

创建 week05

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repopo
$ ls -l
total 8
drwxr-xr-x 1 yangzihan 197121 0 3月 23 12:57 myproject/
drwxr-xr-x 1 yangzihan 197121 0 3月 28 13:04 week04/
drwxr-xr-x 1 yangzihan 197121 0 4月 14 13:58 week05/

(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repopo
$
```

复制 week04 environment.yml 到 week05

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repopo
$ cp week04/environment.yml week05/
```

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repopo
$ ls -l week05
total 25
-rw-r--r-- 1 yangzihan 197121 91 4月 14 14:02 environment.yml
-rw-r--r-- 1 yangzihan 197121 18805 4月 14 13:58 LICENSE
-rw-r--r-- 1 yangzihan 197121 2239 4月 14 13:58 README.md
```

Week05 下创建 conda 环境

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repopo
$ cd week05

(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repopo/week05 (main)
$ conda env create
D:\Anaconda\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
 - https://repo.anaconda.com/pkg/main
 - https://repo.anaconda.com/pkg/r
 - https://repo.anaconda.com/pkg/msys2
Platform: win-64
Collecting package metadata (repodata.json): |
```

```
! environment.yml U X use_of_s

! environment.yml
1 name: week05
2 channels:
3 - conda-forge
4 dependencies:
5 - python=3.12
6 - wat-inspector
```

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)

```
(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ conda activate week05
(yweek05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
hello
(yweek05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
1979565759520
(yweek05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$
(yweek05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
2534404676640
(yweek05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ a
```

```
use_of_str.py > ...
1  a = "hello"
2  b = "hello"
3  x = id(a)
4  print(x)
5  y = id(b)
6  print(y)
7
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
2262350650400
2262350650400
(yweek05)
```

```
7  a = [2, 5]
8  b = [2, 5]
9  x = id(a)
10 print(x)
11 y = id(b)
12 print(y)
```

```
(yweek05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
2533228239904
2533228239904
2533226125568
2533226123584
(yweek05)
```

```
2817912019200
2817912017216
[9, 5]
[2, 5]
(week05)
```

type() -- 返回对象的类型

```
18 print(type(a))
19
```

```
2817912019200
2817912017216
[9, 5]
[2, 5]
2091835070720
2091835068736
<class 'list'>
(week05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$
```

isinstance() -- 判断对象是否属于某个 (或某些) 类型

```
print(id(a))
print(id(b))
print(type(a))
print("isinstance(a, str): ", isinstance(a, str))
```

```
$ 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 inform
ation.
>>> print(1, 2, 6, 10)
1 2 6 10
>>>
```

```
19 print("isinstance(a, str): ", isinstance(a, str))
20 print("isinstance(a, list): ", isinstance(a, list))
```

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

```
20 print("isinstance(a, list): ", is
21 print("dir(a):", dir(a))
```

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

```
19 print("isinstance(a, str): ", isinstance(a, str))
20 print("isinstance(a, list): ", isinstance(a, list))
21 print(isinstance(a, (str, float)))
22 assert isinstance(a, str)
23 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

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

```
23 try:
24     assert isinstance(a, str)
25 except AssertionError:
26     breakpoint()
27     print("type error")
```

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)

```
30 print("字面值")
31 s = "university"
32 print(s)
33 print(isinstance(s, str))
34 assert type(s) is str
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
字面值
university
True
(week05)
```

```
6 print("f-string")
7 x = "Tom"
8 s = f"name: {x}"
9 print(s)
10
11 s = "a\tb"
12 print("TAB", s)
13
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name: Tom
TAB a    b
(week05)
```

```
11 s = "a\tb"
12 print("TAB", s)
13 s = "aaa\nbbb"
14 print("New Line", s)
15
```

```
$ python use_of_str.py
字面值
university
True
f-string
name: Tom
TAB a    b
New Line aaa
bbb
(week05)
```

```
15 s = """xyz
16 abc
17 |   eee
18 aaa
19 """
20 print(s)
21 |
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/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)
```

初始化 (init)

```
21
22 print("初始化")
23 s = str
24 print(s)
25 s = str([5, 8, 2])
26 print(s)
27 |
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/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)
```



```

(week05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/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]
Traceback (most recent call last):
  File "C:\Users\yangzihan\repoo\week05\use_of_str.py", line 28, in <module>
    assert str(1.1 + 2.2) == "3.3"
           ^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError
(week05)

```

运算符 (operator)

