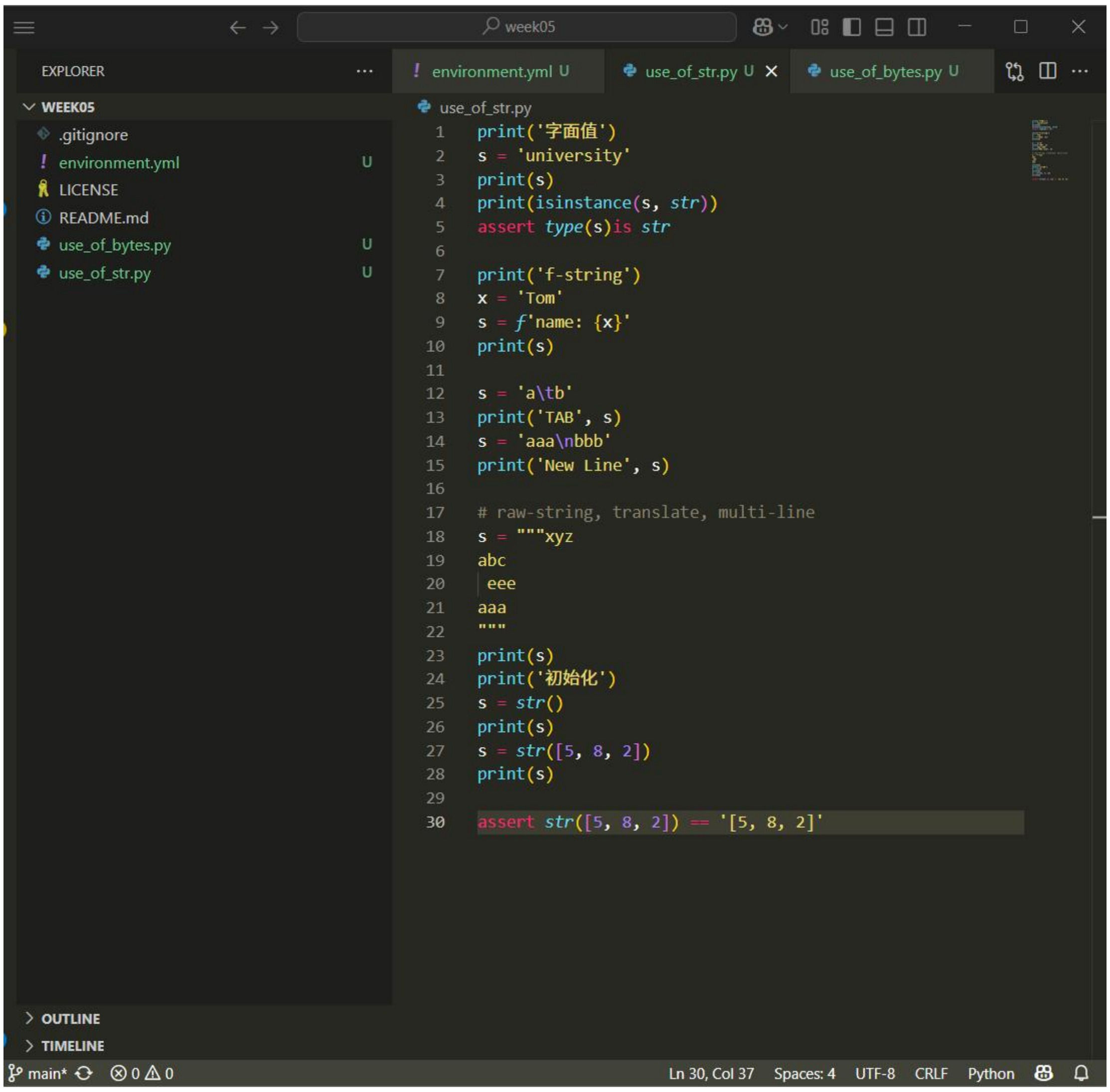
```
(base) 14332@□□□沨□□□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
1720705227008
1720705225024
[9, 5]
[2, 5]
1720705227008
1720705225024
<class 'list'>
isinstance(a, str): False
isinstance(a, str): True
False
Traceback (most recent call last):
  File "C:\Users\14332\repo\week05\use_of_str.py", line 16, in <module>
    assert isinstance(a, str)
           ^^^^^^^^^^^^^^^^^^
AssertionError
```



