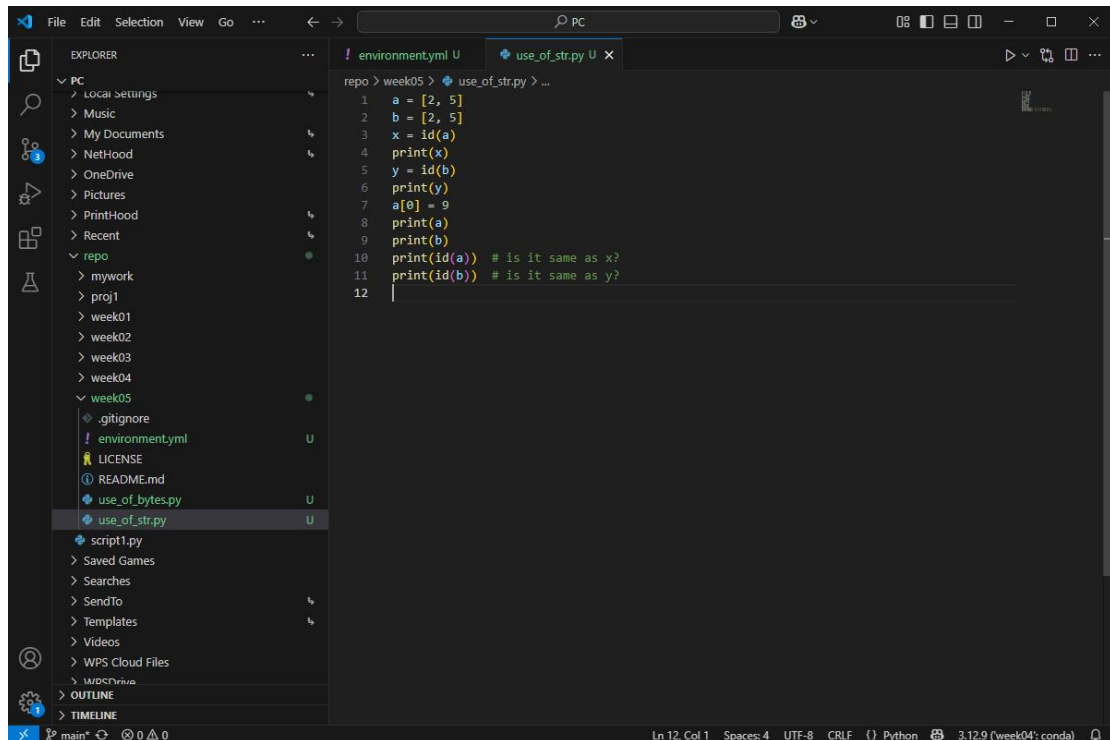
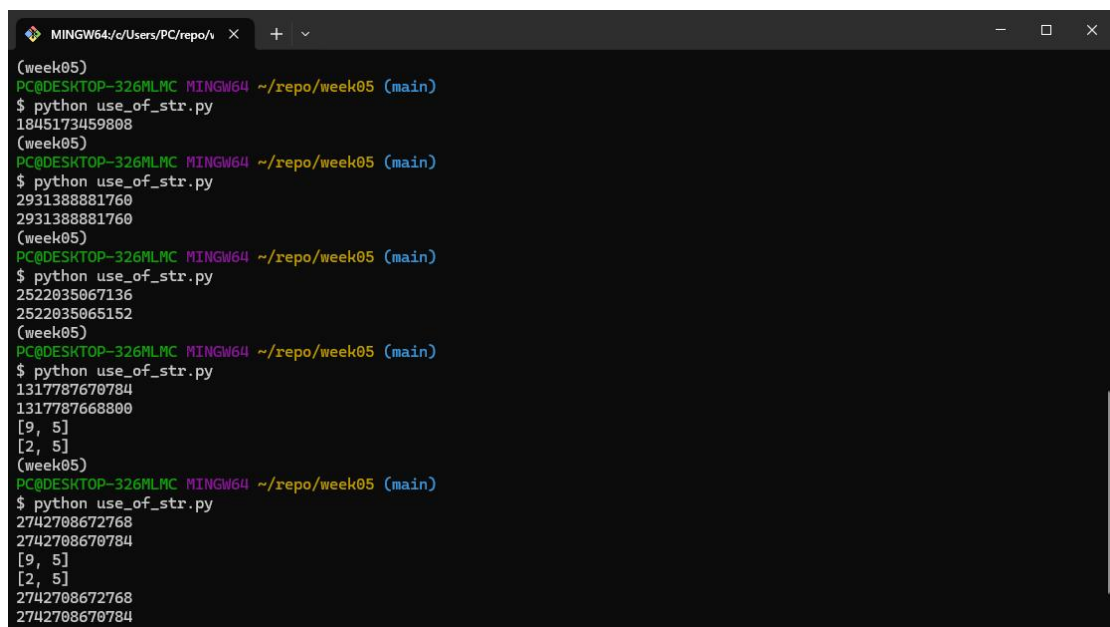


1. `id()` -- 返回对象在虚拟内存中的地址（正整数），如果 `id(a) == id(b)`，那么 `a` is `b`



```
! environment.yml U  use_of_str.py U x
repo > week05 > 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)) # is it same as x?
11 print(id(b)) # is it same as y?
12
```



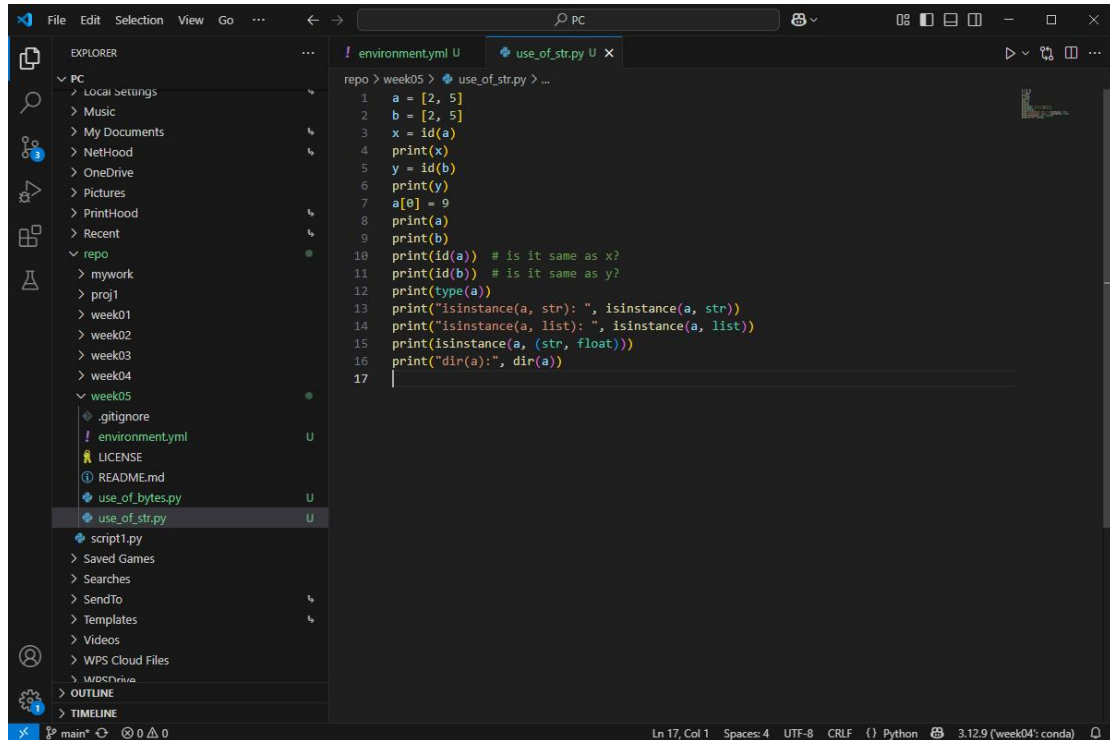
```
MINGW64~/Users/PC/repo/week05
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1845173459808
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2931388881760
2931388881760
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2522035067136
2522035065152
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1317787670784
1317787668800
[9, 5]
[2, 5]
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2742708672768
2742708670784
[9, 5]
[2, 5]
2742708672768
2742708670784
```

2. type() -- 返回对象的类型

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的名称列表

调用 print() 函数将表达式输出到终端，查看结果是否符合预期



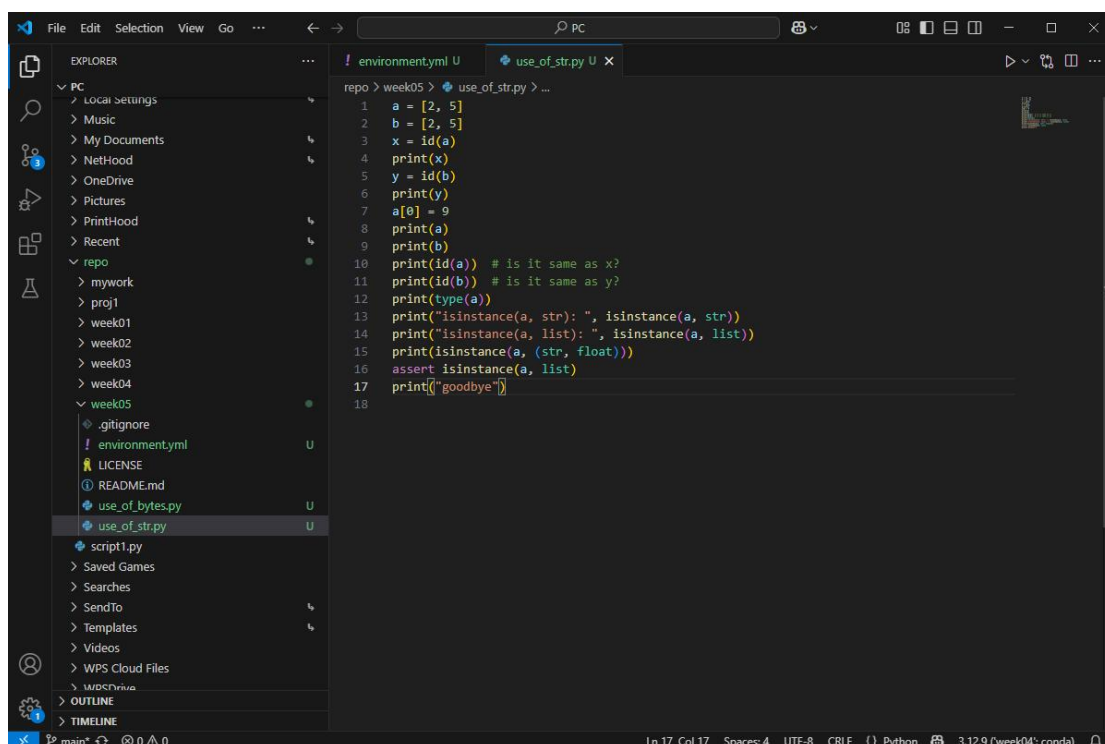
```
File Edit Selection View Go ... PC
EXPLORER
  PC
  > Local Settings
  > Music
  > My Documents
  > NetHood
  > OneDrive
  > Pictures
  > PrintHood
  > Recent
  > repo
    > mywork
    > proj1
    > week01
    > week02
    > week03
    > week04
    > week05
      .gitignore
      ! environment.yml
      LICENSE
      README.md
      use_of_bytes.py
      use_of_str.py
      script1.py
    > Saved Games
    > Searches
    > SendTo
    > Templates
    > Videos
    > WPS Cloud Files
    > WPSDrive
    > OUTLINE
    > TIMELINE
  > main
  0 0 0

! environment.yml U
use_of_str.py U X
repo > week05 > 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)) # is it same as x?
11 print(id(b)) # is it same as y?
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 print("dir(a):", dir(a))
17
```



```
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2416339065088
2416339063104
[9, 5]
[2, 5]
2416339065088
2416339063104
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattr__', '__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']
```