```

29 s = "="
30 x = id(s)
31 s = s * 20
32 y = id(s)
33 print(s)
34 assert x != y
35
36 s = "hello"
37 assert s[3] == "l"
38 assert s[-1] == "o"
39 assert s[:3] == "hel"
40 assert s[4] == s[-1]
41 try:
42     s[5]
43 except IndexError as e:
44     print(e)
45

```

返回值

```

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

```

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

```

46 s1 = "abc"
47 s2 = "ghi"
48 s = s1 + s2
49 assert s == "abcghi"
50 print(s2 + s1)
51
52 try:
53     print(s2 + s1)
54 except TypeError as e:
55     print(e)
56
57 s = "aaaa"
58 try:
59     s = s / 2
60 except TypeError as e:
61     print(e)
62

```

如何判断相等 (==)

```

62 assert s == "aaaa"
63

```

对于比较运算符 (>、=、<=) 有没有支持  
什么值被当作 True, 什么值被当作 False

```

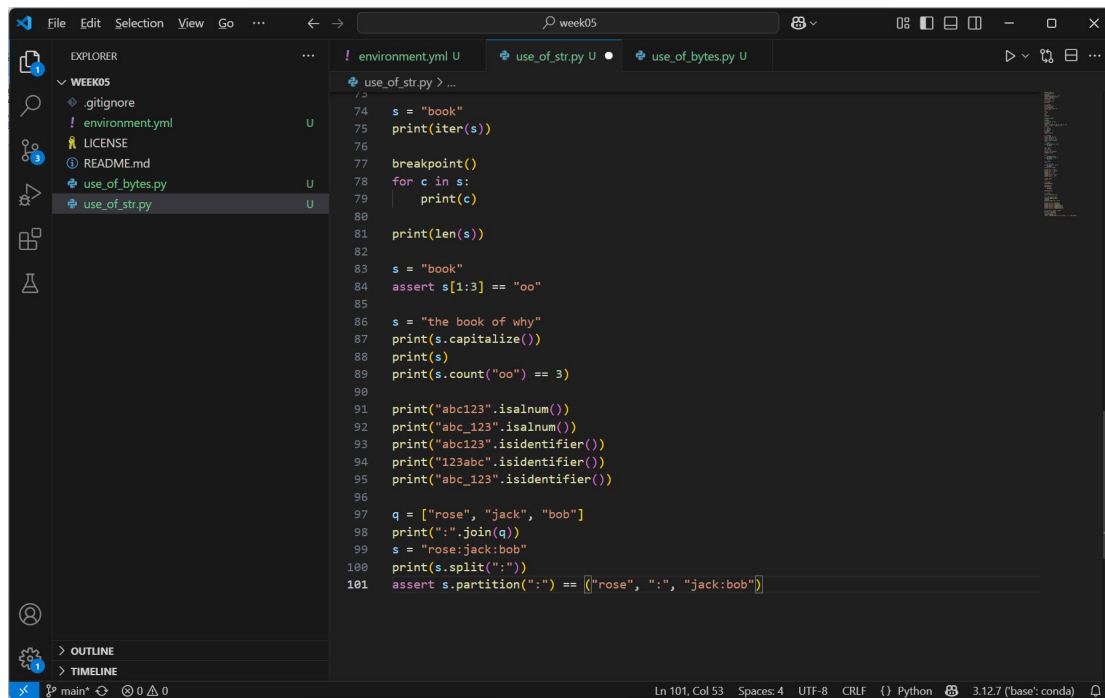
64 print("abc" > "ABC")
65 print("123" > "abc")
66 print("9" > ".")
67 print("9" > ":")
68 print("book" > "box")
69 print("book" > "{")
70

```

是否支持返回长度 (len)

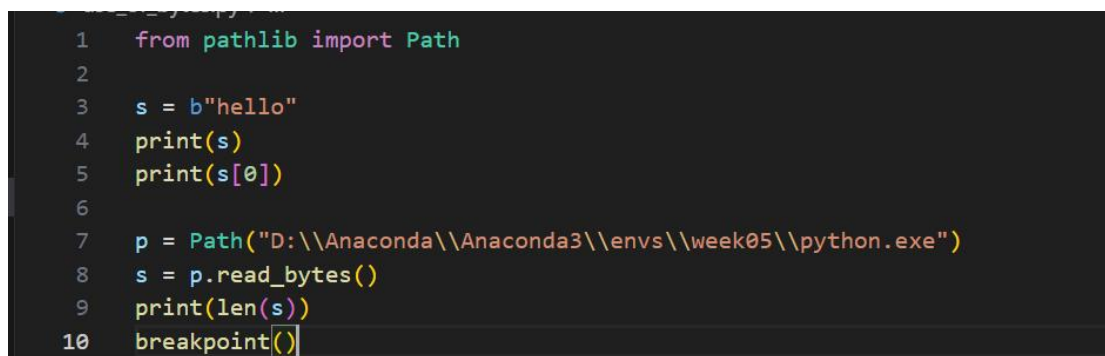
是否 (如何) 支持索引操作 (subscription) ([]) 运算符

拥有哪些常用方法 (method) 可供调用 (()) 运算符