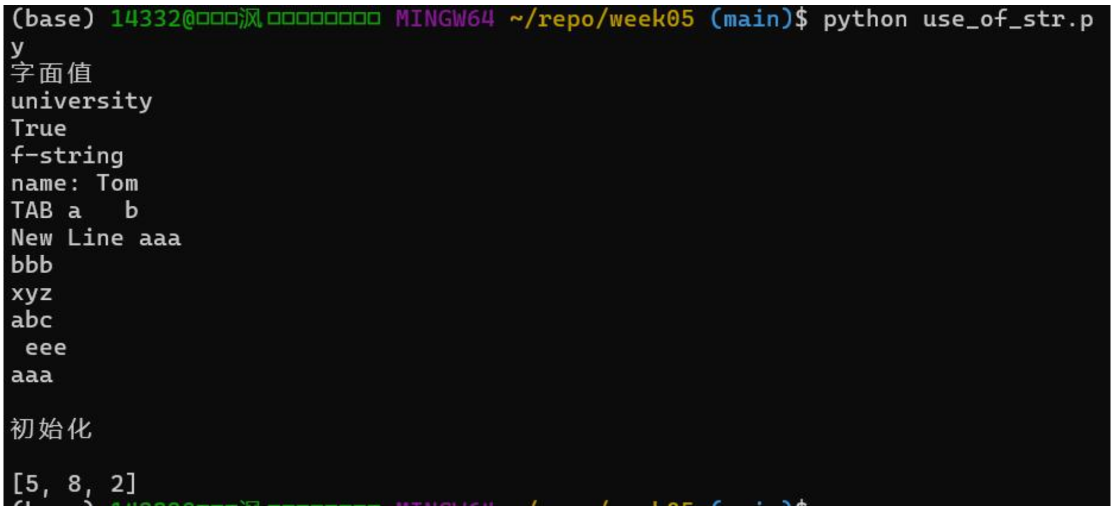
```
(base) 14332@□□□沨□□□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
1704276138240
1704276136256
[9, 5]
[2, 5]
1704276138240
1704276136256
<class 'list'>
isinstance(a, str): False
isinstance(a, str): True
False
goodbye
(base) 14332@□□□沨□□□□□□□□□ MINGW64 ~/repo/week05 (main)$
```

part2-获得 str 类型实例的几种途径

熟悉如何通过表达式（expression）得到对象类型的实例（instance）



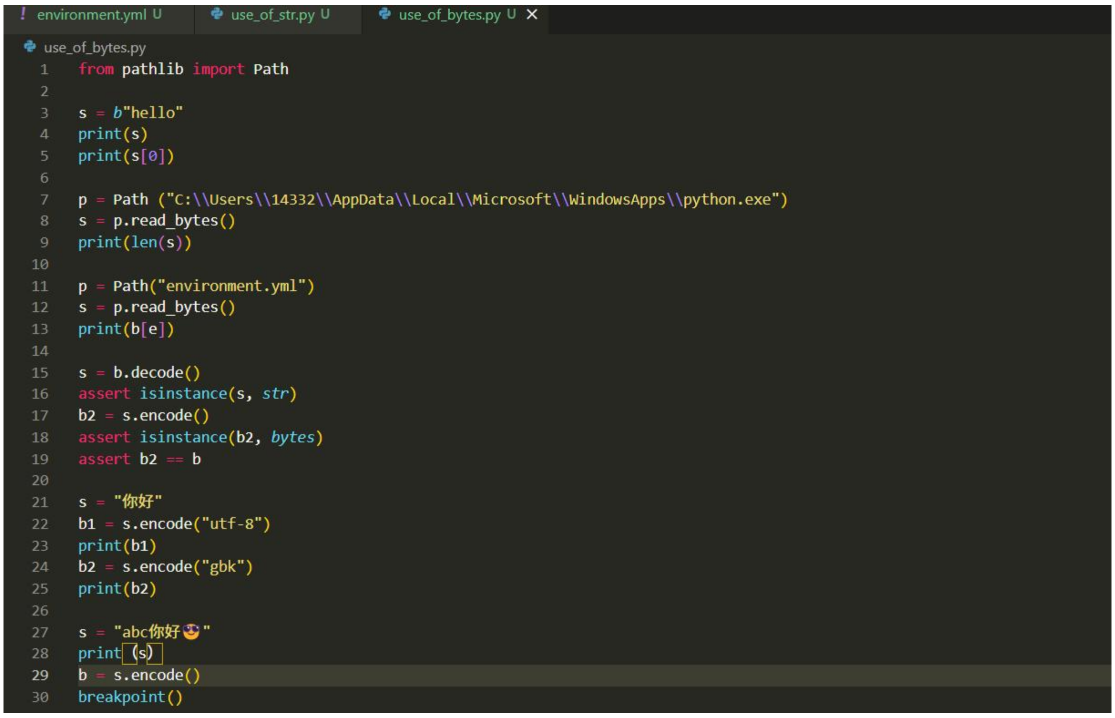
```python
print('字面值')
s = 'university'
print(s)
print(isinstance(s, str))
assert type(s)is str

print('f-string')
x = 'Tom'
s = f'name: {x}'
print(s)

s = 'a\tb'
print('TAB', s)
s = 'aaa\nbbb'
print('New Line', s)

# raw-string, translate, multi-line
s = """xyz
abc
 eee
aaa
"""
print(s)
print('初始化')
s = str()
print(s)
s = str([5, 8, 2])
print(s)

assert str([5, 8, 2]) == '[5, 8, 2]'
```



