第五周学习报告

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

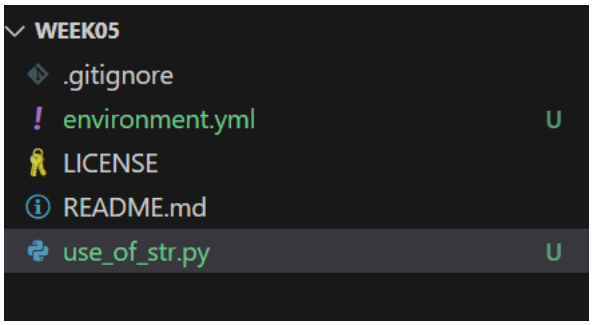
2. 用 VS Code 打开项目目录，新建一个 environment.yml 文件，指定安装 Python 3.12，然后运行 conda env create 命令创建 Conda 环境

复制第四周 environment.yml 至第五周，创建环境。

```
(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo
$ cat week04/environment.yml
name: week04
channels:
  - conda-forge
dependencies:
  - python=3.12
  - wat-inspector
(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo
$ cp week04/environment.yml week05/

(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo
$ ls -l week05
total 25
-rw-r--r-- 1 74567 197609    91  4月  7 08:29 environment.yml
-rw-r--r-- 1 74567 197609 18805  4月  7 08:26 LICENSE
-rw-r--r-- 1 74567 197609  2239  4月  7 08:26 README.md
```

```
(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo
$ cd week05

(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ conda env create
D:\Anaconda\Lib\argparse.py:2006: FutureWarning: `remote_definition` is deprecated and wi
ll be removed in 25.9. Use `conda env create --file=URL` instead.
  action(self, namespace, argument_values, option_string)
Retrieving notices: ...working... done

CondaValueError: prefix already exists: D:\Anaconda\envs\week04
```

3. 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，例如 use_of_str.py:



(1) 在全局作用域（global scope）内尝试键入（活学活用）Python 代码，亲手验证概念（Proof of Concept，PoC）

(2) 对于任何对象，都可以传给以下内置函数（built-in function）用于检视（inspect）：

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）

```
a = "hello"
x = id(a)
print(x)
```

```
(base) 74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ conda activate week05
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1810933439664
(week05)
```

use_of_str.py > ...

```python
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))
print(id(b))
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2692628879616
2692628877632
[9, 5]
[2, 5]
2692628879616
2692628877632
(week05)
```

type() -- 返回对象的类型

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

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1800279693568
1800279691584
[9, 5]
[2, 5]
1800279693568
1800279691584
<class 'list'>
(week05)
```

isinstance() -- 判断对象是否属于某个（或某些）类型

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

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2399903815936
2399903813952
[9, 5]
[2, 5]
2399903815936
2399903813952
<class 'list'>
isinstance(a,str):  False
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [MSC v.1943 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(1,2,6,10)
1 2 6 10
```

use_of_str.py > ...

```python
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))
print(id(b))
print(type(a))
print("isinstance(a,str): ", isinstance(a, str))
print("isinstance(a,str): ", isinstance(a, list))
print(isinstance(a, (str, float, list)))
print("dir(a): ", dir(a))
```

dir() -- 返回对象所支持的属性（attributes）的名称列表

```python
print("dir(a): ", dir(a))
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2967655946496
2967655944512
[9, 5]
[2, 5]
2967655946496
2967655944512
<class 'list'>
isinstance(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__', '__ne
w__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__
', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy',
 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
(week05)
```

str() -- 返回对象 print 时要显示在终端的字符串

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python
Python 3.12.9 | packaged by conda-forge | (main, Mar  4 2025, 22:37:18) [MSC v.1943 64 bi
t (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print(32)
32
>>> print(str(32))
32
```

(3) 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

(4) 可以利用 assert 语句查验某个表达式（expression）为真，否则报错（AssertionError）退出，流程控制语句。

```
17  assert isinstance(a, list)
18  print("goodbye")
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$  python use_of_str.py
1580376201472
1580376199488
[9, 5]
[2, 5]
1580376201472
1580376199488
<class 'list'>
isinstance(a,str):  False
isinstance(a,str):  True
True
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__', '__ne
w__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__
', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy',
 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
goodbye
(week05)
```

(5) 可以利用 try 语句拦截报错，避免退出，将流程（flow）转入 except 语句。

锁定报错语句：

```
17  try:
18      assert isinstance(a, list)
19  except AssertionError:
20      print("type error")
21  print("goodbye")
```

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

断点调试：

```
17  try:
18      assert isinstance(a, list)
19  except AssertionError:
20      breakpoint()
21      print("type error")
22  print("goodbye")
```

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），一般包括以下途径：

字面值（literal）（包括 f-string 语法）

use_of_expression.py > ...

