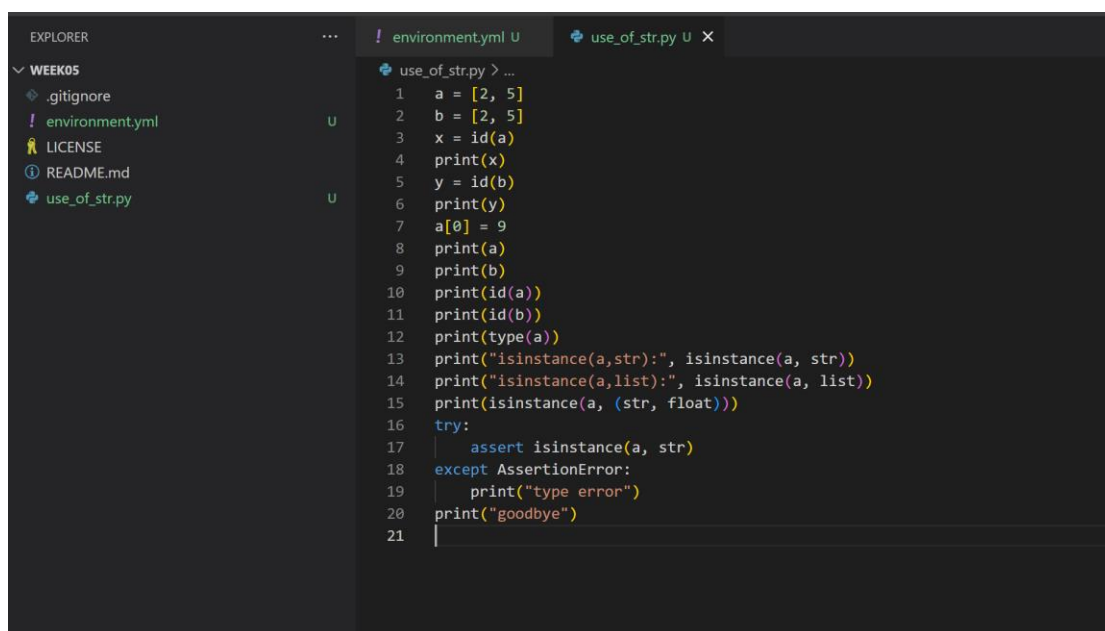


```
(base) xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ conda activate week05
(week05)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
(week05)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
```

```
(week05)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2389197216944
(week05)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1458660859056
(week05)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
```



The screenshot shows a code editor interface. On the left, the 'EXPLORER' sidebar displays the file structure of a project named 'WEEK05'. The files listed are '.gitignore', 'environment.yml', 'LICENSE', 'README.md', and 'use_of_str.py'. The 'use_of_str.py' file is selected. On the right, the code editor shows the contents of 'use_of_str.py'. The code defines two lists, 'a' and 'b', both containing the values [2, 5]. It then prints the memory addresses of 'a' and 'b' using 'id(a)' and 'id(b)', which are 2389197216944 and 1458660859056 respectively. The code also prints the type of 'a' (list) and checks if 'a' is an instance of 'str', 'list', and 'float' using 'isinstance'. A try-except block attempts to assert that 'a' is an instance of 'str', which will raise an 'AssertionError' because 'a' is a list.

```
EXPLORER
WEEK05
  .gitignore
  ! environment.yml
  LICENSE
  README.md
  use_of_str.py

! environment.yml U
use_of_str.py U X

use_of_str.py > ...
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     print("type error")
20 print("goodbye")
21
```

```
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
你好
eason
True
(main)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$
```

```
use_of_str.py > ...
1 print("你好")
2 s = "eason"
3 print(s)
4 print(isinstance(s, str))
5 assert type(s) is str
6
7 print("f-string")
8 x = "Slicence"
9 s = f"name:{x}"
10 print(s)
11
12 s = "a\tb"
13 print("TAB", s)
14
15 s = "aaa\nbbb"
16 print("New Line", s)
17

name:Slicence
TAB a b
(main)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
你好
eason
True
f-string
name:Slicence
TAB a b
New Line aaa
bbb
(main)
xuan@LAPTOP-00K17TT3 MINGW64 ~/repo/week05 (main)
$
```

id(): 能看到对象在内存中的地址 (一个整数), 如果俩对象的 id 一样, 那用 is 判断也会是 True, 因为 is 就是比内存地址的。

type(): 直接返回对象的类型, 比如 int、str 这些。

isinstance(): 判断对象是不是属于某个类型, 比如 isinstance(3, int) 会返回 True, 也能同时判断多个类型, 像 isinstance(3, (int, str))。

dir(): 列出对象所有能调用的属性和方法名, 比如对一个列表用 dir(), 会显示 append、pop 这些方法。

str(): 返回对象打印时显示的字符串, 比如 str([1,2]) 就是 '[1, 2]', 和 print 输出的样子一样。

print(): 把表达式结果打印到终端, 方便看是不是和自己想的一样。

assert: 用来检查某个条件是不是真的, 要是条件不成立就会报错 AssertionError 并停止运行, 比如 assert 2+2==4 没问题, 但 assert 2+2==5 就会报错。

try-except: 用 try 把可能出错的代码包起来, 万一报错了, 程序不会直接退出, 而是跳到 except 里处理, 比如可以捕获特定错误做补救。

breakpoint(): 调试的时候在代码里加这一句, 程序运行到这里会暂停, 进入 pdb 调试模式, 这时候可以一行一行检查代码, 看变量值对不对。