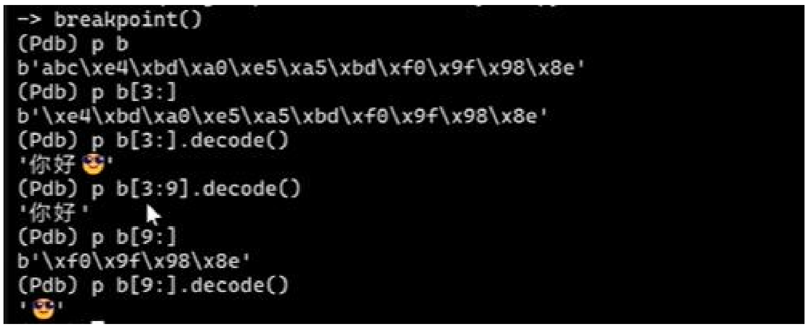
```
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_str.p
y
字面值
university
True
f-string
name: Tom
TAB a	b
New Line aaa
bbb
xyz
abc
 eee
aaa

初始化

[5, 8, 2]
```

part4-bytes 编解码和 int 整数

1.字节 bytes 模式的练习

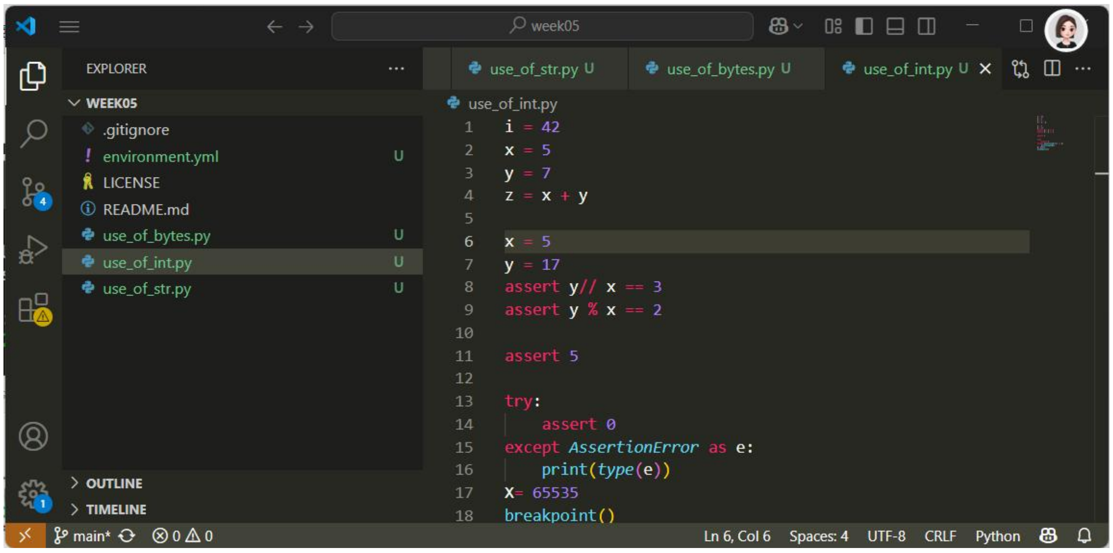


```python
from pathlib import Path

s = b"hello"
print(s)
print(s[0])

p = Path ("C:\\Users\\14332\\AppData\\Local\\Microsoft\\WindowsApps\\python.exe")
s = p.read_bytes()
print(len(s))

p = Path("environment.yml")
s = p.read_bytes()
print(b[e])

s = b.decode()
assert isinstance(s, str)
b2 = s.encode()
assert isinstance(b2, bytes)
assert b2 == b

s = "你好"
b1 = s.encode("utf-8")
print(b1)
b2 = s.encode("gbk")
print(b2)

s = "abc你好😎"
print (s)
b = s.encode()
breakpoint()
```

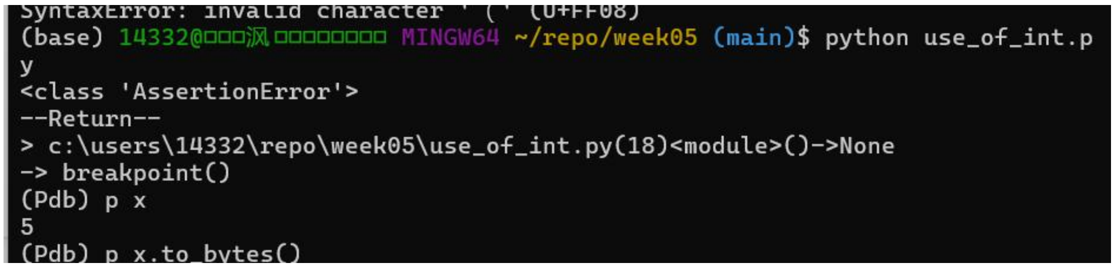


```
-> breakpoint()
(Pdb) p b
b'abc\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\x8e'
(Pdb) p b[3:]
b'\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\x8e'
(Pdb) p b[3:].decode()
'你好😎'
(Pdb) p b[3:9].decode()
'你好'
(Pdb) p b[9:]
b'\xf0\x9f\x98\x8e'
(Pdb) p b[9:].decode()
'😎'
```

