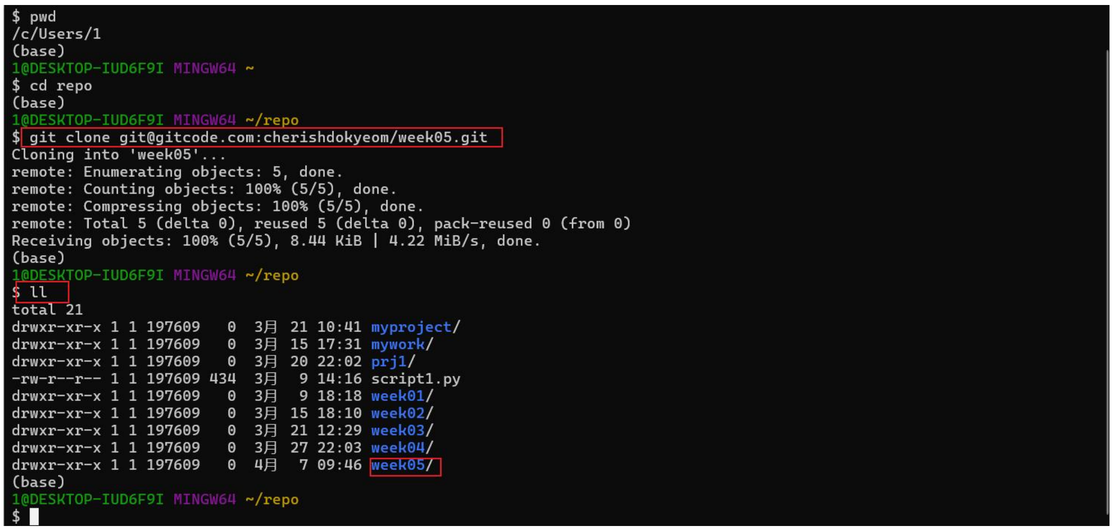
金融编程与计算-学习报告-week05

1.Fork 第 05 周打卡仓库至你的名下，然后将你名下的这个仓库 Clone 到你的本地计算机

若不小心将 week05 放到 week05 下，可以回到 repo 下，将 week05 整个文件夹删掉并重新 Clone。

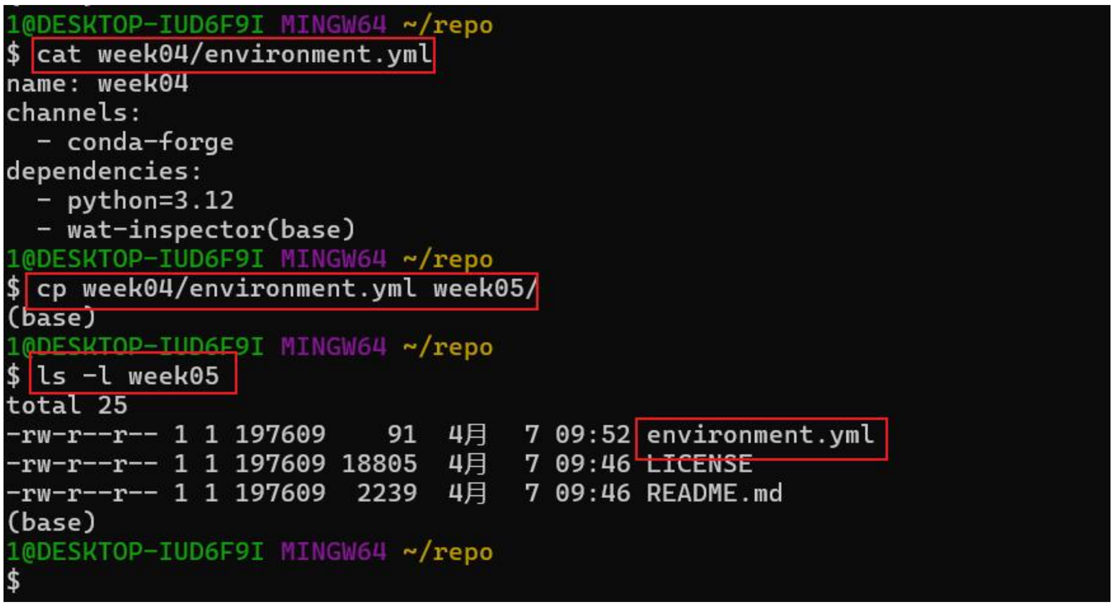
rm -rf weeko5



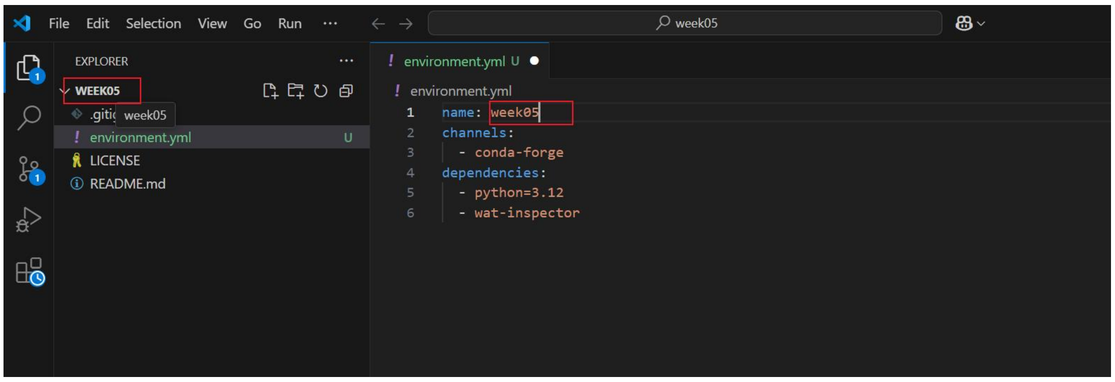
```
$ pwd
/c/Users/1
(base)
1@DESKTOP-IUD6F9I MINGW64 ~
$ cd repo
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$ git clone git@gitcode.com:cherishdokyeom/week05.git
Cloning into 'week05'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 5 (delta 0), reused 5 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), 8.44 KiB | 4.22 MiB/s, done.
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$ ll
total 21
drwxr-xr-x 1 1 197609   0  3月  21 10:41 myproject/
drwxr-xr-x 1 1 197609   0  3月  15 17:31 mywork/
drwxr-xr-x 1 1 197609   0  3月  20 22:02 prj1/
-rw-r--r-- 1 1 197609 434  3月   9 14:16 script1.py
drwxr-xr-x 1 1 197609   0  3月   9 18:18 week01/
drwxr-xr-x 1 1 197609   0  3月  15 18:10 week02/
drwxr-xr-x 1 1 197609   0  3月  21 12:29 week03/
drwxr-xr-x 1 1 197609   0  3月  27 22:03 week04/
drwxr-xr-x 1 1 197609   0  4月   7 09:46 week05/
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$ 
```

2.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

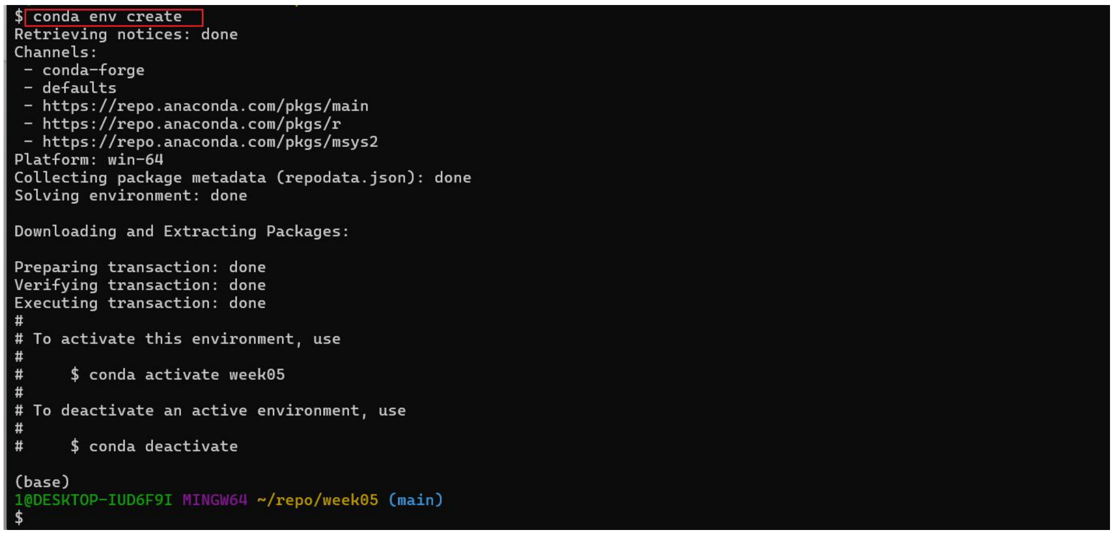
将 week04 下的 environment.yml 复制到 week05，再进行相应修改。



```
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$ cat week04/environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$ cp week04/environment.yml week05/
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$ ls -l week05
total 25
-rw-r--r-- 1 1 197609    91  4月   7 09:52 environment.yml
-rw-r--r-- 1 1 197609 18805  4月   7 09:46 LICENSE
-rw-r--r-- 1 1 197609  2239  4月   7 09:46 README.md
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo
$
```



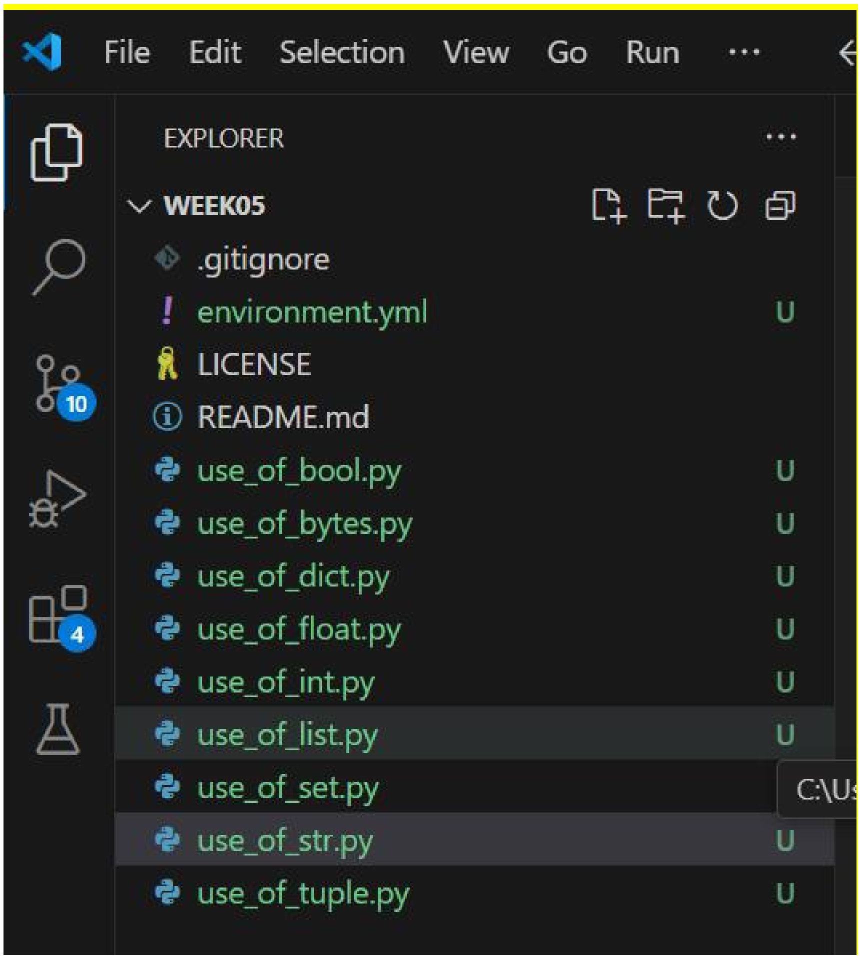
```
name: week05
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector
```



```
$ conda env create
Retrieving notices: 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): done
Solving environment: done

Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate week05
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

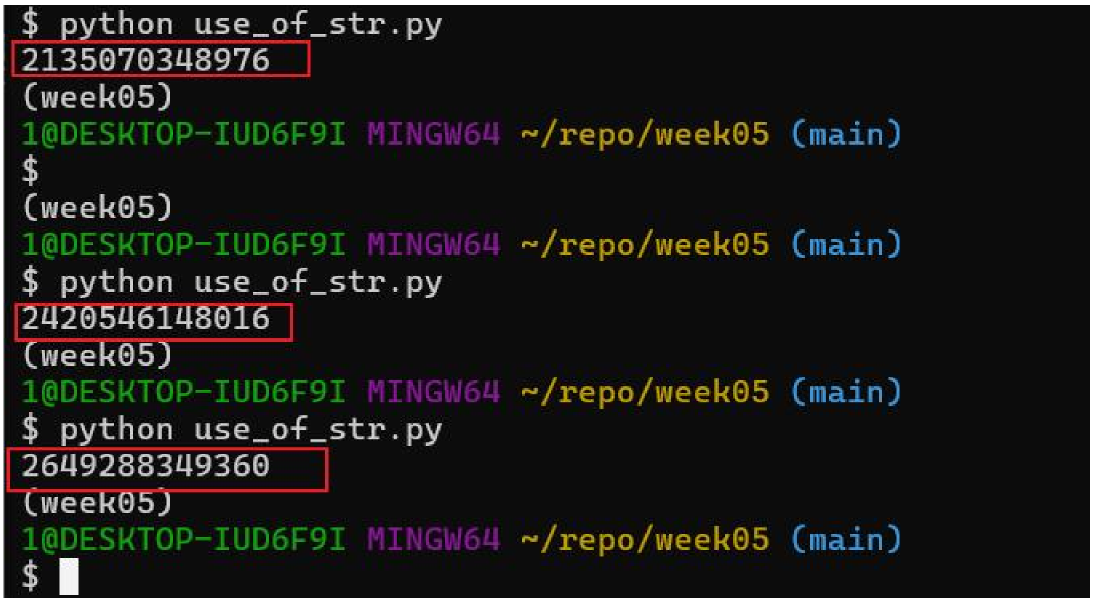
3.逐个创 use_of_{name}.py 文件，其中{name}替换为上述要求掌握的对象类型

虚拟地址，每次运行显示的结果都不同。



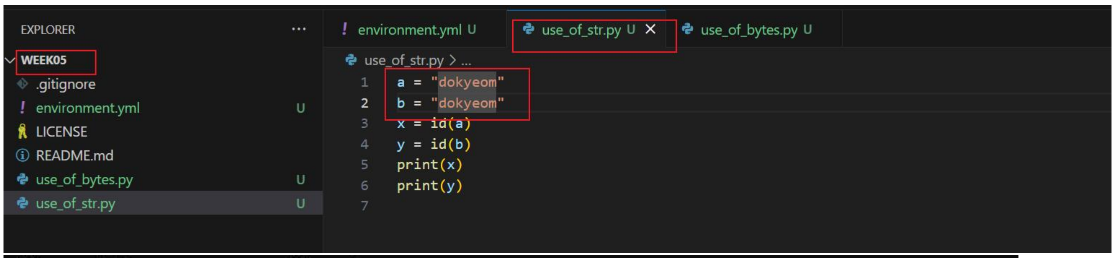
```
.gitignore
environment.yml      U
LICENSE
README.md
use_of_bool.py       U
use_of_bytes.py      U
use_of_dict.py       U
use_of_float.py      U
use_of_int.py        U
use_of_list.py       U
use_of_set.py
use_of_str.py        U
use_of_tuple.py      U
```

Str 字符串

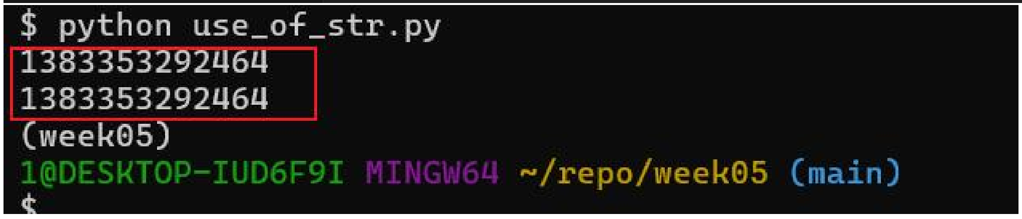


```
$ python use_of_str.py
2135070348976
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2420546148016
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2649288349360
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

虚拟地址，每次运行显示的结果都不同。

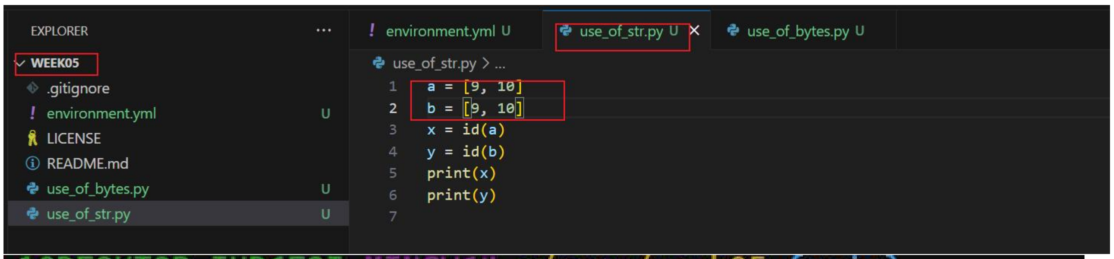


```python
a = "dokyeom"
b = "dokyeom"
x = id(a)
y = id(b)
print(x)
print(y)
```

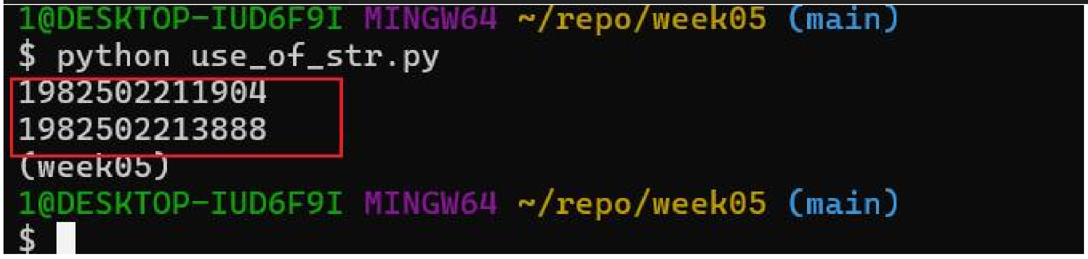


```
$ python use_of_str.py
1383353292464
1383353292464
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

虚拟地址一样的原因：存在缓存

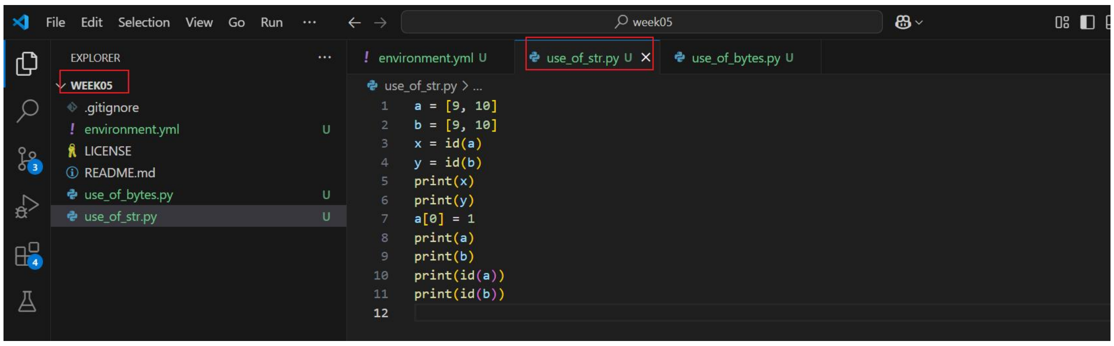


