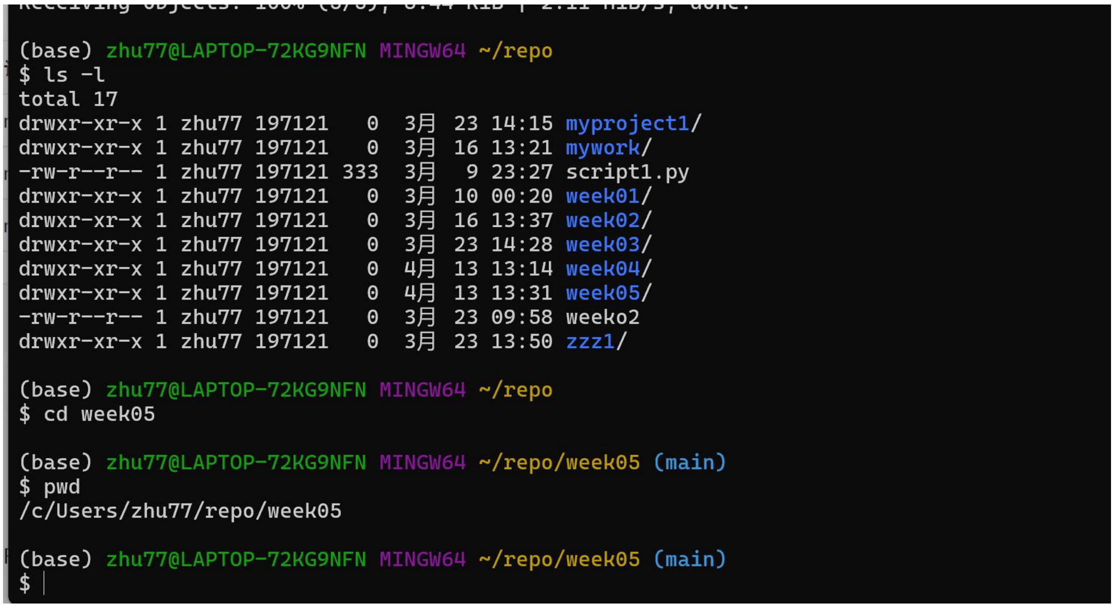
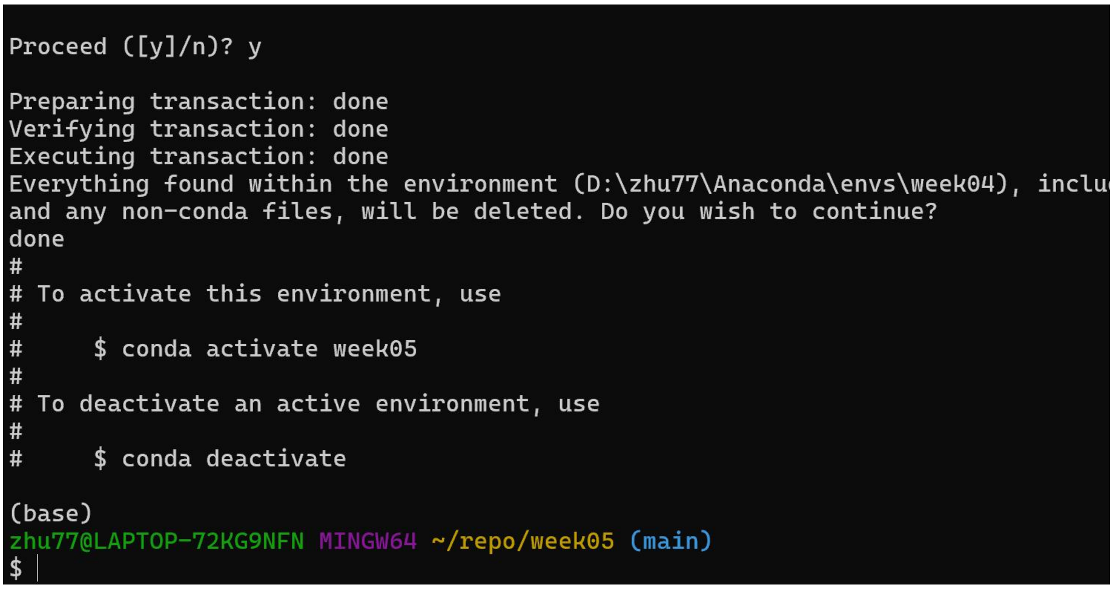
金融编程作业 week05

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



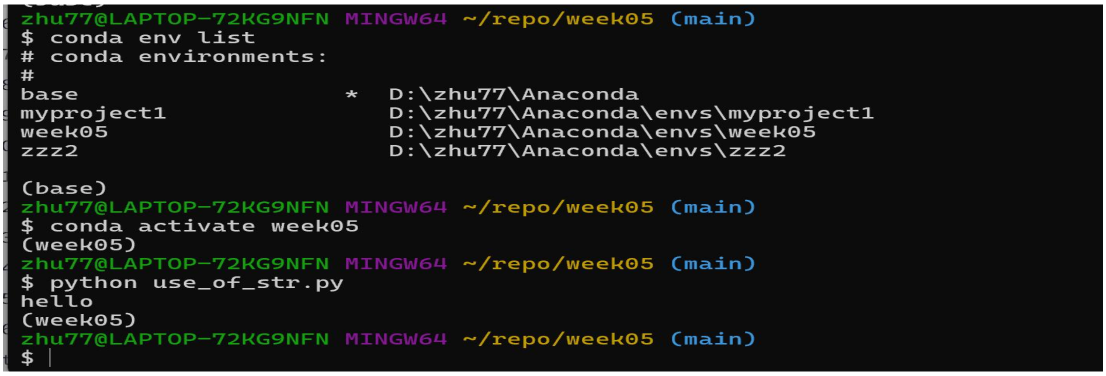
```
Receiving objects: 100% (8/8), 8.44 KiB | 2.11 MiB/s, done.

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo
$ ls -l
total 17
drwxr-xr-x 1 zhu77 197121   0  3月  23 14:15 myproject1/
drwxr-xr-x 1 zhu77 197121   0  3月  16 13:21 mywork/
-rw-r--r-- 1 zhu77 197121 333  3月   9 23:27 script1.py
drwxr-xr-x 1 zhu77 197121   0  3月  10 00:20 week01/
drwxr-xr-x 1 zhu77 197121   0  3月  16 13:37 week02/
drwxr-xr-x 1 zhu77 197121   0  3月  23 14:28 week03/
drwxr-xr-x 1 zhu77 197121   0  4月  13 13:14 week04/
drwxr-xr-x 1 zhu77 197121   0  4月  13 13:31 week05/
-rw-r--r-- 1 zhu77 197121   0  3月  23 09:58 weeko2
drwxr-xr-x 1 zhu77 197121   0  3月  23 13:50 zzz1/

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo
$ cd week05

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ pwd
/c/Users/zhu77/repo/week05

(base) zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ |
```

