创建 week05

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

复制 week04 environment.yml 到 week05

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

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

(base) yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ conda env create
D:\Anaconda\Lib\argparse.py:2006: FutureWarning: `remote_definition` is deprecated and will be removed in 25.9. Use `con
da env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - 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): |
```

```
! environment.yml U ×   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
(week05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_str.py
hello
(week05)
```

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

use_of_str.py > ...

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

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

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

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

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

type() -- 返回对象的类型

```
print(type(a))
```

```
[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
>>>
```

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

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

```
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

```
print("isinstance(a, str): ", isinstance(a, str))
print("isinstance(a, list): ", isinstance(a, list))
print(isinstance(a, (str, float)))
assert isinstance(a, str)
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

```
try:
    assert isinstance(a, str)
except AssertionError:
    breakpoint()
    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)

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

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

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

s = "a\tb"
print("TAB", s)

```

```
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)
```

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

```

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

```
s = """xyz
abc
  eee
aaa
"""
print(s)
```

```
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)

```
print("初始化")
s = str
print(s)
s = str([5, 8, 2])
print(s)
```

```
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)
```

运算值 (operator)

```
s = "="
x = id(s)
s = s * 20
y = id(s)
print(s)
assert x != y

```

```
s = "hello"
assert s[3] == "1"
assert s[-1] == "o"
assert s[:3] == "hel"
assert s[4] == s[-1]
try:
    s[5]
except IndexError as e:
    print(e)

```

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

```python
s1 = "abc"
s2 = "ghi"
s = s1 + s2
assert s == "abcghi"
print(s2 + s1)

try:
    print(s2 + s1)
except TypeError as e:
    print(e)

s = "aaaa"
try:
    s = s / 2
except TypeError as e:
    print(e)

```

如何判断相等 (==)

```python
assert s == "aaaa"

```

对于比较运算符 (>、=、<=) 有没有支持

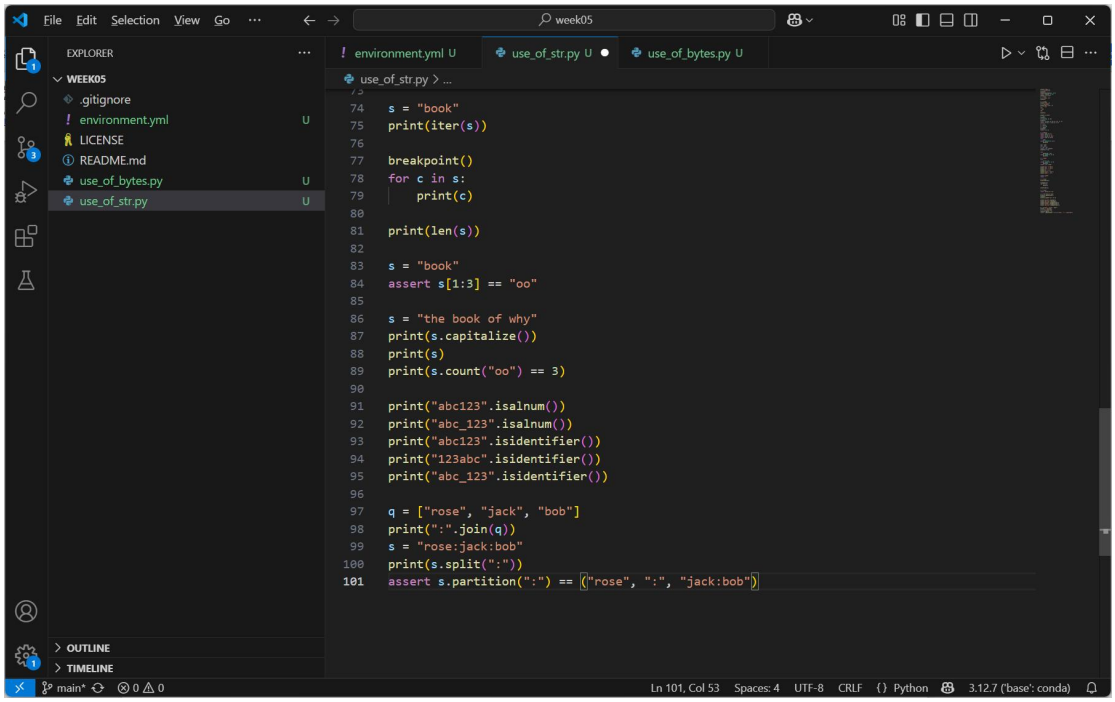
什么值被当作 True，什么值被当作 False