```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
```

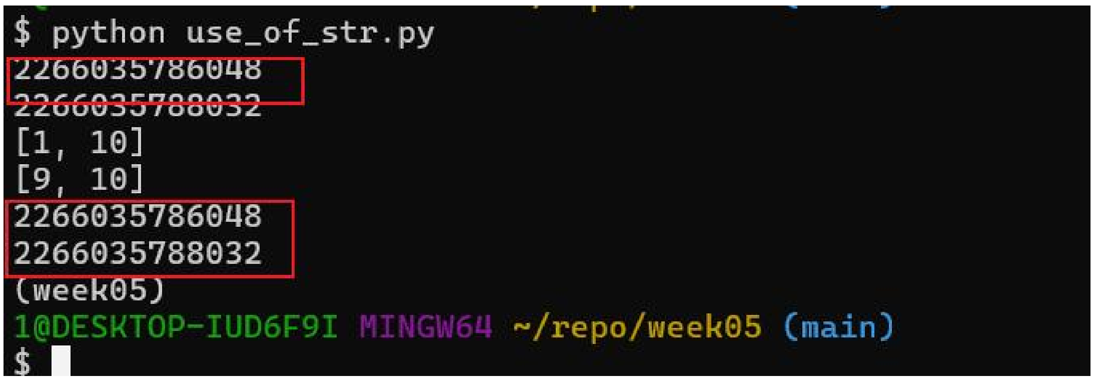


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1982502211904
1982502213888
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

虚拟地址不同

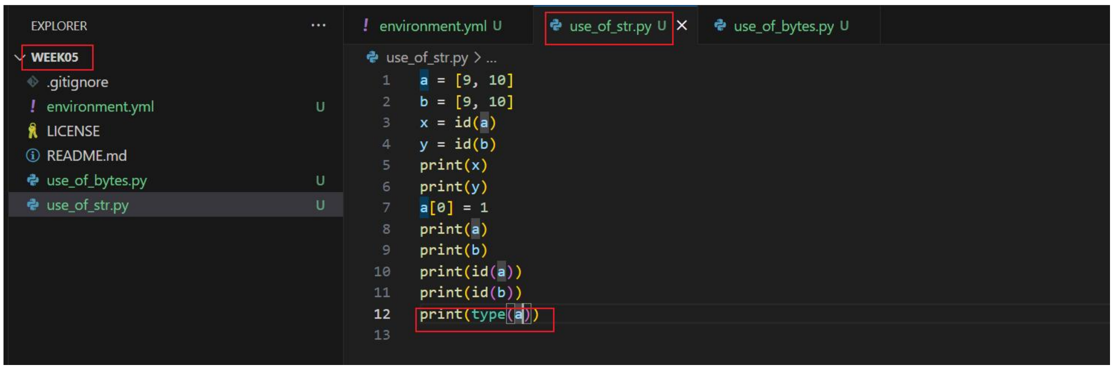


```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
```

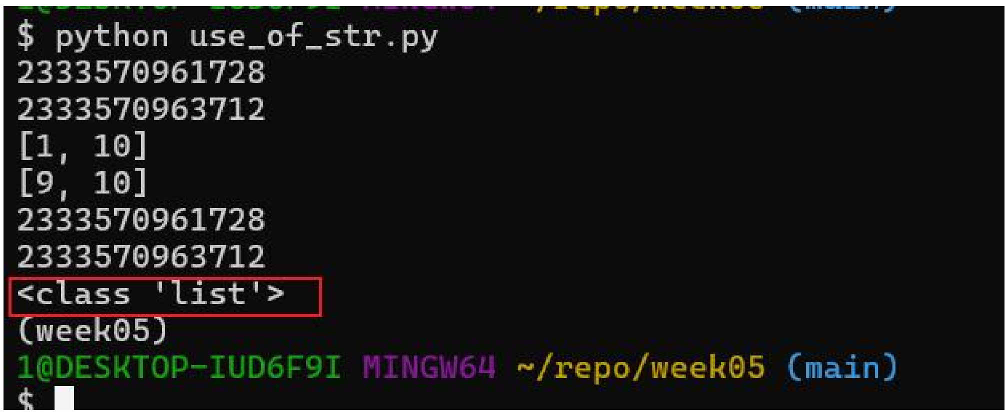


```
$ python use_of_str.py
2266035786048
2266035788032
[1, 10]
[9, 10]
2266035786048
2266035788032
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

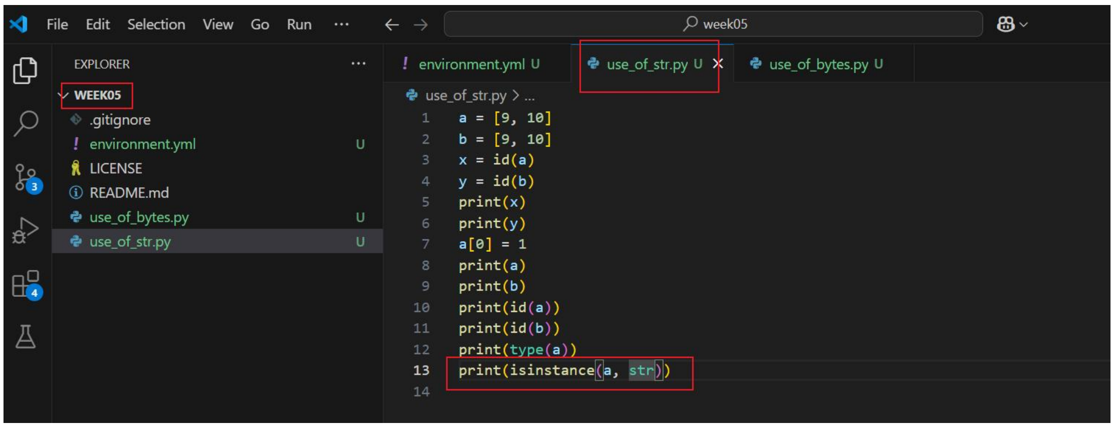
虽做了一些修改，但是虚拟地址未变。



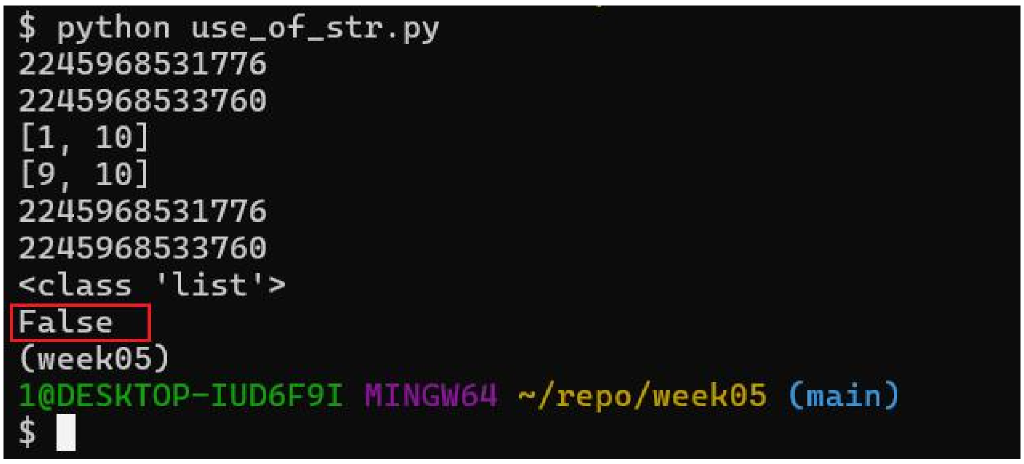
```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
```



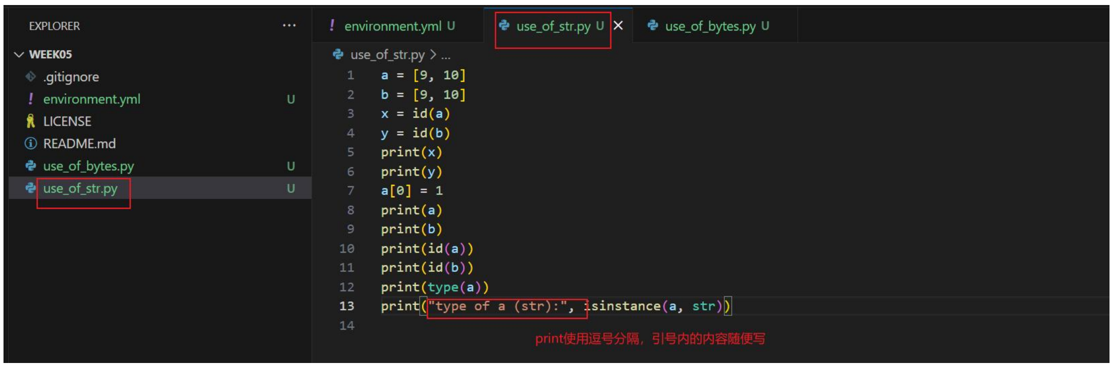
```
$ python use_of_str.py
2333570961728
2333570963712
[1, 10]
[9, 10]
2333570961728
2333570963712
<class 'list'>
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ 
```



```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print(isinstance(a, str))
```

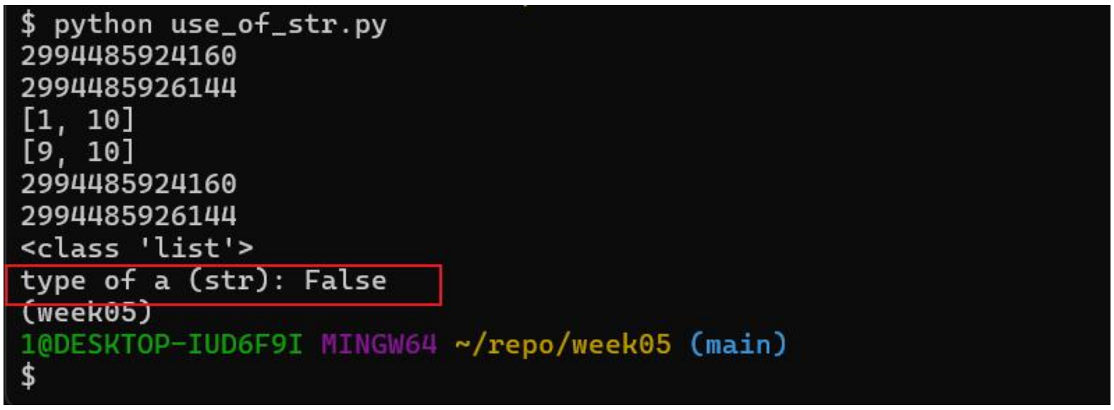


```
$ python use_of_str.py
2245968531776
2245968533760
[1, 10]
[9, 10]
2245968531776
2245968533760
<class 'list'>
False
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ 
```

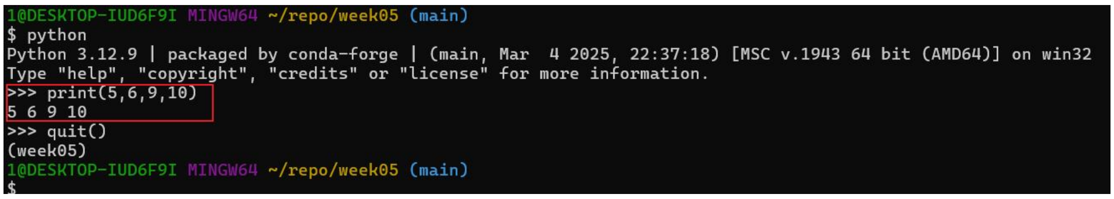


```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print("type of a (str):", isinstance(a, str))
```

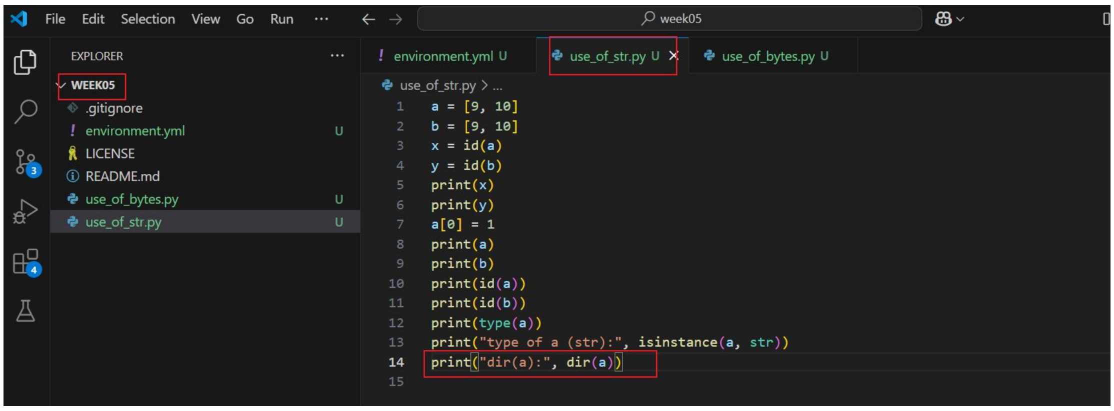
print使用逗号分隔，引号内的内容随便写



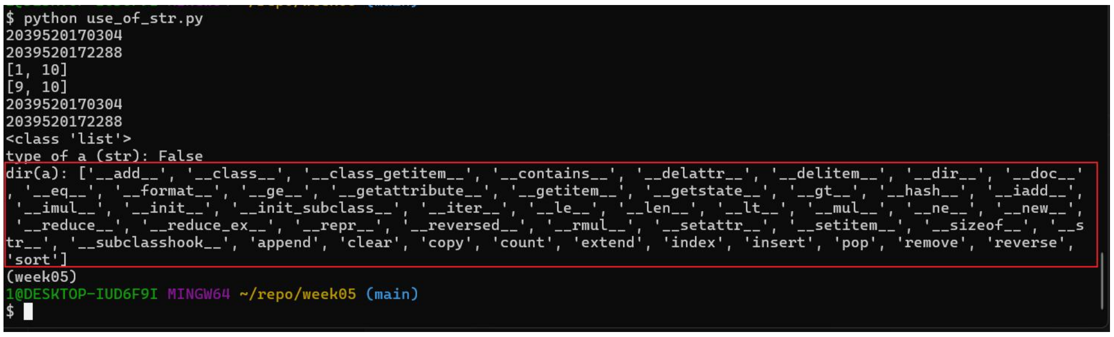
```
$ python use_of_str.py
2994485924160
2994485926144
[1, 10]
[9, 10]
2994485924160
2994485926144
<class 'list'>
type of a (str): False
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```



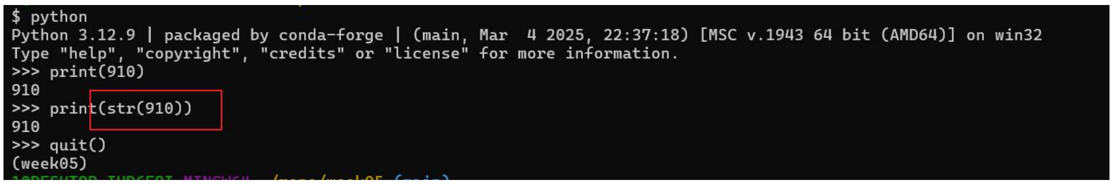
```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(5,6,9,10)
5 6 9 10
>>> quit()
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```



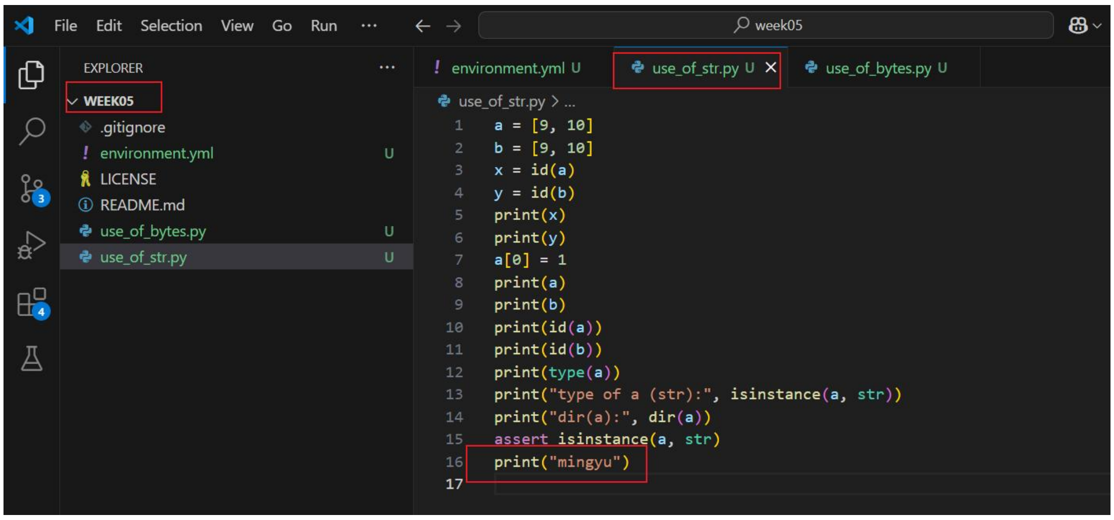
```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print("type of a (str):", isinstance(a, str))
print("dir(a):", dir(a))
```



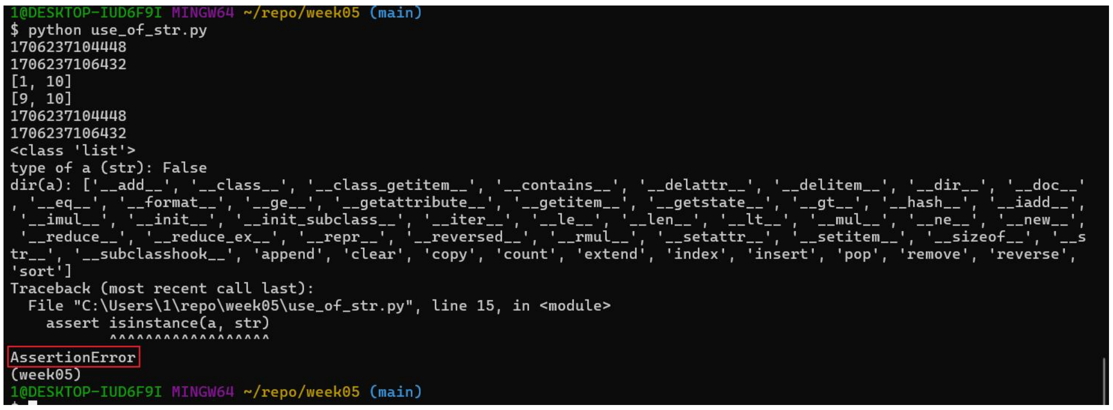
```
$ python use_of_str.py
2039520170304
2039520172288
[1, 10]
[9, 10]
2039520170304
2039520172288
<class 'list'>
type of a (str): False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__'
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__',
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__s
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```



```
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(910)
910
>>> print(str(910))
910
>>> quit()
(week05)
```

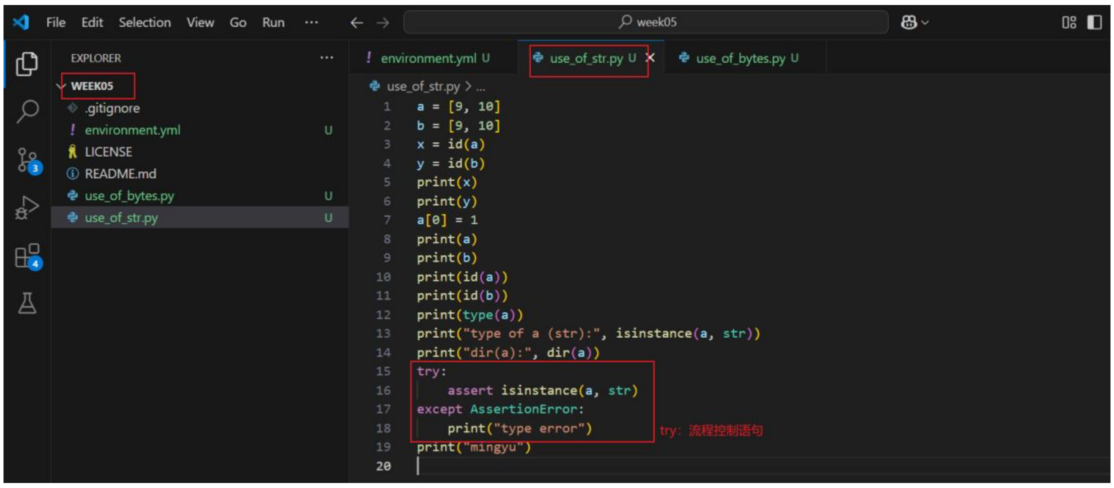