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

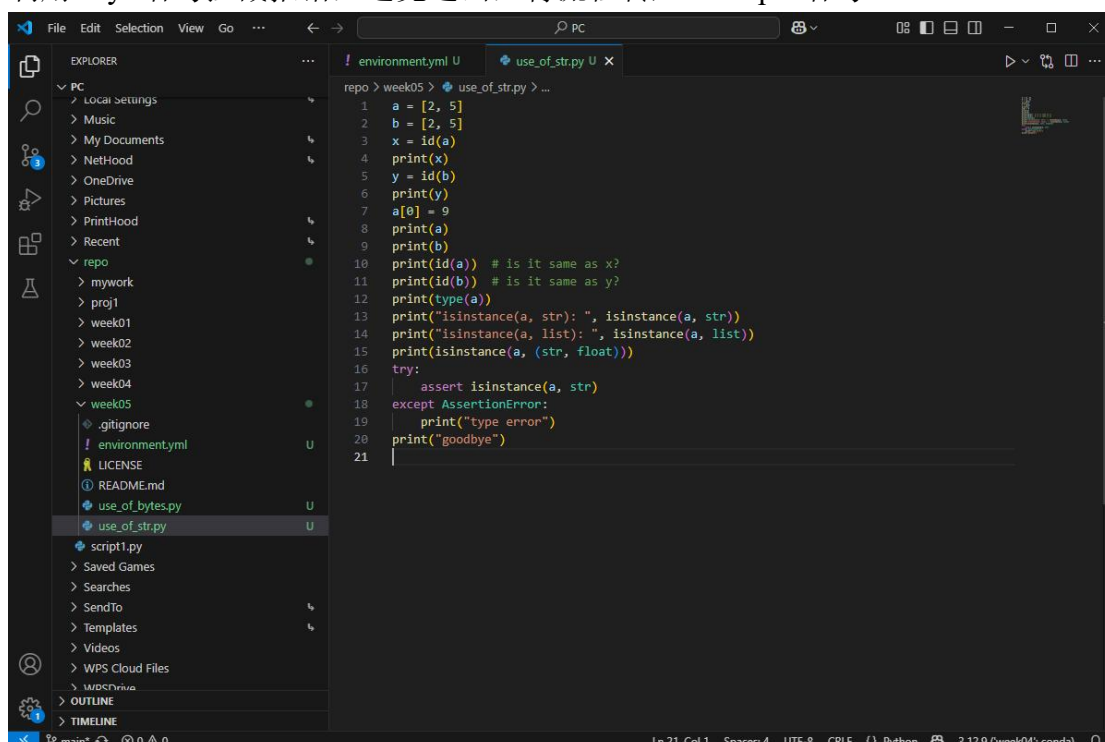


The screenshot shows the VS Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'PC', 'repo', and 'week05'. The code editor displays a Python script named 'use_of_str.py' with the following code:

```
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)) # is it same as x?
11 print(id(b)) # is it same as y?
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")
18
```

The status bar at the bottom indicates the file is at line 17, column 17, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python 3.12.9 (week04: conda) environment.

利用 `try` 语句拦截报错，避免退出，将流程转入 `except` 语句



The screenshot shows the VS Code interface with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'PC', 'repo', and 'week05'. The code editor displays a Python script named 'use_of_str.py' with the following code:

```
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)) # is it same as x?
11 print(id(b)) # is it same as y?
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
```

The status bar at the bottom indicates the file is at line 21, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and Python 3.12.9 (week04: conda) environment.

```
MINGW64/c/Users/PC/repo/week05
$ python use_of_str.py
1517922949376
1517922947392
[9, 5]
[2, 5]
1517922949376
1517922947392
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
False
Traceback (most recent call last):
  File "C:\Users\PC\repo\week05\use_of_str.py", line 16, in <module>
    assert isinstance(a, str)
AssertionError
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1732525562112
1732525560128
[9, 5]
[2, 5]
1732525562112
1732525560128
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
False
goodbye
```

4. 调用 breakpoint() 函数暂停程序运行，进入 pdb 调试模式

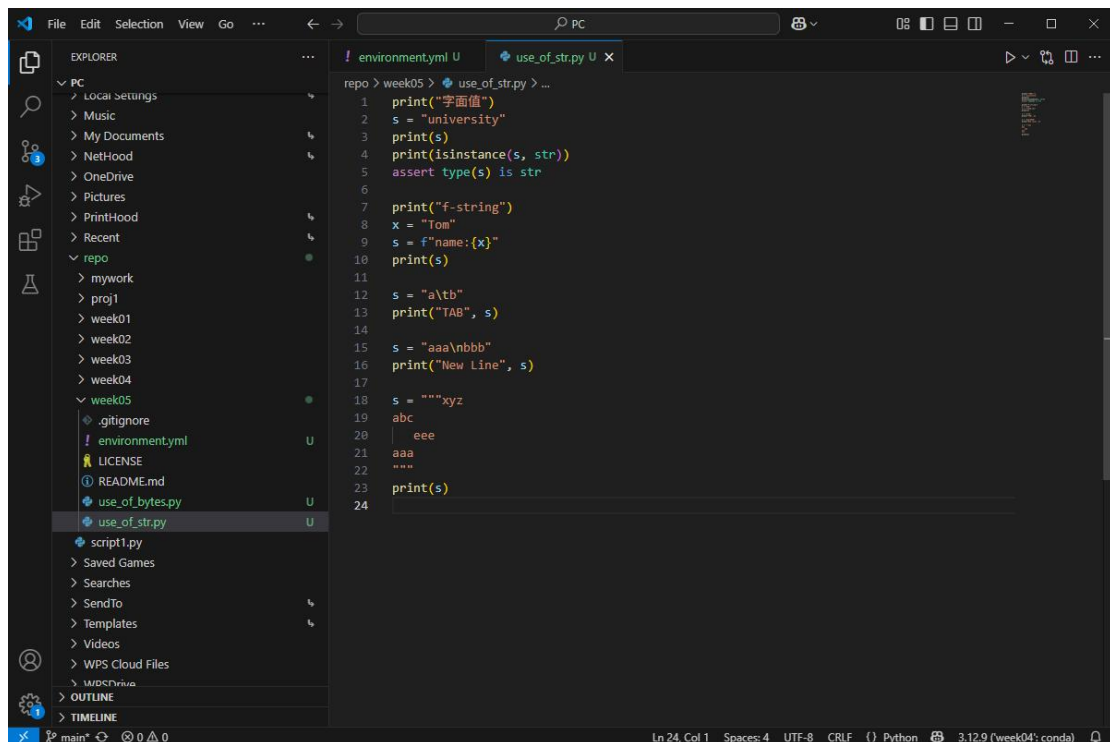
```
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2621753727232
2621753725248
[9, 5]
[2, 5]
2621753727232
2621753725248
<class 'list'>
isinstance(a, str): False
isinstance(a, list): True
False
> c:\users\pc\repo\week05\use_of_str.py(20)<module>()
-> print("type error")
(Pdb) l
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")
[EOF]
(Pdb) p a
[9, 5]
(Pdb) p isinstance(a, str)
False
(Pdb)
```

5. 字符串的表达式 (expression) 和实例 (instance)

(1) 字面值 (Literal) 是直接写在代码里的「固定值」，无需计算即可确定内容，包括 f-string 这种「动态字面值」，即运行时计算表达式。

基础字面值：123（整数）、3.14（浮点数）、'hello'（字符串）、True、None

f-string 字面值：用 f""" 包裹，含 {表达式}，运行时动态渲染。



```
repo > week05 > use_of_str.py > ...
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 x = "Tom"
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
18 s = """xyz
19 abc
20 eee
21 aaa
22 """
23 print(s)
24
```

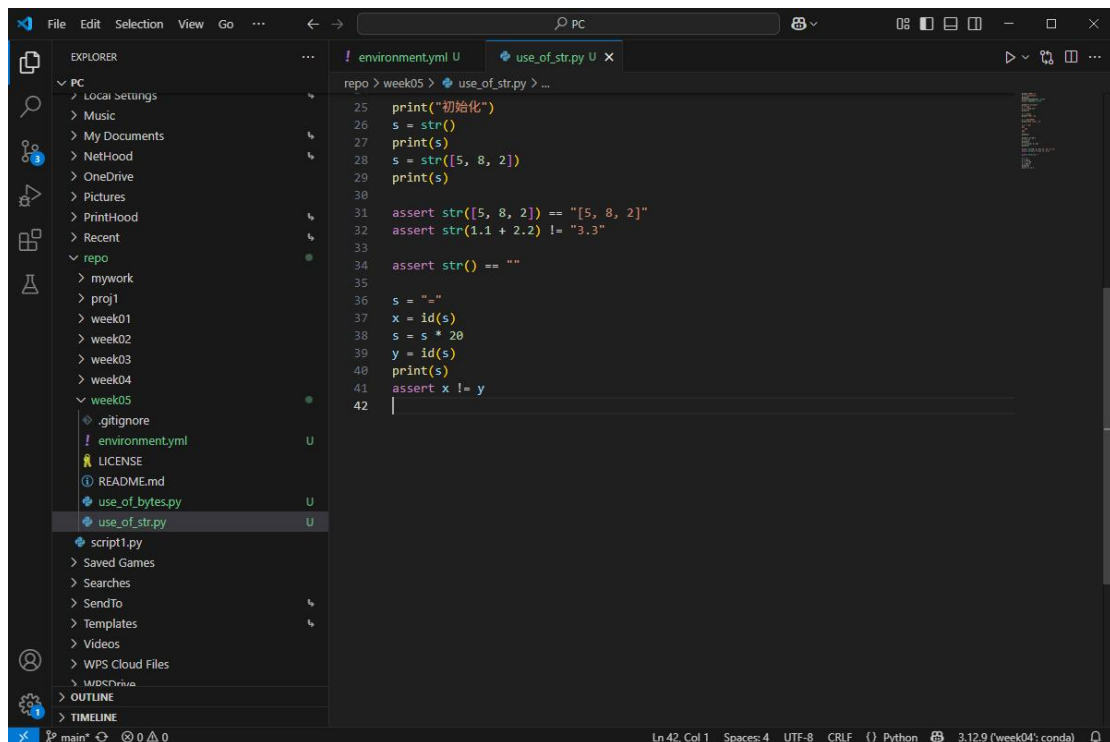


```
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name:Tom
TAB a b
New Line aaa
bbb
xyz
abc
eee
aaa
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$
```