2.对于掌握的对象类型进行对数学运算符（+、-、*、/、//、%、@）有没有支持的练习

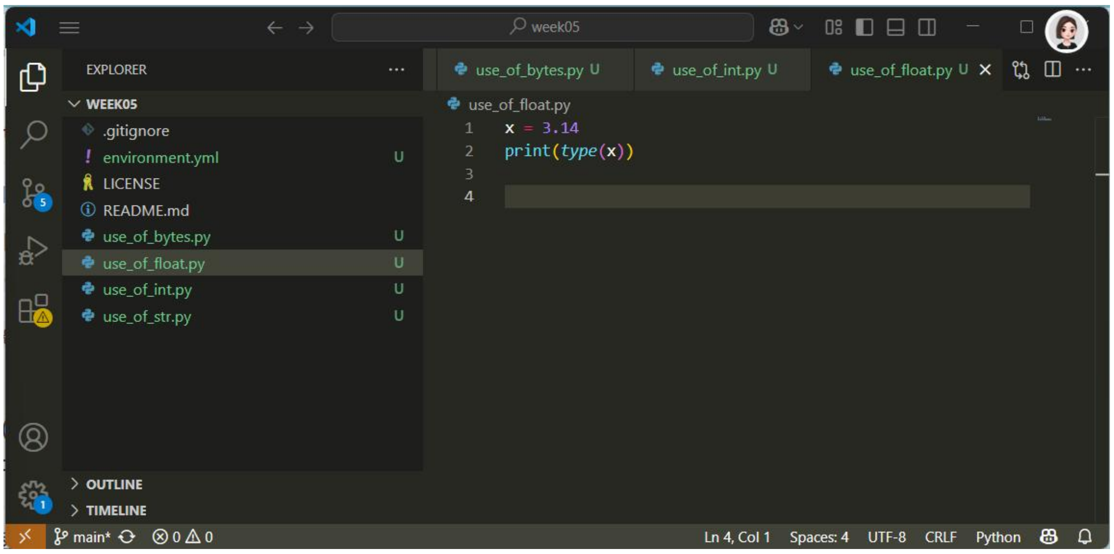


```python
i = 42
x = 5
y = 7
z = x + y

x = 5
y = 17
assert y// x == 3
assert y % x == 2

assert 5

try:
    assert 0
except AssertionError as e:
    print(type(e))
X= 65535
breakpoint()
```

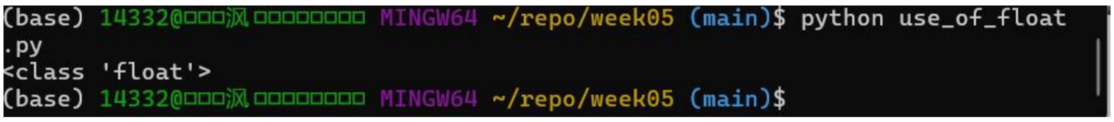

```
SyntaxError: invalid character '（' (U+FF08)
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_int.p
y
<class 'AssertionError'>
--Return--
> c:\users\14332\repo\week05\use_of_int.py(18)<module>()->None
-> breakpoint()
(Pdb) p x
5
(Pdb) p x.to_bytes()
```

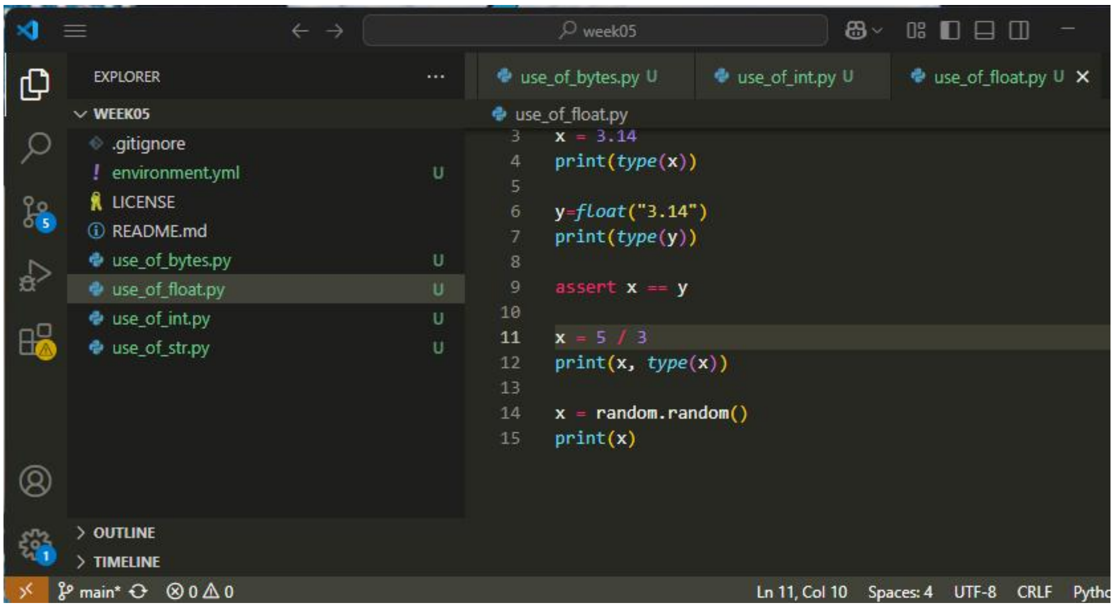
part5-float~dict 等类型

1. float 模式练习

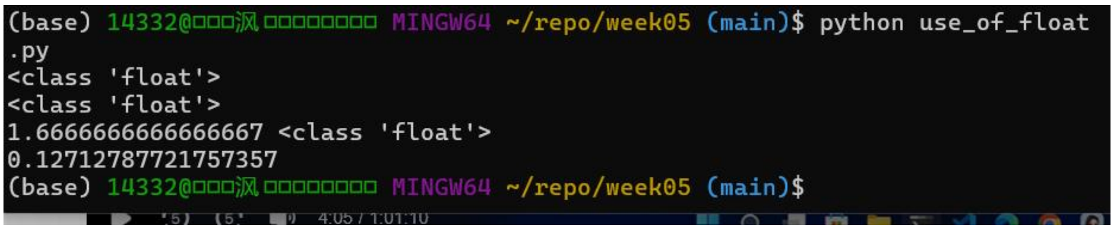

use_of_float.py

```python
x = 3.14
print(type(x))