```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print("type of a (str):", isinstance(a, str))
print("dir(a):", dir(a))
assert isinstance(a, str)
print("mingyu")
```



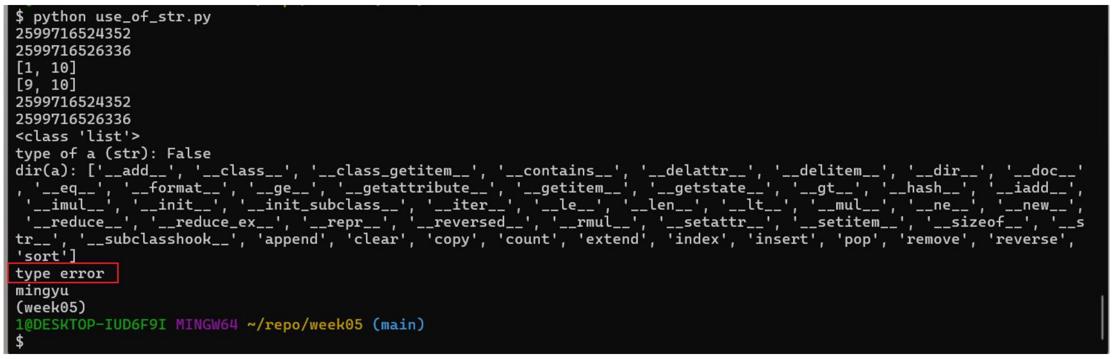
```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1706237104448
1706237106432
[1, 10]
[9, 10]
1706237104448
1706237106432
<class 'list'>
type of a (str): False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__'
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__',
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__s
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
Traceback (most recent call last):
  File "C:\Users\1\repo\week05\use_of_str.py", line 15, in <module>
    assert isinstance(a, str)
           ^^^^^^^^^^^^^^^^^^
AssertionError
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
```

assert 报错后直接退出，不执行后续语句；若不报错，不显示任何结果，直接显示下一条代码的运行结果。



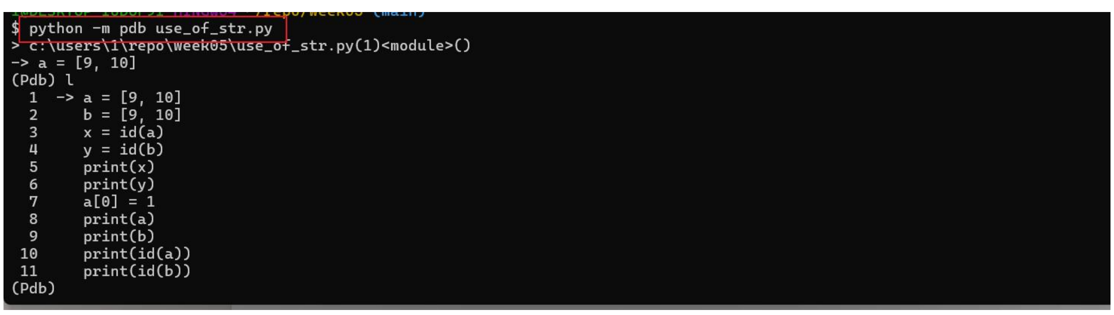
```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print("type of a (str):", isinstance(a, str))
print("dir(a):", dir(a))
try:
    assert isinstance(a, str)
except AssertionError:
    print("type error")
print("mingyu")
```

try：流程控制语句



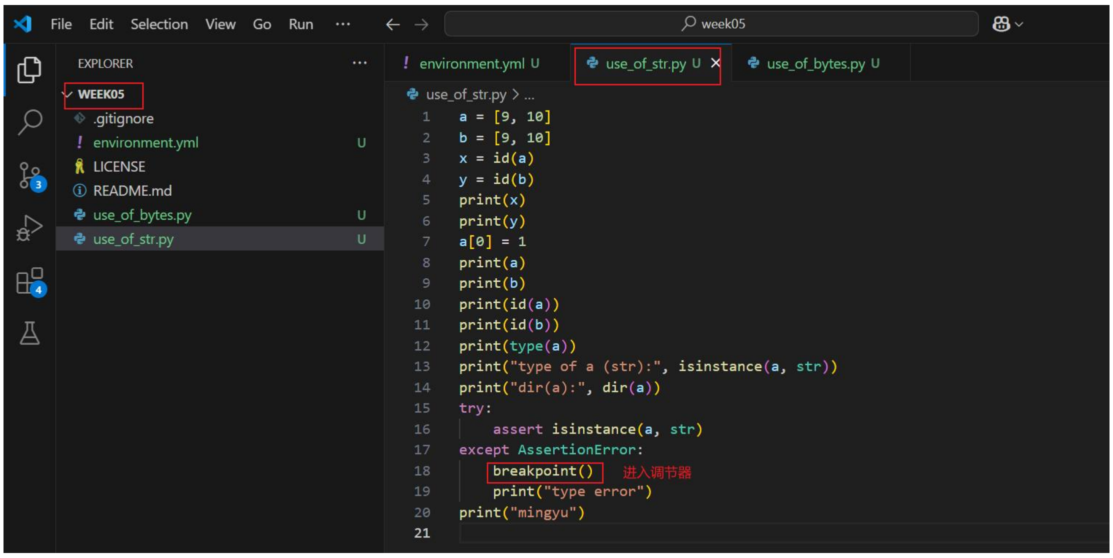
```
$ python use_of_str.py
2599716524352
2599716526336
[1, 10]
[9, 10]
2599716524352
2599716526336
<class 'list'>
type of a (str): False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__'
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__',
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__s
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
type error
mingyu
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

调试

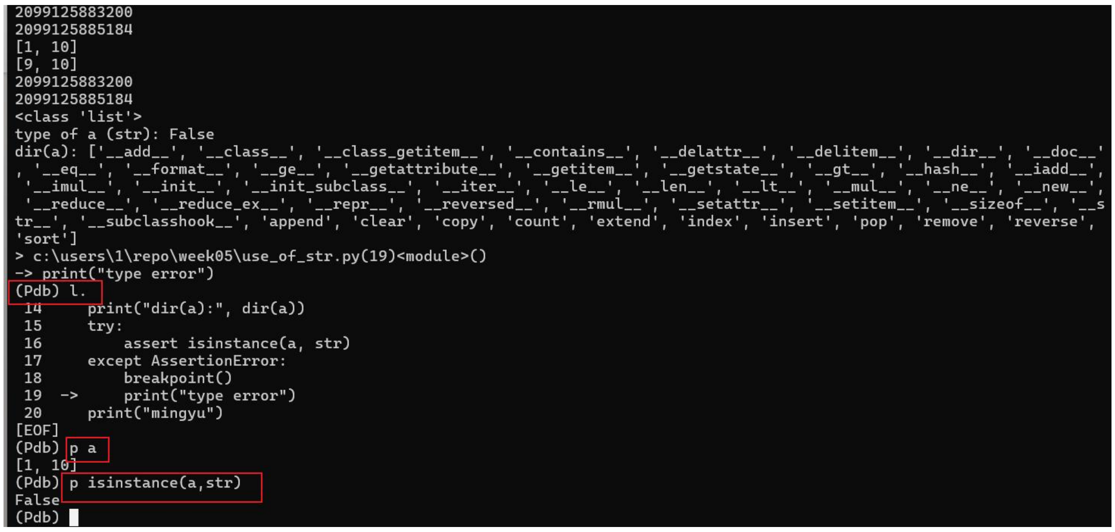


```
$ python -m pdb use_of_str.py
> c:\users\1\repo\week05\use_of_str.py(1)<module>()
-> a = [9, 10]
(Pdb) l
  1  -> a = [9, 10]
  2     b = [9, 10]
  3     x = id(a)
  4     y = id(b)
  5     print(x)
  6     print(y)
  7     a[0] = 1
  8     print(a)
  9     print(b)
 10     print(id(a))
 11     print(id(b))
(Pdb)
```

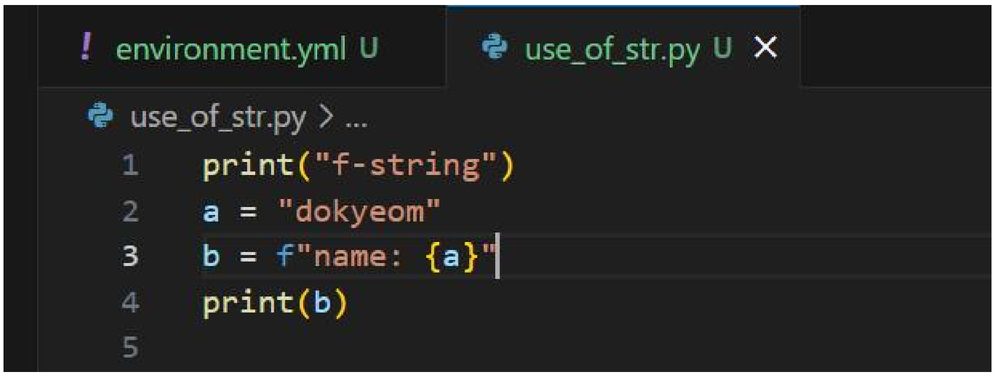
当代码较长时，



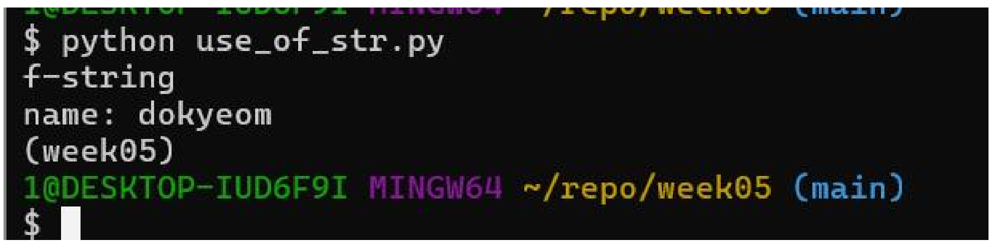
```python
a = [9, 10]
b = [9, 10]
x = id(a)
y = id(b)
print(x)
print(y)
a[0] = 1
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print("type of a (str):", isinstance(a, str))
print("dir(a):", dir(a))
try:
    assert isinstance(a, str)
except AssertionError:
    breakpoint()   进入调节器
    print("type error")
print("mingyu")
```

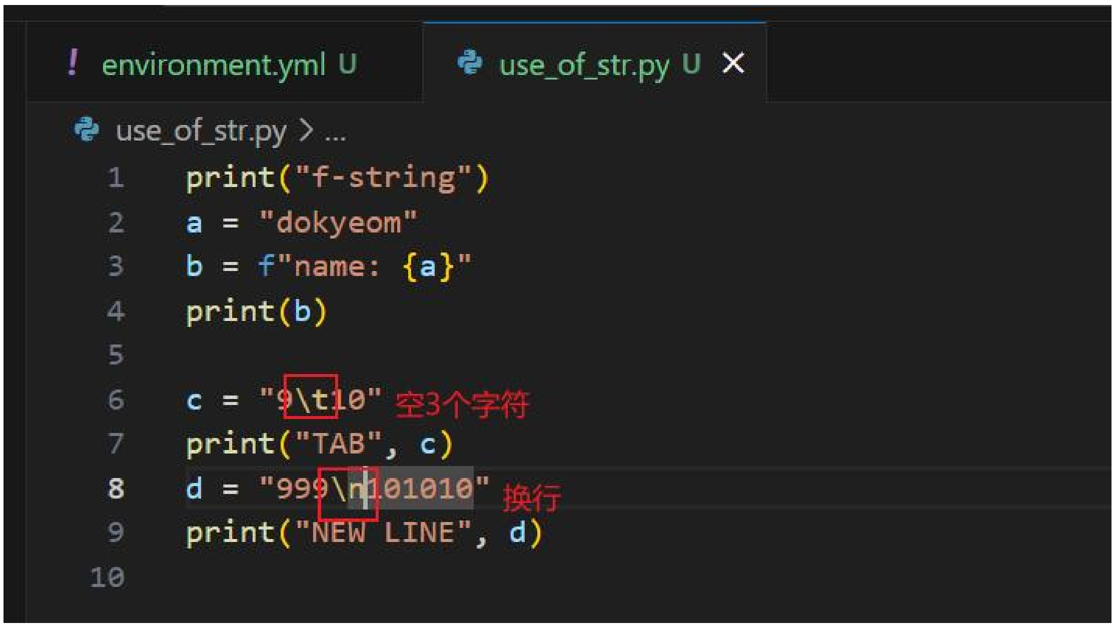

```
2099125883200
2099125885184
[1, 10]
[9, 10]
2099125883200
2099125885184
<class 'list'>
type of a (str): False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__'
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__', '__hash__', '__iadd__',
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__s
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
> c:\users\1\repo\week05\use_of_str.py(19)<module>()
-> print("type error")
(Pdb) l.
 14     print("dir(a):", dir(a))
 15     try:
 16         assert isinstance(a, str)
 17     except AssertionError:
 18         breakpoint()
 19  ->     print("type error")
 20     print("mingyu")
[EOF]
(Pdb) p a
[1, 10]
(Pdb) p isinstance(a,str)
False
(Pdb) 
```

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）

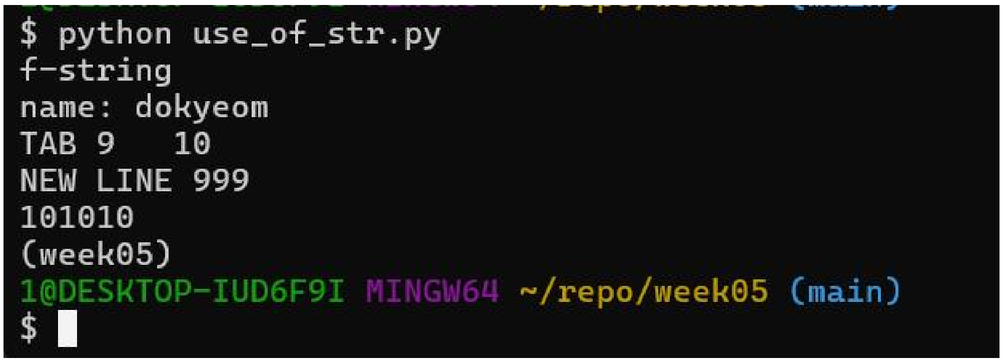


```python
print("f-string")
a = "dokyeom"
b = f"name: {a}"
print(b)
```

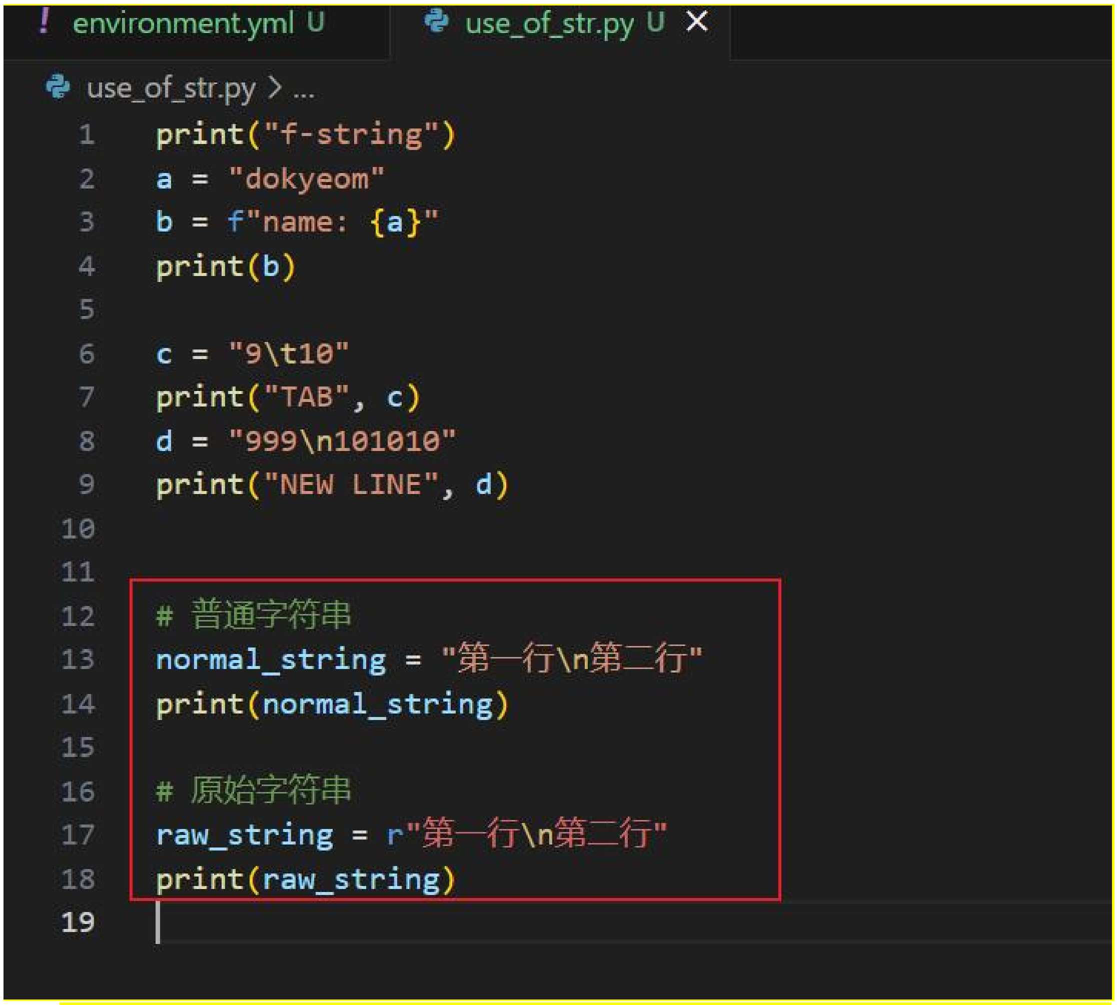

```
$ python use_of_str.py
f-string
name: dokyeom
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ 
```



```python
print("f-string")
a = "dokyeom"
b = f"name: {a}"
print(b)

c = "9\t10" 空3个字符
print("TAB", c)
d = "999\n101010" 换行
print("NEW LINE", d)
```

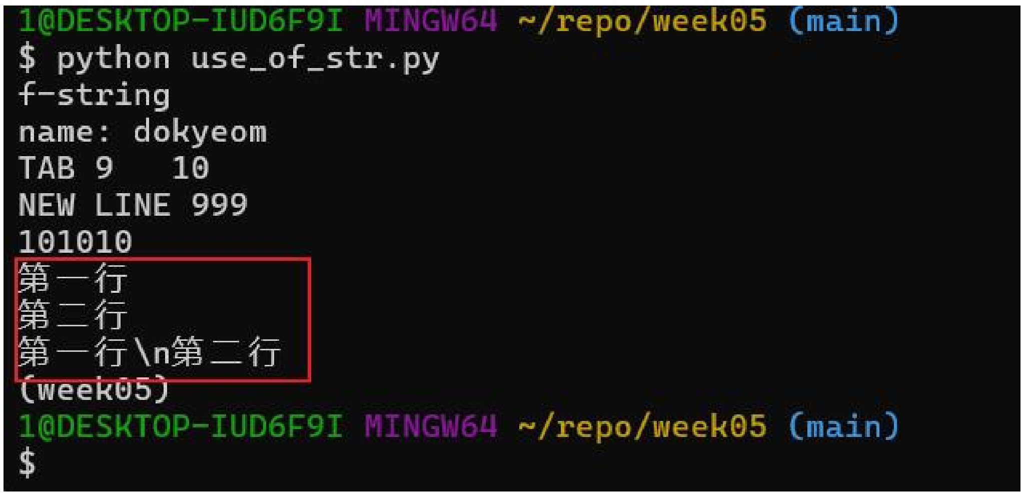


```
$ python use_of_str.py
f-string
name: dokyeom
TAB 9    10
NEW LINE 999
101010
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ 
```

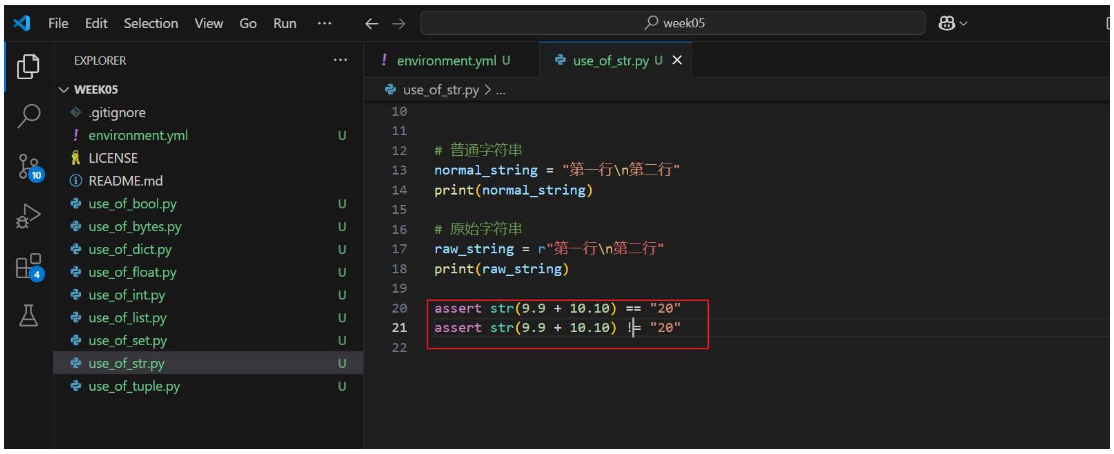


```python
print("f-string")
a = "dokyeom"
b = f"name: {a}"
print(b)

c = "9\t10"
print("TAB", c)
d = "999\n101010"
print("NEW LINE", d)


# 普通字符串
normal_string = "第一行\n第二行"
print(normal_string)

# 原始字符串
raw_string = r"第一行\n第二行"
print(raw_string)
```