```
1  print("字面值")
2  s = "university"
3  print(s)
4  print(isinstance(s, str))
5  assert type(s) is str
6
```

use_of_expression.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)
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_expression.py
字面值
university
True
f-string
name:Tom
TAB a	b
new line aaa
bbb
(week05)
```

推导式（comprehension）（仅限 list、dict、set）

初始化（init）

```
25    print("初始化")
26    s = str()
27    print(s)
28    s = [5, 8, 2]
29    print(s)
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_expression.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)
```

运算值（operator）

```
34    assert str() == ""
35
36    s = " ="
37    s = s = 20
38    print(s)
```

```
36    s = "="
37    x = id(s)
38    s = s * 20
39    y = id(s)
40    print(s)
41    assert x != y
42
```

索引值（subscription）

```
43    s = "hello"
44    assert s[3] == "l"
45    assert s[-1] == "o"
46    assert s[:3] == "hel"
47    assert s[4] == s[-1]
48    try:
49        s[s]
50    except ImportError as e:
51        print(e)
```

返回值（return value of function/method call）

```
53    s ="hello"
54    u = s.upper()
55    print(u)
56    print(s)
```

```
58    t = "name:{},age{}"
59    print(t)
60    t1 =t.format("Jack",21)
61    print(t1)
```

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)：

对数学运算符（+、-、*、/、//、%、@）有没有支持

```
63  s1 ="abc"
64  s2 ="ghi"
65  s =s1+s2
66  assert s =="zbcghi"
67  print(s1+s2)
68
69  print(s1-s2)
70  assert ImportError as e:
71      print(e)
```

如何判断相等（==）

```
73  s = "=*="
74  s = = 10
75  print(s)
76  s =="aaa"
77  try:
78      s = s/2
79  except TypeError as e:
80      print(e)
81
82  assert s == "aaaa"
```

对于比较运算符（>、<、>=、<=）有没有支持

```
84  print('abc'<'ABC')
85  print('123'>'abcd')
86  print('9'>',')
87  print('9'<':')
88  print("book"<'box')
89  print("book"<'{')
```

什么值被当作 True，什么值被当作 False

```
assert "book"
assert not ""

s = "book"
print(iter(s))
```

是否可迭代（iterable），如何做迭代（for 循环）

是否支持返回长度（len）

```
for c in s:
    print(c)

print(len(s))
```

是否（如何）支持索引操作（subscription）（[] 运算符）

```
s = "book"
assert s[1:3] == "oo"
```

拥有哪些常用方法（method）可供调用（() 运算符）

```
s = "the book of why took nooo"
print(s.capitalize())
print(s)
print(s.count("oo") == 3)

print("abc123".isalnum())
print("abc123".isalnum())
print("abc123".isidentifier())

q = ["rose", "jack", "bob"]
print(";".join(q))
s = "rose:jack:bob"
print(s.split(":"))
assert s.partition(":") == ("rose", ":", "jack:bob")
```

6. 字节串

```
1  from pathlib import Path
2
3  s = b"hello"
4  print(s)
5
6  p = Path("/d/Anaconda/envs/week05/python")
7  breakpoint()
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'hello'
--Return--
> c:\users\74567\repo\week05\use_of_bytes.py(7)<module>()->None
-> breakpoint()
(Pdb) p p
WindowsPath('/d/Anaconda/envs/week05/python')
(Pdb) p p.exists
<bound method Path.exists of WindowsPath('/d/Anaconda/envs/week05/python')>
```

use_of_bytes.py > ...

```python
from pathlib import Path

s = b"hello"
print(s)

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

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

```
*** NameError: name 'wat' is not defined
(Pdb) import wat
(Pdb) wat /p
str: \d\Anaconda\envs\week05\python
repr: WindowsPath('/d/Anaconda/envs/week05/python')
type: pathlib.WindowsPath
parents: pathlib.Path, pathlib.PureWindowsPath, pathlib.PurePath

Public attributes:
  anchor: str = '\'
  drive: str = ''
  name: str = 'python'
  parent: pathlib.WindowsPath = \d\Anaconda\envs\week05
  parents: pathlib._PathParents = <WindowsPath.parents>
  parts: tuple = ('\\', 'd', 'Anaconda', 'envs', 'week05', 'python')
  root: str = '\'
  stem: str = 'python'
  suffix: str = ''
  suffixes: list = []

  def absolute() # Return an absolute version of this path by prepending 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 permissions 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.…
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
C:\Users\74567\repo\week05\use_of_bytes.py:6: SyntaxWarning: invalid escape sequence '\A'
  p = Path("D:\Anaconda\envs\week05\python.exe")
b'hello'
93184
110
--Return--
> c:\users\74567\repo\week05\use_of_bytes.py(13)<module>()->None
-> breakpoint()
(Pdb) p s[0]
110
(Pdb) p str(s[0])
'110'
(Pdb) p s
b'name: week05\r\nchannels:\r\n  - conda-forge\r\ndependencies:\r\n  - python=3.12\r\n  - wat-inspector'
(Pdb) p s.decode()
'name: week05\r\nchannels:\r\n  - conda-forge\r\ndependencies:\r\n  - python=3.12\r\n  - wat-inspector'
```

use_of_bytes.py > ...