```


```
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_float
.py
<class 'float'>
(base) 14332@□□□沨 □□□□□□□□ MINGW64 ~/repo/week05 (main)$
```

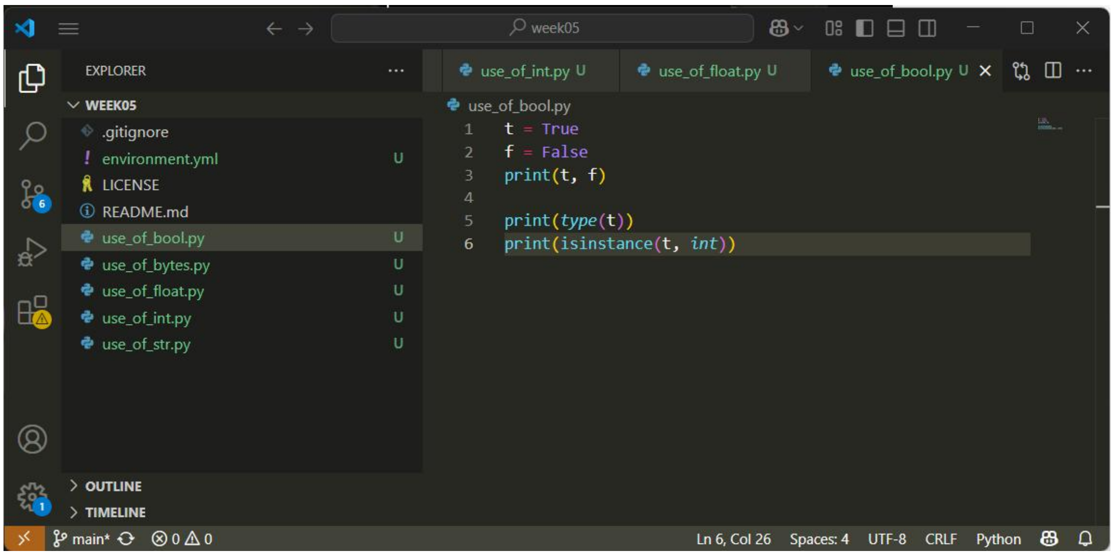

use_of_float.py

```python
x = 3.14
print(type(x))

y=float("3.14")
print(type(y))

assert x == y

x = 5 / 3
print(x, type(x))

x = random.random()
print(x)
```

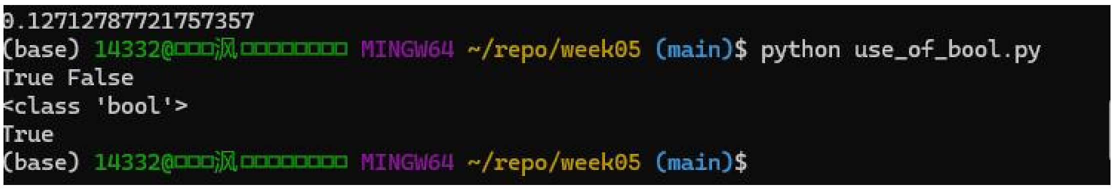


```
(base) 14332@□□□汎□□□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_float
.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.12712787721757357
(base) 14332@□□□汎□□□□□□□□□ MINGW64 ~/repo/week05 (main)$
```

2. Bool 类型

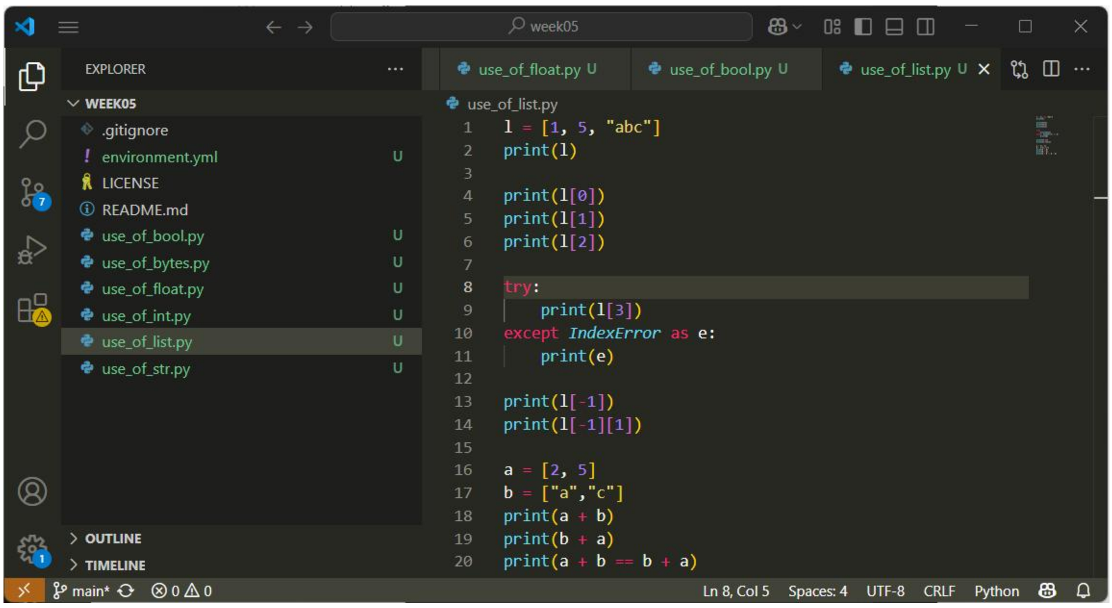


```python
t = True
f = False
print(t, f)

print(type(t))
print(isinstance(t, int))
```