(2) 推导式 (Comprehension) 是一行代码创建容器，替代 for 循环的冗长写法，仅限 list/dict/set 三种类型。

(3) 初始化 (Init) 是创建对象并赋予初始值的过程，分「字面量初始化」和「构造函数初始化」。

(4) 运算值 (Operator) 是运算符作用于操作数后得到的结果，分「算术运算」和「特殊运算」(如拼接、比较)。



```
(week05)
PC@DESKTOP-326MLMC 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

[5, 8, 2]
=====
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$
```

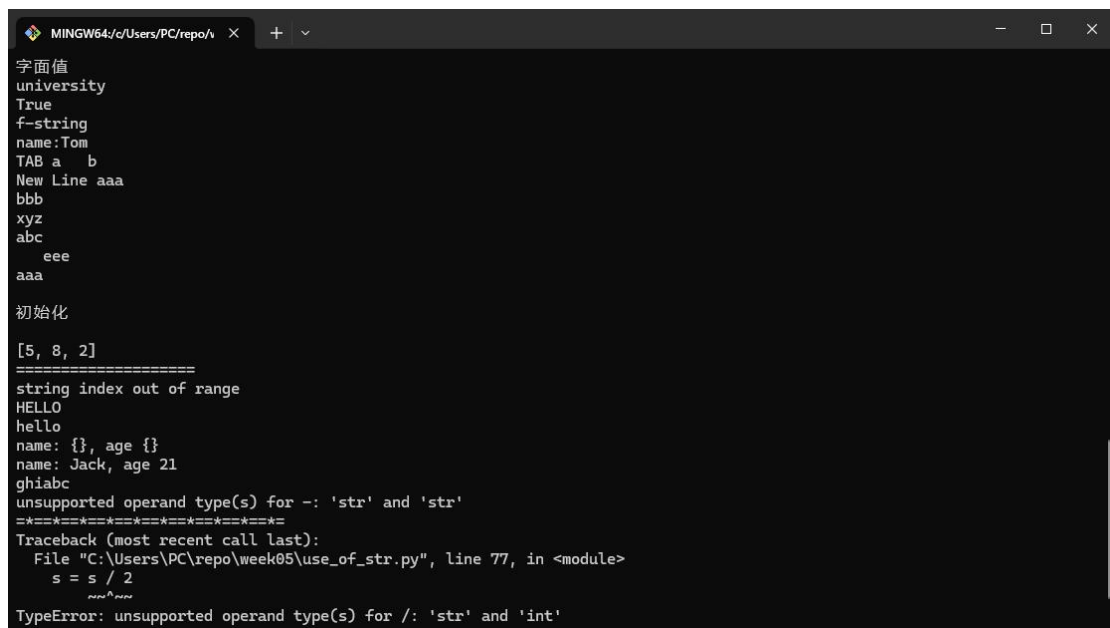
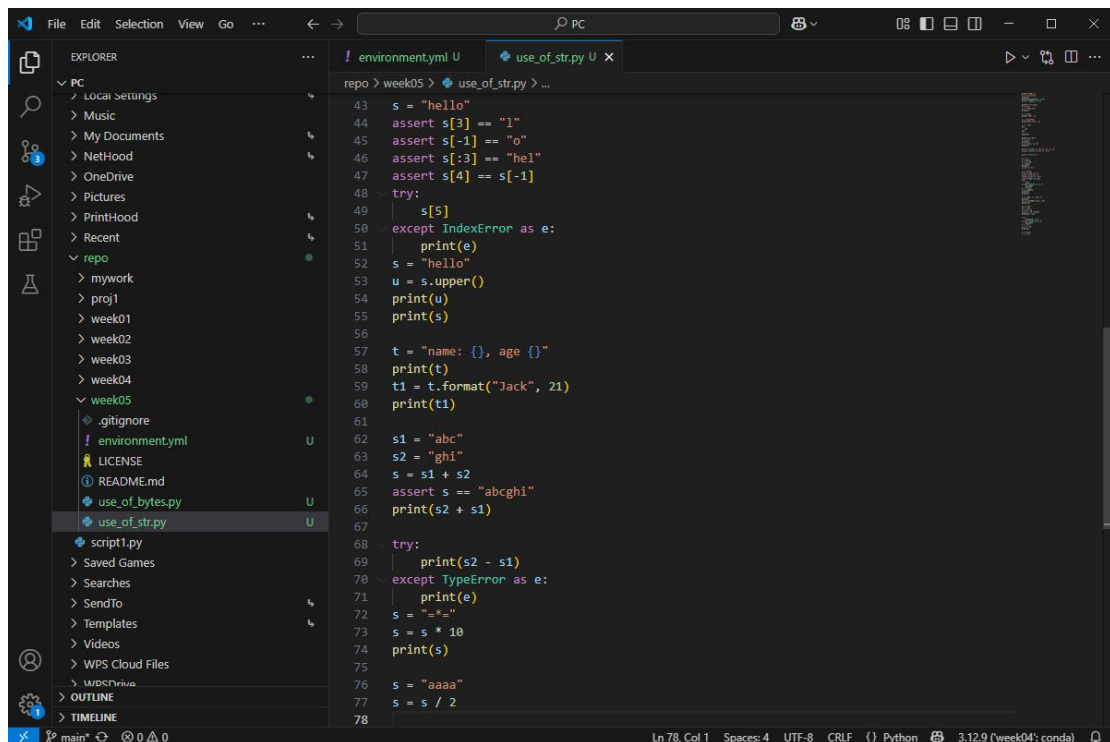
(5) 索引值 (Subscription) 是通过 `[]` 运算符获取对象中的元素，支持「索引」和「切片」。

序列类型：列表、元组、字符串（按位置索引）

映射类型：字典（按键索引）

(6) 返回值 (Return Value) 是函数或方法执行后返回的结果，用 `return` 语句指定，默认返回 `None`。

区分「修改原对象的方法」（如 `list.append()` 返回 `None`）和「纯计算方法」（如 `str.strip()` 返回新字符串）。



6. 字符串的属性

(1) 数学运算符支持情况

- +: 加法运算符，用于数值相加或字符串拼接
- : 减法运算符，用于数值相减
- *: 乘法运算符，用于数值相乘或字符串重复

/: 除法运算符，执行普通除法，结果为浮点数

(2) 判断相等，== 用于比较两个对象的值是否相等

(3) 比较运算符支持情况

>、<、>=、<= 这些比较运算符，用于比较两个对象的大小关系

(4) 在 Python 中，以下值被当作 False，其他值一般被当作 True:

False、None、数值 0（包括 0、0.0、0j）、空序列（如 ''、[]、()）、空映射（如 {}）、空集合（如 set()）

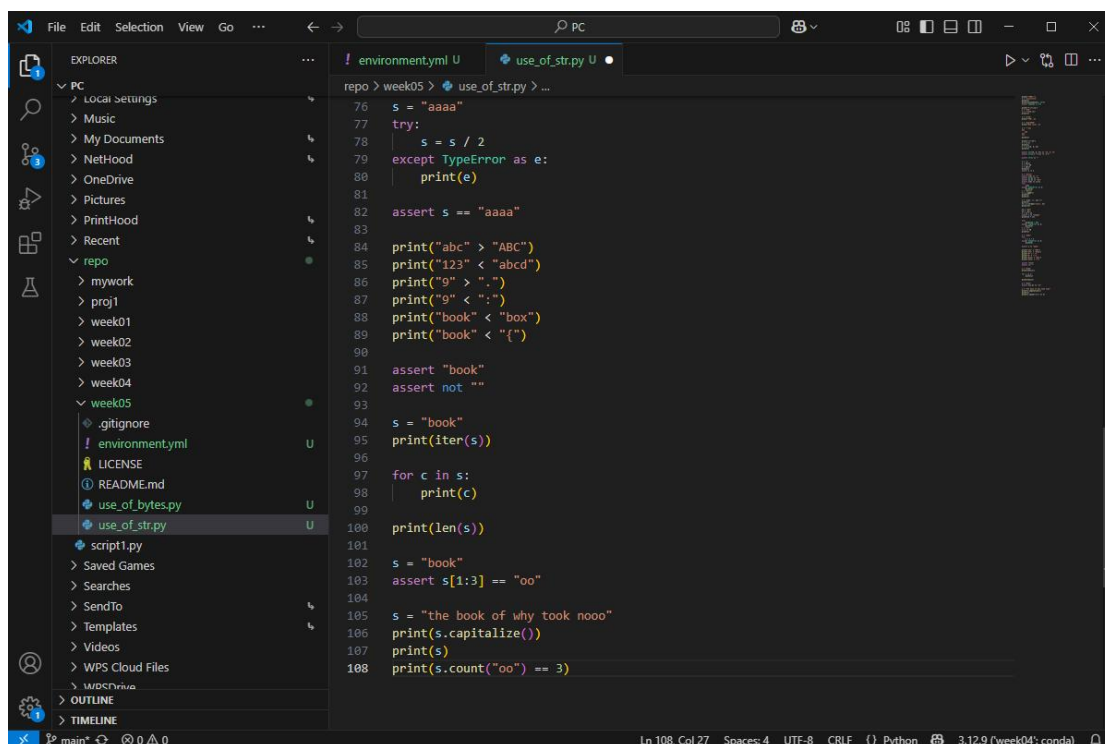
(5) 可迭代对象与迭代

在 Python 中，可迭代对象是指实现了 iter() 方法的对象，像列表、元组、字符串、字典等都是可迭代对象，可以使用 for 循环对可迭代对象进行迭代。

(6) Python 支持使用 len() 函数返回可迭代对象的长度

(7) 索引操作

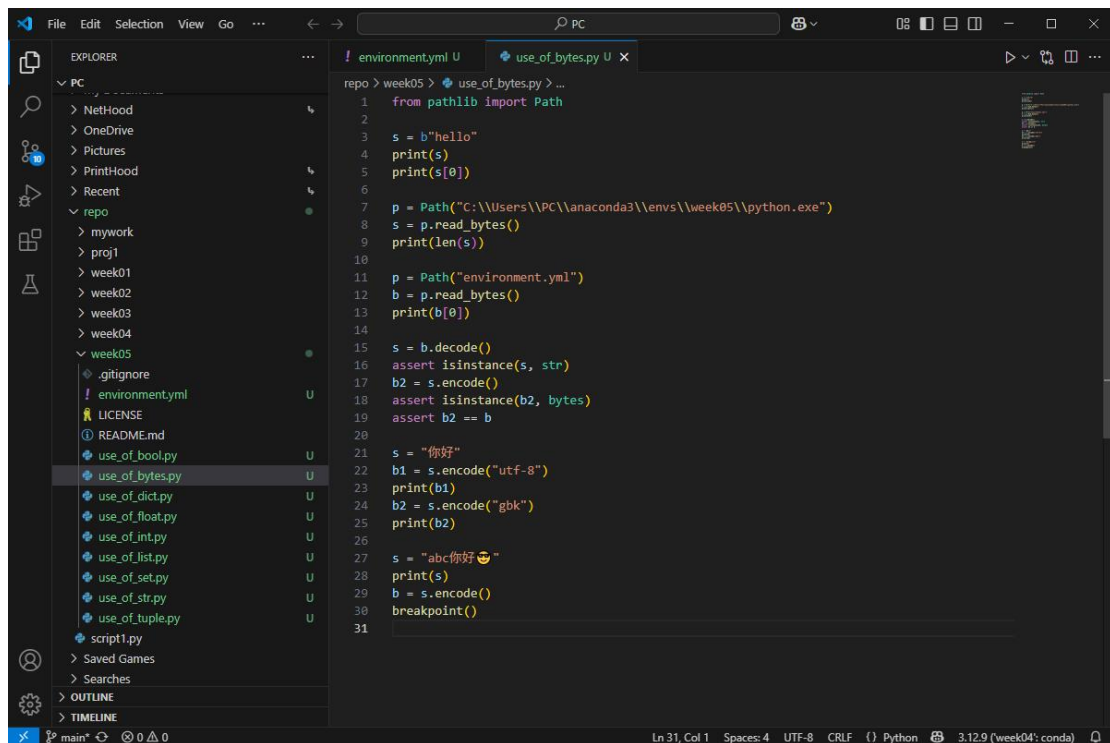
Python 支持使用 [] 运算符对序列（如列表、元组、字符串）和映射（如字典）进行索引操作。



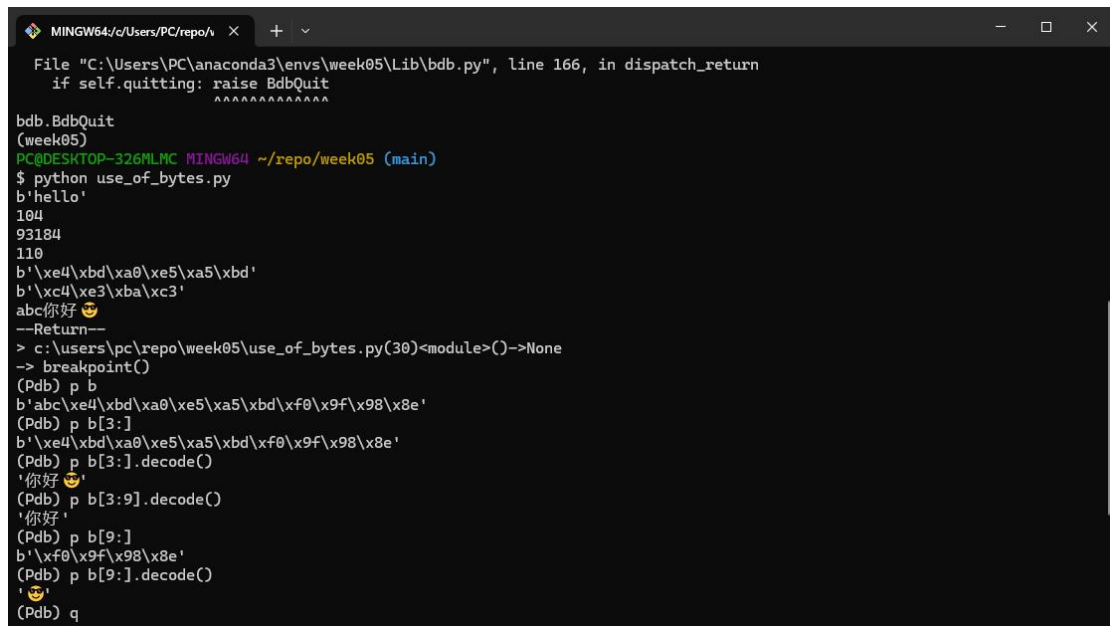
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like 'PC', 'Music', 'My Documents', 'Pictures', 'PrintHood', 'Recent', 'repo', 'mywork', 'proj1', 'week01', 'week02', 'week03', 'week04', and 'week05'. The 'week05' folder is expanded, showing files like '.gitignore', 'environment.yml', 'LICENSE', 'README.md', 'use_of_bytes.py', 'use_of_str.py', 'script1.py', 'Saved Games', 'Searches', 'SendTo', 'Templates', 'Videos', and 'WPS Cloud Files'. The 'use_of_str.py' file is selected. The code editor shows the following Python code:

```
76 s = "aaaa"
77 try:
78     s = s / 2
79 except TypeError as e:
80     print(e)
81
82 assert s == "aaaa"
83
84 print("abc" > "ABC")
85 print("123" < "abcd")
86 print("9" > ".")
87 print("9" < ":")
88 print("book" < "box")
89 print("book" < "{}")
90
91 assert "book"
92 assert not ""
93
94 s = "book"
95 print(iter(s))
96
97 for c in s:
98     print(c)
99
100 print(len(s))
101
102 s = "book"
103 assert s[1:3] == "oo"
104
105 s = "the book of why took nooo"
106 print(s.capitalize())
107 print(s)
108 print(s.count("oo") == 3)
```


7. 字节串



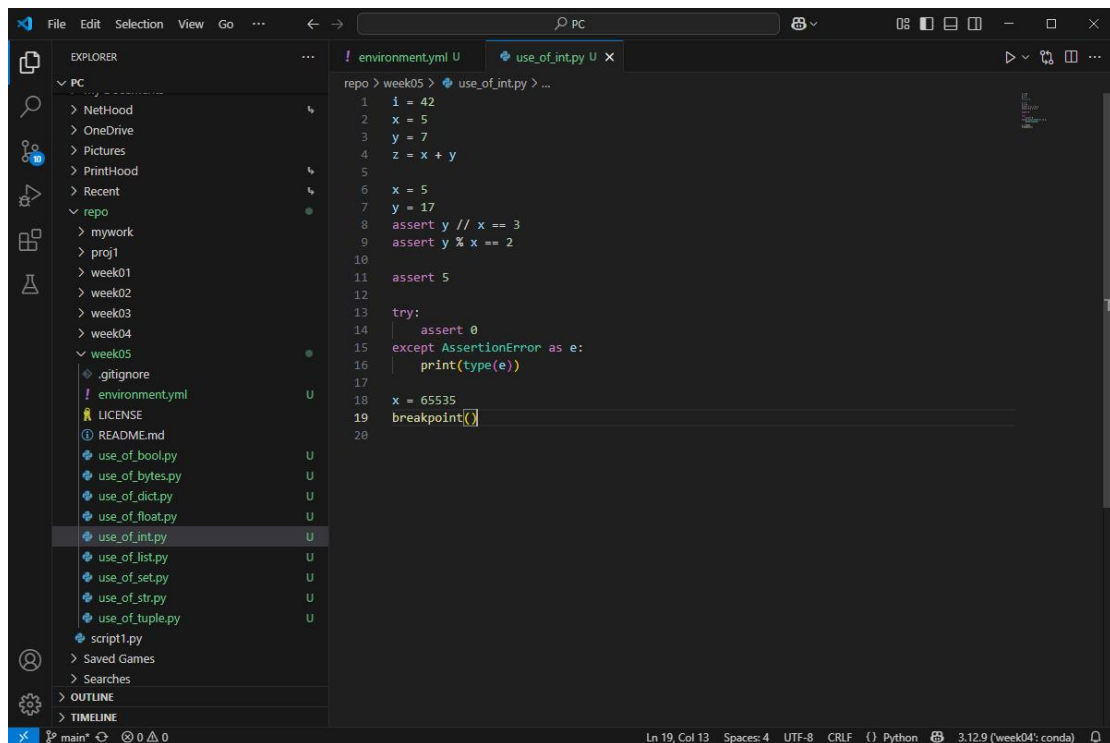
```
environment.yml U  use_of_bytes.py U X
repo > week05 > use_of_bytes.py > ...
1  from pathlib import Path
2
3  s = b"hello"
4  print(s)
5  print(s[0])
6
7  p = Path("C:\\Users\\PC\\anaconda3\\envs\\week05\\python.exe")
8  s = p.read_bytes()
9  print(len(s))
10
11 p = Path("environment.yml")
12 b = 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 b1 = s.encode("utf-8")
23 print(b1)
24 b2 = s.encode("gbk")
25 print(b2)
26
27 s = "abc你好😁"
28 print(s)
29 b = s.encode()
30 breakpoint()
31
```



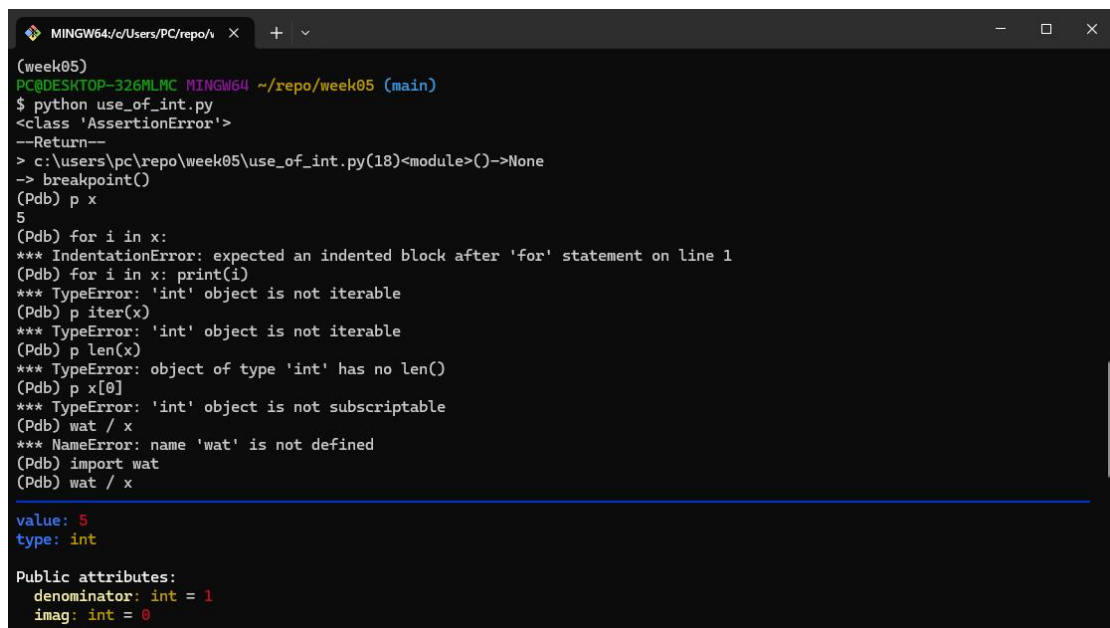
```
File "C:\Users\PC\anaconda3\envs\week05\Lib\bdb.py", line 166, in dispatch_return
if self.quitting: raise BdbQuit
^^^^^^^^^^^^^^^^

bdb.BdbQuit
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'hello'
104
93184
110
b'\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\xe'
b'\xc4\xe3\xba\xc3'
abc你好😁
--Return--
> c:\users\pc\repo\week05\use_of_bytes.py(30)<module>()->None
-> breakpoint()
(Pdb) p b
b'abc\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\xe'
(Pdb) p b[3:]
b'\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\xe'
(Pdb) p b[3:].decode()
'你好😁'
(Pdb) p b[3:9].decode()
'你好'
(Pdb) p b[9:]
b'\xf0\x9f\x98\xe'
(Pdb) p b[9:].decode()
'😁'
(Pdb) q
```