```python
from pathlib import Path

s = b"hello"
print(s)

p = Path("D:\Anaconda\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"
b = s.encode()
breakpoint()
```

```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
C:\Users\74567\repo\week05\use_of_bytes.py:6: SyntaxWarning: invalid escape sequence '\A'
  p = Path("D:\Anaconda\envs\week05\python.exe")
b'hello'
93184
110
--Return--
> c:\users\74567\repo\week05\use_of_bytes.py(22)<module>()->None
-> breakpoint()
(Pdb) p b
b'\xe4\xbd\xa0\xe5\xa5\xbd'
(Pdb) p s
'你好'
(Pdb) p b
b'\xe4\xbd\xa0\xe5\xa5\xbd'
(Pdb) p b[0]
228
(Pdb) p b[1]
189
(Pdb) p b[2]
160
(Pdb) p b[3]
229
```

```
(Pdb) import wat
(Pdb) wat/s.encode
```

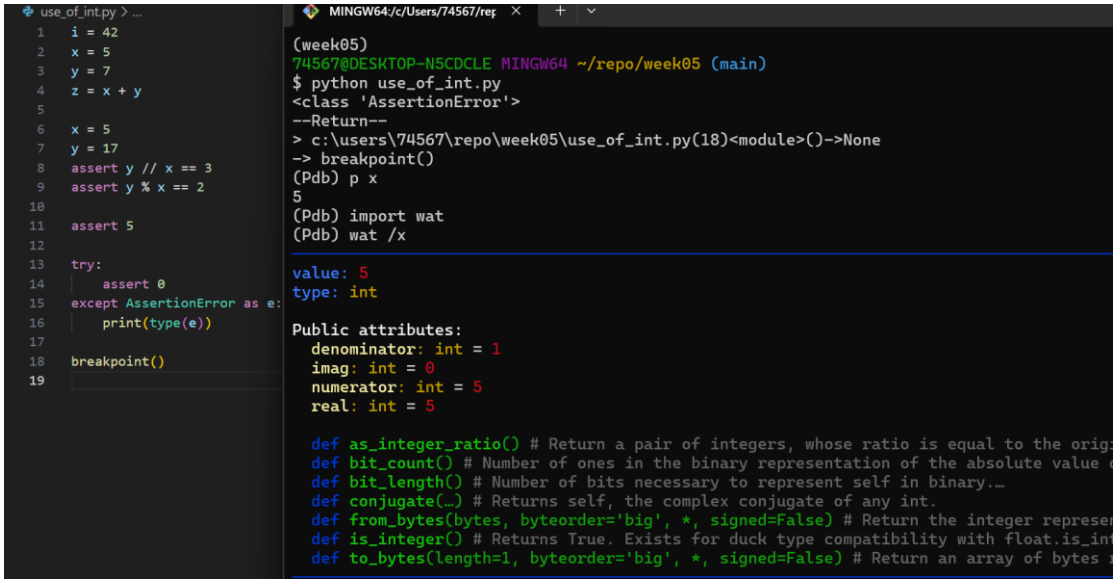
```
value: <built-in method encode of str object at 0x000001C087DC69C0>
type: builtin_function_or_method
signature: def encode(encoding='utf-8', errors='strict')
"""
Encode the string using the codec registered for encoding.

encoding
  The encoding in which to encode the string.
errors
  The error handling scheme to use for encoding errors.
  The default is 'strict' meaning that encoding errors raise a
  UnicodeEncodeError.  Other possible values are 'ignore', 'replace' and
  'xmlcharrefreplace' as well as any other name registered with
  codecs.register_error that can handle UnicodeEncodeErrors.
"""
```

```
(Pdb) p b
b'\xe4\xbd\xa0\xe5\xa5\xbd'
(Pdb) p b.decode()
'你好'
```

```python
s = "你好"
b1 = s.encode("utf-8")
print(b1)
b2 = s.encode("gbk")
print(b2)
s = "abc你好😍"
print(s)
b = s.encode()
breakpoint()
```

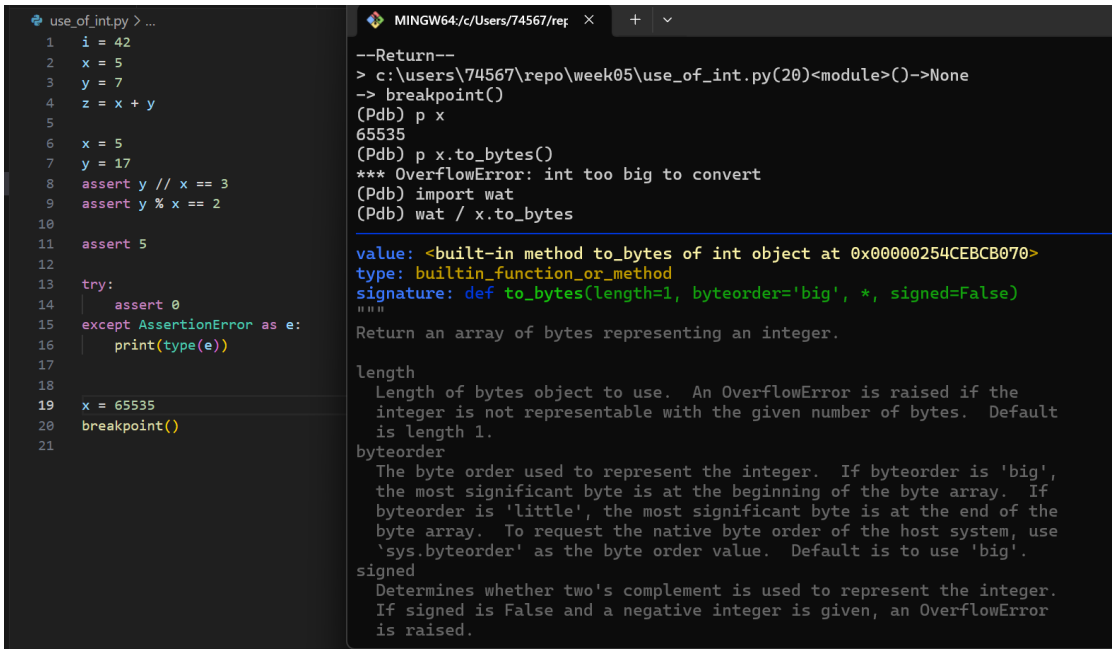
```
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
C:\Users\74567\repo\week05\use_of_bytes.py:6: SyntaxWarning: invalid escape sequence '\A'
  p = Path("D:\Anaconda\envs\week05\python.exe")
b'hello'
93184
110
b'\xe4\xbd\xa0\xe5\xa5\xbd'
b'\xc4\xe3\xba\xc3'
abc你好😍
--Return--
> c:\users\74567\repo\week05\use_of_bytes.py(28)<module>()->None
-> breakpoint()
(Pdb) p b
b'abc\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\x8e'
(Pdb) p b[3:]
b'\xe4\xbd\xa0\xe5\xa5\xbd\xf0\x9f\x98\x8e'
(Pdb) p b[3:].encode()
*** AttributeError: 'bytes' object has no attribute 'encode'. Did you mean: 'decode'?
(Pdb) p b[3:].decode()
'你好😍'
(Pdb) p b[3:9].decode()
'你好'
```

7. 整数