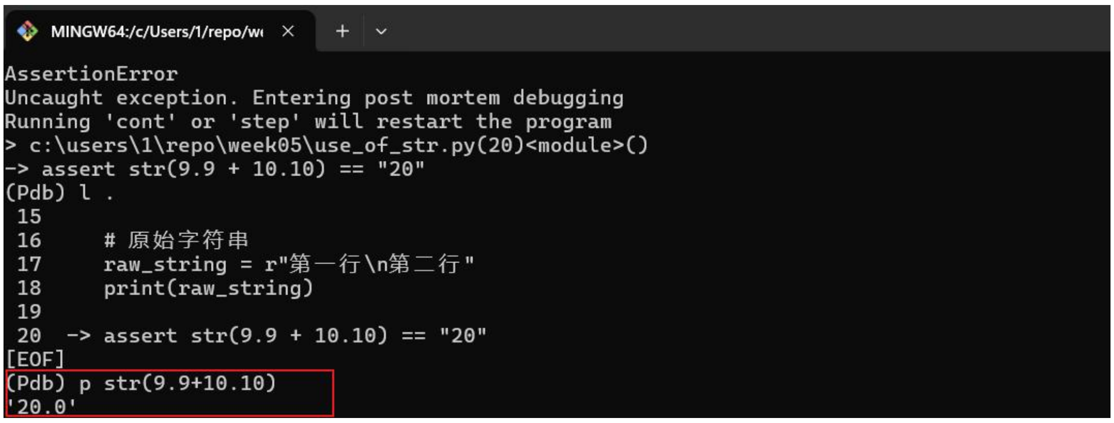
在 Python 里，原始字符串（raw string）是一种特殊字符串，它会把反斜杠 \ 视为普通字符，而非转义字符的起始。



```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
f-string
name: dokyeom
TAB 9	10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

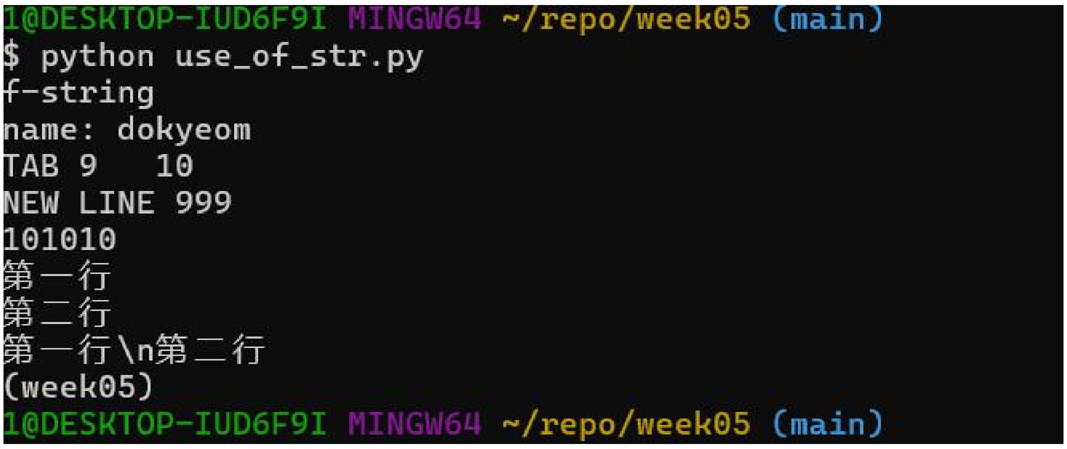


```python


# 普通字符串
normal_string = "第一行\n第二行"
print(normal_string)

# 原始字符串
raw_string = r"第一行\n第二行"
print(raw_string)

assert str(9.9 + 10.10) == "20"
assert str(9.9 + 10.10) != "20"

```

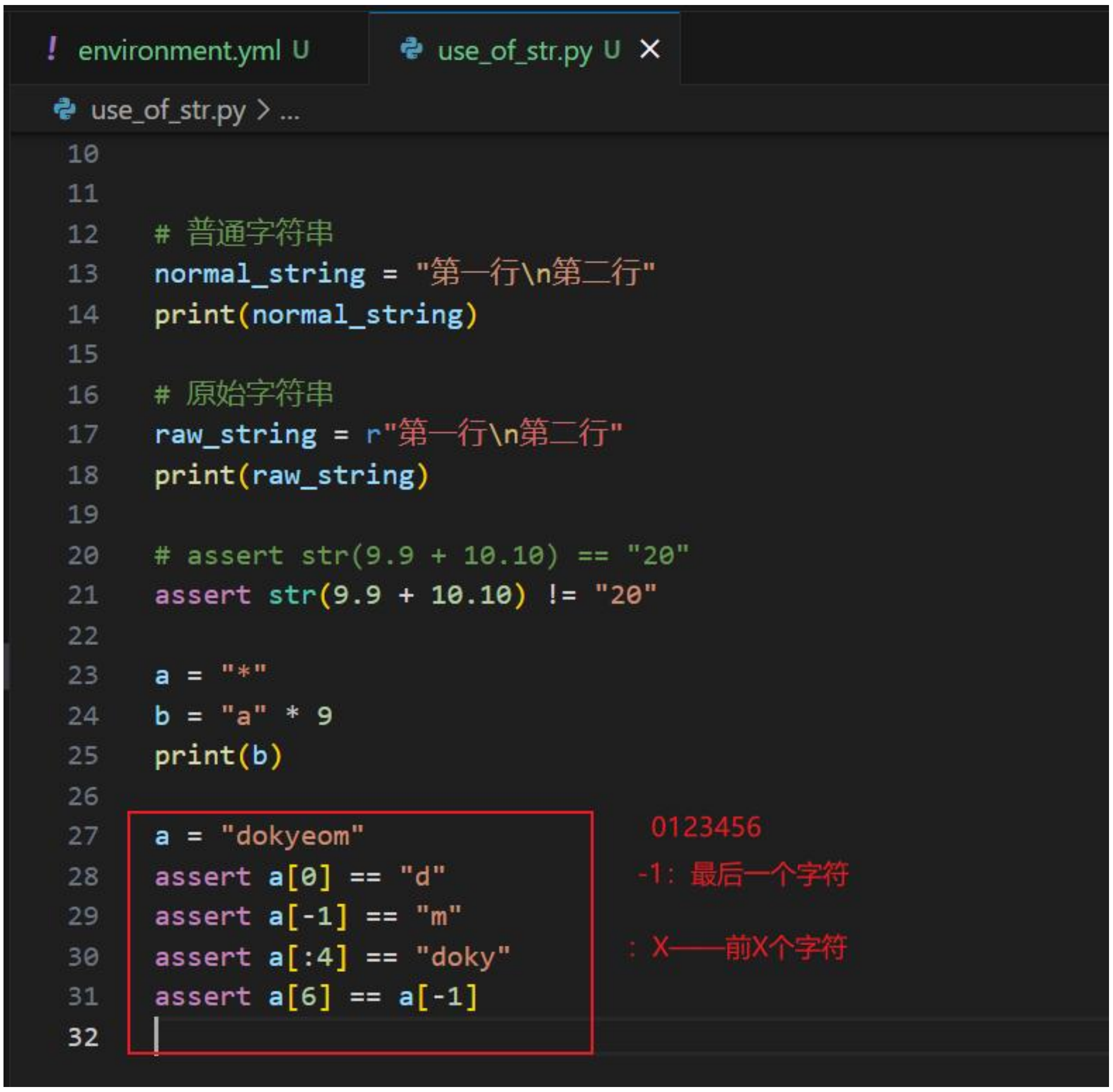


```
AssertionError
Uncaught exception. Entering post mortem debugging
Running 'cont' or 'step' will restart the program
> c:\users\1\repo\week05\use_of_str.py(20)<module>()
-> assert str(9.9 + 10.10) == "20"
(Pdb) l .
 15
 16     # 原始字符串
 17     raw_string = r"第一行\n第二行"
 18     print(raw_string)
 19
 20  -> assert str(9.9 + 10.10) == "20"
[EOF]
(Pdb) p str(9.9+10.10)
'20.0'
```

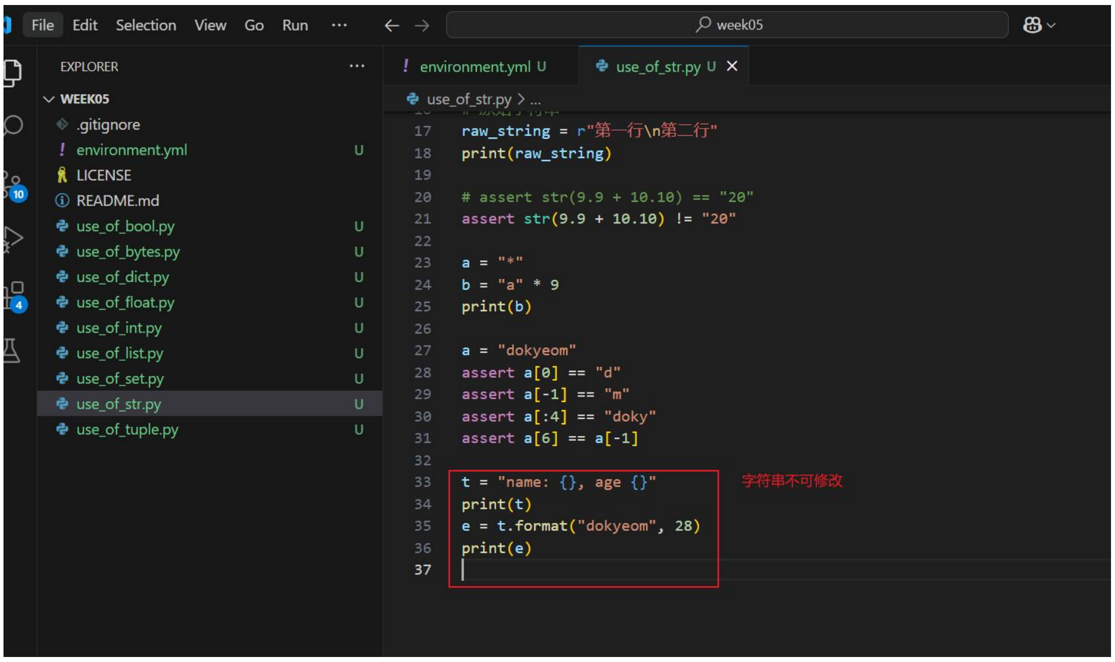
第二条代码未报错，pass



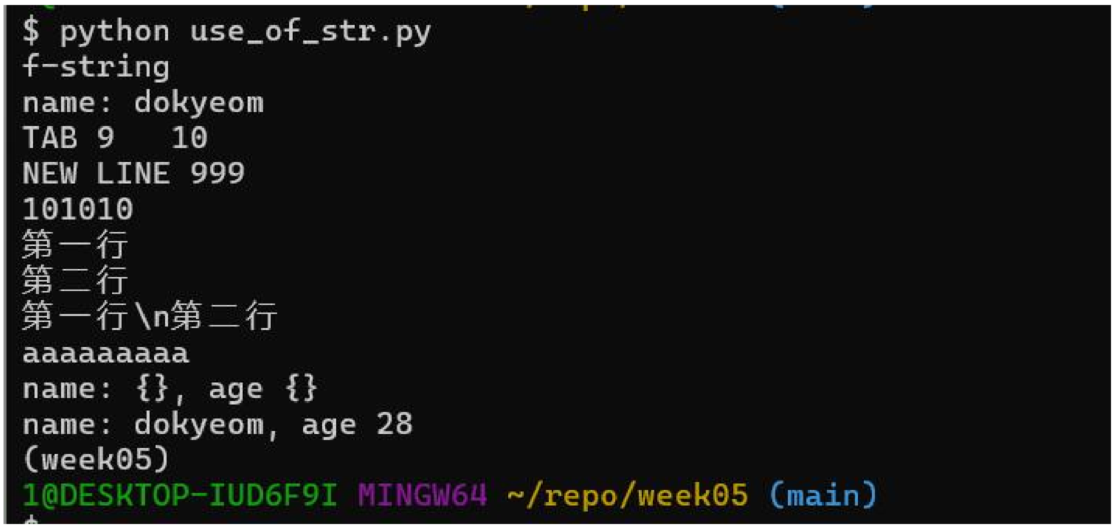
```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
f-string
name: dokyeom
TAB 9    10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
```

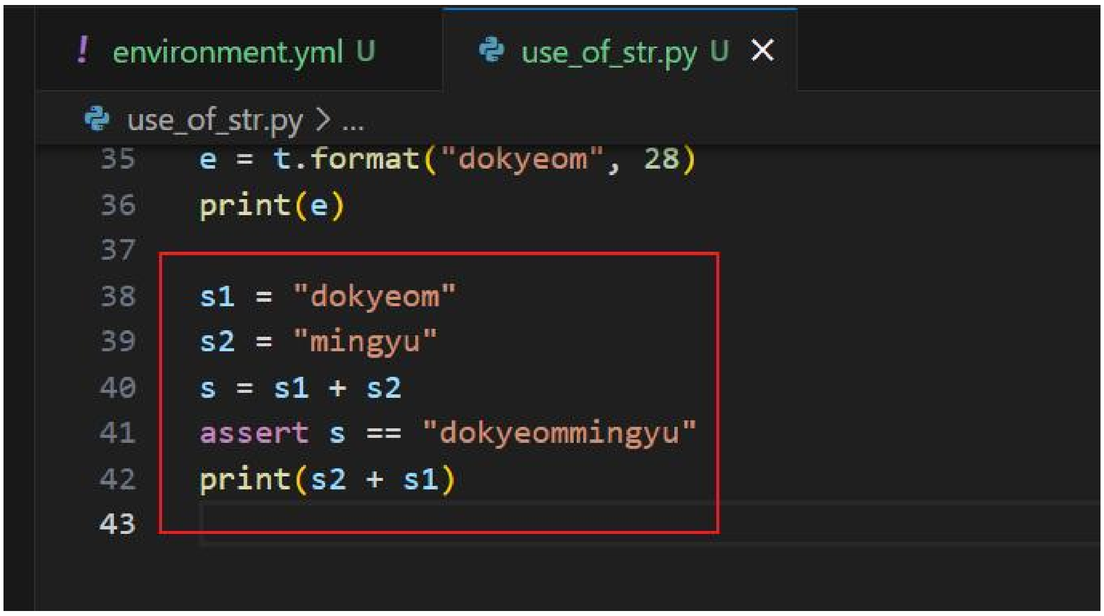


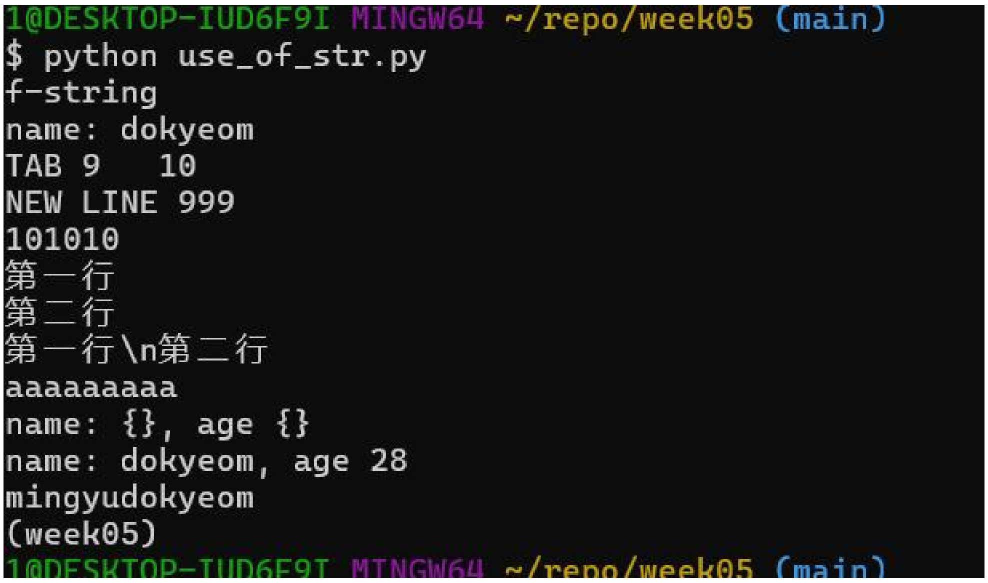
```python
# 普通字符串
normal_string = "第一行\n第二行"
print(normal_string)

# 原始字符串
raw_string = r"第一行\n第二行"
print(raw_string)

# assert str(9.9 + 10.10) == "20"
assert str(9.9 + 10.10) != "20"

a = "*"
b = "a" * 9
print(b)

a = "dokyeom"
assert a[0] == "d"
assert a[-1] == "m"
assert a[:4] == "doky"
assert a[6] == a[-1]
```

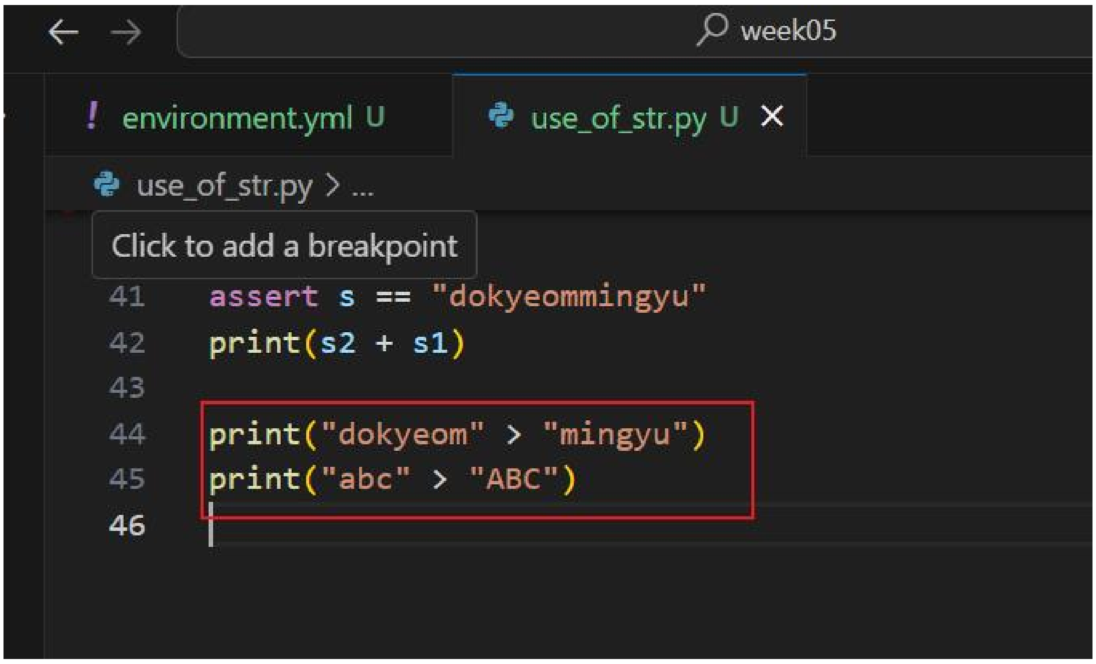


```python
raw_string = r"第一行\n第二行"
print(raw_string)

# assert str(9.9 + 10.10) == "20"
assert str(9.9 + 10.10) != "20"

a = "*"
b = "a" * 9
print(b)

a = "dokyeom"
assert a[0] == "d"
assert a[-1] == "m"
assert a[:4] == "doky"
assert a[6] == a[-1]

t = "name: {}, age {}"
print(t)
e = t.format("dokyeom", 28)
print(e)
```

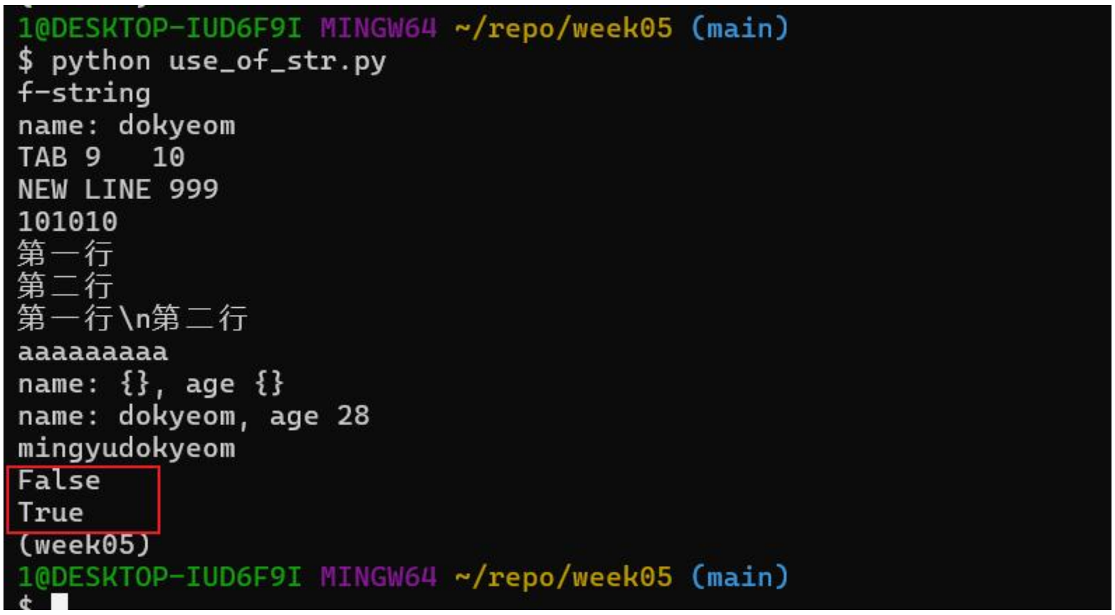