8. 整数



```
File Edit Selection View Go ... PC
! environment.yml U use_of_int.py U X
repo > week05 > use_of_int.py > ...
1 i = 42
2 x = 5
3 y = 7
4 z = x + y
5
6 x = 5
7 y = 17
8 assert y // x == 3
9 assert y % x == 2
10
11 assert 5
12
13 try:
14     assert 0
15 except AssertionError as e:
16     print(type(e))
17
18 x = 65535
19 breakpoint()
20
```

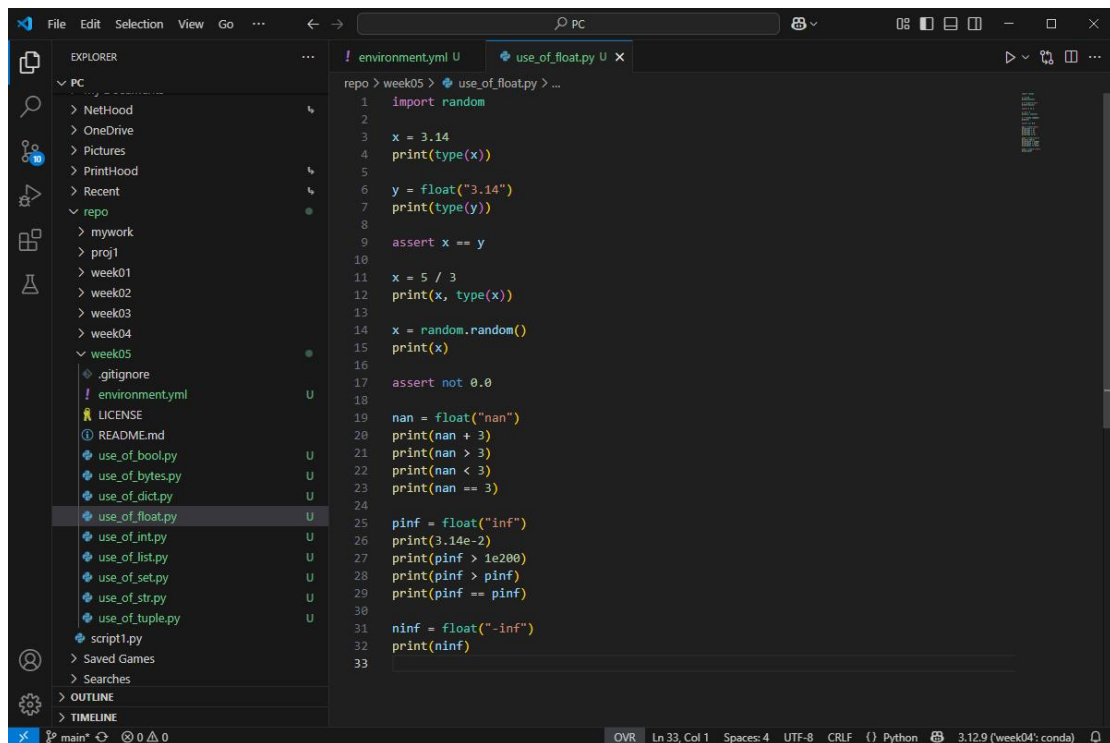


```
MINGW64/c/Users/PC/repo/  X + -
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_int.py
<class 'AssertionError'>
--Return--
> c:\users\pc\repo\week05\use_of_int.py(18)<module>()->None
-> breakpoint()
(Pdb) p x
5
(Pdb) for i in x:
*** IndentationError: expected an indented block after 'for' statement 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) wat / x
*** NameError: name 'wat' is not defined
(Pdb) import wat
(Pdb) wat / x

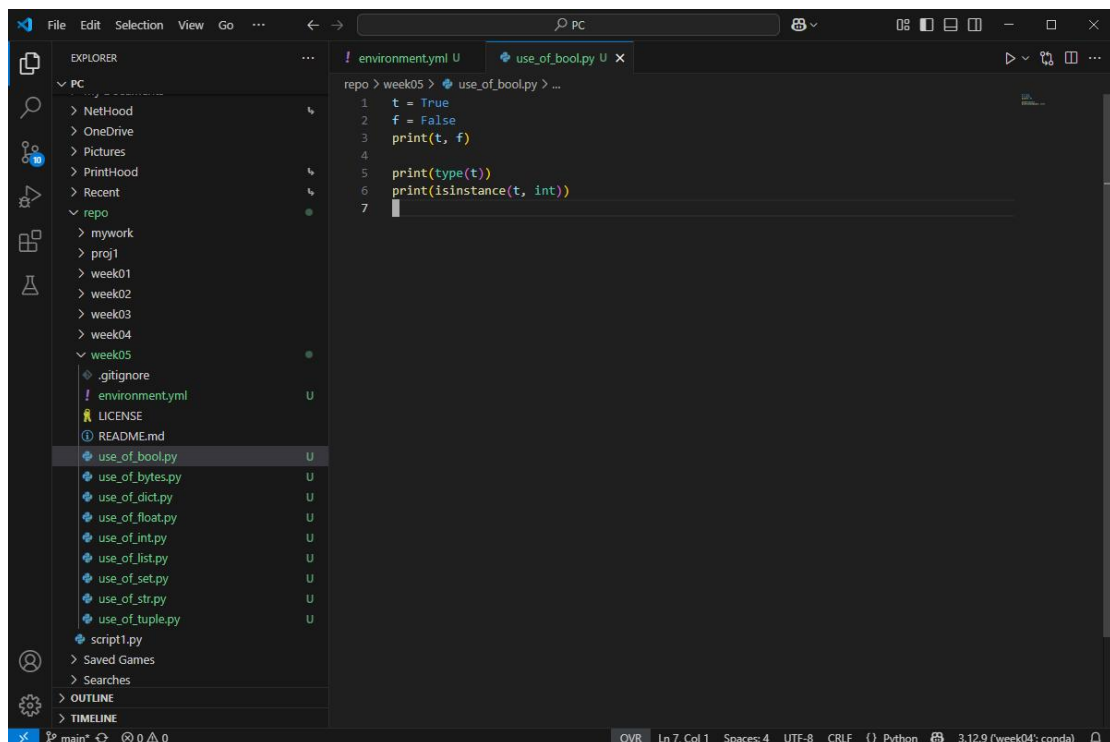
value: 5
type: int

Public attributes:
  denominator: int = 1
  imag: int = 0
```

9. 浮点数和布尔值



```
environment.yml U  use_of_float.py U x
repo > week05 > 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 / 3
12 print(x, type(x))
13
14 x = random.random()
15 print(x)
16
17 assert not 0.0
18
19 nan = float("nan")
20 print(nan + 3)
21 print(nan > 3)
22 print(nan < 3)
23 print(nan == 3)
24
25 pinf = float("inf")
26 print(3.14e-2)
27 print(pinf > 1e200)
28 print(pinf > pinf)
29 print(pinf == pinf)
30
31 ninf = float("-inf")
32 print(ninf)
33
```



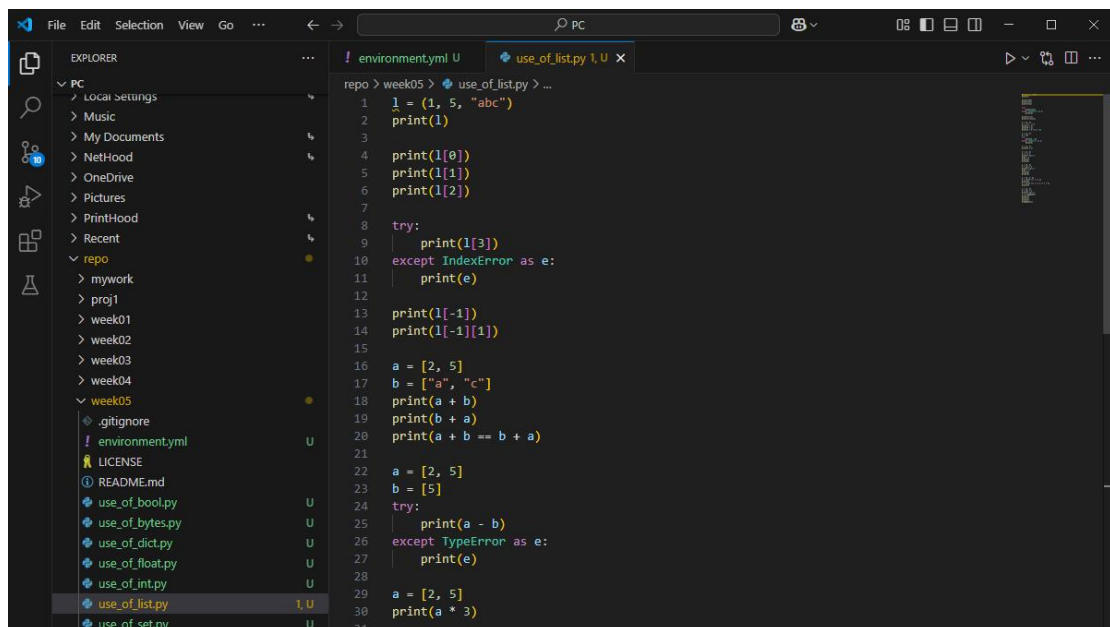
```
environment.yml U  use_of_bool.py U x
repo > week05 > use_of_bool.py > ...
1  t = True
2  f = False
3  print(t, f)
4
5  print(type(t))
6  print(isinstance(t, int))
7
```

```

PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.666666666666667 <class 'float'>
0.2174845816334181
nan
False
False
False
0.0314
True
False
True
-inf
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_bool.py
True False
<class 'bool'>
True
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$