```
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 breakpoint()
19
```

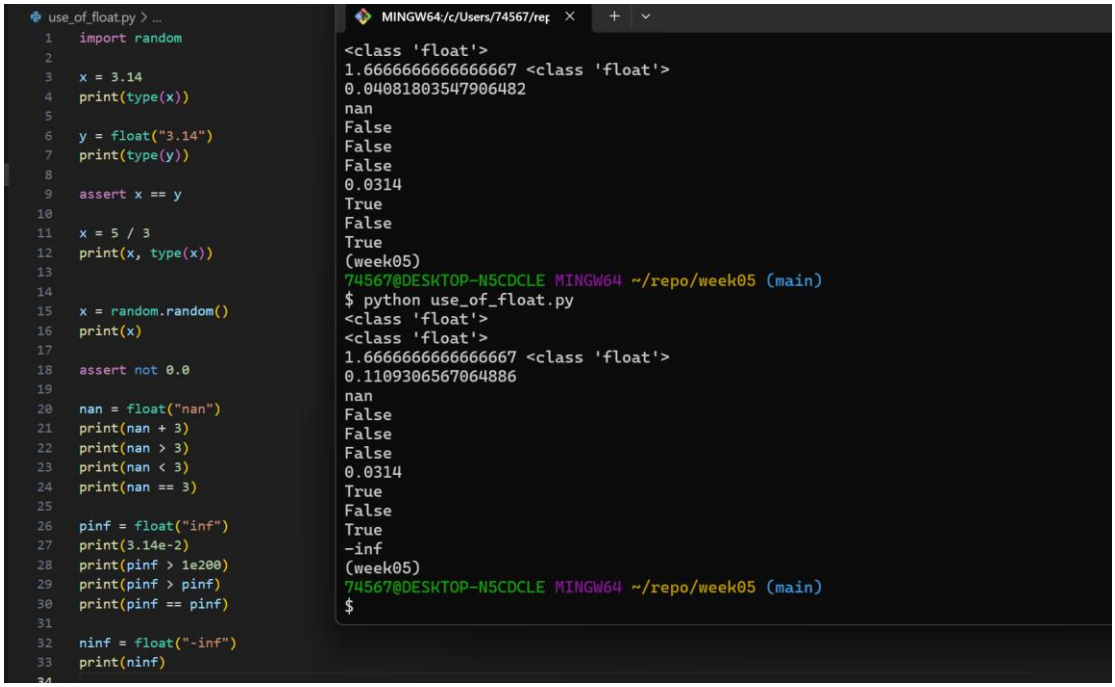
```
MINGW64:/c/Users/74567/rep
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_int.py
<class 'AssertionError'>
--Return--
> c:\users\74567\repo\week05\use_of_int.py(18)<module>()->None
-> breakpoint()
(Pdb) p x
5
(Pdb) import wat
(Pdb) wat /x

value: 5
type: int

Public attributes:
  denominator: int = 1
  imag: int = 0
  numerator: int = 5
  real: int = 5