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

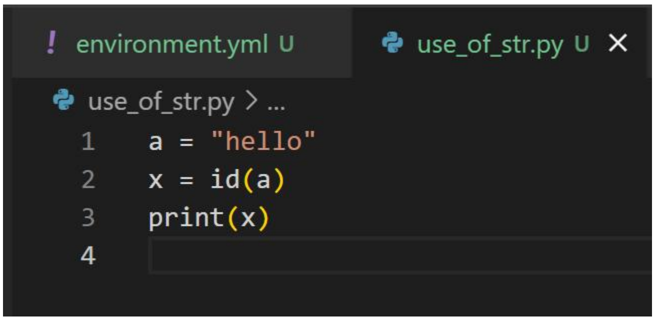


```

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Everything found within the environment (D:\zhu77\Anaconda\envs\week04), inclu
and any non-conda files, will be deleted. Do you wish to continue?
done
#
# To activate this environment, use
#
#     $ conda activate week05
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ |
```

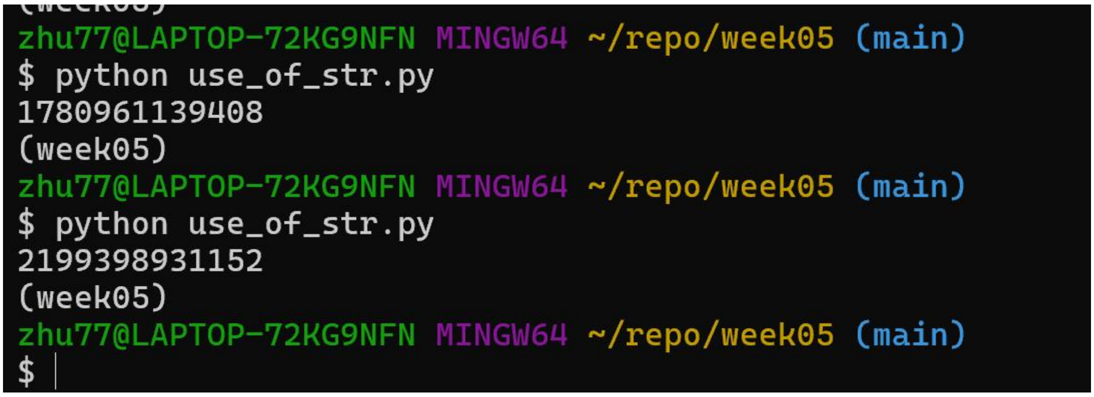
3. 逐个 创建 use_of_{name}.py 文件，其中 {name} 替换为上述要求掌握的对象类型，例如 use_of_str.py:



```
(base)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ conda env list
# conda environments:
#
base                  *  D:\zhu77\Anaconda
myproject1               D:\zhu77\Anaconda\envs\myproject1
week05                   D:\zhu77\Anaconda\envs\week05
zzz2                     D:\zhu77\Anaconda\envs\zzz2

(base)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ conda activate week05
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ |
```

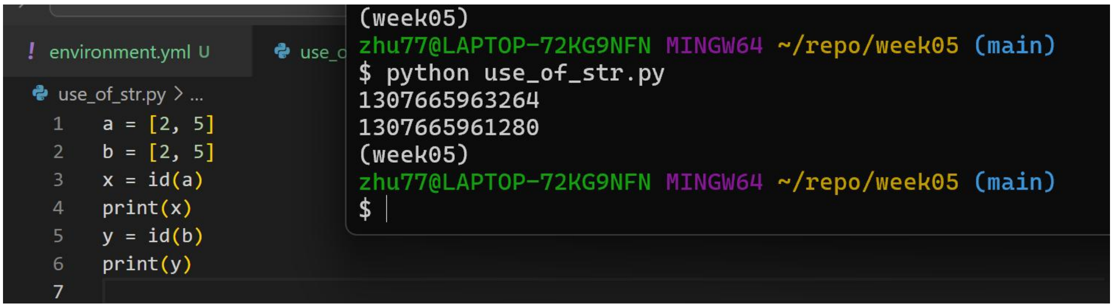


```python
a = "hello"
x = id(a)
print(x)

```



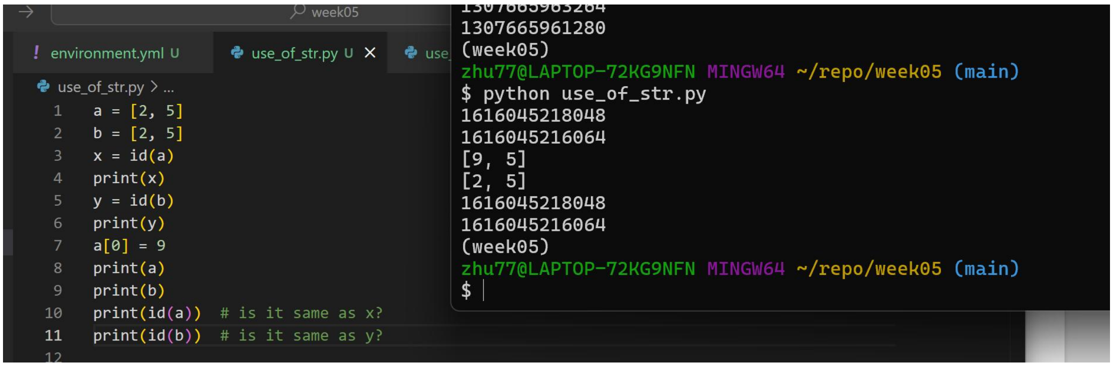
```
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1780961139408
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2199398931152
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

每次运行的 id 不一样。



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

```

```
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1307665963264
1307665961280
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```



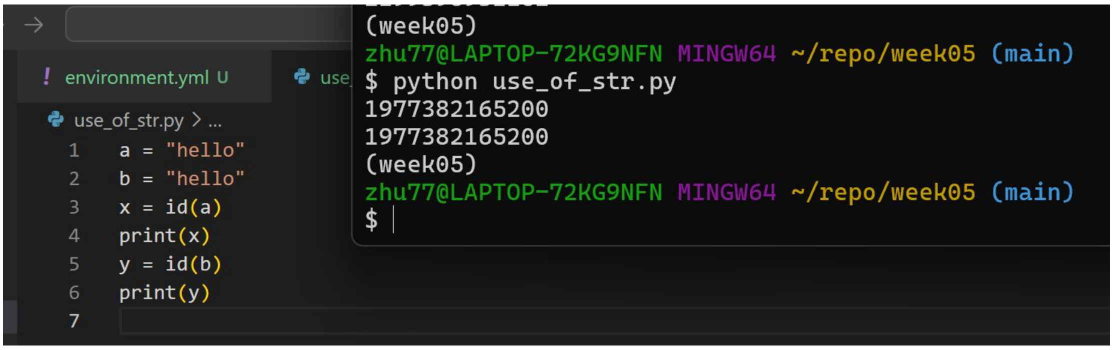
```python
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))  # is it same as x?
print(id(b))  # is it same as y?
```

```
1307665963264
1307665961280
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1616045218048
1616045216064
[9, 5]
[2, 5]
1616045218048
1616045216064
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