```

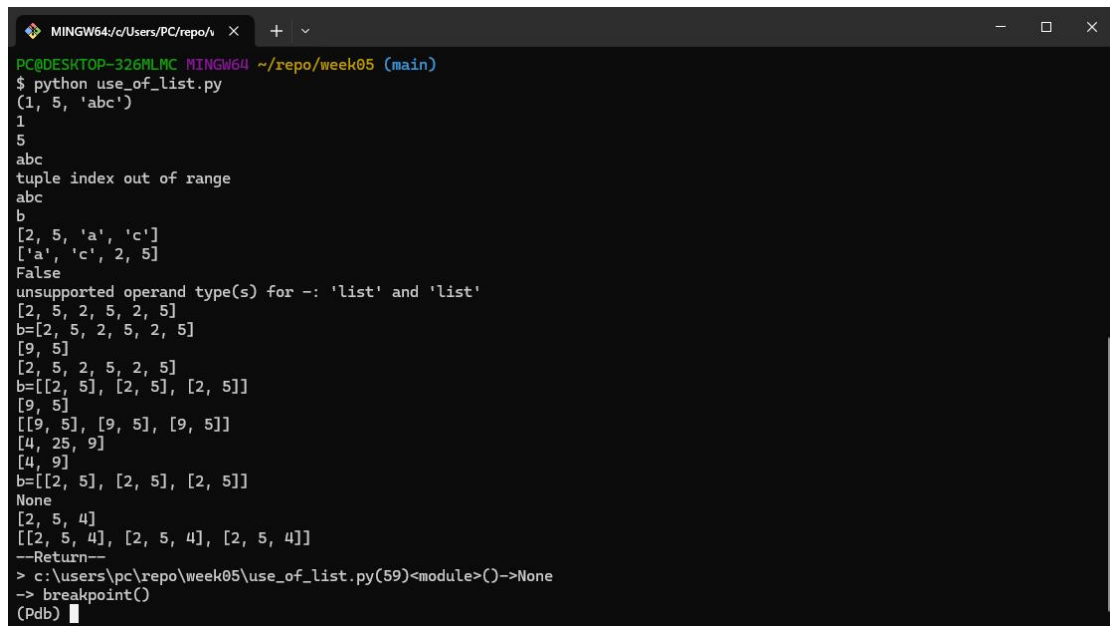
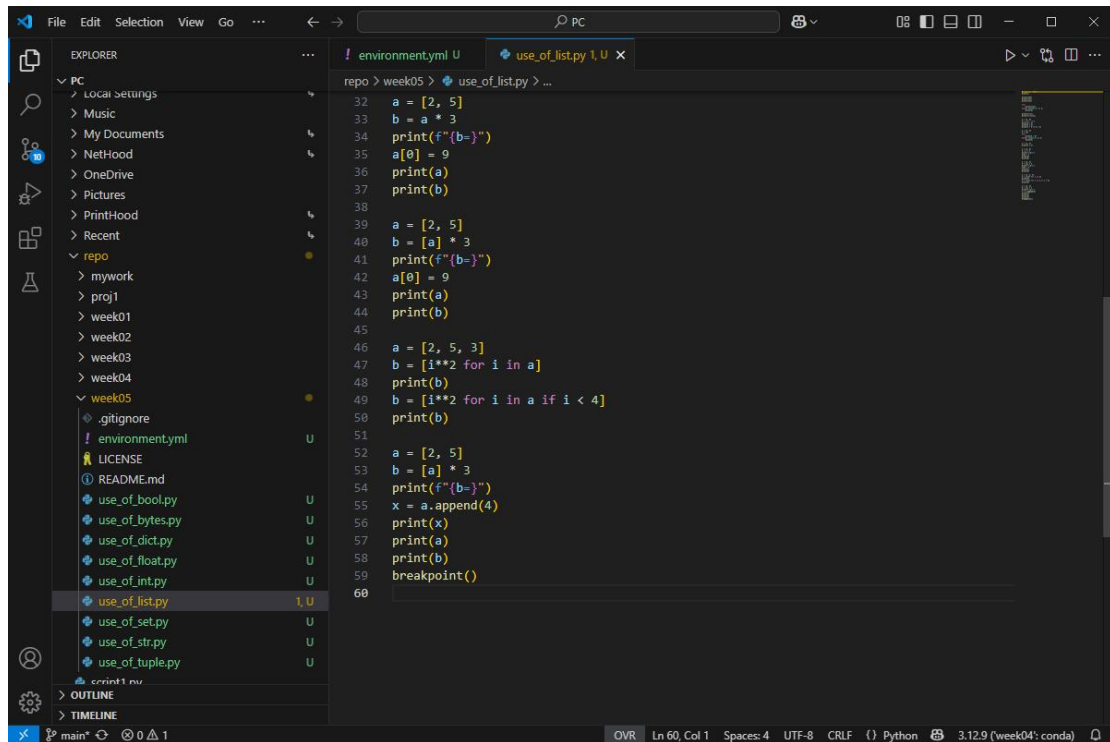
10. 列表



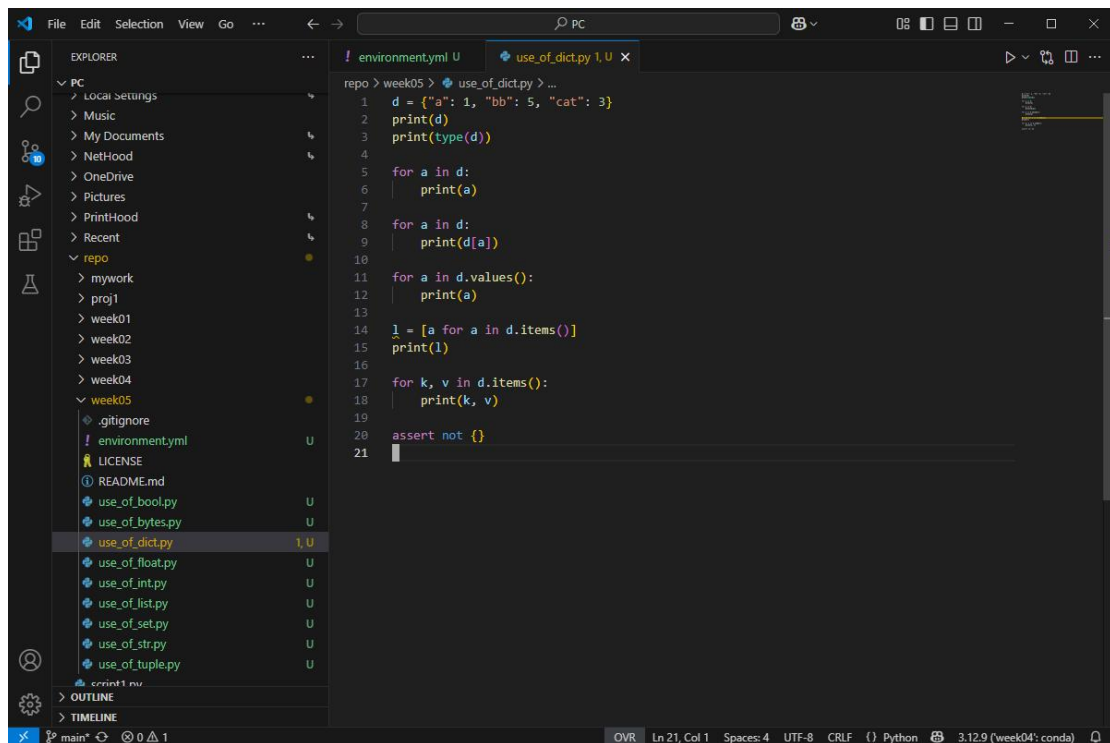
```

! environment.yml U
use_of_list.py 1, U X
repo > week05 > use_of_list.py > ...
1  l = (1, 5, "abc")
2  print(l)
3
4  print(l[0])
5  print(l[1])
6  print(l[2])
7
8  try:
9      print(l[3])
10 except IndexError as e:
11     print(e)
12
13 print(l[-1])
14 print(l[-1][1])
15
16 a = [2, 5]
17 b = ["a", "c"]
18 print(a + b)
19 print(b + a)
20 print(a + b == b + a)
21
22 a = [2, 5]
23 b = [5]
24 try:
25     print(a - b)
26 except TypeError as e:
27     print(e)
28
29 a = [2, 5]
30 print(a * 3)
31

```



11. 字典



```
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_dict.py
{'a': 1, 'bb': 5, 'cat': 3}
<class 'dict'>
a
bb
cat
1
5
3
1
5
3
[('a', 1), ('bb', 5), ('cat', 3)]
a 1
bb 5
cat 3
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$
```

12. 元组及与列表的区别

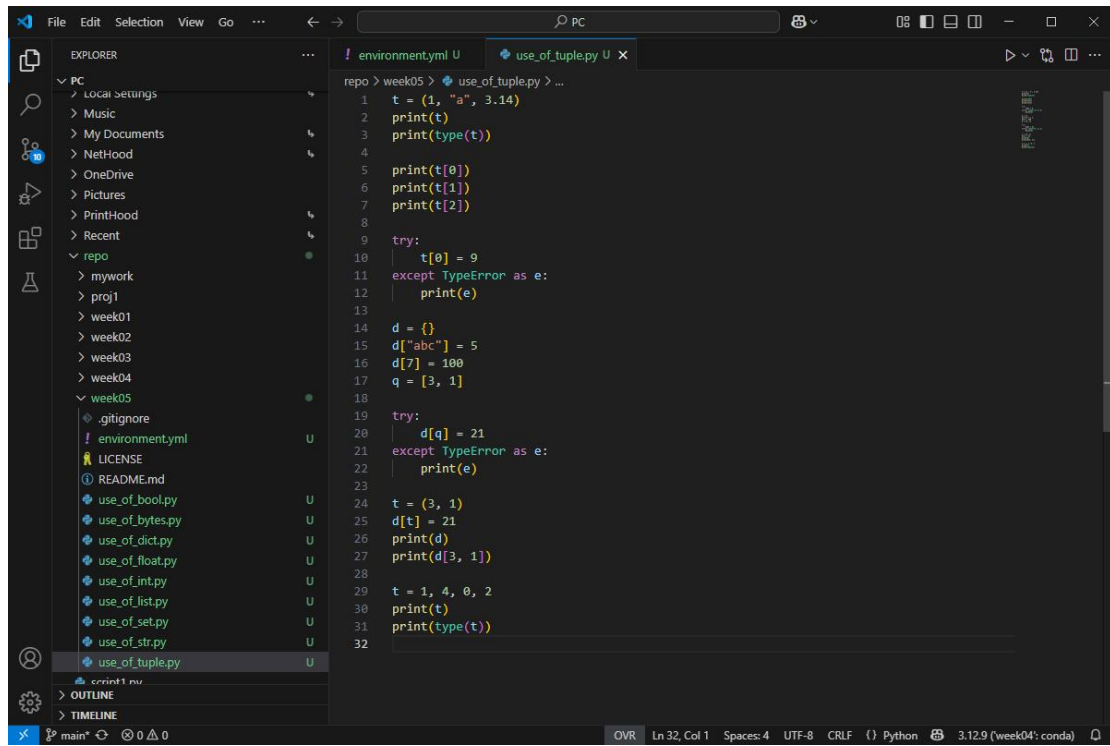
(1) 列表使用方括号 `[]` 来定义，元素之间用逗号分隔；元组使用圆括号 `()` 来定义，元素之间用逗号分隔。定义只有一个元素的元组时，需要在元素后面加逗号。