  def as_integer_ratio() # Return a pair of integers, whose ratio is equal to the orig
  def bit_count() # Number of ones in the binary representation of the absolute value
  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 represe
  def is_integer() # Returns True. Exists for duck type compatibility with float.is_in
  def to_bytes(length=1, byteorder='big', *, signed=False) # Return an array of bytes
```



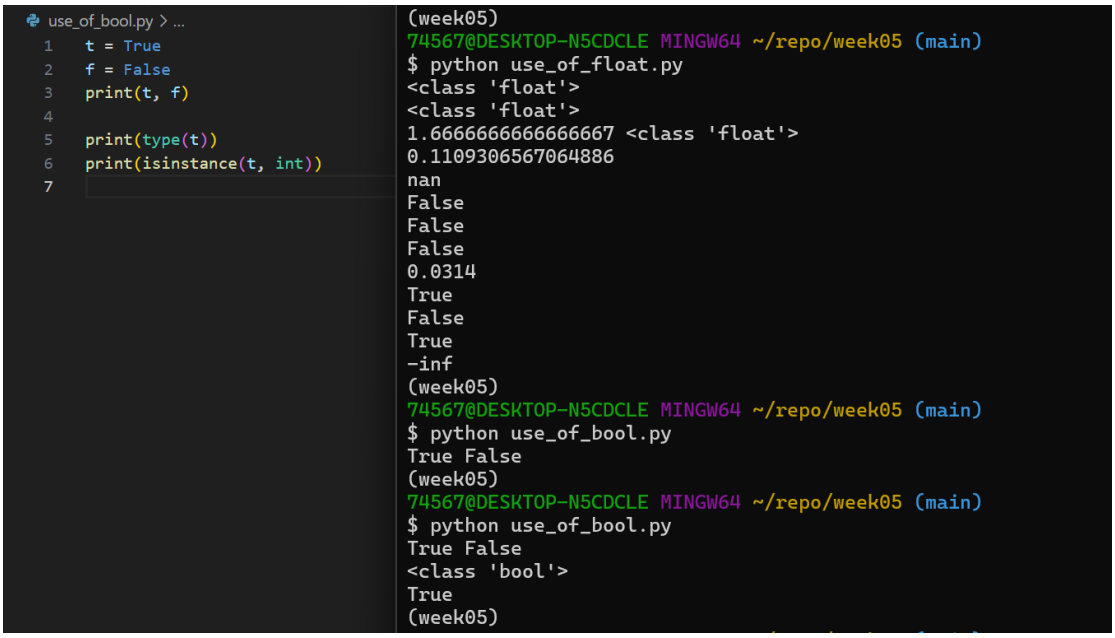
```
use_of_int.py > ...
1  i = 42
2  x = 5
3  y = 7
4  z = x + y
5
6  x = 5
7  y = 17
8  assert y // x == 3
9  assert y % x == 2
10
11 assert 5
12
13 try:
14     assert 0
15 except AssertionError as e:
16     print(type(e))
17
18
19 x = 65535
20 breakpoint()
21
```

```
MINGW64:/c/Users/74567/rep
--Return--
> c:\users\74567\repo\week05\use_of_int.py(20)<module>()->None
-> breakpoint()
(Pdb) p x
65535
(Pdb) p x.to_bytes()
*** OverflowError: int too big to convert
(Pdb) import wat
(Pdb) wat / x.to_bytes

value: <built-in method to_bytes of int object at 0x00000254CEBCB070>
type: builtin_function_or_method
signature: def to_bytes(length=1, byteorder='big', *, signed=False)
"""
Return an array of bytes representing an integer.