```
74 s = "book"
75 print(iter(s))
76
77 breakpoint()
78 for c in s:
79     print(c)
80
81 print(len(s))
82
83 s = "book"
84 assert s[1:3] == "oo"
85
86 s = "the book of why"
87 print(s.capitalize())
88 print(s)
89 print(s.count("oo") == 3)
90
91 print("abc123".isalnum())
92 print("abc_123".isalnum())
93 print("abc123".isidentifier())
94 print("123abc".isidentifier())
95 print("abc_123".isidentifier())
96
97 q = ["rose", "jack", "bob"]
98 print(":".join(q))
99 s = "rose:jack:bob"
100 print(s.split(":"))
101 assert s.partition(":") == ("rose", ":", "jack:bob")
```

逐个创建 `use_of_{name}.py` 文件，例如 `use_of_bytes.py`：  
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()
```

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://www.ascii-code.com>，为其中一种



```
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("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  b1 = s.encode("utf-8")
23  breakpoint()
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_bytes.py
b'hello'
```

创建 use\_of\_int.py

```
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 # 整除
9  assert y % x == 2 # 余除
10
11  assert 5
12
13  try:
14      assert 0
15  except AssertionError as e:
16      print(type(e))
17
18  x = 3
19  breakpoint()
20
```

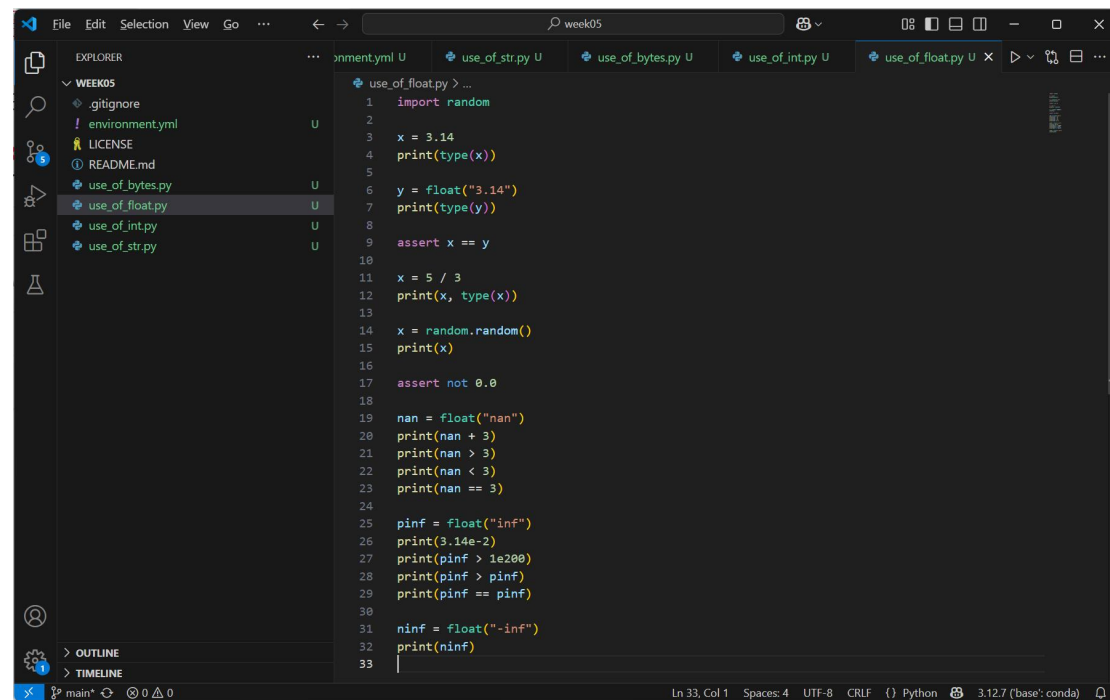
```
(week05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_int.py
```

```

(Pdb) for i in x:
*** IndentationError: expected an indented block after 'for' sta
tatement 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)