(2) 列表是可变对象，这意味着能够对其元素进行添加、删除、修改操作；元组是不可变对象，一旦创建，其元素就不能被修改、添加或删除。

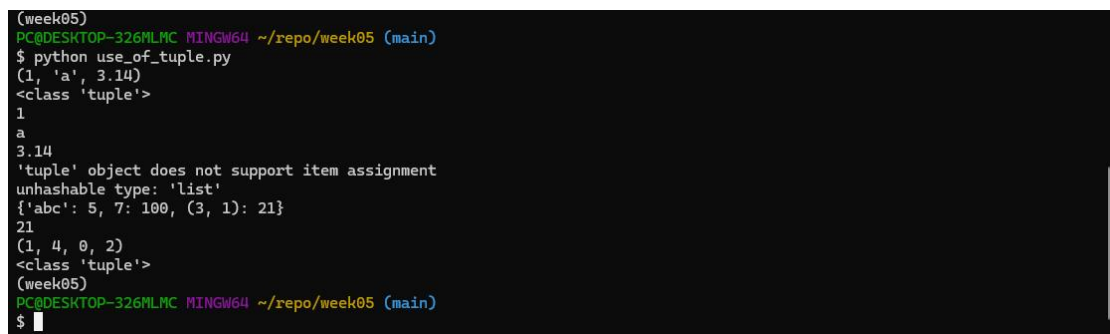
(3) 列表适用于需要动态修改元素的场景，像存储用户输入的一系列数据、动态更新的数据集等；元组适合存储不可变的数据，如坐标、数据库查询结果等，

也可用于函数返回多个值。

(4) 列表拥有较多的内置方法，如 `append()`、`extend()`、`remove()`、`sort()` 等，方便对列表进行操作；元组方法较少，只有 `count()` 和 `index()` 两个常用方法。

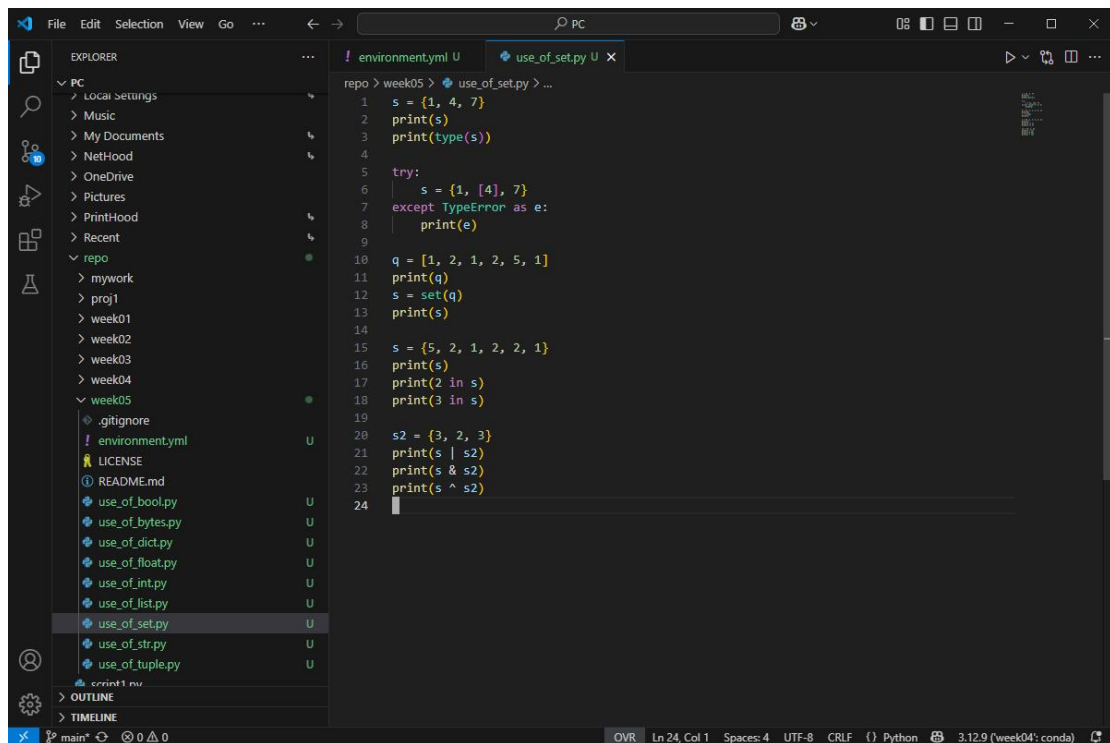


```
1 t = (1, "a", 3.14)
2 print(t)
3 print(type(t))
4
5 print(t[0])
6 print(t[1])
7 print(t[2])
8
9
10 try:
11     t[0] = 9
12 except TypeError as e:
13     print(e)
14
15 d = {}
16 d["abc"] = 5
17 d[7] = 100
18 q = [3, 1]
19
20 try:
21     d[q] = 21
22 except TypeError as e:
23     print(e)
24
25 t = (3, 1)
26 d[t] = 21
27 print(d)
28 print(d[3, 1])
29
30 t = 1, 4, 0, 2
31 print(t)
32 print(type(t))
```

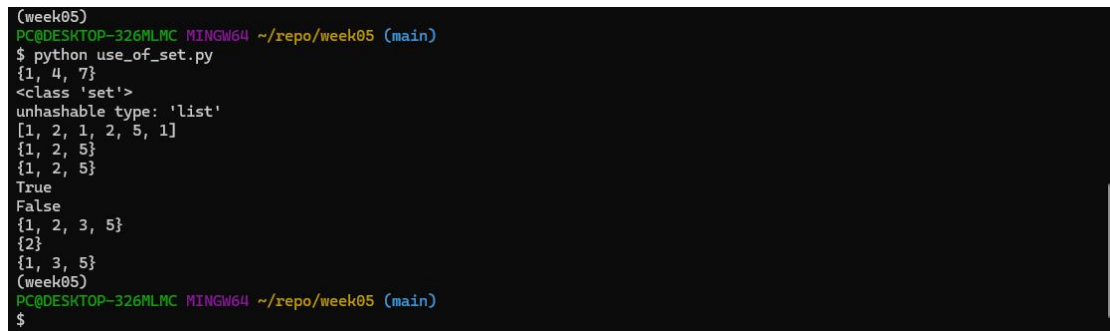


```
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_tuple.py
(1, 'a', 3.14)
<class 'tuple'>
1
a
3.14
'tuple' object does not support item assignment
unhashable type: 'list'
{'abc': 5, 7: 100, (3, 1): 21}
21
(1, 4, 0, 2)
<class 'tuple'>
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$
```

13. 集合

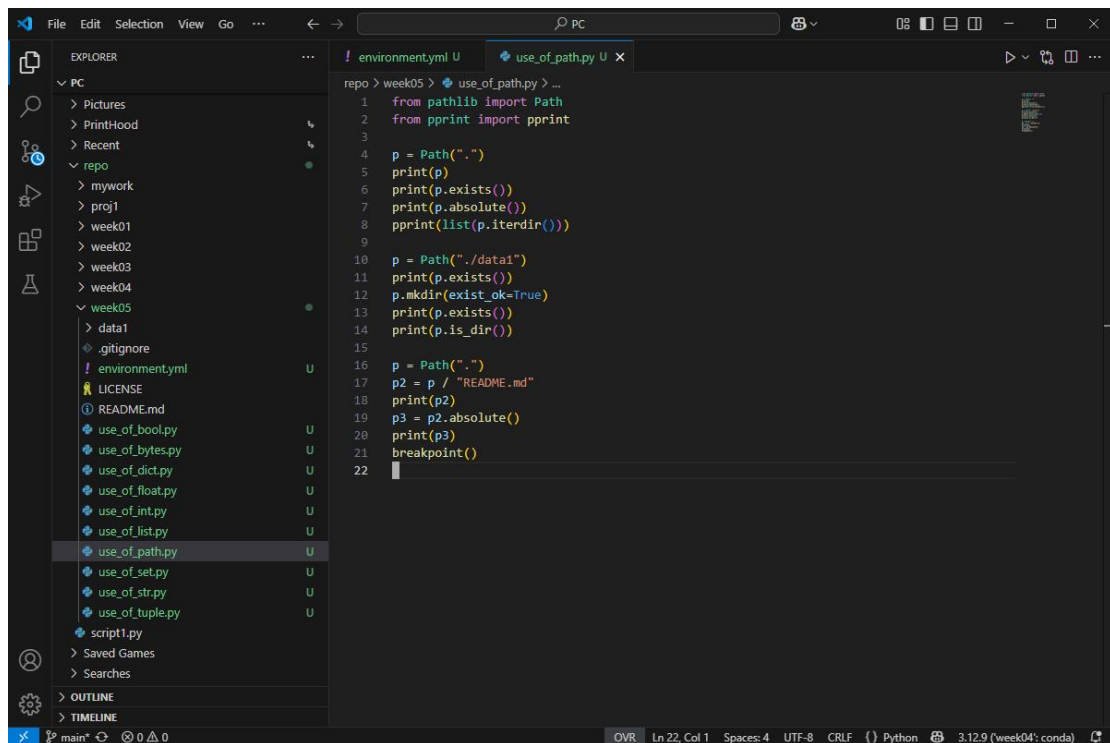


```
1 s = {1, 4, 7}
2 print(s)
3 print(type(s))
4
5 try:
6     s = {1, [4], 7}
7 except TypeError as e:
8     print(e)
9
10 q = [1, 2, 1, 2, 5, 1]
11 print(q)
12 s = set(q)
13 print(s)
14
15 s = {5, 2, 1, 2, 2, 1}
16 print(s)
17 print(2 in s)
18 print(3 in s)
19
20 s2 = {3, 2, 3}
21 print(s | s2)
22 print(s & s2)
23 print(s ^ s2)
24
```