```
$ python use_of_str.py
f-string
name: dokyeom
TAB 9	10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
aaaaaaaaa
name: {}, age {}
name: dokyeom, age 28
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
```
5.对于每一个上述要求掌握的对象类型 （将来遇到新的对象类型也应该如此），我们也要尝试验证其以下几个方面的属性 （attributes）



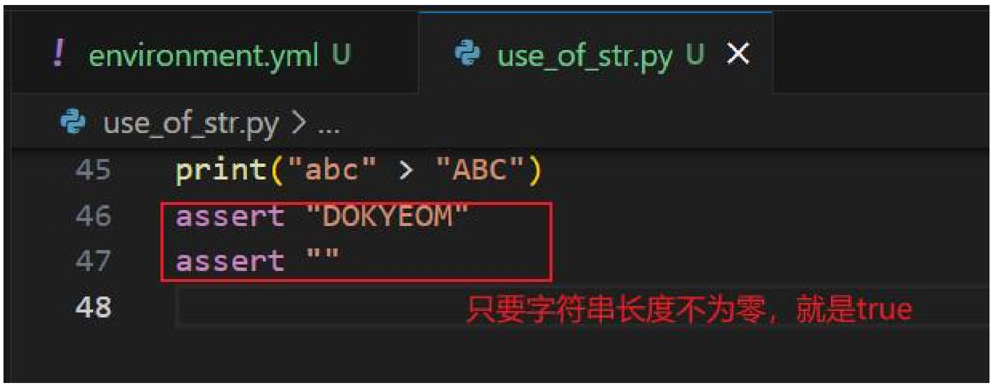
```python
e = t.format("dokyeom", 28)
print(e)

s1 = "dokyeom"
s2 = "mingyu"
s = s1 + s2
assert s == "dokyeommingyu"
print(s2 + s1)

```

```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
f-string
name: dokyeom
TAB 9	10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
aaaaaaaaa
name: {}, age {}
name: dokyeom, age 28
mingyudokyeom
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
```



```python
assert s == "dokyeommingyu"
print(s2 + s1)

print("dokyeom" > "mingyu")
print("abc" > "ABC")

```

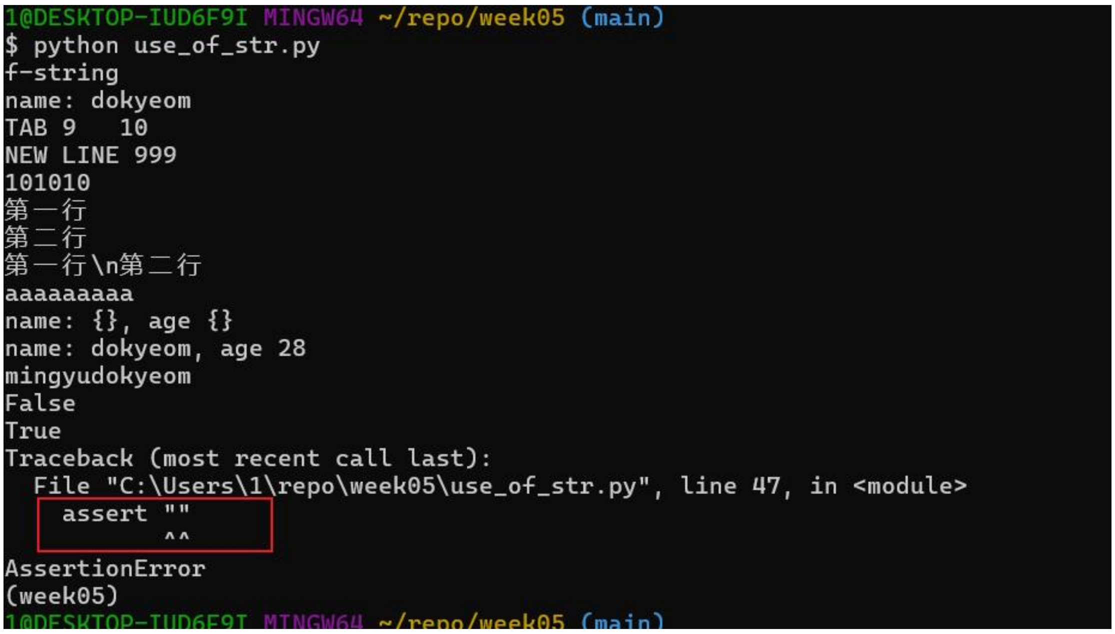


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
f-string
name: dokyeom
TAB 9	10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
aaaaaaaaa
name: {}, age {}
name: dokyeom, age 28
mingyudokyeom
False
True
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ 
```

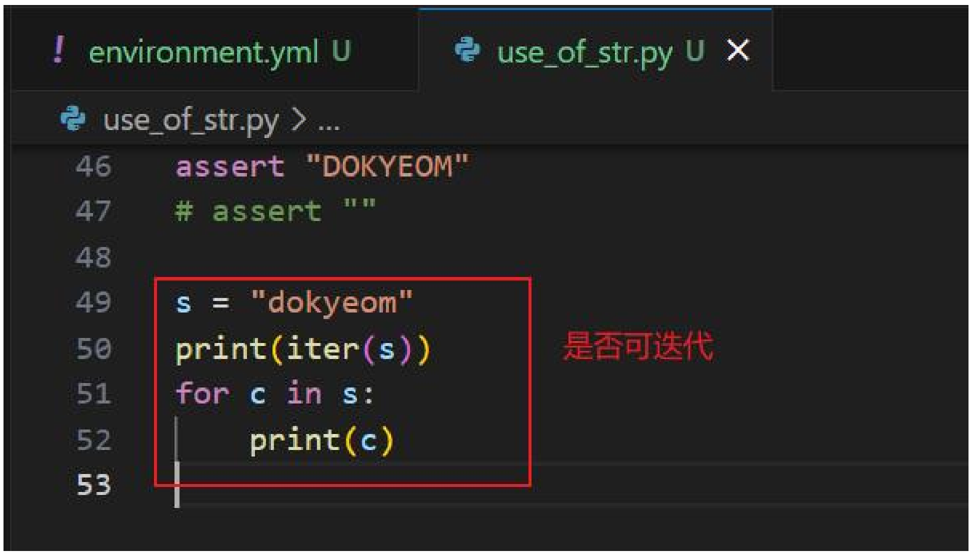


```python
print("abc" > "ABC")
assert "DOKYEOM"
assert ""

```

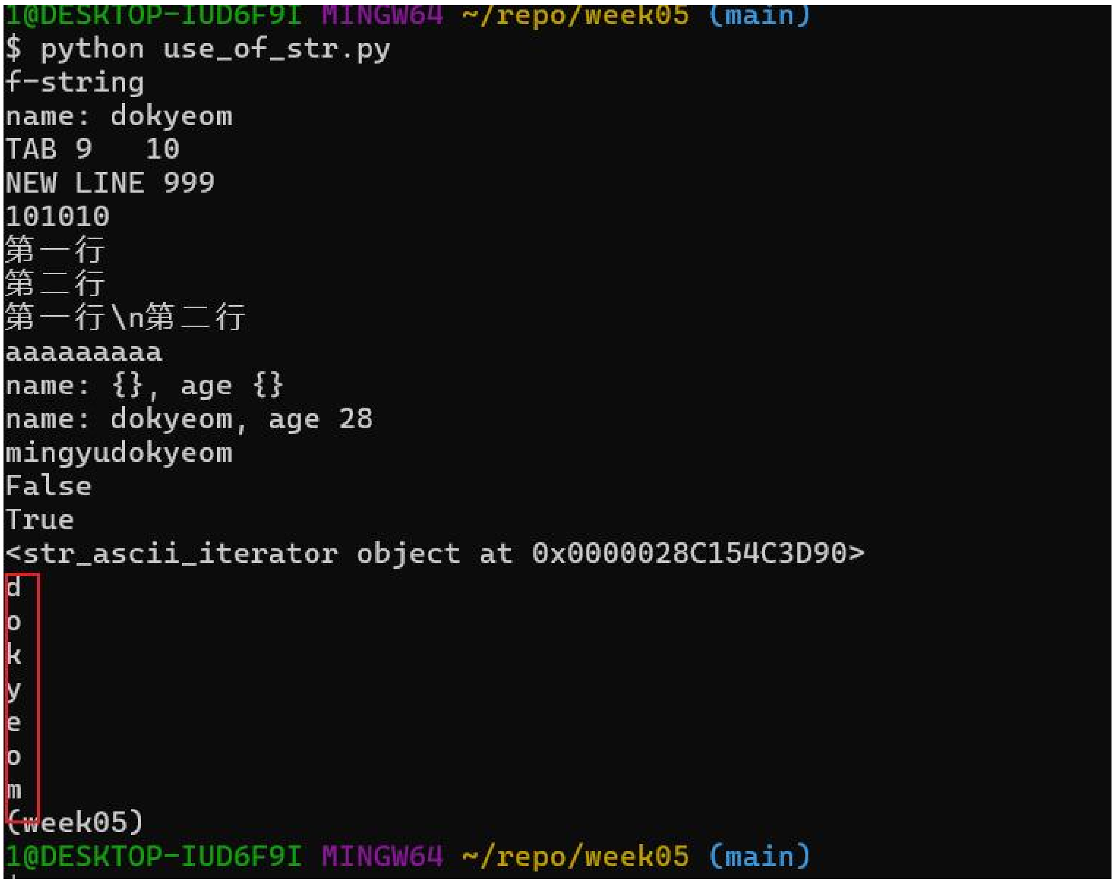
只要字符串长度不为零，就是true



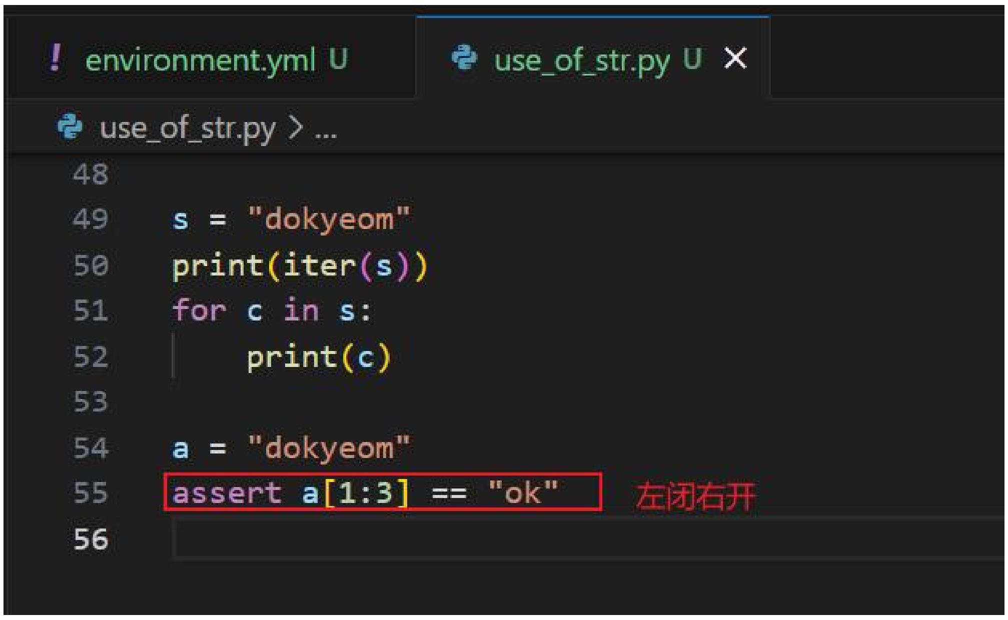
```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
f-string
name: dokyeom
TAB 9	10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
aaaaaaaaa
name: {}, age {}
name: dokyeom, age 28
mingyudokyeom
False
True
Traceback (most recent call last):
  File "C:\Users\1\repo\week05\use_of_str.py", line 47, in <module>
    assert ""
           ^^
AssertionError
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
```



```
assert "DOKYEOM"
# assert ""

s = "dokyeom"
print(iter(s))
for c in s:
    print(c)

```

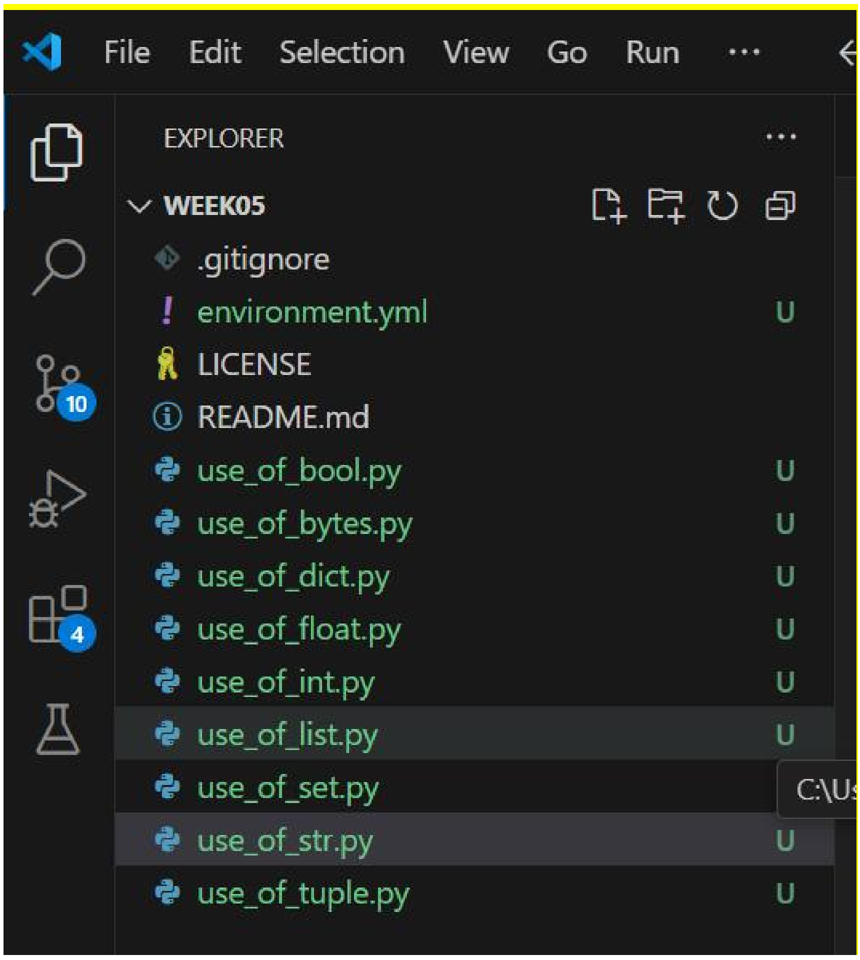


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
f-string
name: dokyeom
TAB 9	10
NEW LINE 999
101010
第一行
第二行
第一行\n第二行
aaaaaaaaa
name: {}, age {}
name: dokyeom, age 28
mingyudokyeom
False
True
<str_ascii_iterator object at 0x0000028C154C3D90>
d
o
k
y
e
o
m
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
```



```python
s = "dokyeom"
print(iter(s))
for c in s:
    print(c)

a = "dokyeom"
assert a[1:3] == "ok"
```

左闭右开

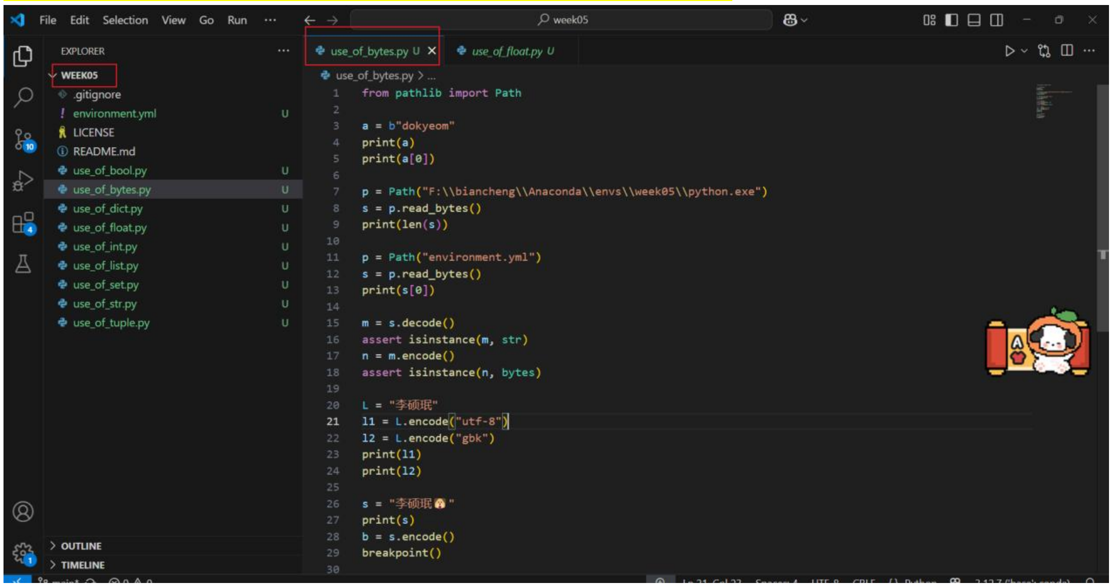