```python
print("abc" > "ABC")
print("123" > "abc")
print("9" > ".")
print("9" > ":")
print("book" > "box")
print("book" > "{")

```

是否支持返回长度 (len)

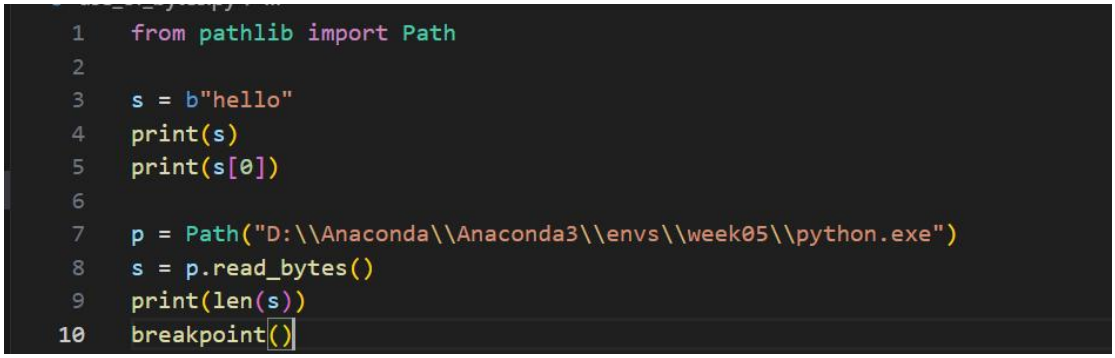
是否 (如何) 支持索引操作 (subscription) ([] 运算符)

拥有哪些常用方法 (method) 可供调用 (() 运算符)



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

字节和返回长度 (len)



```
from pathlib import Path

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

p = Path("D:\\Anaconda\\Anaconda3\\envs\\week05\\python.exe")
s = p.read_bytes()
print(len(s))
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://wwwh.ascii-code.com，为其中一种

use_of_bytes.py > ...

```python
from pathlib import Path

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

p = Path("D:\\Anaconda\\Anaconda3\\envs\\week05\\python.exe")
s = p.read_bytes()
print(len(s))

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

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")
breakpoint()
```

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

创建 use_of_int.py

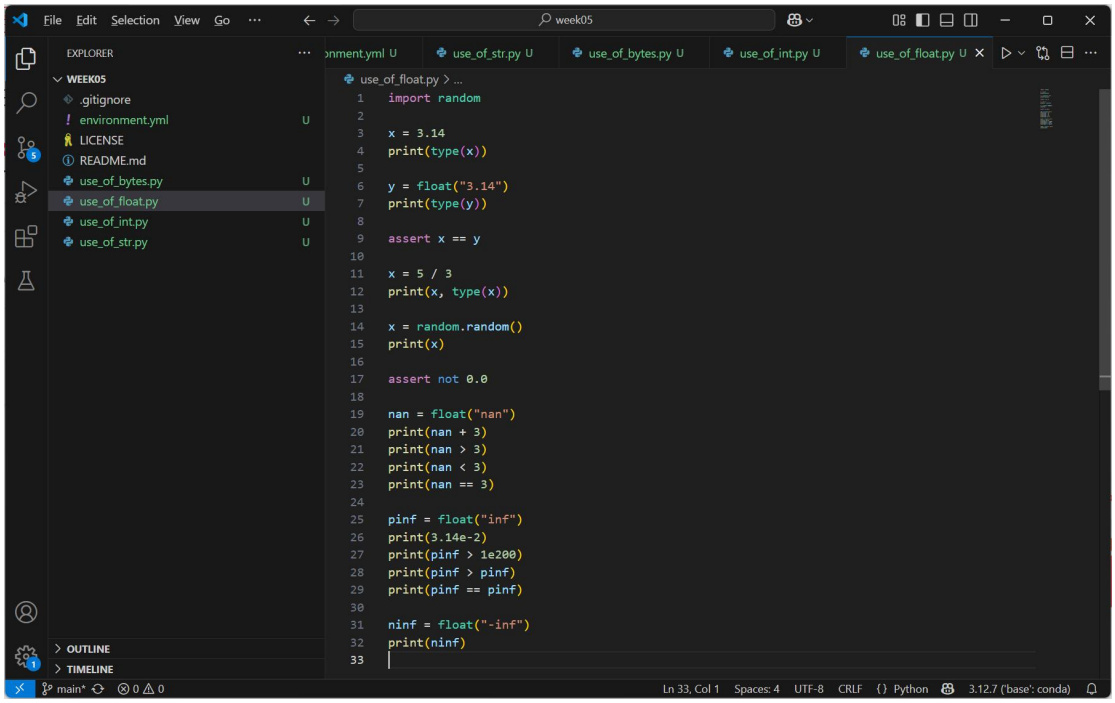
use_of_int.py > ...