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)

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):

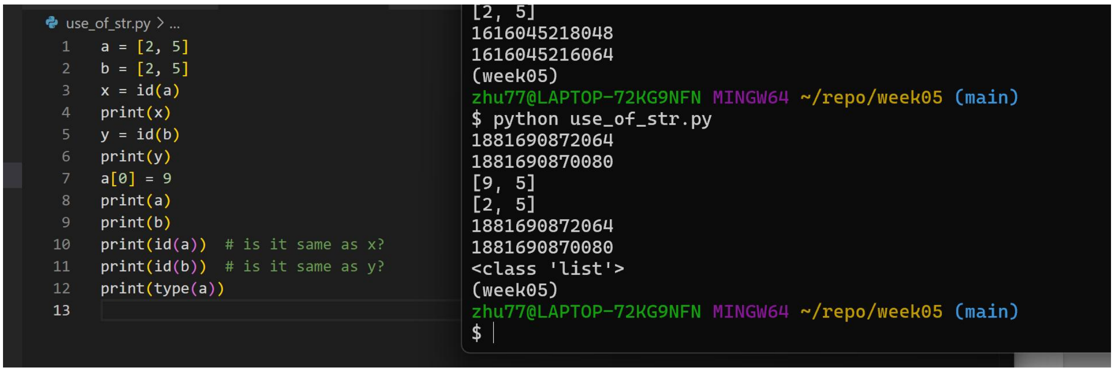
id() -- 返回对象在虚拟内存中的地址（正整数)，如果 id(a) == id(b)，那么 a is b (is 是个运算符)



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

```
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1977382165200
1977382165200
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

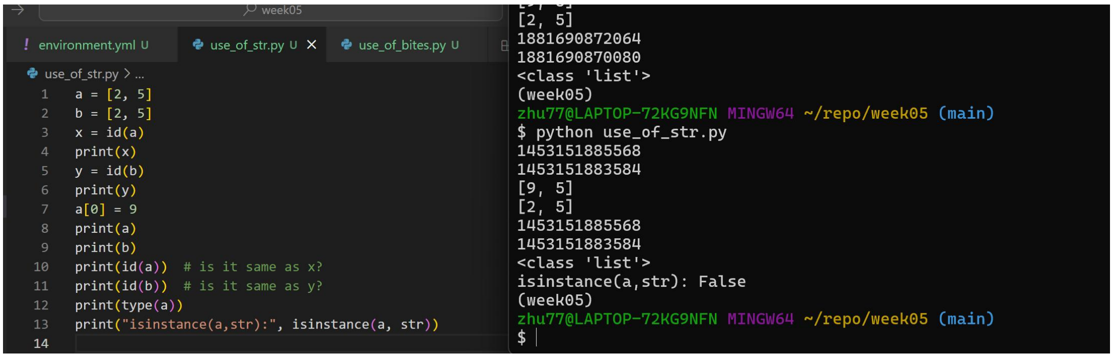
type() -- 返回对象的类型



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

```
[2, 5]
1616045218048
1616045216064
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1881690872064
1881690870080
[9, 5]
[2, 5]
1881690872064
1881690870080
<class 'list'>
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

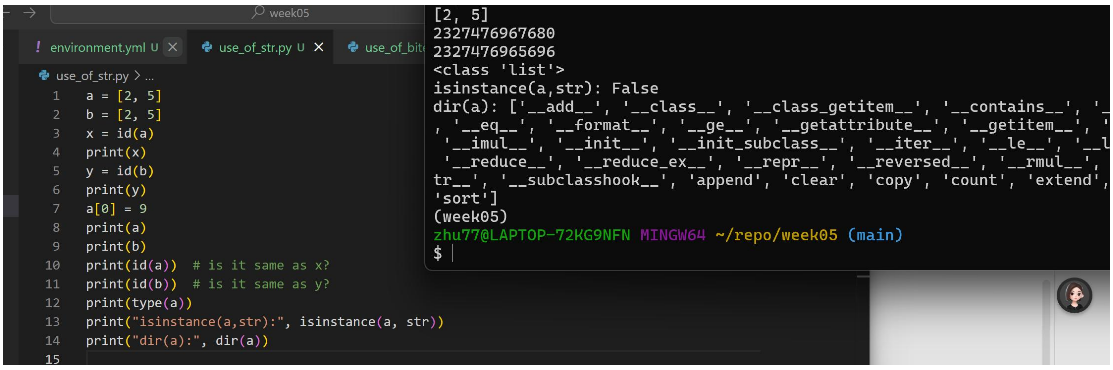
isinstance() -- 判断对象是否属于某个（或某些）类型



```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))  # is it same as x?
print(id(b))  # is it same as y?
print(type(a))
print("isinstance(a,str):", isinstance(a, str))
```

```
[2, 5]
1881690872064
1881690870080
<class 'list'>
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1453151885568
1453151883584
[9, 5]
[2, 5]
1453151885568
1453151883584
<class 'list'>
isinstance(a,str): False
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

dir() -- 返回对象所支持的属性（attributes）的名称列表



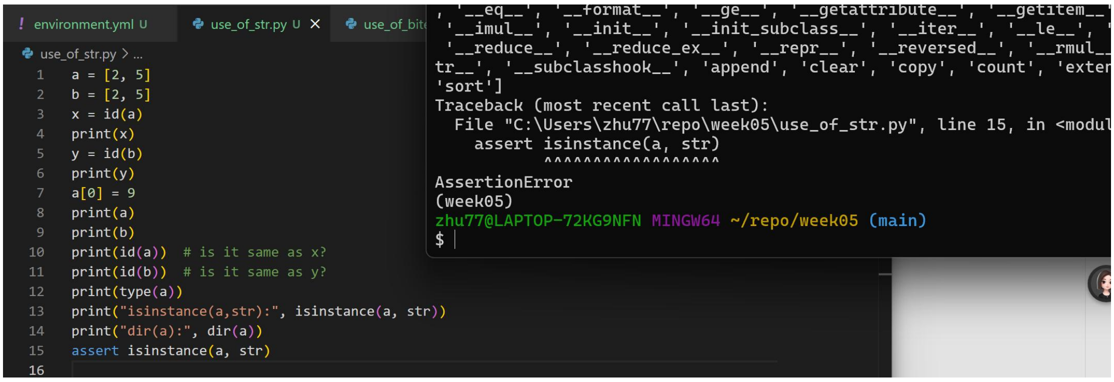
```python
a = [2, 5]
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))  # is it same as x?
print(id(b))  # is it same as y?
print(type(a))
print("isinstance(a,str):", isinstance(a, str))
print("dir(a):", dir(a))
```

```
[2, 5]
2327476967680
2327476965696
<class 'list'>
isinstance(a,str): False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '_
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__l
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend',
'sort']
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