```

创建 use\_of\_float.py:



```

File Edit Selection View Go ... week05
EXPLORER
WEEK05
  .gitignore
  ! environment.yml
  LICENSE
  README.md
  use_of_bytes.py
  use_of_float.py
  use_of_int.py
  use_of_str.py
  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
Ln 33, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.7 (base: conda)

```

创建 use\_of\_bool.py

The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays a project structure for 'WEEK05' with files: .gitignore, environment.yml, LICENSE, README.md, use\_of\_bool.py, use\_of\_bytes.py, use\_of\_float.py, use\_of\_int.py, and use\_of\_str.py. The file 'use\_of\_bool.py' is selected and open in the editor. The code in the editor is as follows:

```
1 t = True
2 f = False
3 print(t, f)
4
5 print(type(t))
6 print(isinstance(t, int))
7
```

The status bar at the bottom indicates 'Ln 7, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.12.7 (base: conda)'.

```
yangzihan@LAPTOP-B9DHBGEG MINGW64 ~/repor/week05 (main)
$ python use_of_bool.py
True False
<class 'bool'>
True
(week05)
```

创建 use\_of\_list.py

The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the same project structure as the previous image, with 'use\_of\_list.py' now added to the list of files. The file 'use\_of\_list.py' is selected and open in the editor. The code in the editor is as follows:

```
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(b + a == a + b)
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
32 a = [2, 5]
33 b = a * 3
34 print(c=f'h={a}')
```

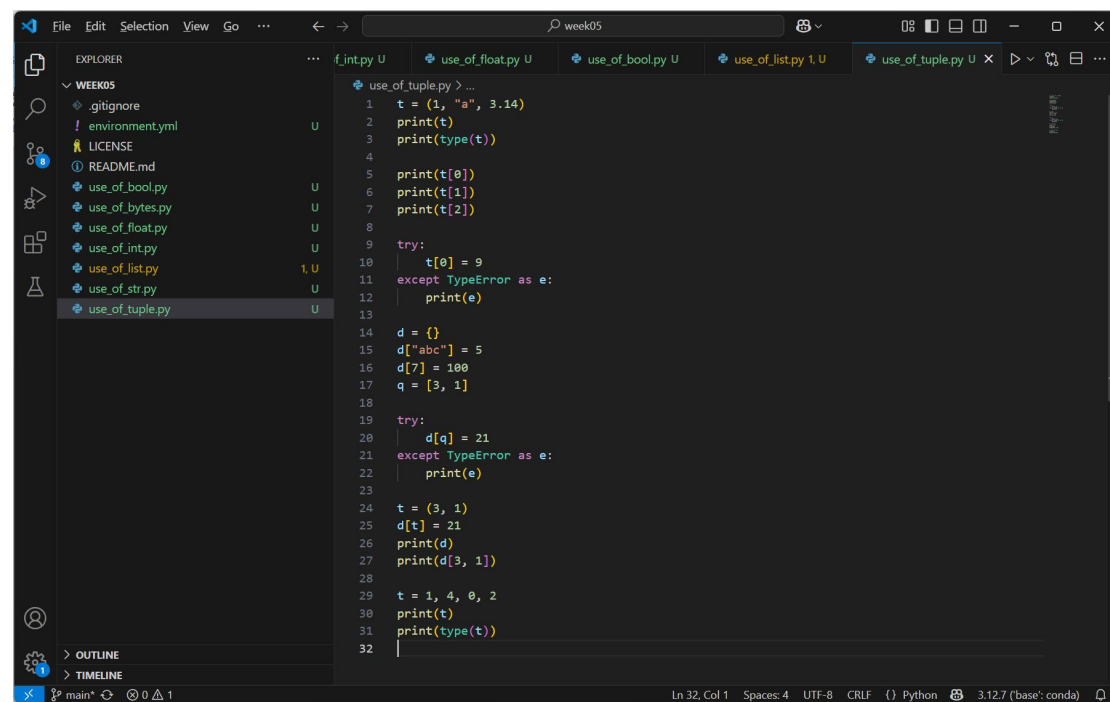
The status bar at the bottom indicates 'Ln 27, Col 13', 'Spaces: 4', 'UTF-8', 'CRLF', 'Python', and '3.12.7 (base: conda)'.

```

$ 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
[2, 5] [5]
[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]]
(week05)