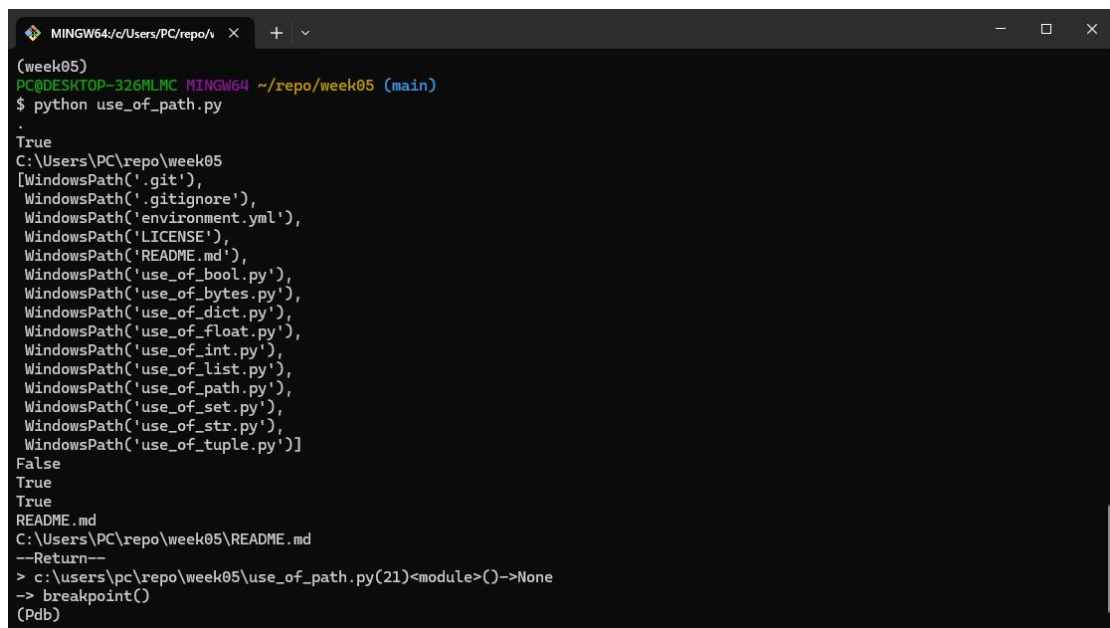
```
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_set.py
{1, 4, 7}
<class 'set'>
unhashable type: 'list'
[1, 2, 1, 2, 5, 1]
{1, 2, 5}
{1, 2, 5}
True
False
{1, 2, 3, 5}
{2}
{1, 3, 5}
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$
```

14. 路径



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a file tree for a project named 'repo'. The tree is expanded to show the 'week05' directory, which contains a 'data1' subdirectory and several Python files, including 'use_of_path.py'. The main editor window displays the contents of 'use_of_path.py'. The script uses the 'pathlib' module to create a Path object for the current directory, check for the existence of files, and list the contents of the 'data1' directory. It also demonstrates creating a new directory and checking its existence.

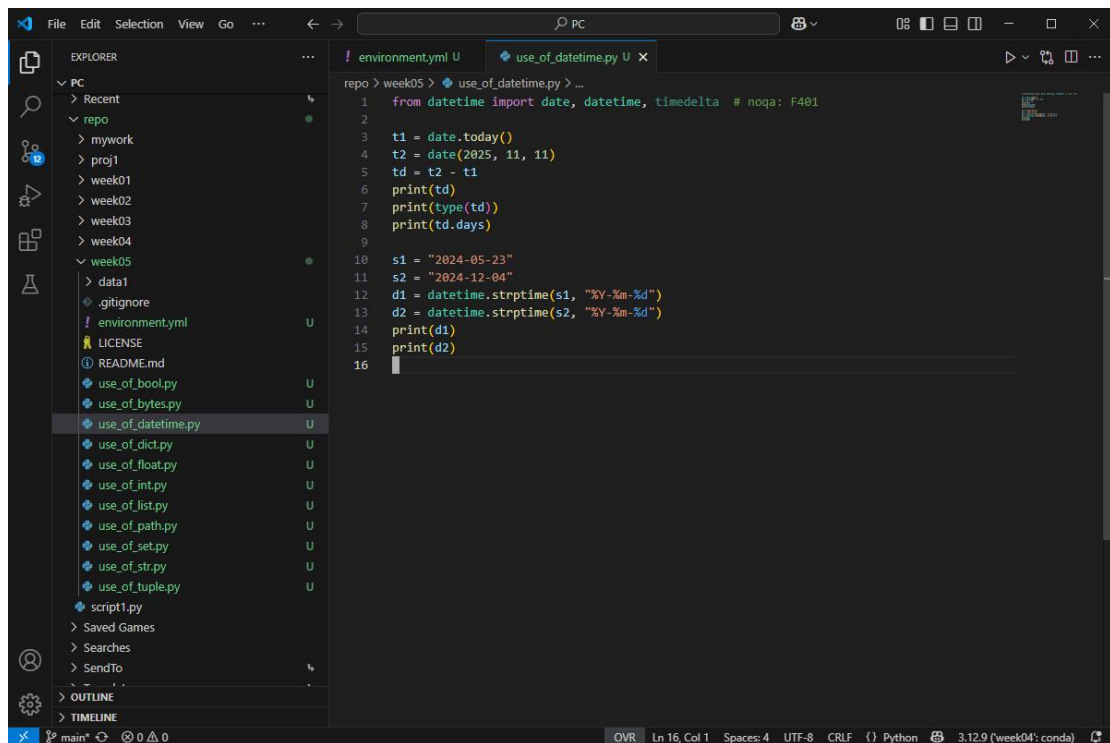
```
environment.yml U use_of_path.py U x
repo > week05 > use_of_path.py > ...
1 from pathlib import Path
2 from pprint import pprint
3
4 p = Path(".")
5 print(p)
6 print(p.exists())
7 print(p.absolute())
8 pprint(list(p.iterdir()))
9
10 p = Path("./data1")
11 print(p.exists())
12 p.mkdir(exist_ok=True)
13 print(p.exists())
14 print(p.is_dir())
15
16 p = Path(".")
17 p2 = p / "README.md"
18 print(p2)
19 p3 = p2.absolute()
20 print(p3)
21 breakpoint()
22
```



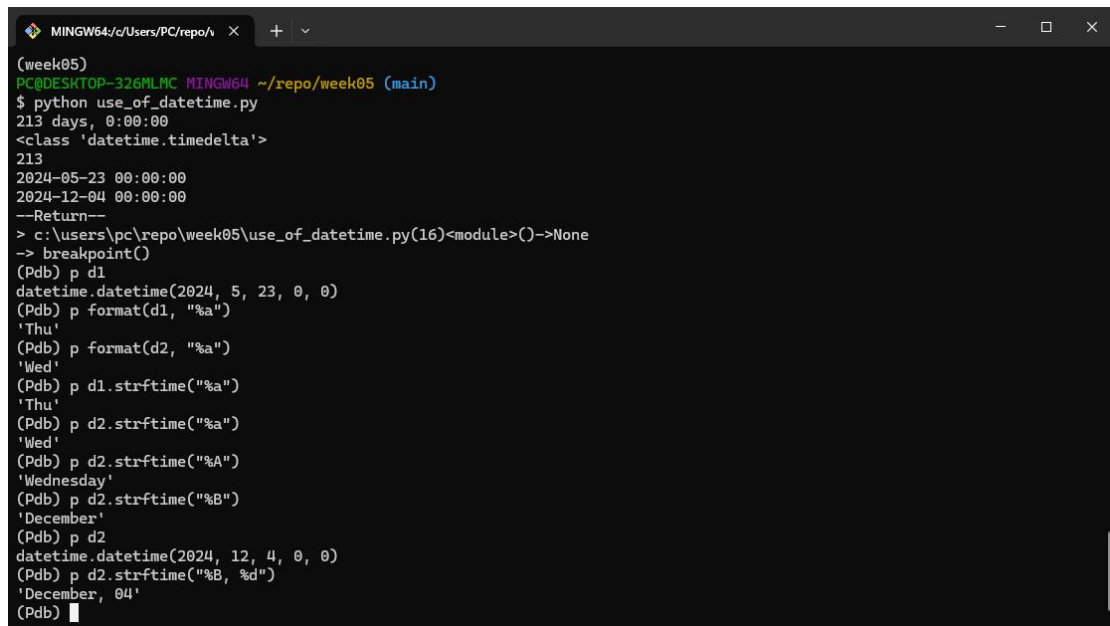
The screenshot shows a Windows command prompt window titled 'MINGW64/c/Users/PC/repo/'. The prompt is at the 'week05' directory. The user has run the command 'python use_of_path.py'. The output shows the script's execution, including the current directory path, a list of files in the directory, and the creation of a new directory 'data1'. The script then prints the absolute path of the 'README.md' file and hits a breakpoint.

```
MINGW64/c/Users/PC/repo/ X + v
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_path.py
.
True
C:\Users\PC\repo\week05
[WindowsPath('.git'),
 WindowsPath('.gitignore'),
 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')]
False
True
True
README.md
C:\Users\PC\repo\week05\README.md
--Return--
> c:\users\pc\repo\week05\use_of_path.py(21)<module>()-->None
-> breakpoint()
(Pdb)
```

15. 日期时间



```
1 from datetime import date, datetime, timedelta # noqa: F401
2
3 t1 = date.today()
4 t2 = date(2025, 11, 11)
5 td = t2 - t1
6 print(td)
7 print(type(td))
8 print(td.days)
9
10 s1 = "2024-05-23"
11 s2 = "2024-12-04"
12 d1 = datetime.strptime(s1, "%Y-%m-%d")
13 d2 = datetime.strptime(s2, "%Y-%m-%d")
14 print(d1)
15 print(d2)
16
```



```
(week05)
PC@DESKTOP-326MLMC MINGW64 ~/repo/week05 (main)
$ python use_of_datetime.py
213 days, 0:00:00
<class 'datetime.timedelta'>
213
2024-05-23 00:00:00
2024-12-04 00:00:00
--Return--
> c:\users\pc\repo\week05\use_of_datetime.py(16)<module>()->None
-> breakpoint()
(Pdb) p d1
datetime.datetime(2024, 5, 23, 0, 0)
(Pdb) p format(d1, "%a")
'Thu'
(Pdb) p format(d2, "%a")
'Wed'
(Pdb) p d1.strftime("%a")
'Thu'
(Pdb) p d2.strftime("%a")
'Wed'
(Pdb) p d2.strftime("%A")
'Wednesday'
(Pdb) p d2.strftime("%B")
'December'
(Pdb) p d2
datetime.datetime(2024, 12, 4, 0, 0)
(Pdb) p d2.strftime("%B, %d")
'December, 04'
(Pdb)
```