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

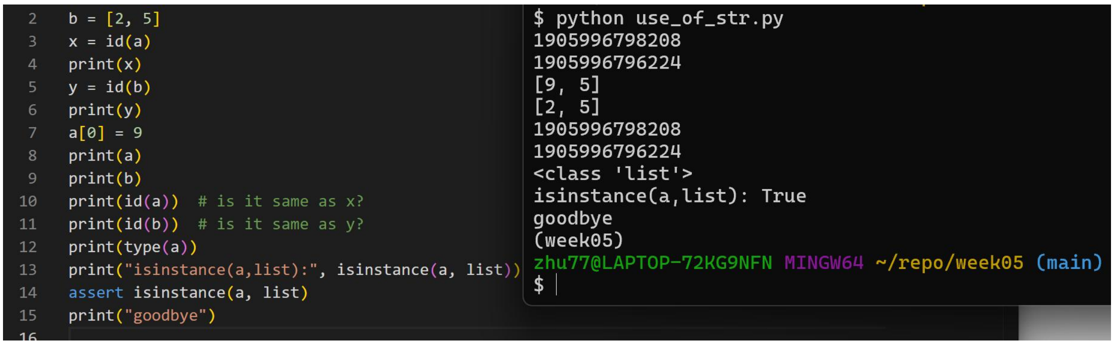
可以利用 assert 语句查验某个表达式（expression）为真，否则报错（AssertionError）退出



```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))  # is it same as x?
print(id(b))  # is it same as y?
print(type(a))
print("isinstance(a,str):", isinstance(a, str))
print("dir(a):", dir(a))
assert isinstance(a, str)
```

```
, '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__'
 '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'exter
'sort']
Traceback (most recent call last):
  File "C:\Users\zhu77\repo\week05\use_of_str.py", line 15, in <modul
    assert isinstance(a, str)
           ^^^^^^^^^^^^^^^^^^

AssertionError
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

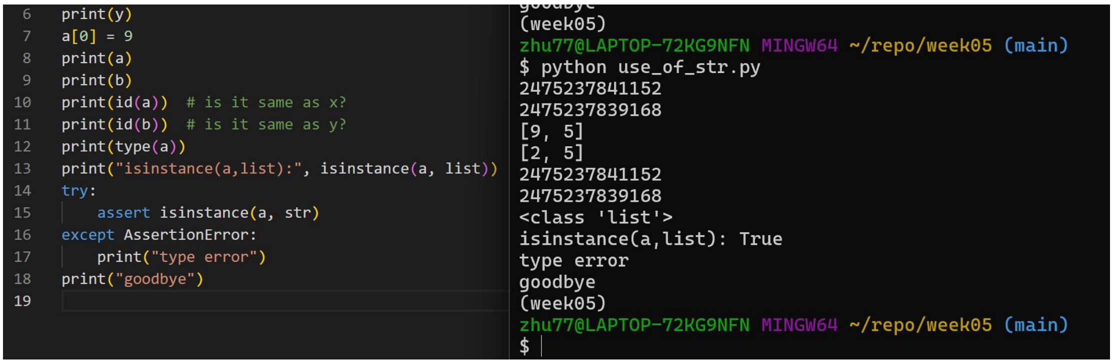
如果 assert 语句报错，就退出，无法运行下面的程序。



```python
b = [2, 5]
x = id(a)
print(x)
y = id(b)
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))  # is it same as x?
print(id(b))  # is it same as y?
print(type(a))
print("isinstance(a,list):", isinstance(a, list))
assert isinstance(a, list)
print("goodbye")
```

```
$ python use_of_str.py
1905996798208
1905996796224
[9, 5]
[2, 5]
1905996798208
1905996796224
<class 'list'>
isinstance(a,list): True
goodbye
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

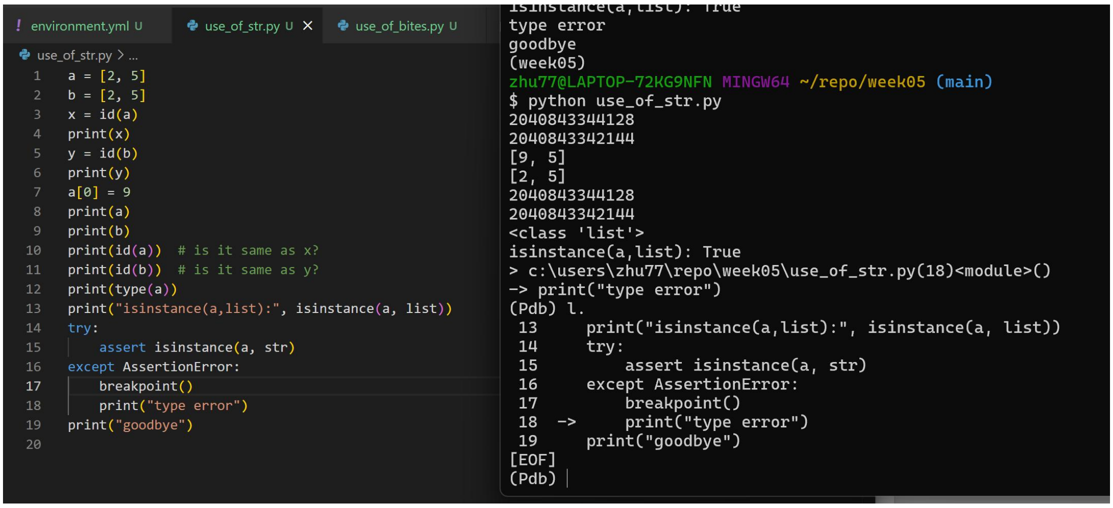
可以利用 try 语句拦截报错，避免退出，将流程 (flow) 转入 except 语句



```python
print(y)
a[0] = 9
print(a)
print(b)
print(id(a))  # is it same as x?
print(id(b))  # is it same as y?
print(type(a))
print("isinstance(a,list):", isinstance(a, list))
try:
    assert isinstance(a, str)
except AssertionError:
    print("type error")
print("goodbye")
```

```
goodbye
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2475237841152
2475237839168
[9, 5]
[2, 5]
2475237841152
2475237839168
<class 'list'>
isinstance(a,list): True
type error
goodbye
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

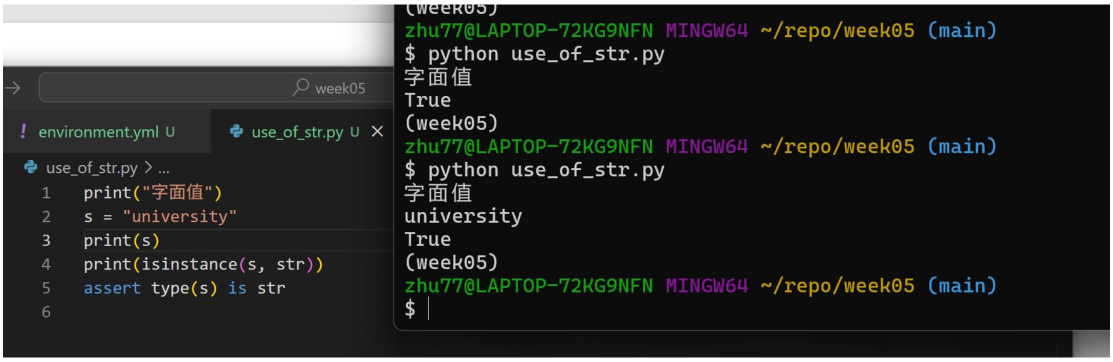
可以调用 breakpoint() 函数暂停程序运行，进入 pdb 调试 (debug) 模式