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

x = 5
y = 17
assert y // x == 3  # 整除
assert y % x == 2  # 余除

assert 5

try:
    assert 0
except AssertionError as e:
    print(type(e))

x = 3
breakpoint()
```

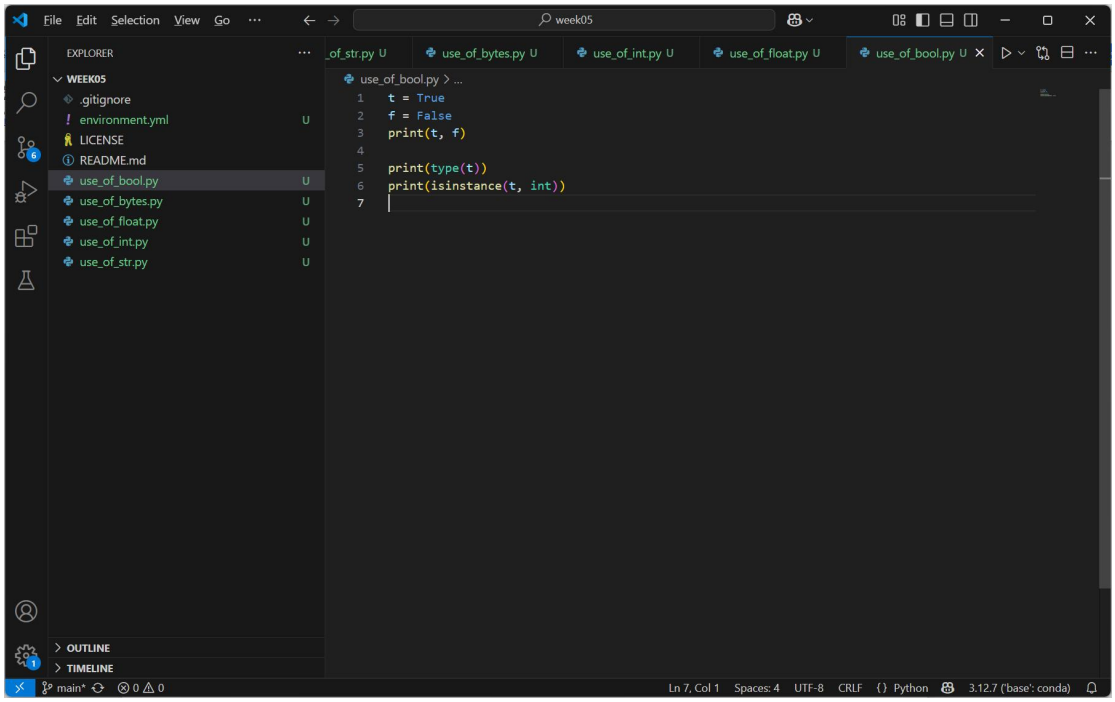
```
(week05)
yangzihan@LAPTOP-B9DHBGED MINGW64 ~/repoo/week05 (main)
$ python use_of_int.py
```

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