```

创建 use\_of\_tuple.py



```

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

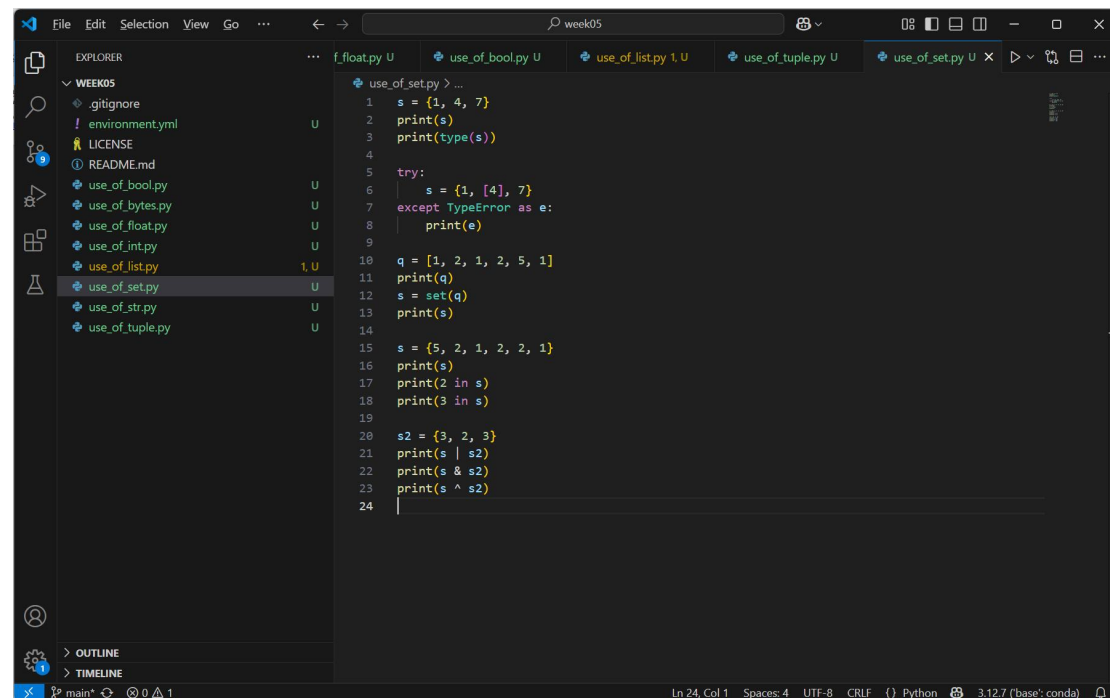
```

```

$ 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)

```

创建 use\_of\_set.py



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 named 'WEEK05' with several files, including 'use\_of\_set.py'. The code editor shows the content of 'use\_of\_set.py'.

```

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

```

```

$ 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)

```

use\_of\_path.py:

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

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_path.py
.
True
C:\Users\yangzihan\repoo\week05
```



```

[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
    depending 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

```

创建 use\_of\_datetime.py

```
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 = "2025-05-23"
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 breakpoint()
17
```

```
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_datetime.py
```

```
value: <class 'datetime.date'>
type: type
signature: class date(...)
    "date(year, month, day) --> date object""

Public attributes:
  day: getset_descriptor = <attribute 'day' of 'datetime.date' o
objects>
  max: datetime.date = 9999-12-31
  min: datetime.date = 0001-01-01
  month: getset_descriptor = <attribute 'month' of 'datetime.dat
e' objects>
  resolution: datetime.timedelta = 1 day, 0:00:00
  year: getset_descriptor = <attribute 'year' of 'datetime.date'
objects>

  def ctime(...) # Return ctime() style string.
  def fromisocalendar(...) # int, int, int -> Construct a date fro
the ISO year, week number and weekday....
  def fromisoformat(...) # str -> Construct a date from a string i
ISO 8601 format.
  def fromordinal(...) # int -> date corresponding to a proleptic
gregorian ordinal.
  def fromtimestamp(timestamp, /) # Create a date from a POSIX t
imestamp....
  def isocalendar(...) # Return a named tuple containing ISO year,
week number, and weekday.
  def isoformat(...) # Return string in ISO 8601 format, YYYY-MM-D
.
  def isoweekday(...) # Return the day of the week represented by
the date....
  def replace(...) # Return date with new specified fields.
  def strftime(...) # format -> strftime() style string.
  def timetuple(...) # Return time tuple, compatible with time.loc
```