```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))  # is it same as x?
print(id(b))  # is it same as y?
print(type(a))
print("isinstance(a,list):", isinstance(a, list))
try:
    assert isinstance(a, str)
except AssertionError:
    breakpoint()
    print("type error")
print("goodbye")
```

```
isinstance(a,list): True
type error
goodbye
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2040843344128
2040843342144
[9, 5]
[2, 5]
2040843344128
2040843342144
<class 'list'>
isinstance(a,list): True
> c:\users\zhu77\repo\week05\use_of_str.py(18)<module>()
-> print("type error")
(Pdb) l.
 13         print("isinstance(a,list):", isinstance(a, list))
 14         try:
 15             assert isinstance(a, str)
 16         except AssertionError:
 17             breakpoint()
 18  ->         print("type error")
 19         print("goodbye")
[EOF]
(Pdb) 
```

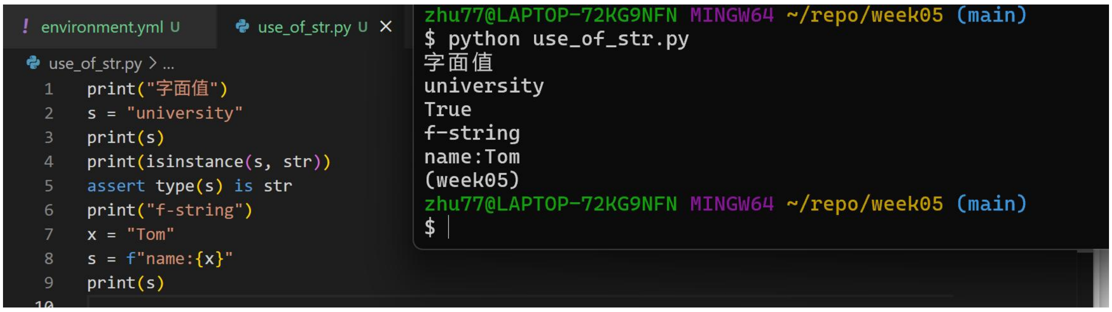
4. 对于 每一个 上述要求掌握的对象类型 (将来遇到新的对象类型也应该如此)，我们首先应该熟悉如何通过 表达式 (expression) 得到他们的 实例 (instance)，一般包括以下途径:



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

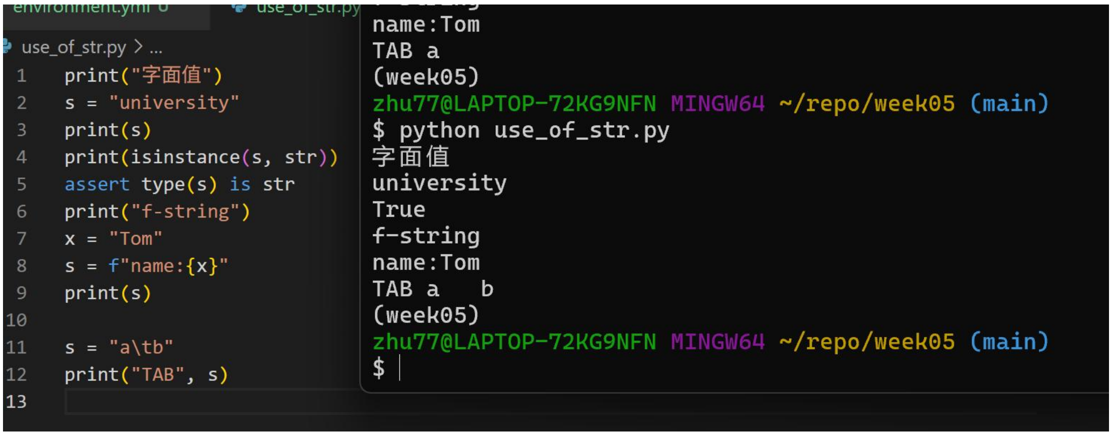
```
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
True
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$
```

字面值 (literal) (包括 f-string 语法)



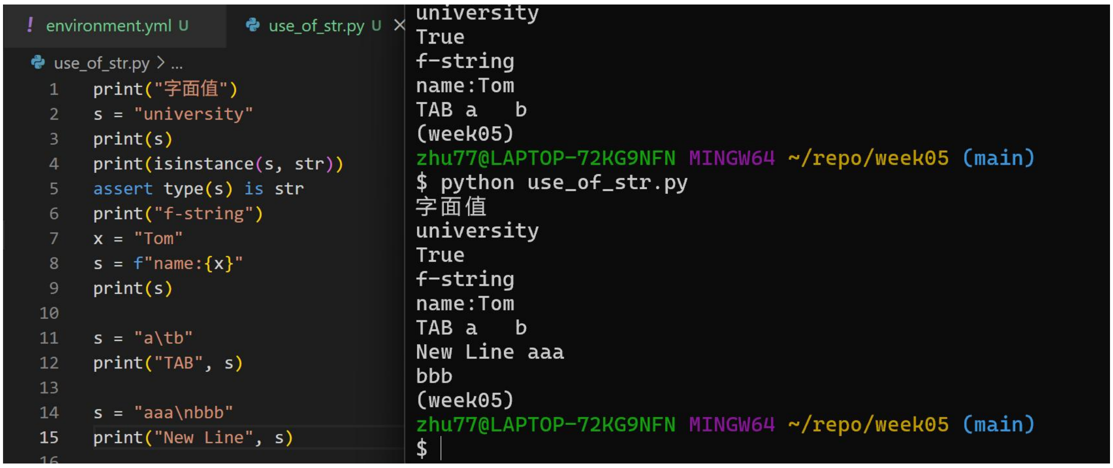
```python
print("字面值")
s = "university"
print(s)
print(isinstance(s, str))
assert type(s) is str
print("f-string")
x = "Tom"
s = f"name:{x}"
print(s)
```

```
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name:Tom
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$
```



```python
print("字面值")
s = "university"
print(s)
print(isinstance(s, str))
assert type(s) is str
print("f-string")
x = "Tom"
s = f"name:{x}"
print(s)

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

```
name:Tom
TAB a
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f-string
name:Tom
TAB a    b
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$
```

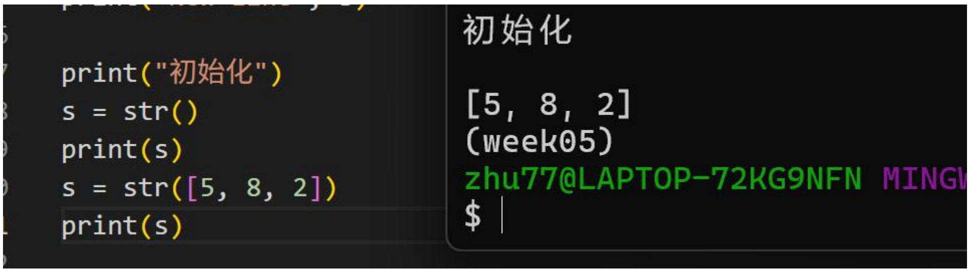


```python
print("字面值")
s = "university"
print(s)
print(isinstance(s, str))
assert type(s) is str
print("f-string")
x = "Tom"
s = f"name:{x}"
print(s)

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

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