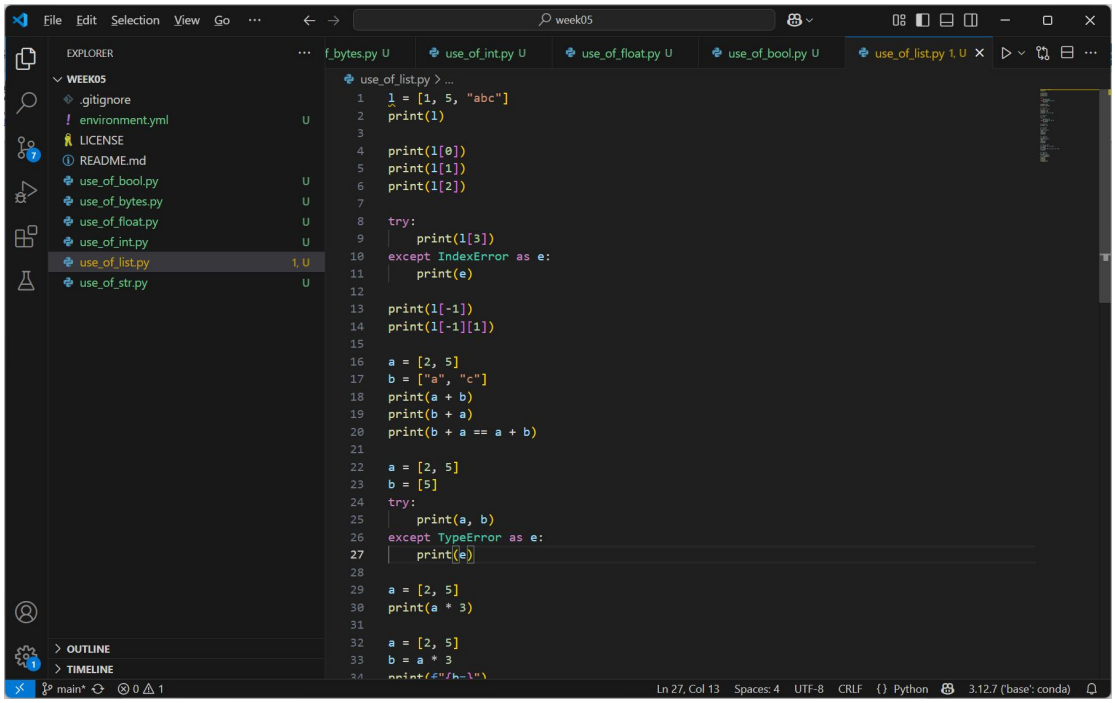
创建 use_of_float.py:



创建 use_of_bool.py

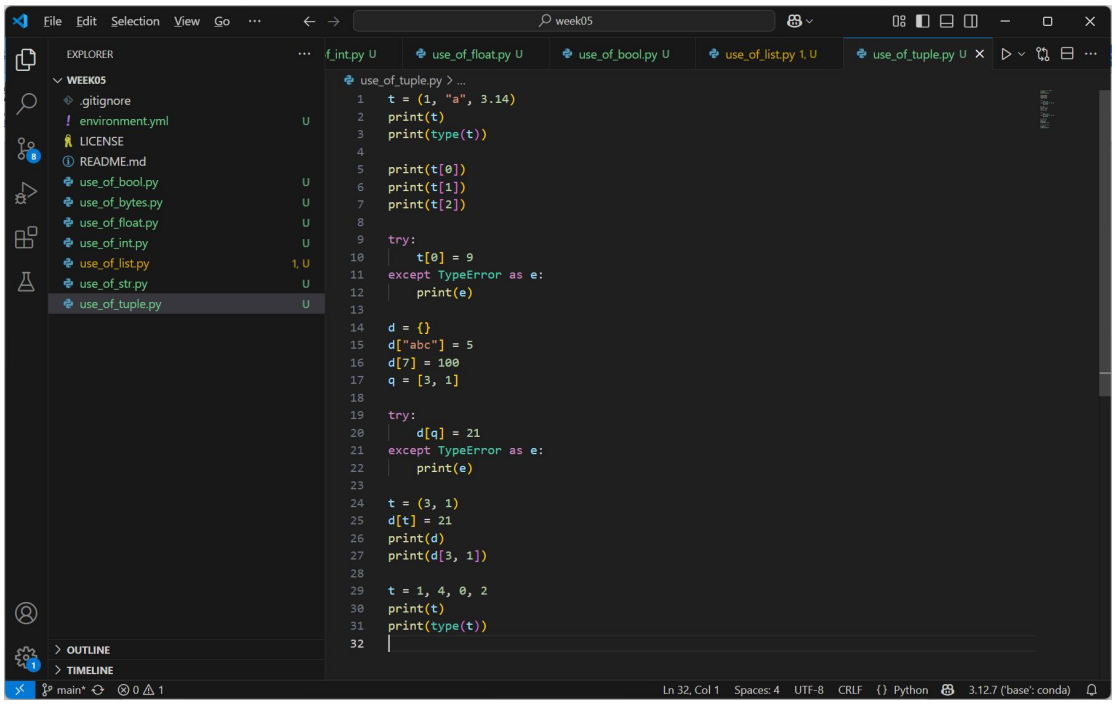


创建 use_of_list.py



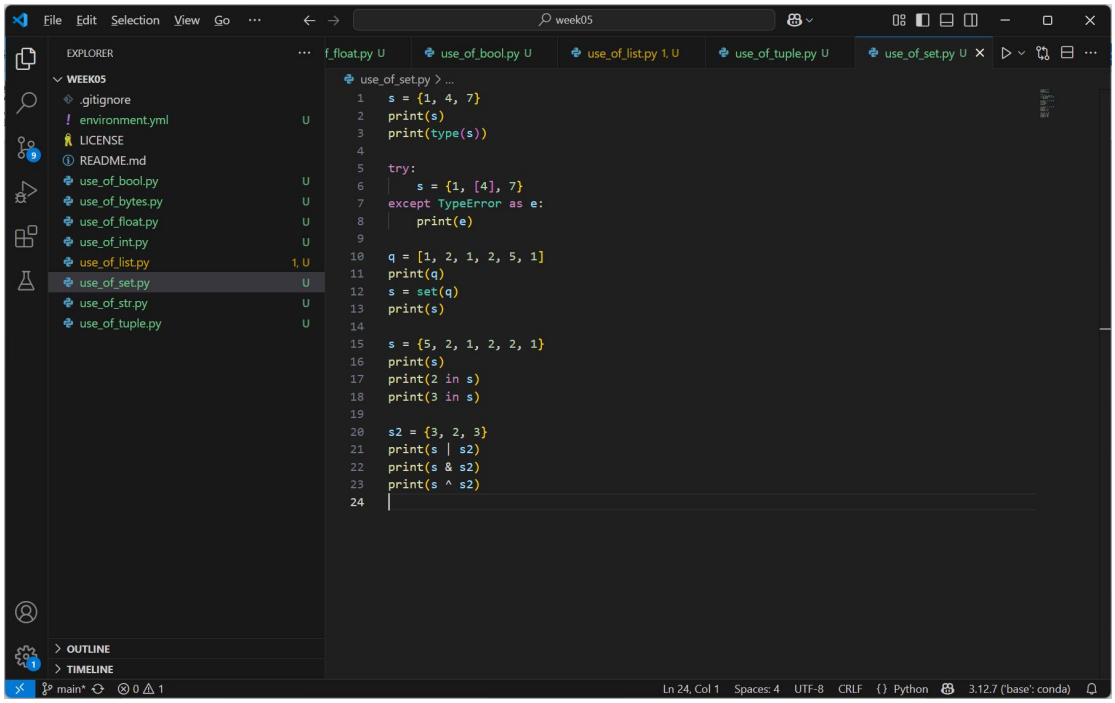
```
$ python use_of_list.py
[1, 5, 'abc']
1
5
abc
list index out of range
abc
b
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
False
[2, 5] [5]
[2, 5, 2, 5, 2, 5]
b=[2, 5, 2, 5, 2, 5]
[9, 5]