```
EXPLORER
WEEK05
.gitignore
environment.yml        U
LICENSE
README.md
use_of_bool.py         U
use_of_bytes.py        U
use_of_dict.py         U
use_of_float.py        U
use_of_int.py          U
use_of_list.py         U
use_of_set.py
use_of_str.py          U
use_of_tuple.py        U
```

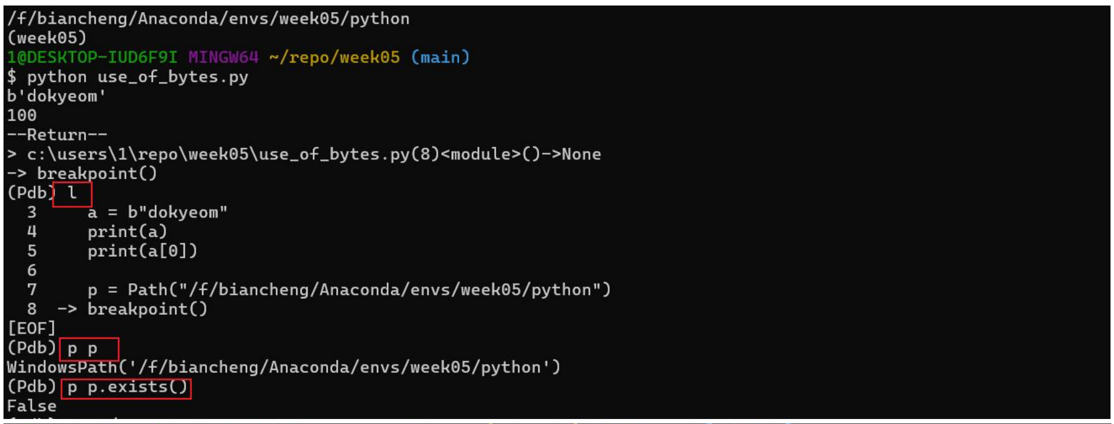
Bytes 字节串

**：代表次幂

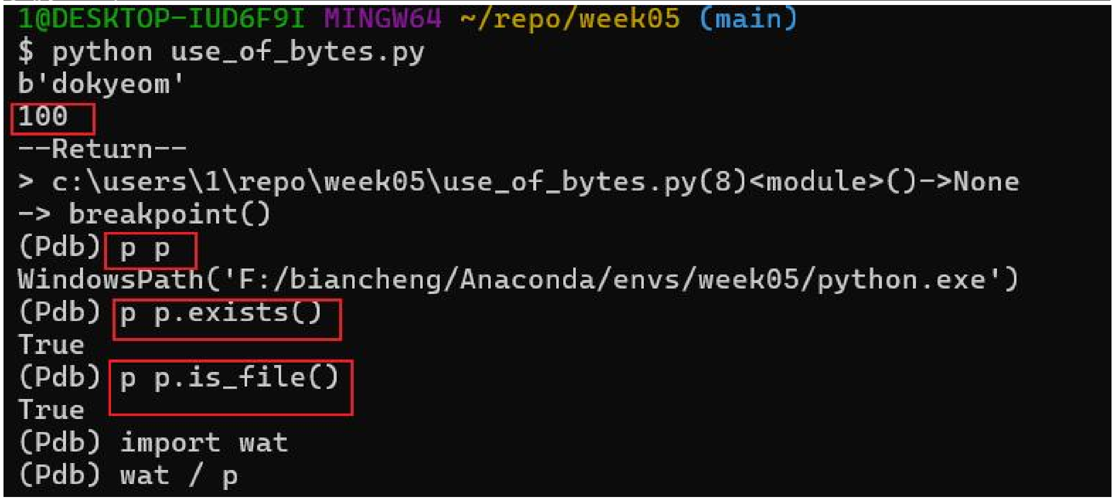
字符串经过编码得到字节串，字节串经过解码变成字符串。



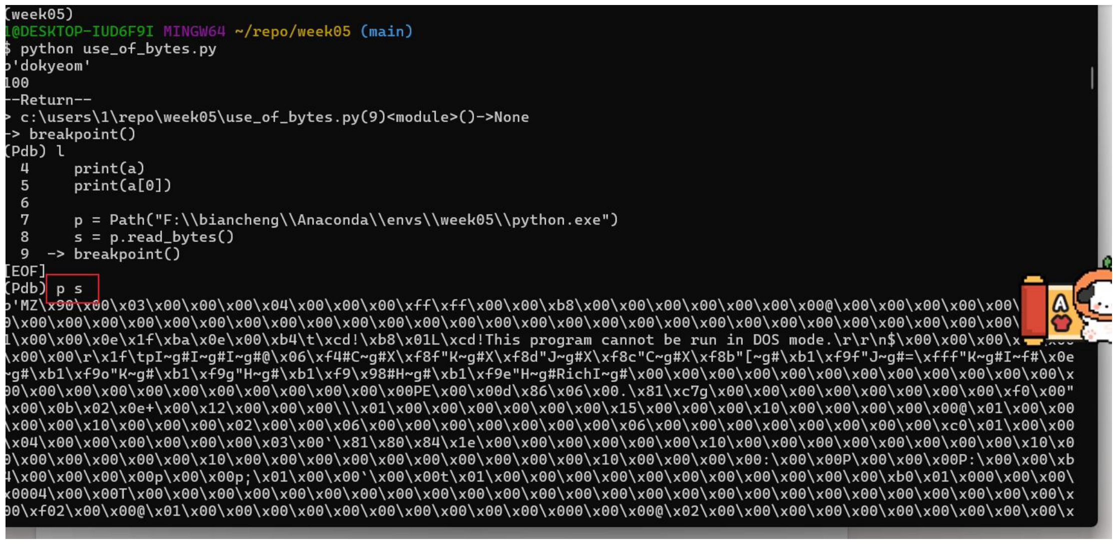
```python
from pathlib import Path

a = b"dokyeom"
print(a)
print(a[0])

p = Path("F:\\biancheng\\Anaconda\\envs\\week05\\python.exe")
s = p.read_bytes()
print(len(s))

p = Path("environment.yml")
s = p.read_bytes()
print(s[0])

m = s.decode()
assert isinstance(m, str)
n = m.encode()
assert isinstance(n, bytes)

L = "李硕珉"
l1 = L.encode("utf-8")
l2 = L.encode("gbk")
print(l1)
print(l2)

s = "李硕珉🙈"
print(s)
b = s.encode()
breakpoint()
```



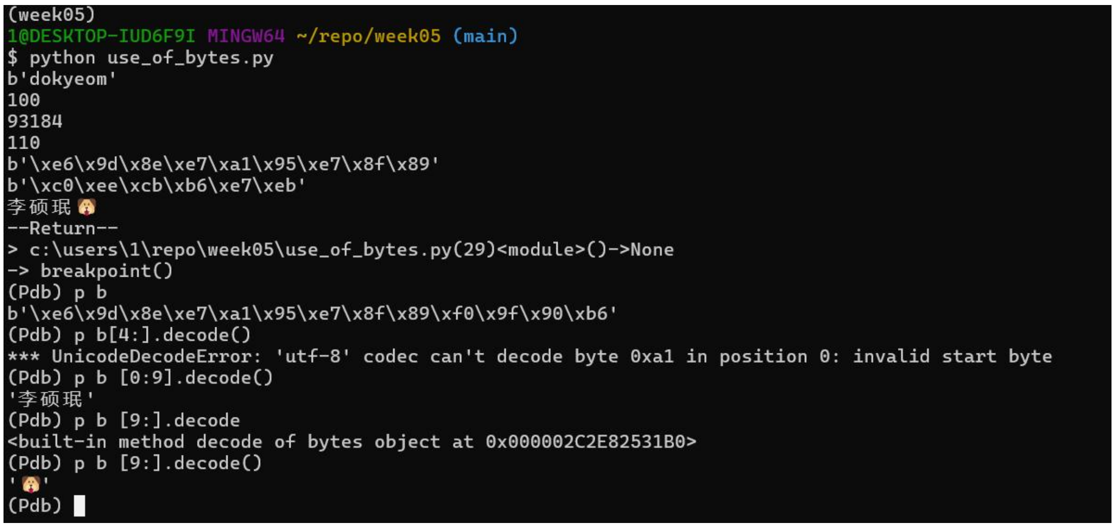
```
/f/biancheng/Anaconda/envs/week05/python
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'dokyeom'
100
--Return--
> c:\users\1\repo\week05\use_of_bytes.py(8)<module>()->None
-> breakpoint()
(Pdb) l
  3       a = b"dokyeom"
  4       print(a)
  5       print(a[0])
  6
  7       p = Path("/f/biancheng/Anaconda/envs/week05/python")
  8   ->  breakpoint()
[EOF]
(Pdb) p p
WindowsPath('/f/biancheng/Anaconda/envs/week05/python')
(Pdb) p p.exists()
False
```



```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'dokyeom'
100
--Return--
> c:\users\1\repo\week05\use_of_bytes.py(8)<module>()->None
-> breakpoint()
(Pdb) p p
WindowsPath('F:/biancheng/Anaconda/envs/week05/python.exe')
(Pdb) p p.exists()
True
(Pdb) p p.is_file()
True
(Pdb) import wat
(Pdb) wat / p
```



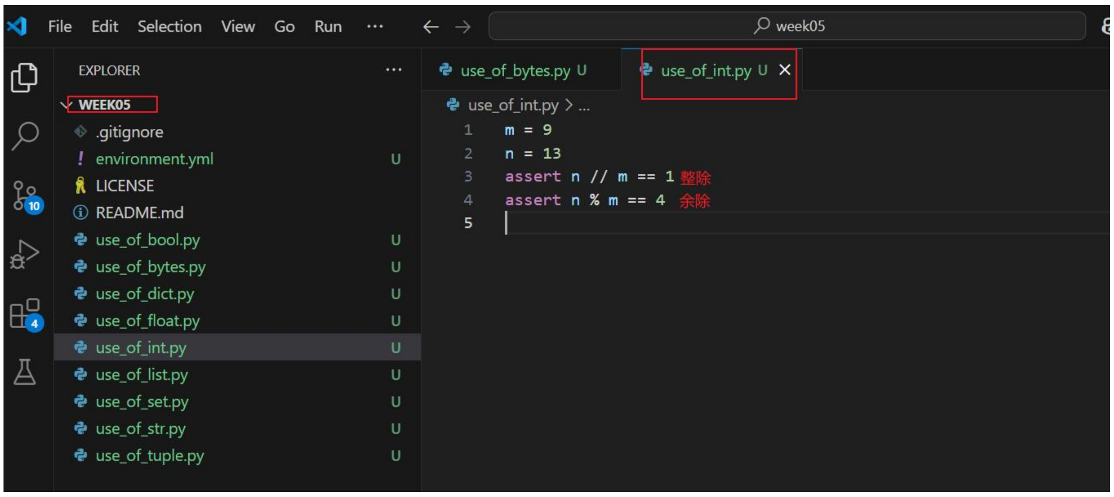
```
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'dokyeom'
100
--Return--
> c:\users\1\repo\week05\use_of_bytes.py(9)<module>()->None
-> breakpoint()
(Pdb) l
  4       print(a)
  5       print(a[0])
  6
  7       p = Path("F:\\biancheng\\Anaconda\\envs\\week05\\python.exe")
  8       s = p.read_bytes()
  9   ->  breakpoint()
[EOF]
(Pdb) p s
b'MZ\x90\x00\x03\x00\x00\x00\x04\x00\x00\x00\xff\xff\x00\x00\xb8\x00\x00\x00\x00\x00\x00\x00@\x00\x00\x00\x00\x00\
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
1\x00\x00\x0e\x1f\xba\x0e\x00\xb4\t\xcd!\xb8\x01L\xcd!This program cannot be run in DOS mode.\r\r\n$\x00\x00\x00\x
\x00\x00\r\x1f\tpI~g#I~g#I~g#@\x06\xf4#C~g#X\xf8f"K~g#X\xf8d"J~g#X\xf8c"C~g#X\xf8b"[~g#\xb1\xf9f"J~g#=\xfff"K~g#I~f#\x0e
~g#\xb1\xf9o"K~g#\xb1\xf9g"H~g#\xb1\xf9\x98#H~g#\xb1\xf9e"H~g#RichI~g#\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00PE\x00\x00d\x86\x06\x00.\x81\xc7g\x00\x00\x00\x00\x00\x00\x00\x00\xf0\x00"
\x00\x0b\x02\x0e+\x00\x12\x00\x00\x00\\\x01\x00\x00\x00\x00\x00\x00\x00\x15\x00\x00\x00\x10\x00\x00\x00\x00\x00@\x01\x00\x00
\x00\x00\x10\x00\x00\x00\x02\x00\x00\x06\x00\x00\x00\x00\x00\x00\x00\x06\x00\x00\x00\x00\x00\x00\x00\x00\xc0\x01\x00\x00
\x04\x00\x00\x00\x00\x00\x00\x03\x00`\x81\x80\x84\x1e\x00\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x00\x10\x0
0\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00:\x00\x00P\x00\x00\x00P:\x00\x00\xb
4\x00\x00\x00\x00p\x00\x00p;\x01\x00\x00`\x00\x00t\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xb0\x01\x000\x00\x00\
x0004\x00\x00T\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
00\xf02\x00\x00@\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x000\x00\x00@\x02\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
```



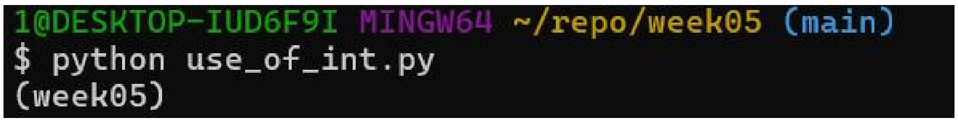
```
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'dokyeom'
100
93184
110
b'\xe6\x9d\x8e\xe7\xa1\x95\xe7\x8f\x89'
b'\xc0\xee\xcb\xb6\xe7\xeb'
李硕珉🙈
--Return--
> c:\users\1\repo\week05\use_of_bytes.py(29)<module>()->None
-> breakpoint()
(Pdb) p b
b'\xe6\x9d\x8e\xe7\xa1\x95\xe7\x8f\x89\xf0\x9f\x90\xb6'
(Pdb) p b[4:].decode()
*** UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa1 in position 0: invalid start byte
(Pdb) p b [0:9].decode()
'李硕珉'
(Pdb) p b [9:].decode
<built-in method decode of bytes object at 0x000002C2E82531B0>
(Pdb) p b [9:].decode()
'🙈'
(Pdb)
```

在 UTF-8 编码中，通常一个汉字占用 3 个字节

Int 整数

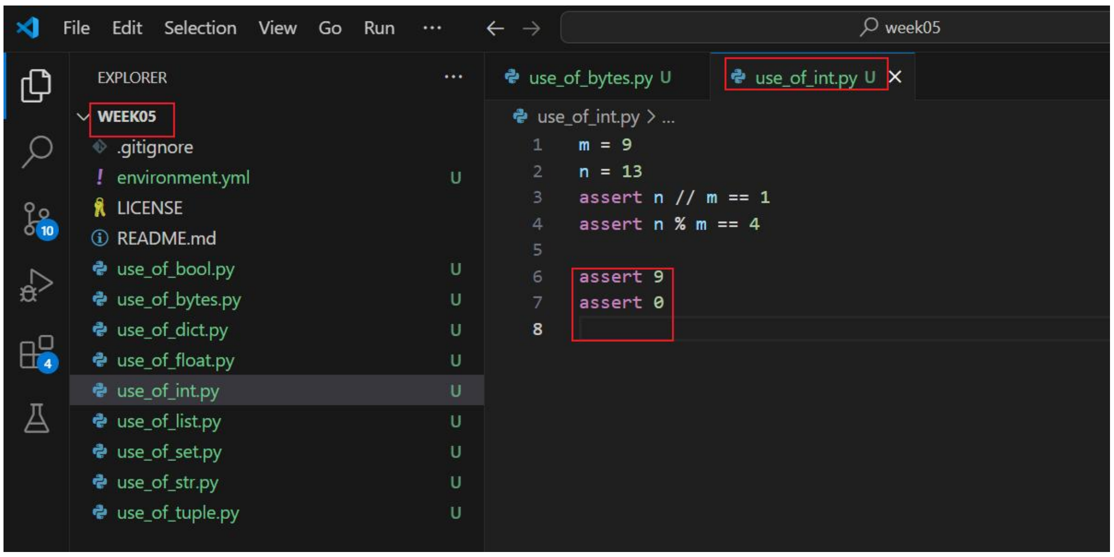


```
m = 9
n = 13
assert n // m == 1 整除
assert n % m == 4  余除
```

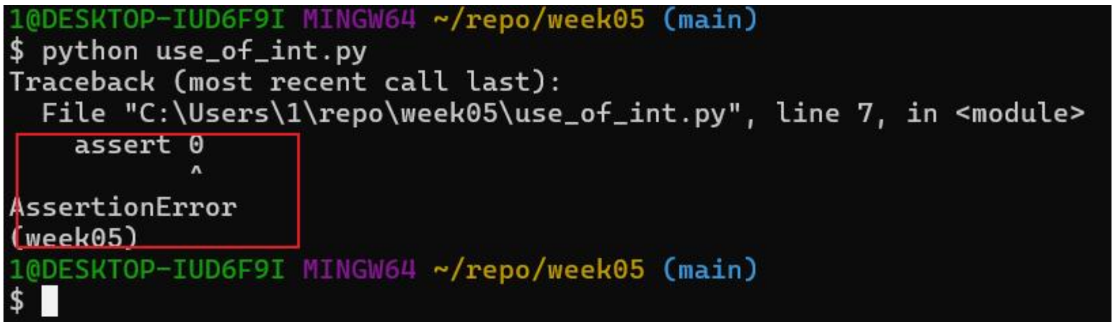


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_int.py
(week05)
```

未报错，pass。

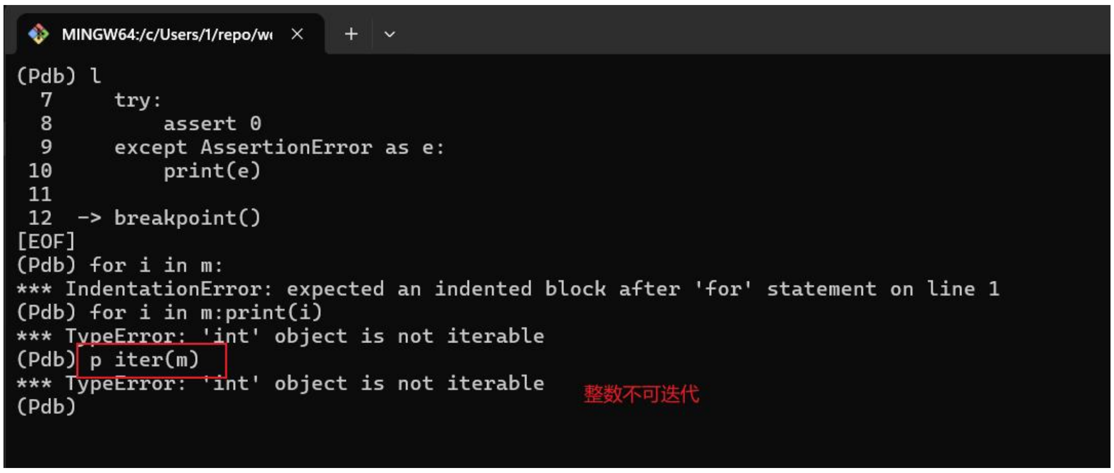


```python
m = 9
n = 13
assert n // m == 1
assert n % m == 4

assert 9
assert 0

```



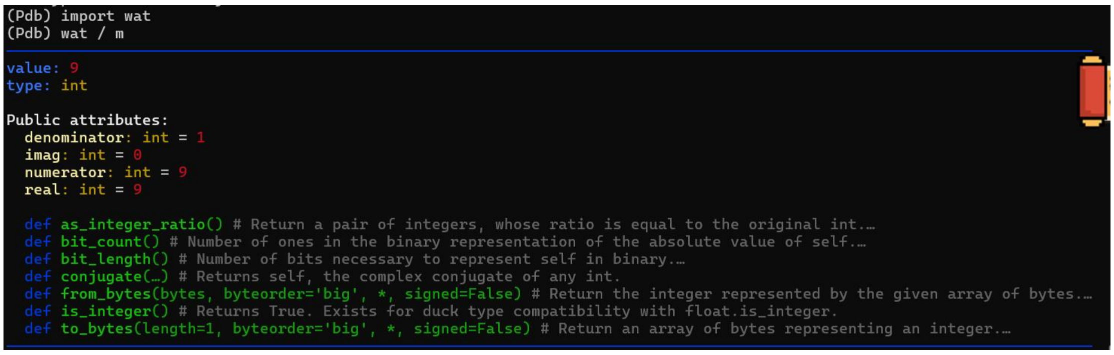
```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_int.py
Traceback (most recent call last):
  File "C:\Users\1\repo\week05\use_of_int.py", line 7, in <module>
    assert 0
           ^
AssertionError
(week05)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ 
```

0 报错

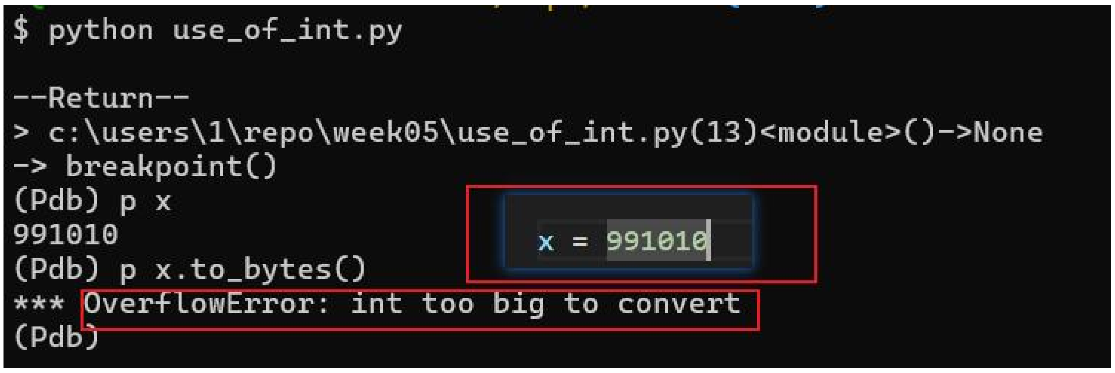


```
(Pdb) l
  7     try:
  8         assert 0
  9     except AssertionError as e:
 10         print(e)
 11
 12  -> breakpoint()
[EOF]
(Pdb) for i in m:
*** IndentationError: expected an indented block after 'for' statement on line 1
(Pdb) for i in m:print(i)
*** TypeError: 'int' object is not iterable
(Pdb) p iter(m)
*** TypeError: 'int' object is not iterable
(Pdb)
```

整数不支持提取以及返回长度。



```
(Pdb) import wat
(Pdb) wat / m

value: 9
type: int

Public attributes:
  denominator: int = 1
  imag: int = 0
  numerator: int = 9
  real: int = 9

  def as_integer_ratio() # Return a pair of integers, whose ratio is equal to the original int.…
  def bit_count() # Number of ones in the binary representation of the absolute value of self.…
  def bit_length() # Number of bits necessary to represent self in binary.…
  def conjugate(…) # Returns self, the complex conjugate of any int.
  def from_bytes(bytes, byteorder='big', *, signed=False) # Return the integer represented by the given array of bytes.…
  def is_integer() # Returns True. Exists for duck type compatibility with float.is_integer.
  def to_bytes(length=1, byteorder='big', *, signed=False) # Return an array of bytes representing an integer.…
```