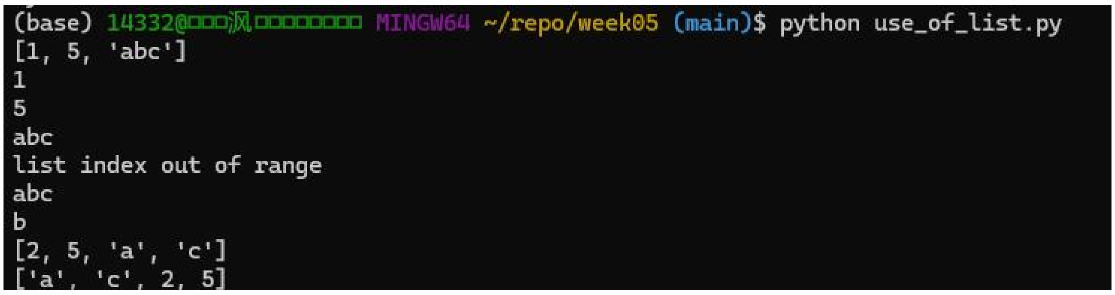


```
0.12712787721757357
(base) 14332@□□□汎□□□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_bool.py
True False
<class 'bool'>
True
(base) 14332@□□□汎□□□□□□□□□ MINGW64 ~/repo/week05 (main)$
```