length
  Length of bytes object to use.  An OverflowError is raised if the
  integer is not representable with the given number of bytes.  Default
  is length 1.
byteorder
  The byte order used to represent the integer.  If byteorder is 'big',
  the most significant byte is at the beginning of the byte array.  If
  byteorder is 'little', the most significant byte is at the end of the
  byte array.  To request the native byte order of the host system, use
  `sys.byteorder' as the byte order value.  Default is to use 'big'.
signed
  Determines whether two's complement is used to represent the integer.
  If signed is False and a negative integer is given, an OverflowError
  is raised.
```

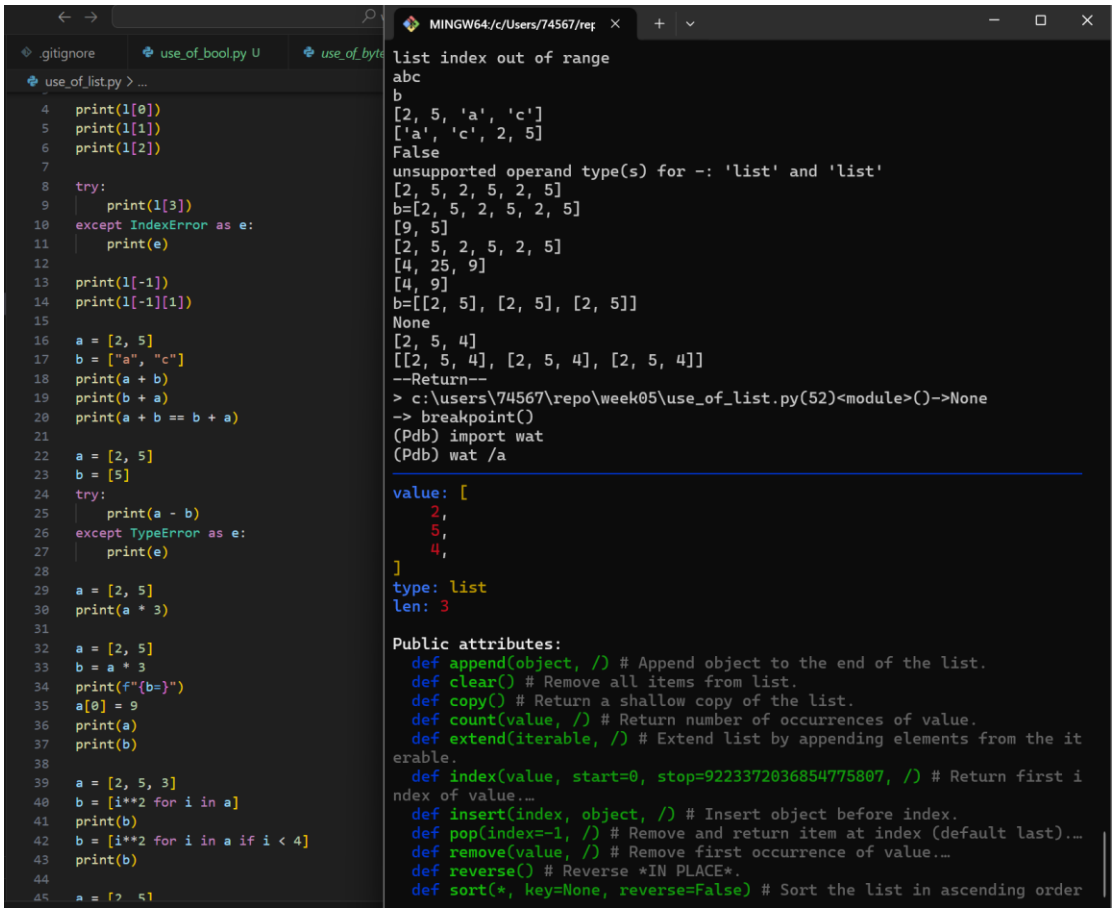
8. 浮点数


use_of_float.py > ...
import random
x = 3.14
print(type(x))
y = float("3.14")
print(type(y))
assert x == y
x = 5 / 3
print(x, type(x))
x = random.random()
print(x)
assert not 0.0
nan = float("nan")
print(nan + 3)
print(nan > 3)
print(nan < 3)
print(nan == 3)
pinf = float("inf")
print(3.14e-2)
print(pinf > 1e200)
print(pinf > pinf)
print(pinf == pinf)
ninf = float("-inf")
print(ninf)
MINGW64:/c/Users/74567/rep
<class 'float'>
1.6666666666666667 <class 'float'>
0.04081803547906482
nan
False
False
False
0.0314
True
False
True
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.1109306567064886
nan
False
False
False
0.0314
True
False
True
-inf
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$

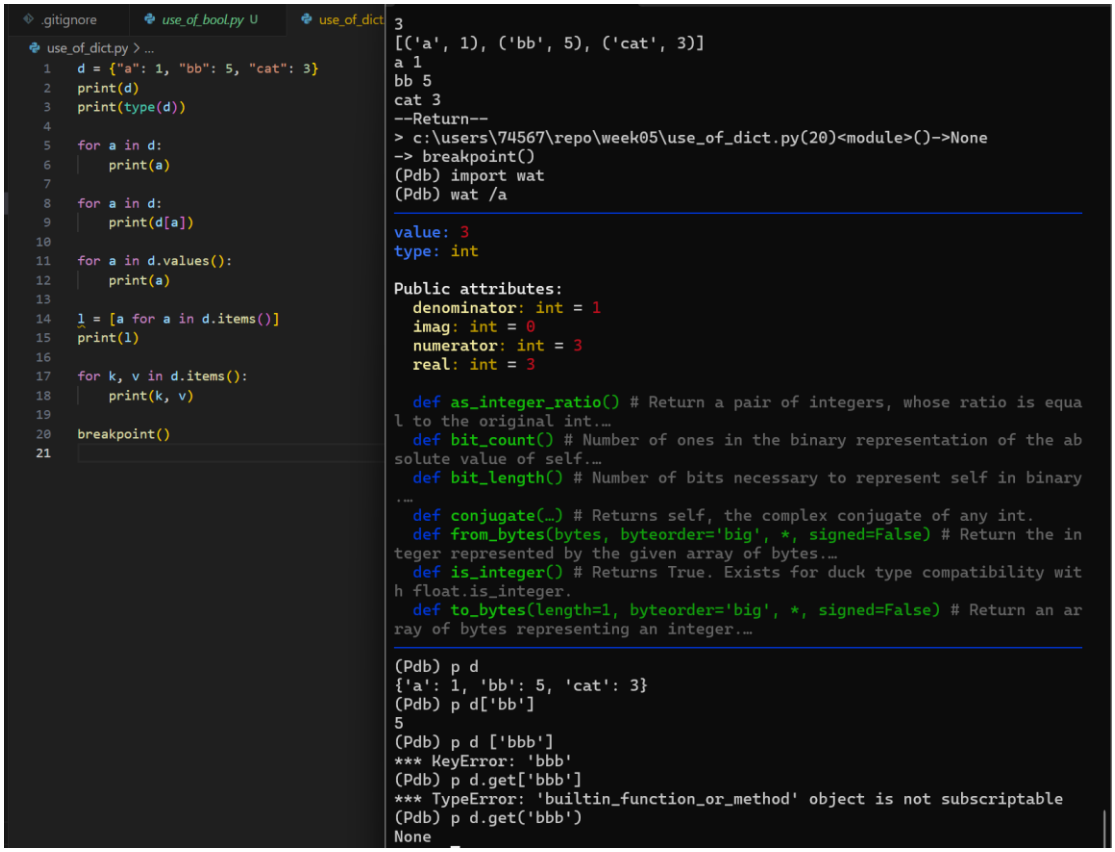
9. 布尔值


use_of_bool.py > ...
t = True
f = False
print(t, f)
print(type(t))
print(isinstance(t, int))
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_float.py
<class 'float'>
<class 'float'>
1.6666666666666667 <class 'float'>
0.1109306567064886
nan
False
False
False
0.0314
True
False
True
-inf
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_bool.py
True False
(week05)
74567@DESKTOP-N5CDCLE MINGW64 ~/repo/week05 (main)
$ python use_of_bool.py
True False
<class 'bool'>
True
(week05)

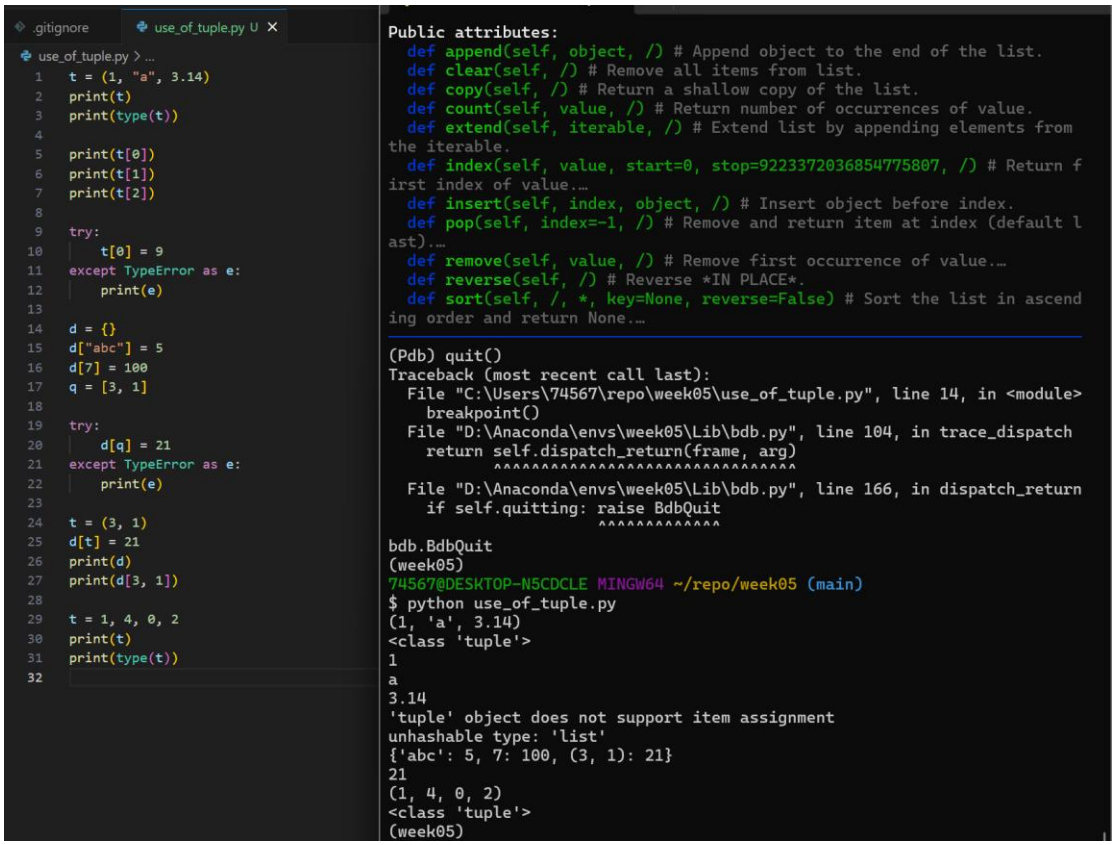
10. 列表


.gitignore
use_of_bool.py U
use_of_list.py > ...
print(l[0])
print(l[1])
print(l[2])
try:
print(l[3])
except IndexError as e:
print(e)
print(l[-1])
print(l[-1][1])
a = [2, 5]
b = ["a", "c"]
print(a + b)
print(b + a)
print(a + b == b + a)
a = [2, 5]
b = [5]
try:
print(a - b)
except TypeError as e:
print(e)
a = [2, 5]
print(a * 3)
a = [2, 5]
b = a * 3
print(f"{b=}")
a[0] = 9
print(a)
print(b)
a = [2, 5, 3]
b = [i**2 for i in a]
print(b)