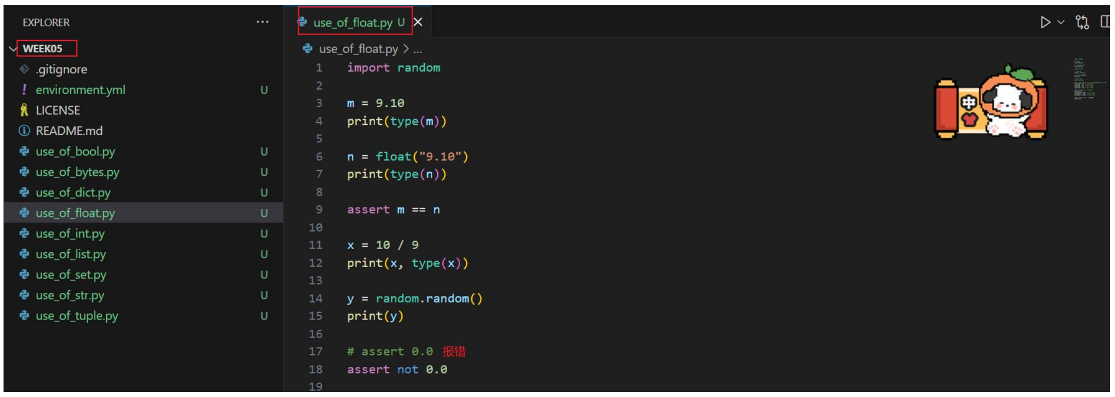
```
$ python use_of_int.py

--Return--
> c:\users\1\repo\week05\use_of_int.py(13)<module>()->None
-> breakpoint()
(Pdb) p x
991010
(Pdb) p x.to_bytes()
*** OverflowError: int too big to convert
(Pdb)
```

```
x = 991010
```

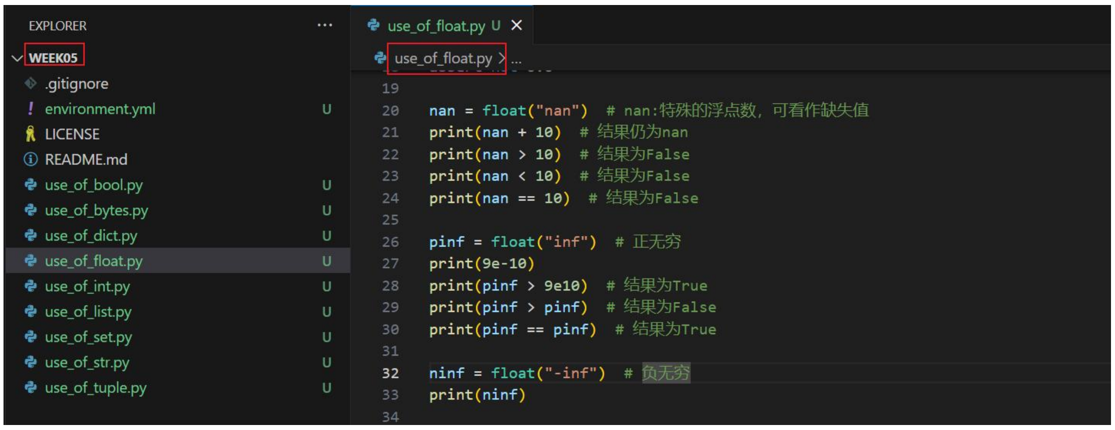
Float 浮点数

不要对浮点数做“=”的判断。

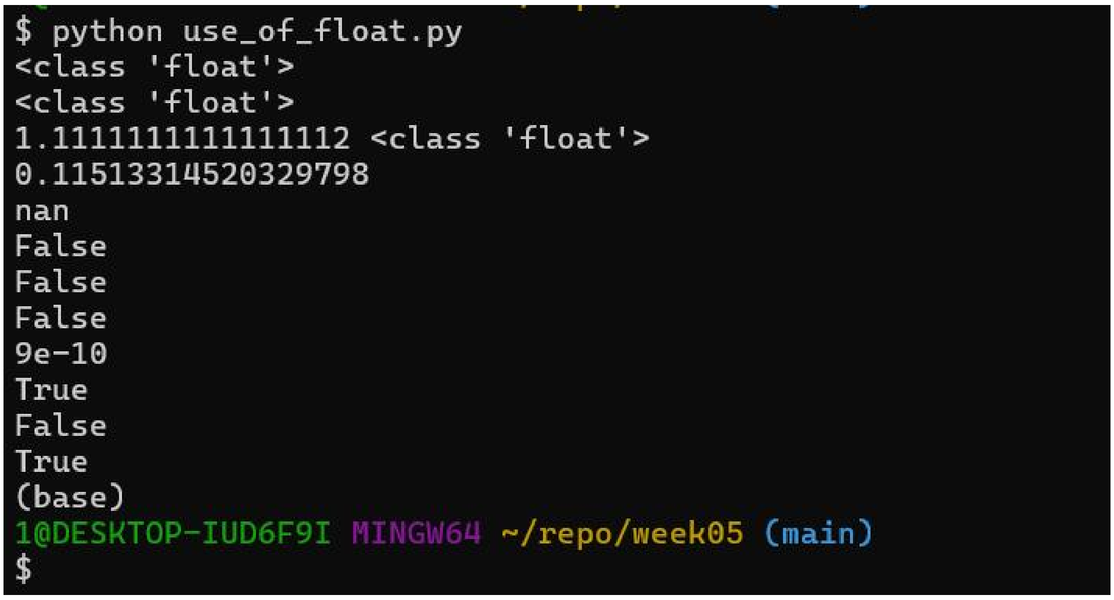
浮点数是零点几开头时，可以将小数点前的零省去。



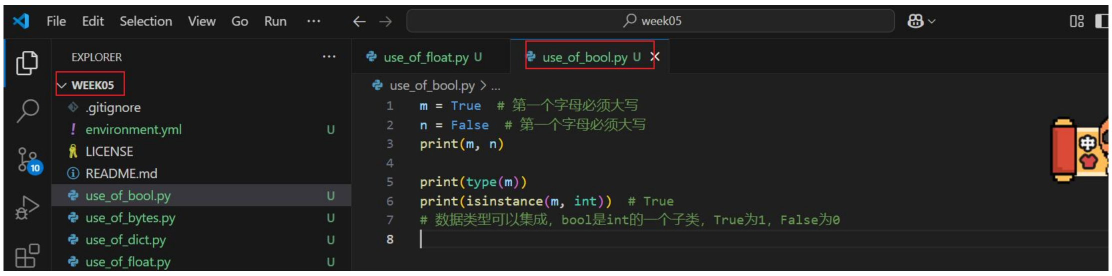
```python
import random

m = 9.10
print(type(m))

n = float("9.10")
print(type(n))

assert m == n

x = 10 / 9
print(x, type(x))

y = random.random()
print(y)

# assert 0.0 报错
assert not 0.0
```

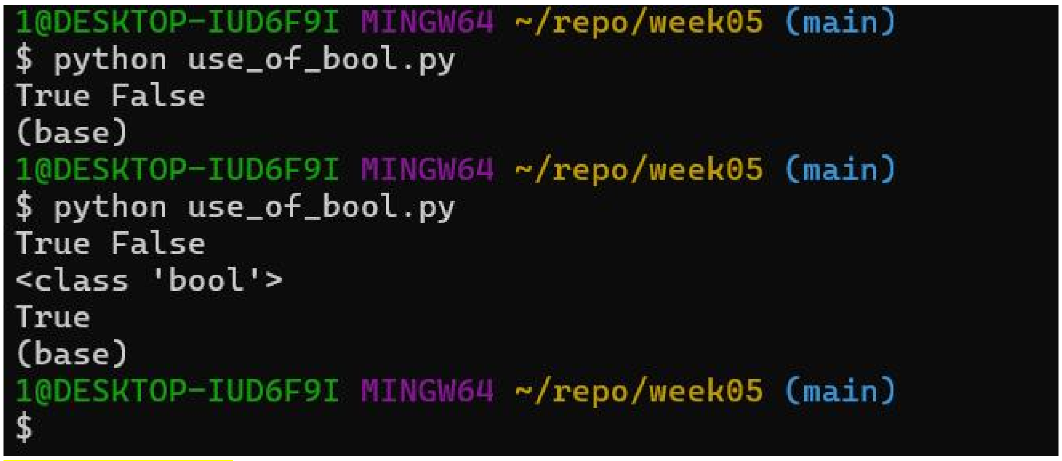


```python
nan = float("nan")  # nan:特殊的浮点数，可看作缺失值
print(nan + 10)  # 结果仍为nan
print(nan > 10)  # 结果为False
print(nan < 10)  # 结果为False
print(nan == 10)  # 结果为False

pinf = float("inf")  # 正无穷
print(9e-10)
print(pinf > 9e10)  # 结果为True
print(pinf > pinf)  # 结果为False
print(pinf == pinf)  # 结果为True

ninf = float("-inf")  # 负无穷
print(ninf)
```

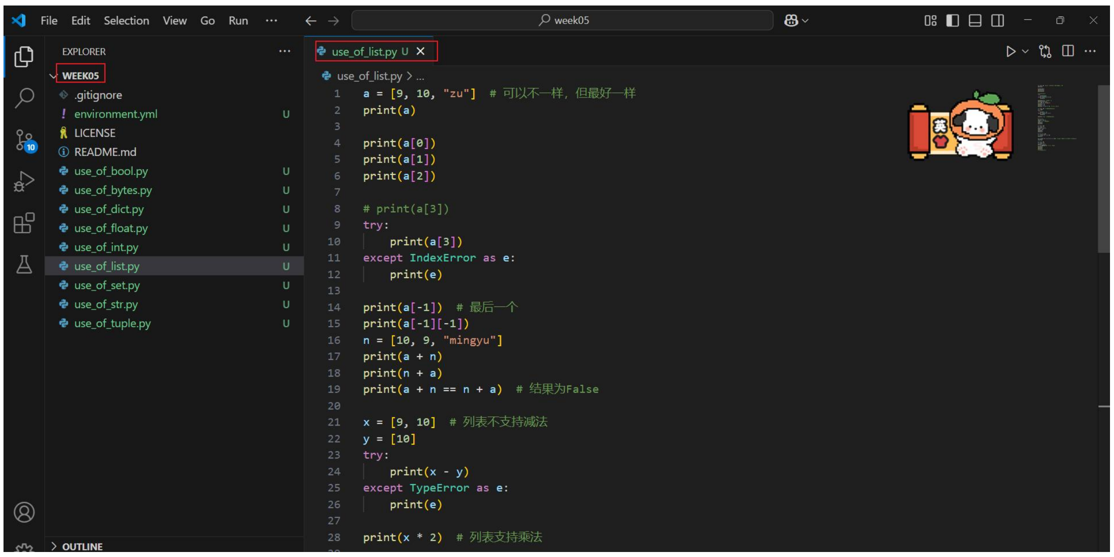

```
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.1111111111111112 <class 'float'>
0.11513314520329798
nan
False
False
False
9e-10
True
False
True
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

Bool 布尔值

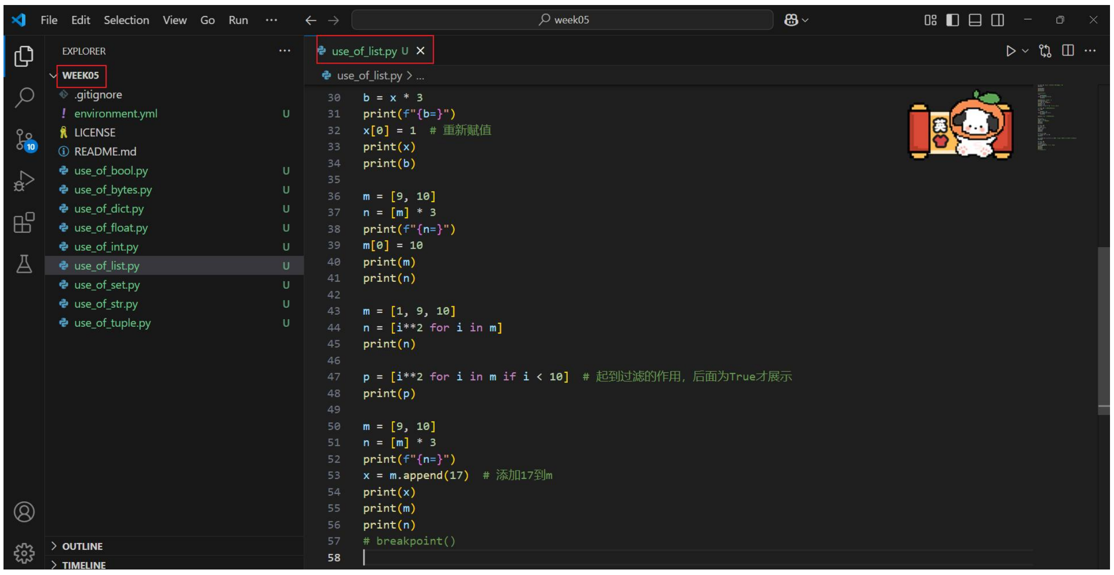

```
use_of_bool.py > ...
m = True  # 第一个字母必须大写
n = False  # 第一个字母必须大写
print(m, n)

print(type(m))
print(isinstance(m, int))  # True
# 数据类型可以集成，bool是int的一个子类，True为1，False为0
```


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_bool.py
True False
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_bool.py
True False
<class 'bool'>
True
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

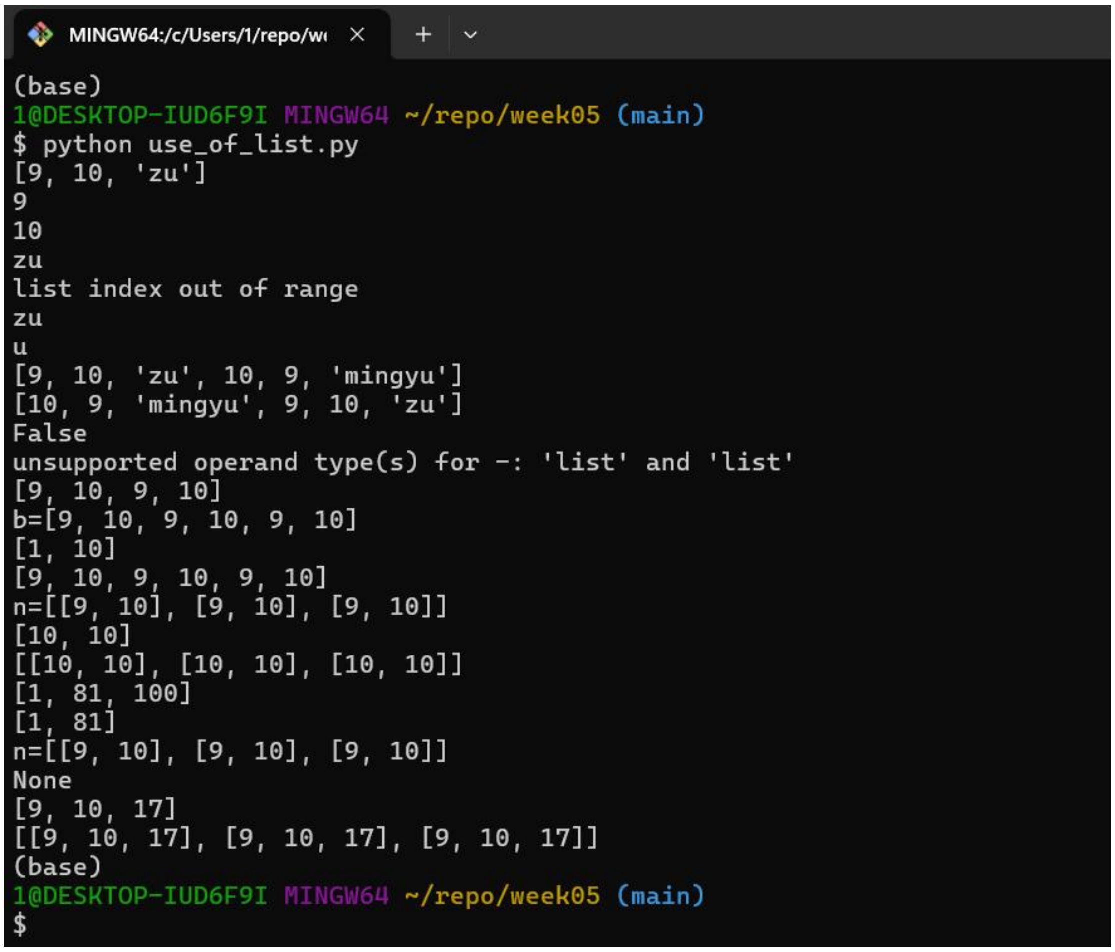
List 列表



```python
a = [9, 10, "zu"]  # 可以不一样，但最好一样
print(a)

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

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

print(a[-1])  # 最后一个
print(a[-1][-1])
n = [10, 9, "mingyu"]
print(a + n)
print(n + a)
print(a + n == n + a)  # 结果为False

x = [9, 10]  # 列表不支持减法
y = [10]
try:
    print(x - y)
except TypeError as e:
    print(e)

print(x * 2)  # 列表支持乘法
```

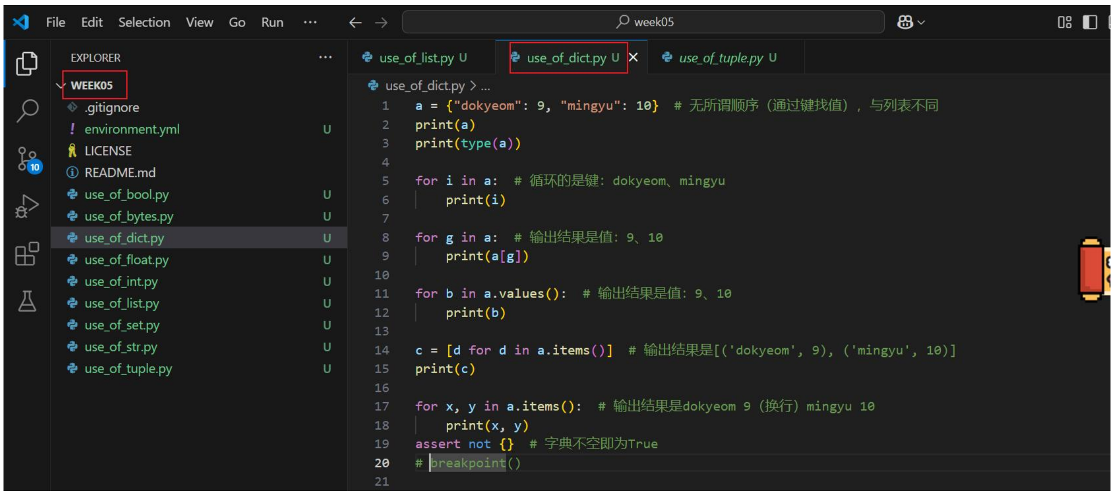


```python
b = x * 3
print(f"{b=}")
x[0] = 1  # 重新赋值
print(x)
print(b)

m = [9, 10]
n = [m] * 3
print(f"{n=}")
m[0] = 10
print(m)
print(n)

m = [1, 9, 10]
n = [i**2 for i in m]
print(n)

p = [i**2 for i in m if i < 10]  # 起到过滤的作用，后面为True才展示
print(p)

m = [9, 10]
n = [m] * 3
print(f"{n=}")
x = m.append(17)  # 添加17到m
print(x)
print(m)
print(n)
# breakpoint()

```

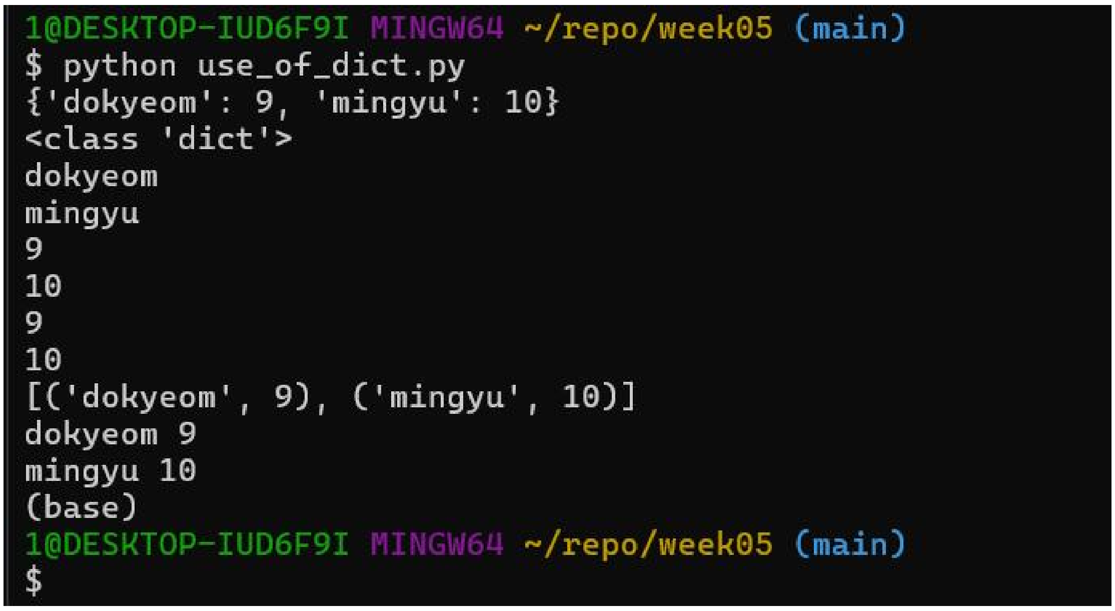