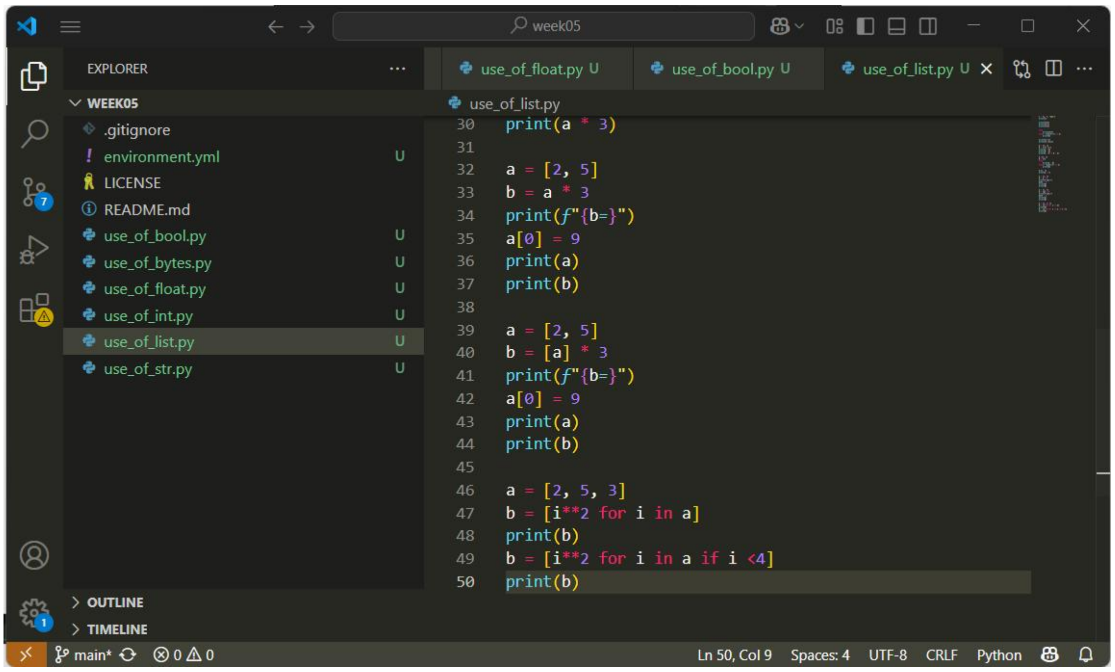
3. List 类型



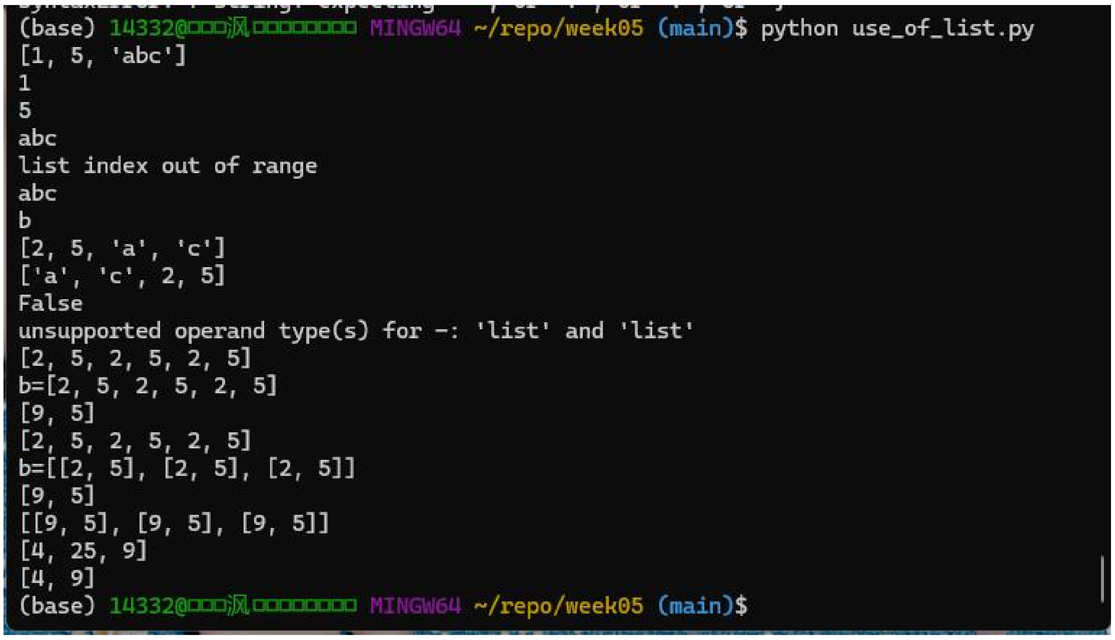
```python
l = [1, 5, "abc"]
print(l)

print(l[0])
print(l[1])
print(l[2])

try:
    print(l[3])
except IndexError as e:
    print(e)

print(l[-1])
print(l[-1][1])

a = [2, 5]
b = ["a","c"]
print(a + b)
print(b + a)
print(a + b == b + a)
```



```
(base) 14332@□□□沨□□□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_list.py
[1, 5, 'abc']
1
5
abc
list index out of range
abc
b
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
```

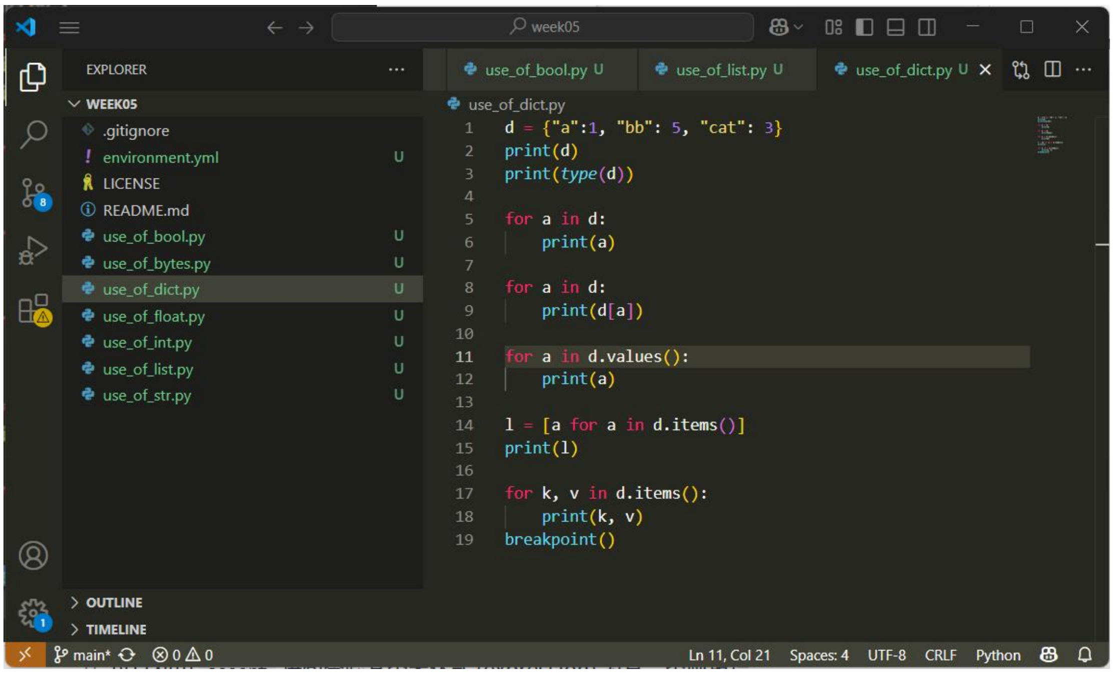


```python
print(a * 3)

a = [2, 5]
b = a * 3
print(f"{b=}")
a[0] = 9
print(a)
print(b)

a = [2, 5]
b = [a] * 3
print(f"{b=}")
a[0] = 9
print(a)
print(b)

a = [2, 5, 3]
b = [i**2 for i in a]
print(b)
b = [i**2 for i in a if i <4]
print(b)
```