```
university
True
f-string
name:Tom
TAB a    b
(week05)
zhu77@LAPTOP-72KG9NFN 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
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$
```

推导式 (comprehension) (仅限 list、dict、set)

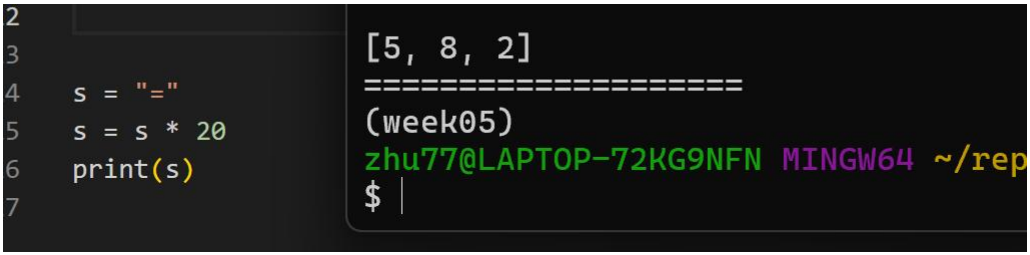
初始化 (init)



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

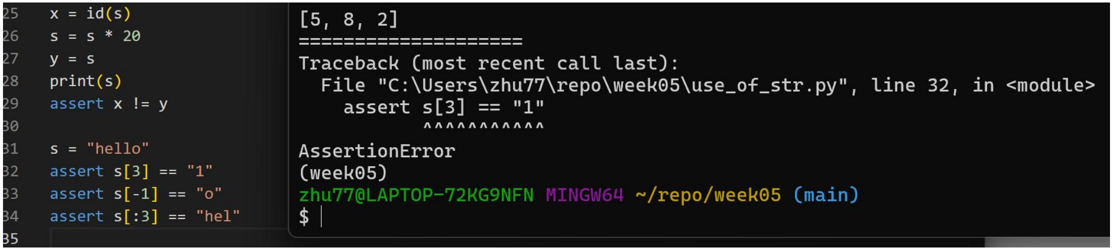
```
初始化

[5, 8, 2]
(week05)
zhu77@LAPTOP-72KG9NFN MINGW
$ 
```

运算值 (operator)



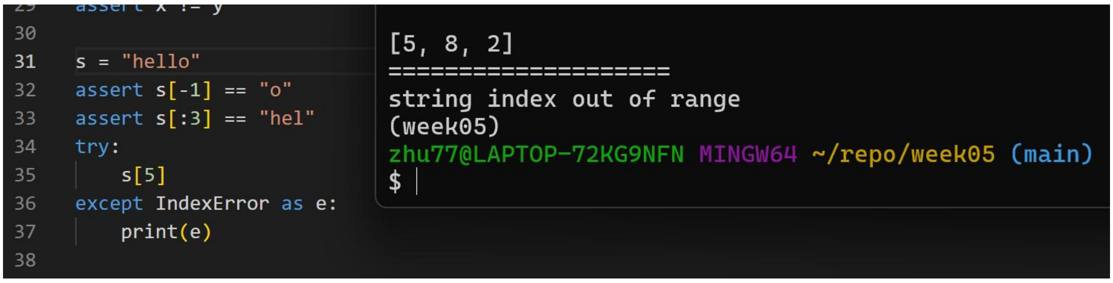
```
s = "="
s = s * 20
print(s)
```

```
[5, 8, 2]
====================
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/rep
$ 
```



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

s = "hello"
assert s[3] == "1"
assert s[-1] == "o"
assert s[:3] == "hel"
```

```
[5, 8, 2]
====================
Traceback (most recent call last):
  File "C:\Users\zhu77\repo\week05\use_of_str.py", line 32, in <module>
    assert s[3] == "1"
           ^^^^^^^^^^^
AssertionError
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

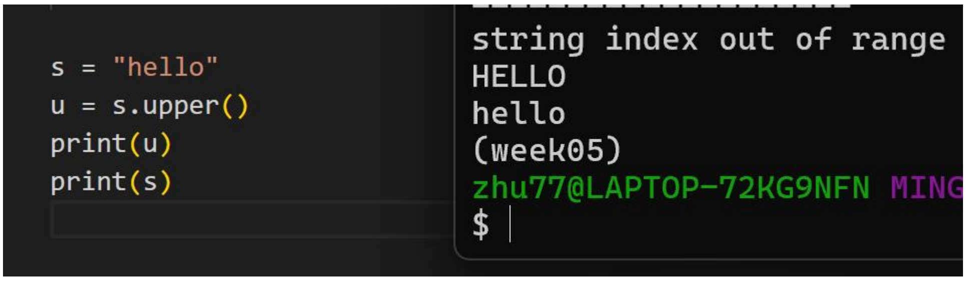


```
assert x != y

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

```
[5, 8, 2]
====================
string index out of range
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

提取值 (subscription)

返回值 (return value of function/method call)

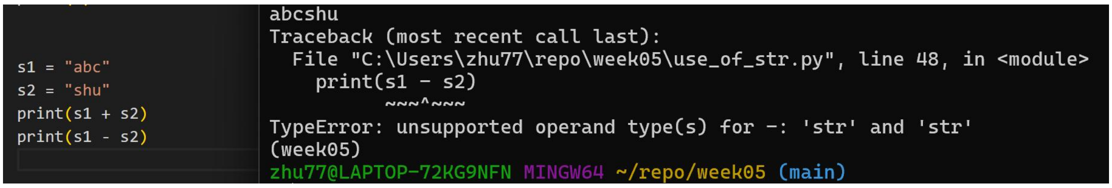


```
s = "hello"
u = s.upper()
print(u)
print(s)
```

```
string index out of range
HELLO
hello
(week05)
zhu77@LAPTOP-72KG9NFN MING
$ 
```

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):

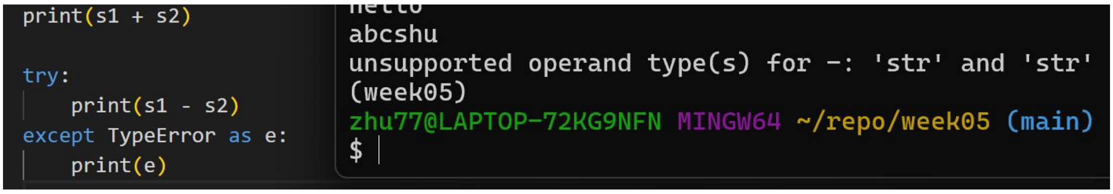
对数学运算符 (+、-、*、**、/、//、%、@) 有没有支持