[2, 5, 2, 5, 2, 5]
b=[[2, 5], [2, 5], [2, 5]]
[9, 5]
[[9, 5], [9, 5], [9, 5]]
(week05)
```

创建 use_of_tuple.py



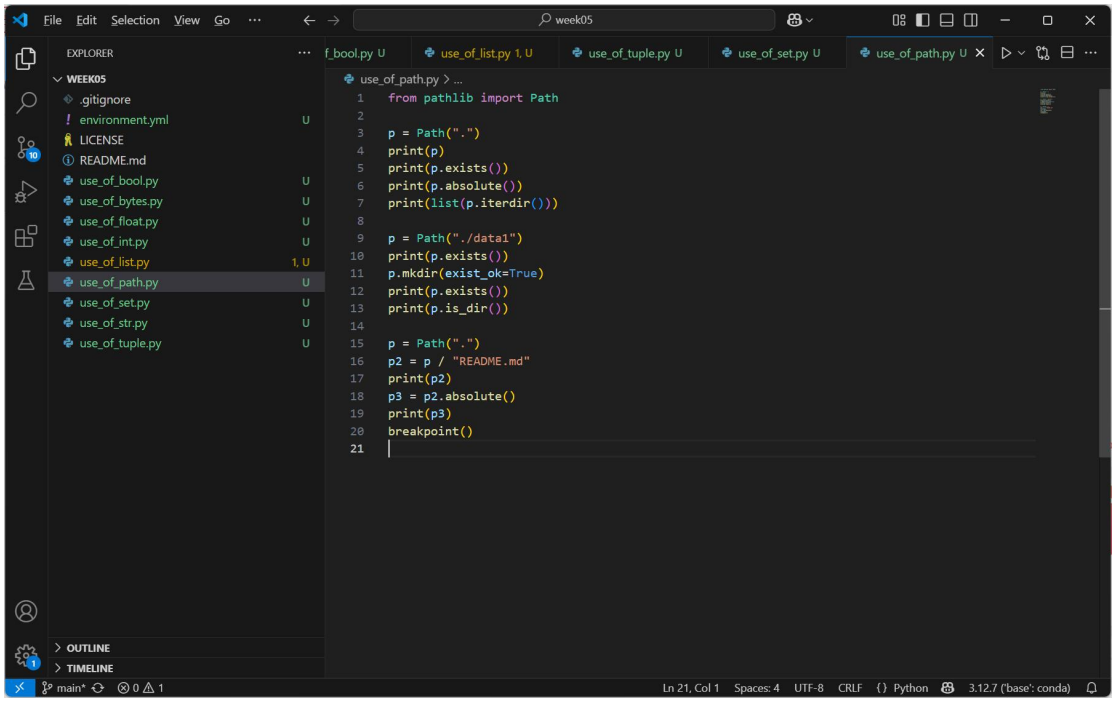
```
$ python use_of_tuple.py
(1, 'a', 3.14)
<class 'tuple'>
1
a
3.14
'tuple' object does not support item assignment
unhashable type: 'list'
{'abc': 5, 7: 100, (3, 1): 21}
21
(1, 4, 0, 2)
<class 'tuple'>
(week05)
```

创建 use_of_set.py