```
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_list.py
[9, 10, 'zu']
9
10
zu
list index out of range
zu
u
[9, 10, 'zu', 10, 9, 'mingyu']
[10, 9, 'mingyu', 9, 10, 'zu']
False
unsupported operand type(s) for -: 'list' and 'list'
[9, 10, 9, 10]
b=[9, 10, 9, 10, 9, 10]
[1, 10]
[9, 10, 9, 10, 9, 10]
n=[[9, 10], [9, 10], [9, 10]]
[10, 10]
[[10, 10], [10, 10], [10, 10]]
[1, 81, 100]
[1, 81]
n=[[9, 10], [9, 10], [9, 10]]
None
[9, 10, 17]
[[9, 10, 17], [9, 10, 17], [9, 10, 17]]
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

Dict 字典

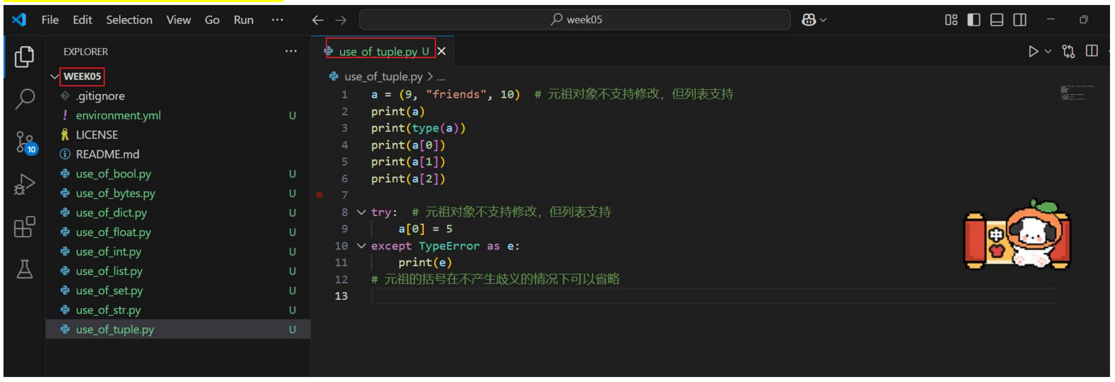


```python
a = {"dokyeom": 9, "mingyu": 10}  # 无所谓顺序（通过键找值），与列表不同
print(a)
print(type(a))

for i in a:  # 循环的是键: dokyeom、mingyu
    print(i)

for g in a:  # 输出结果是值: 9、10
    print(a[g])

for b in a.values():  # 输出结果是值: 9、10
    print(b)

c = [d for d in a.items()]  # 输出结果是[('dokyeom', 9), ('mingyu', 10)]
print(c)

for x, y in a.items():  # 输出结果是dokyeom 9 (换行) mingyu 10
    print(x, y)
assert not {}  # 字典不空即为True
# breakpoint()

```

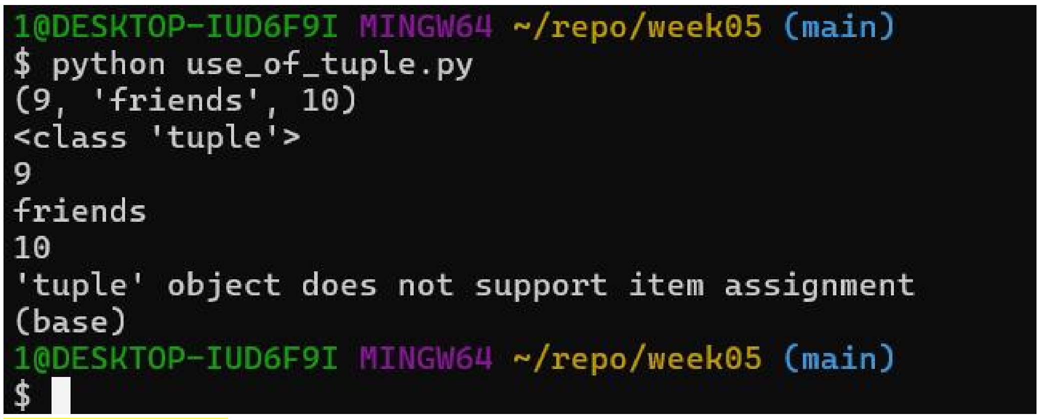


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_dict.py
{'dokyeom': 9, 'mingyu': 10}
<class 'dict'>
dokyeom
mingyu
9
10
9
10
[('dokyeom', 9), ('mingyu', 10)]
dokyeom 9
mingyu 10
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

Tuple 元组

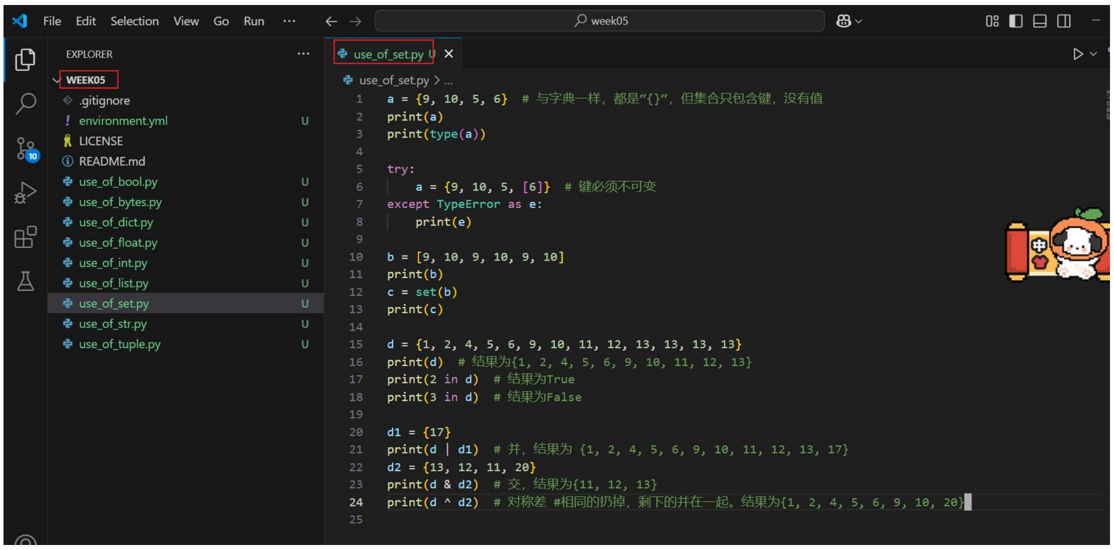


```python
a = (9, "friends", 10)  # 元祖对象不支持修改，但列表支持
print(a)
print(type(a))
print(a[0])
print(a[1])
print(a[2])

try:  # 元祖对象不支持修改，但列表支持
    a[0] = 5
except TypeError as e:
    print(e)
# 元祖的括号在不产生歧义的情况下可以省略

```

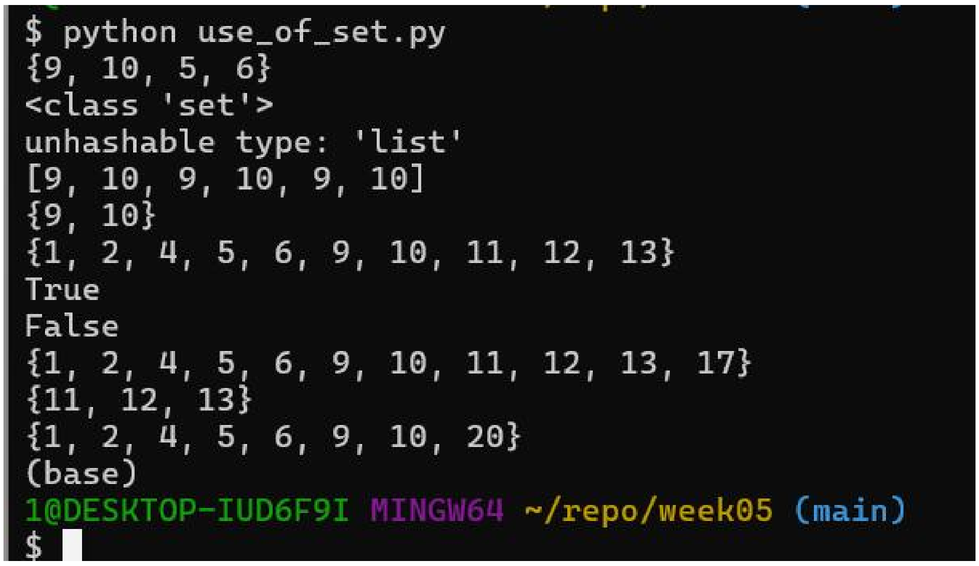


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_tuple.py
(9, 'friends', 10)
<class 'tuple'>
9
friends
10
'tuple' object does not support item assignment
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

Set 集合

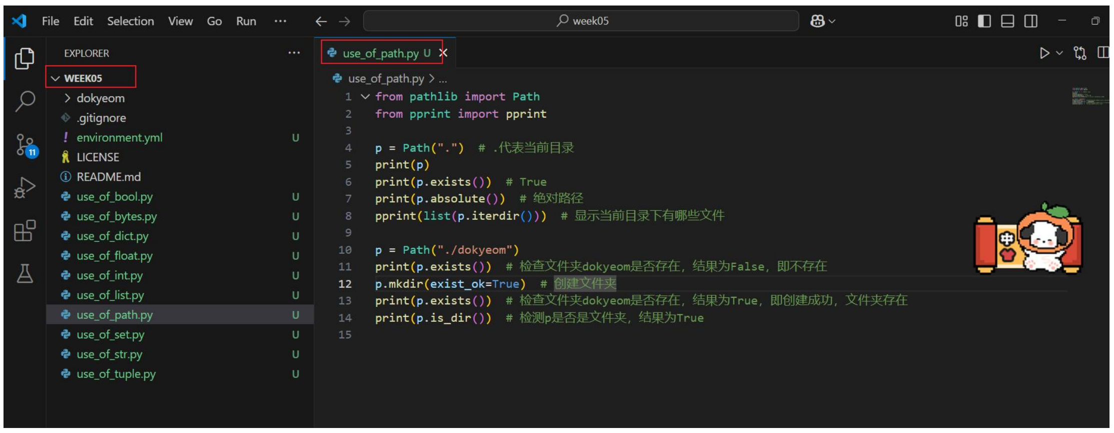


```python
a = {9, 10, 5, 6}  # 与字典一样，都是“{}“，但集合只包含键，没有值
print(a)
print(type(a))

try:
    a = {9, 10, 5, [6]}  # 键必须不可变
except TypeError as e:
    print(e)

b = [9, 10, 9, 10, 9, 10]
print(b)
c = set(b)
print(c)

d = {1, 2, 4, 5, 6, 9, 10, 11, 12, 13, 13, 13, 13}
print(d)  # 结果为{1, 2, 4, 5, 6, 9, 10, 11, 12, 13}
print(2 in d)  # 结果为True
print(3 in d)  # 结果为False

d1 = {17}
print(d | d1)  # 并，结果为 {1, 2, 4, 5, 6, 9, 10, 11, 12, 13, 17}
d2 = {13, 12, 11, 20}
print(d & d2)  # 交，结果为{11, 12, 13}
print(d ^ d2)  # 对称差 #相同的扔掉，剩下的并在一起。结果为{1, 2, 4, 5, 6, 9, 10, 20}
```

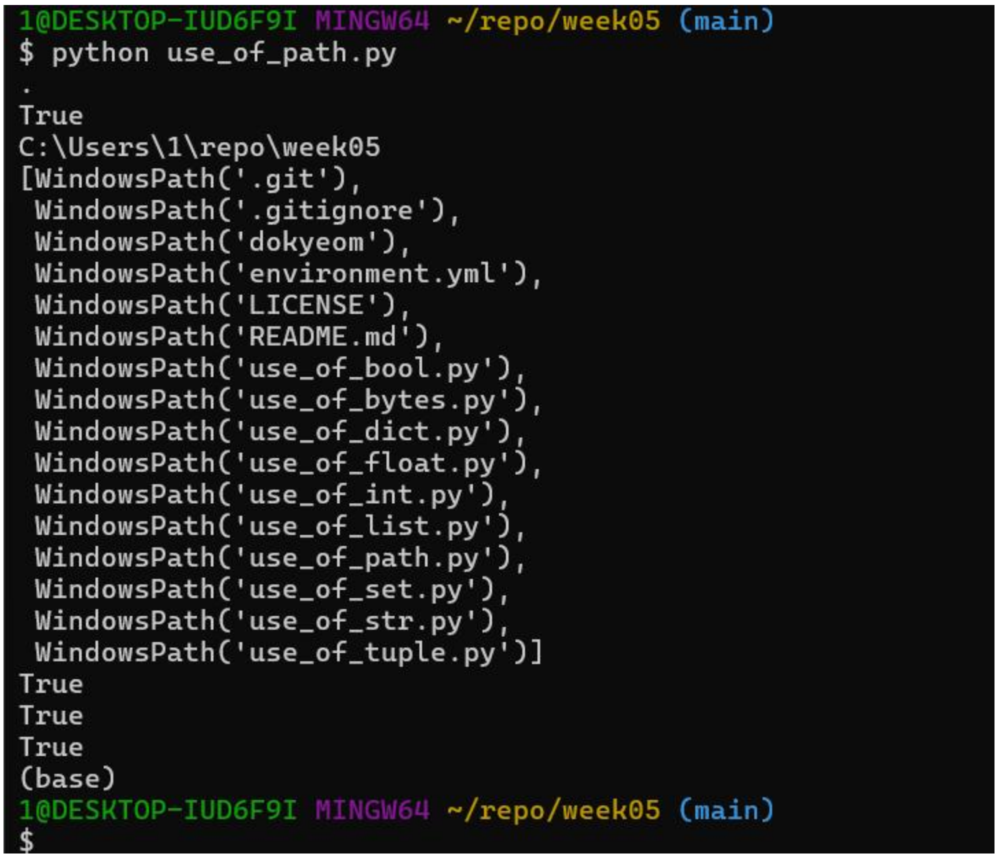


```
$ python use_of_set.py
{9, 10, 5, 6}
<class 'set'>
unhashable type: 'list'
[9, 10, 9, 10, 9, 10]
{9, 10}
{1, 2, 4, 5, 6, 9, 10, 11, 12, 13}
True
False
{1, 2, 4, 5, 6, 9, 10, 11, 12, 13, 17}
{11, 12, 13}
{1, 2, 4, 5, 6, 9, 10, 20}
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

Pathlib 路径

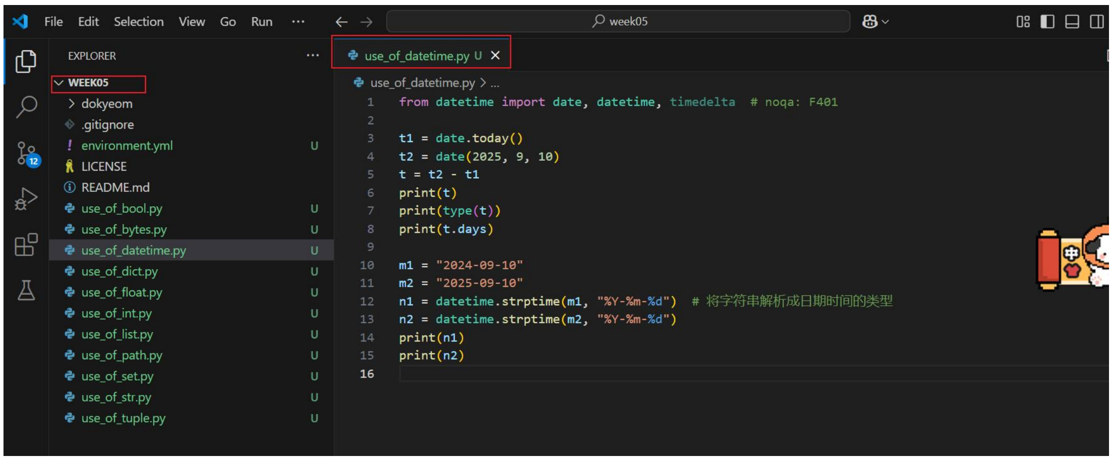


```python
from pathlib import Path
from pprint import pprint

p = Path(".")  # .代表当前目录
print(p)
print(p.exists())  # True
print(p.absolute())  # 绝对路径
pprint(list(p.iterdir()))  # 显示当前目录下有哪些文件

p = Path("./dokyeom")
print(p.exists())  # 检查文件夹dokyeom是否存在，结果为False，即不存在
p.mkdir(exist_ok=True)  # 创建文件夹
print(p.exists())  # 检查文件夹dokyeom是否存在，结果为True，即创建成功，文件夹存在
print(p.is_dir())  # 检测p是否是文件夹，结果为True
```

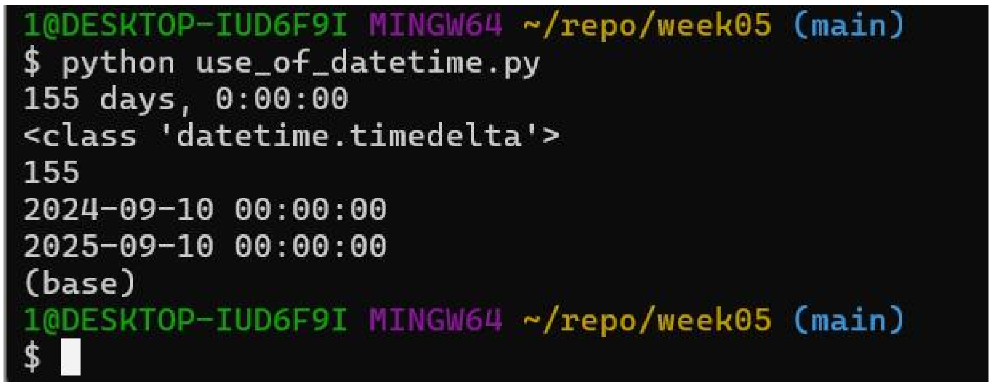


```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_path.py
.
True
C:\Users\1\repo\week05
[WindowsPath('.git'),
 WindowsPath('.gitignore'),
 WindowsPath('dokyeom'),
 WindowsPath('environment.yml'),
 WindowsPath('LICENSE'),
 WindowsPath('README.md'),
 WindowsPath('use_of_bool.py'),
 WindowsPath('use_of_bytes.py'),
 WindowsPath('use_of_dict.py'),
 WindowsPath('use_of_float.py'),
 WindowsPath('use_of_int.py'),
 WindowsPath('use_of_list.py'),
 WindowsPath('use_of_path.py'),
 WindowsPath('use_of_set.py'),
 WindowsPath('use_of_str.py'),
 WindowsPath('use_of_tuple.py')]
True
True
True
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```

Datetime 日期时间



```python
from datetime import date, datetime, timedelta  # noqa: F401

t1 = date.today()
t2 = date(2025, 9, 10)
t = t2 - t1
print(t)
print(type(t))
print(t.days)

m1 = "2024-09-10"
m2 = "2025-09-10"
n1 = datetime.strptime(m1, "%Y-%m-%d")  # 将字符串解析成日期时间的类型
n2 = datetime.strptime(m2, "%Y-%m-%d")
print(n1)
print(n2)
```



```
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$ python use_of_datetime.py
155 days, 0:00:00
<class 'datetime.timedelta'>
155
2024-09-10 00:00:00
2025-09-10 00:00:00
(base)
1@DESKTOP-IUD6F9I MINGW64 ~/repo/week05 (main)
$
```