```
s1 = "abc"
s2 = "shu"
print(s1 + s2)
print(s1 - s2)
```

```
abcshu
Traceback (most recent call last):
  File "C:\Users\zhu77\repo\week05\use_of_str.py", line 48, in <module>
    print(s1 - s2)
          ~~~^~~~
TypeError: unsupported operand type(s) for -: 'str' and 'str'
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
```

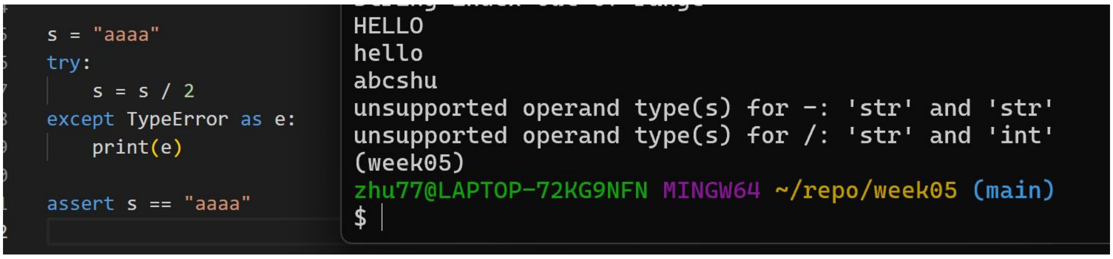
不同的 error 可以用 try 引入不同的走向



```
print(s1 + s2)

try:
    print(s1 - s2)
except TypeError as e:
    print(e)
```

```
hello
abcshu
unsupported operand type(s) for -: 'str' and 'str'
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

如何判断相等（==）：不报错，说明是成功的



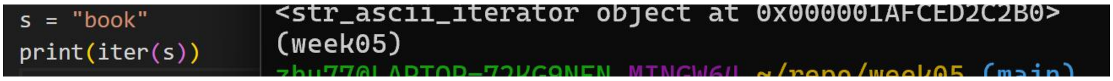
```
s = "aaaa"
try:
    s = s / 2
except TypeError as e:
    print(e)

assert s == "aaaa"
```

```
string index out of range
HELLO
hello
abcshu
unsupported operand type(s) for -: 'str' and 'str'
unsupported operand type(s) for /: 'str' and 'int'
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

对于比较运算符（>、<、>=、<=）有没有支持

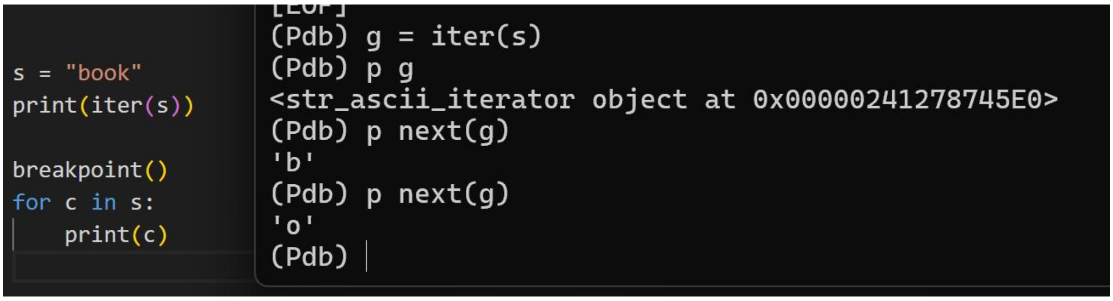
什么值被当作 True，什么值被当作 False

是否可迭代（iterable），如何做迭代（for 循环）



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

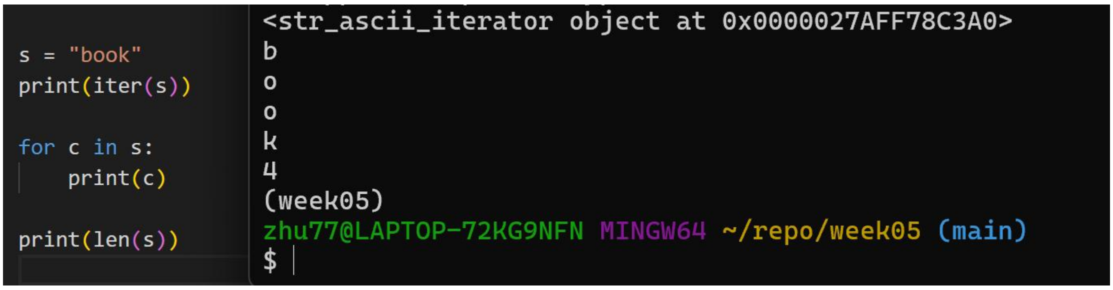
```
<str_ascii_iterator object at 0x000001AFCED2C2B0>
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
```



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

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

```
[EOF]
(Pdb) g = iter(s)
(Pdb) p g
<str_ascii_iterator object at 0x00000241278745E0>
(Pdb) p next(g)
'b'
(Pdb) p next(g)
'o'
(Pdb) 
```

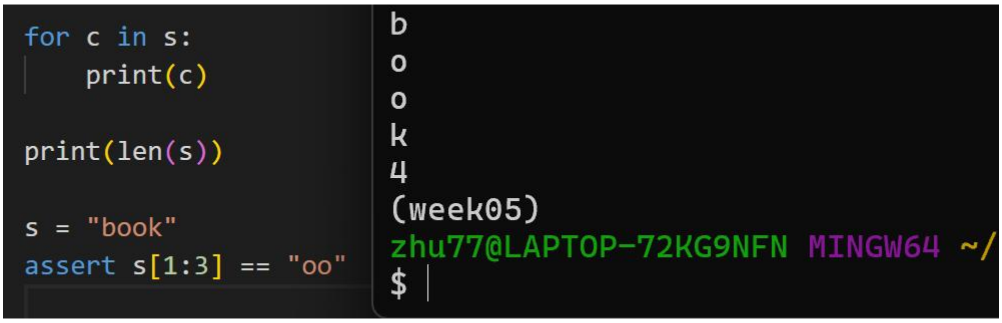
是否支持返回长度（len）



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

for c in s:
    print(c)

print(len(s))
```

```
<str_ascii_iterator object at 0x0000027AFF78C3A0>
b
o
o
k
4
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ 
```