```
$ 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:


File Edit Selection View Go
week05
EXPLORER
WEEK05
.gitignore
environment.yml U
LICENSE
README.md
use_of_bool.py U
use_of_bytes.py U
use_of_float.py U
use_of_int.py U
use_of_list.py 1, U
use_of_path.py U
use_of_set.py U
use_of_str.py U
use_of_tuple.py U
use_of_path.py > ...
from pathlib import Path
p = Path(".")
print(p)
print(p.exists())
print(p.absolute())
print(list(p.iterdir()))
p = Path("./data1")
print(p.exists())
p.mkdir(exist_ok=True)
print(p.exists())
print(p.is_dir())
p = Path(".")
p2 = p / "README.md"
print(p2)
p3 = p2.absolute()
print(p3)
breakpoint()
OUTLINE
TIMELINE
main*
Ln 21, Col 1 Spaces: 4 UTF-8 CRLF Python 3.12.7 ('base': conda)


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
epending 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

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

t1 = date.today()
t2 = date(2025, 11, 11)
td = t2 - t1
print(td)
print(type(td))
print(td.days)

s1 = "2025-05-23"
s2 = "2024-12-04"
d1 = datetime.strptime(s1, "%Y-%m-%d")
d2 = datetime.strptime(s2, "%Y-%m-%d")
print(d1)
print(d2)
breakpoint()
```

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

```
alue: <class 'datetime.date'>
ype: type
ignature: class date(…)
"""date(year, month, day) --> date object"""

ublic attributes:
 day: getset_descriptor = <attribute 'day' of 'datetime.date' o
jects>
 max: datetime.date = 9999-12-31
 min: datetime.date = 0001-01-01
 month: getset_descriptor = <attribute 'month' of 'datetime.dat
' 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
regorian ordinal.
 def fromtimestamp(timestamp, /) # Create a date from a POSIX t
mestamp.…
 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
he date.…
 def replace(…) # Return date with new specified fields.
 def strftime(…) # format -> strftime() style string.
 def timetuple(…) # Return time tuple, compatible with time.loc
```