b = [i**2 for i in a if i < 4]
print(b)
MINGW64:/c/Users/74567/rep
list index out of range
abc
b
[2, 5, 'a', 'c']
['a', 'c', 2, 5]
False
unsupported operand type(s) for -: 'list' and 'list'
[2, 5, 2, 5, 2, 5]
b=[2, 5, 2, 5, 2, 5]
[9, 5]
[2, 5, 2, 5, 2, 5]
[4, 25, 9]
[4, 9]
b=[[2, 5], [2, 5], [2, 5]]
None
[2, 5, 4]
[[2, 5, 4], [2, 5, 4], [2, 5, 4]]
--Return--
> c:\users\74567\repo\week05\use_of_list.py(52)<module>()->None
-> breakpoint()
(Pdb) import wat
(Pdb) wat /a
value: [
2,
5,
4,
]
type: list
len: 3
Public attributes:
def append(object, /) # Append object to the end of the list.
def clear() # Remove all items from list.
def copy() # Return a shallow copy of the list.
def count(value, /) # Return number of occurrences of value.
def extend(iterable, /) # Extend list by appending elements from the it
erable.
def index(value, start=0, stop=9223372036854775807, /) # Return first i
ndex of value.…
def insert(index, object, /) # Insert object before index.
def pop(index=-1, /) # Remove and return item at index (default last).…
def remove(value, /) # Remove first occurrence of value.…
def reverse() # Reverse *IN PLACE*.
def sort(*, key=None, reverse=False) # Sort the list in ascending order

11. 字典


.gitignore
use_of_bool.py U
use_of_dict
use_of_dict.py > ...
d = {"a": 1, "bb": 5, "cat": 3}
print(d)
print(type(d))
for a in d:
print(a)
for a in d:
print(d[a])
for a in d.values():
print(a)
l = [a for a in d.items()]
print(l)
for k, v in d.items():
print(k, v)
breakpoint()
3
[('a', 1), ('bb', 5), ('cat', 3)]
a 1
bb 5
cat 3
--Return--
> c:\users\74567\repo\week05\use_of_dict.py(20)<module>()->None
-> breakpoint()
(Pdb) import wat
(Pdb) wat /a
value: 3
type: int
Public attributes:
denominator: int = 1
imag: int = 0
numerator: int = 3
real: int = 3
def as_integer_ratio() # Return a pair of integers, whose ratio is equa
l to the original int.…
def bit_count() # Number of ones in the binary representation of the ab
solute 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 in
teger represented by the given array of bytes.…
def is_integer() # Returns True. Exists for duck type compatibility wit
h float.is_integer.
def to_bytes(length=1, byteorder='big', *, signed=False) # Return an ar
ray of bytes representing an integer.…
(Pdb) p d
{'a': 1, 'bb': 5, 'cat': 3}
(Pdb) p d['bb']
5
(Pdb) p d ['bbb']
*** KeyError: 'bbb'
(Pdb) p d.get['bbb']
*** TypeError: 'builtin_function_or_method' object is not subscriptable
(Pdb) p d.get('bbb')
None

12. 元组