是否（如何）支持提取操作（subscription）([] 运算符)：没报错，就说明成功。[1:3]不包括 3 这个位置的字符。

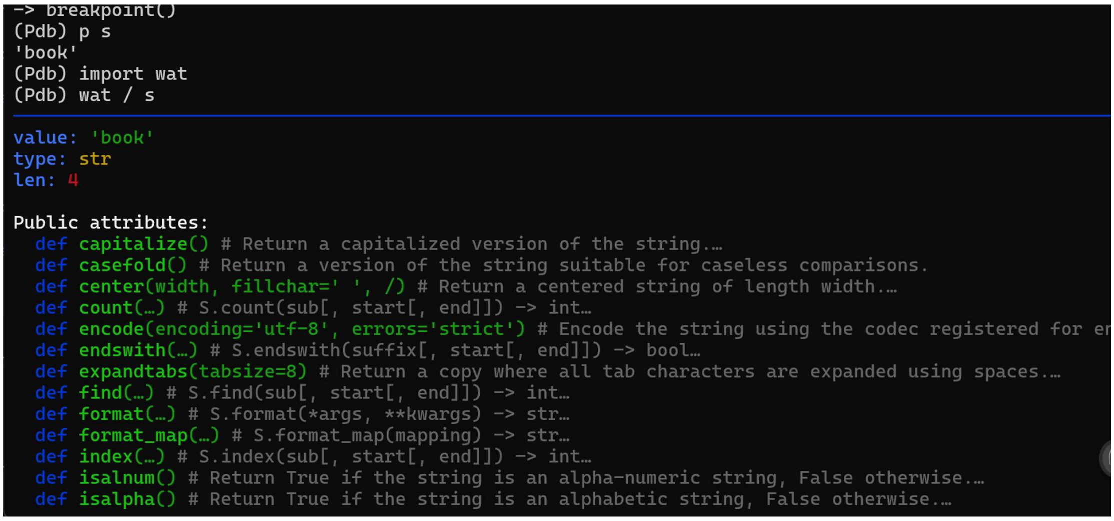


```
for c in s:
    print(c)

print(len(s))

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

```
b
o
o
k
4
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/
$ 
```

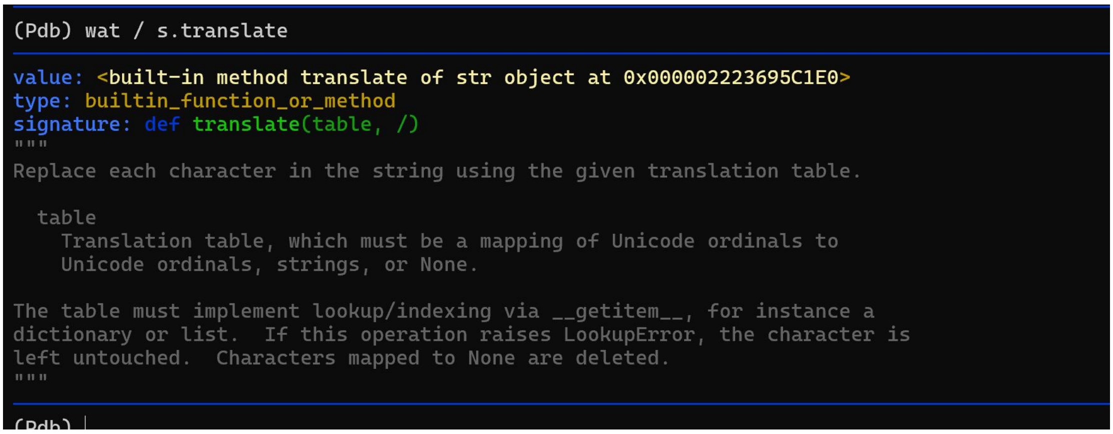
拥有哪些常用方法（method）可供调用（()运算符）

建议先在 pdb 里试验，然后把确定能够运行的代码写在 use_of_{name}.py 文件里

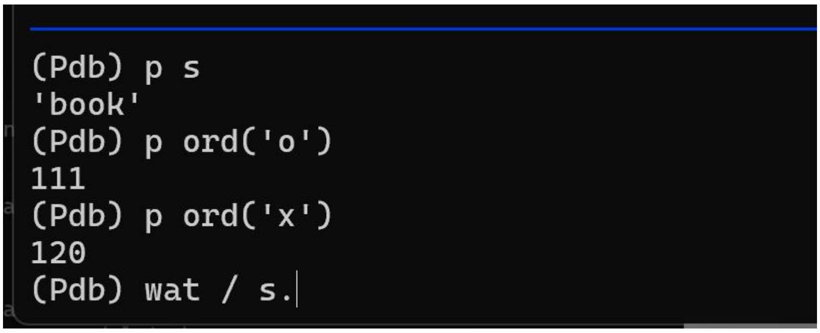


```
-> breakpoint()
(Pdb) p s
'book'
(Pdb) import wat
(Pdb) wat / s

value: 'book'
type: str
len: 4

Public attributes:
  def capitalize() # Return a capitalized version of the string.…
  def casefold() # Return a version of the string suitable for caseless comparisons.
  def center(width, fillchar=' ', /) # Return a centered string of length width.…
  def count(…) # S.count(sub[, start[, end]]) -> int…
  def encode(encoding='utf-8', errors='strict') # Encode the string using the codec registered for en
  def endswith(…) # S.endswith(suffix[, start[, end]]) -> bool…
  def expandtabs(tabsize=8) # Return a copy where all tab characters are expanded using spaces.…
  def find(…) # S.find(sub[, start[, end]]) -> int…
  def format(…) # S.format(*args, **kwargs) -> str…
  def format_map(…) # S.format_map(mapping) -> str…
  def index(…) # S.index(sub[, start[, end]]) -> int…
  def isalnum() # Return True if the string is an alpha-numeric string, False otherwise.…
  def isalpha() # Return True if the string is an alphabetic string, False otherwise.…
```

比如：



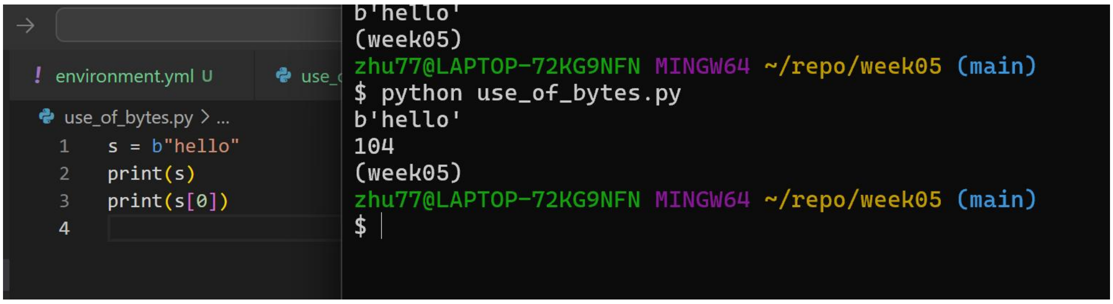
```
(Pdb) wat / s.translate

value: <built-in method translate of str object at 0x000002223695C1E0>
type: builtin_function_or_method
signature: def translate(table, /)
"""
Replace each character in the string using the given translation table.

  table
    Translation table, which must be a mapping of Unicode ordinals to
    Unicode ordinals, strings, or None.

The table must implement lookup/indexing via __getitem__, for instance a
dictionary or list.  If this operation raises LookupError, the character is
left untouched.  Characters mapped to None are deleted.
"""

(Pdb) 
```



```
(Pdb) p s
'book'
(Pdb) p ord('o')
111
(Pdb) p ord('x')
120
(Pdb) wat / s.
```

字节串 bytes

1.



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

```
b'hello'
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$ python use_of_bytes.py
b'hello'
104
(week05)
zhu77@LAPTOP-72KG9NFN MINGW64 ~/repo/week05 (main)
$
```