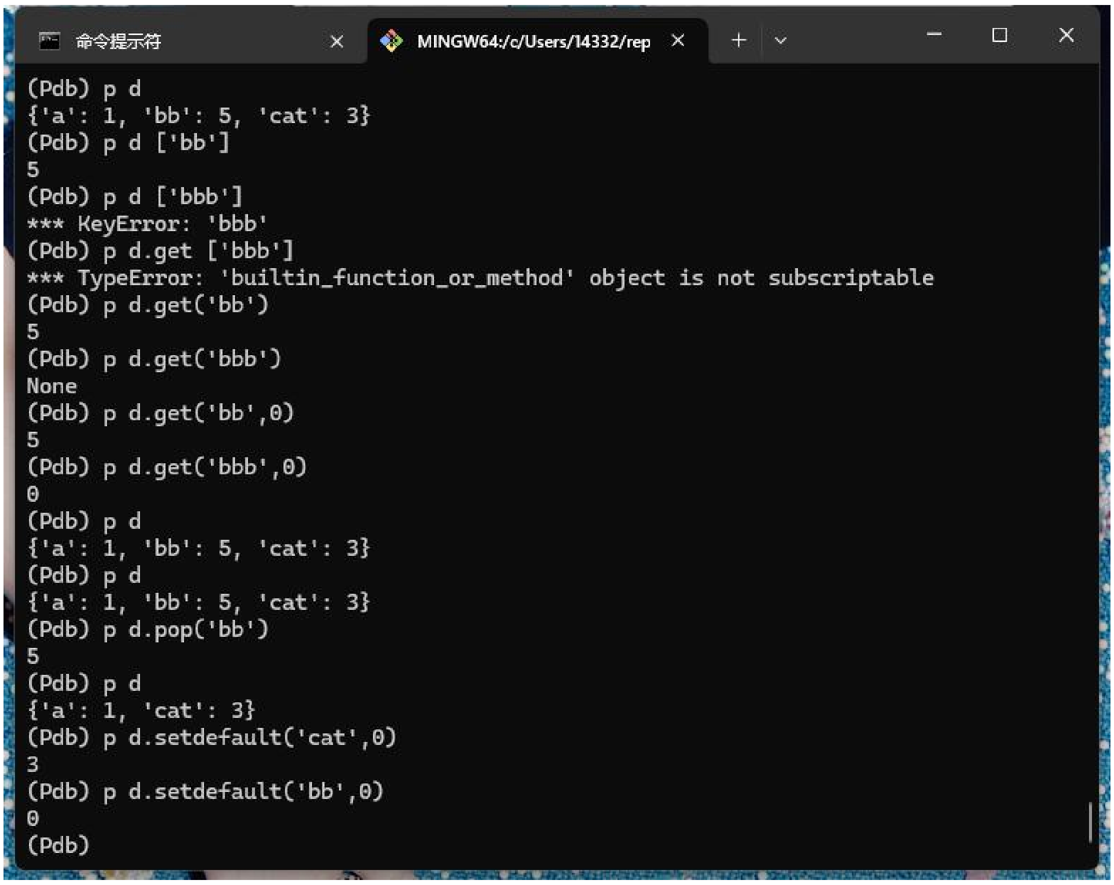
```
(base) 14332@□□□沨□□□□□□□□□ MINGW64 ~/repo/week05 (main)$ python use_of_list.py
[1, 5, 'abc']
1
5
abc
list index out of range
abc
b
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
False
unsupported operand type(s) for -: 'list' and 'list'
[2, 5, 2, 5, 2, 5]
b=[2, 5, 2, 5, 2, 5]
[9, 5]
[2, 5, 2, 5, 2, 5]
b=[[2, 5], [2, 5], [2, 5]]
[9, 5]
[[9, 5], [9, 5], [9, 5]]
[4, 25, 9]
[4, 9]
(base) 14332@□□□沨□□□□□□□□□ MINGW64 ~/repo/week05 (main)$
```

4. Dict 类型



use_of_dict.py

```python
d = {"a":1, "bb": 5, "cat": 3}
print(d)
print(type(d))

for a in d:
    print(a)

for a in d:
    print(d[a])

for a in d.values():
    print(a)

l = [a for a in d.items()]
print(l)

for k, v in d.items():
    print(k, v)
breakpoint()
```



```
(Pdb) p d
{'a': 1, 'bb': 5, 'cat': 3}
(Pdb) p d ['bb']
5
(Pdb) p d ['bbb']
*** KeyError: 'bbb'
(Pdb) p d.get ['bbb']
*** TypeError: 'builtin_function_or_method' object is not subscriptable
(Pdb) p d.get('bb')
5
(Pdb) p d.get('bbb')
None
(Pdb) p d.get('bb',0)
5
(Pdb) p d.get('bbb',0)
0
(Pdb) p d
{'a': 1, 'bb': 5, 'cat': 3}
(Pdb) p d
{'a': 1, 'bb': 5, 'cat': 3}
(Pdb) p d.pop('bb')
5
(Pdb) p d
{'a': 1, 'cat': 3}
(Pdb) p d.setdefault('cat',0)
3
(Pdb) p d.setdefault('bb',0)
0
(Pdb)
```