```
.gitignore   use_of_tuple.py U
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[q] = 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
```

```
Public attributes:
  def append(self, object, /) # Append object to the end of the list.
  def clear(self, /) # Remove all items from list.
  def copy(self, /) # Return a shallow copy of the list.
  def count(self, value, /) # Return number of occurrences of value.
  def extend(self, iterable, /) # Extend list by appending elements from
the iterable.
  def index(self, value, start=0, stop=9223372036854775807, /) # Return f
irst index of value.…
  def insert(self, index, object, /) # Insert object before index.
  def pop(self, index=-1, /) # Remove and return item at index (default l
ast).…
  def remove(self, value, /) # Remove first occurrence of value.…
  def reverse(self, /) # Reverse *IN PLACE*.
  def sort(self, /, *, key=None, reverse=False) # Sort the list in ascend
ing order and return None.…

(Pdb) quit()
Traceback (most recent call last):
  File "C:\Users\74567\repo\week05\use_of_tuple.py", line 14, in <module>
    breakpoint()
  File "D:\Anaconda\envs\week05\Lib\bdb.py", line 104, in trace_dispatch
    return self.dispatch_return(frame, arg)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "D:\Anaconda\envs\week05\Lib\bdb.py", line 166, in dispatch_return
    if self.quitting: raise BdbQuit
                      ^^^^^^^^^^^^^
bdb.BdbQuit
(week05)
74567@DESKTOP-N5CDCLE 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)
```

13. 集合

use_of_set.py > ...

```python
s = {1, 4, 7}
print(s)
print(type(s))

try:
    s = {1, [4], 7}
except TypeError as e:
    print(e)

q = [1, 2, 1, 2, 5, 1]
print(q)
s = set(q)
print(s)

s = (5, 2, 1, 2, 2, 1)
print(s)
print(2 in s)
print(3 in s)

s2 = (3, 2, 3)
print(s | s2)
print(s & s2)
print(s ^ s2)
```

14. Path

use_of_path.py > ...

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

p = Path(".")
print(p)
print(p.exists())
print(p.absolute())
pprint(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()
```

15. Datetime、

use_of_datetime.py > ...

```python
from datetime import date, datetime

t1 = date.today()
t2 = date(2025, 11, 11)
td = t2 - t1
print(td)
print(type(td))
print(td.days)
s1 = "2024-05-23"
s2 = "2024-12-04"
d1 = datetime.strptime(s1, "%Y-%m-%d")
d2 = datetime.strptime(s2, "%Y-%m-%d")
print(d1)
print(d2)
breakpoint()
```