

金融计算与编程第五周学习笔记

1. 把 environment.yml 文件复制到 week05,

```
MINGW64:/c/Users/ASUS/repo
$ ls -l
total 24
drwxr-xr-x 1 ASUS 197121 0 Mar 19 19:50 myproject/
drwxr-xr-x 1 ASUS 197121 0 Mar 13 17:26 mywork/
drwxr-xr-x 1 ASUS 197121 0 Mar 19 17:48 prj1/
-rw-r--r-- 1 ASUS 197121 2003 Mar 7 23:00 script1.py
drwxr-xr-x 1 ASUS 197121 0 Mar 7 22:43 week01/
drwxr-xr-x 1 ASUS 197121 0 Mar 13 18:01 week02/
drwxr-xr-x 1 ASUS 197121 0 Mar 19 20:27 week03/
drwxr-xr-x 1 ASUS 197121 0 Mar 27 23:05 week04/
drwxr-xr-x 1 ASUS 197121 0 Apr 6 19:54 week05/

(base) ASUS@%C[REDACTED] MINGW64 ~/repo
$ cp week04/environment.yml week05/

(base) ASUS@%C[REDACTED] MINGW64 ~/repo
$ ls -l week05
total 25
-rw-r--r-- 1 ASUS 197121 18805 Apr 6 19:54 LICENSE
-rw-r--r-- 1 ASUS 197121 2239 Apr 6 19:54 README.md
-rw-r--r-- 1 ASUS 197121 91 Apr 6 20:01 environment.yml

(base) ASUS@%C[REDACTED] MINGW64 ~/repo
$
```

2. 在 vs code 里创建 use_of_str.py 文件。

```
MINGW64:/c/Users/ASUS/repo/week05
week04 C:\Users\ASUS\anaconda3\envs\week04
week05 C:\Users\ASUS\anaconda3\envs\week05

(base) ASUS@%C[REDACTED] MINGW64 ~/repo/week05 (main)
$ conda activate week05
(week05)
ASUS@%C[REDACTED] MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
(week05)
ASUS@%C[REDACTED] MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
hello
(week05)
ASUS@%C[REDACTED] MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
1705861143392
(week05)
ASUS@%C[REDACTED] MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2419059612512
(week05)
ASUS@%C[REDACTED] MINGW64 ~/repo/week05 (main)
$
```

```
! environment.yml U .gitignore use_of_str.py U X use_of_bytes.py U
use_of_str.py > ...
1 a = "hello"
2 x = id(a)
3 print(x)
4
```

每次运行的结果不一样，地址是虚拟的，用来显示地址。
但是改为列表以后，id 就不重复了。虽然值是相等的，但不是同一个对象。

```

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

```

```

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

```

```

MINGW64:/c/Users/ASUS/repo/week05
$ conda activate week05
(week05)
ASUS@%C:~$ python use_of_str.py
hello
(week05)
ASUS@%C:~$ python use_of_str.py
1705861143392
(week05)
ASUS@%C:~$ python use_of_str.py
2419059612512
(week05)
ASUS@%C:~$ python use_of_str.py
2502127672160
(week05)
ASUS@%C:~$ python use_of_str.py
2502127672160
(week05)
ASUS@%C:~$ python use_of_str.py
2090440857856
(week05)
ASUS@%C:~$ python use_of_str.py
2090440857856
(week05)
ASUS@%C:~$

```

```

ASUS@%C:~$ python use_of_str.py
hello
(week05)
ASUS@%C:~$ python use_of_str.py
1705861143392
(week05)
ASUS@%C:~$ python use_of_str.py
2419059612512
(week05)
ASUS@%C:~$ python use_of_str.py
2502127672160
(week05)
ASUS@%C:~$ python use_of_str.py
2502127672160
(week05)
ASUS@%C:~$ python use_of_str.py
2090440857856
(week05)
ASUS@%C:~$ python use_of_str.py
2090440857856
(week05)
ASUS@%C:~$

```

id 是不变的。

```

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

```

```

(week05)
ASUS@%C:~$ python use_of_str.py
2391273576704
2391273574720
[9, 5]
[2, 5]
2391273576704
2391273574720
(week05)
ASUS@%C:~$

```

type 可以返回对象的类型。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 (attributes) 的名称列表。

```

ASUS@%C:~$ python use_of_str.py
1525042845952
1525042843968
[9, 5]
[2, 5]
1525042845952
1525042843968
<class 'list'>
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__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
(week05)
ASUS@%C:~$

```

```

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))
14 print("dir(a):", dir(a))
15 print("isinstance(a,str):", isinstance(a, str))
16 print(isinstance(a, (str, float)))
17 print(isinstance(a, (str, float, list)))
18

```

```

ASUS@%C:~$ python use_of_str.py
2917094398208
2917094396224
[9, 5]
[2, 5]
2917094398208
2917094396224
<class 'list'>
False
dir(a): ['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__',
         '__delitem__', '__dir__', '__doc__', '__eq__', 'format', '__ge__', 'get',
         'getattribute', '__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__', '__str__',
         '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index',
         'insert', 'pop', 'remove', 'reverse', 'sort']
isinstance(a,str): False
False
True
(week05)
ASUS@%C:~$

```

如果 assert 语句正确，就会什么也不干进行下一步，错误将会终止。

```

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))
14 print("dir(a):", dir(a))
15 print("isinstance(a,str):", isinstance(a, str))
16 print(isinstance(a, (str, float)))
17 print(isinstance(a, (str, float, list)))
18 assert isinstance(a, list)
19 print("goodbye")
20

```

```

$ python use_of_str.py
2393451403520
2393451401536
[9, 5]
[2, 5]
2393451403520
2393451401536
<class 'list'>
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__',
        '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',
        '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append',
        'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
isinstance(a,str): False
False
True
goodbye
(week05)
ASUS@%C:~$ MINGW64 ~/repo/week05 (main)
$

```

3. 字面值，一般用\n 或者\t。

```

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 = "abc\ndef"
16 print("great", s)
17

```

```

ASUS@%C:~$ MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f -string
name: tom
TAB a    b
(week05)
ASUS@%C:~$ MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
字面值
university
True
f -string
name: tom
TAB a    b
great abc
def
(week05)
ASUS@%C:~$ MINGW64 ~/repo/week05 (main)
$

```


在二进制下面，1.1+2.2 不是整数。

```
24 print("初始化")
25 s = str()
26 print(s)
27 s = str([5, 8, 2])
28 print(s)
29 assert str([5, 8, 2]) == "[5, 8, 2]"
30 assert str(1.1 + 2.2) == "3.3"
31
```

```
[5, 8, 2]
Traceback (most recent call last):
  File "C:\Users\ASUS\anaconda3\envs\week05\Lib\pdb.py", line 1960, in main
    pdb._run(target)
  File "C:\Users\ASUS\anaconda3\envs\week05\Lib\pdb.py", line 1754, in _run
    self.run(target.code)
  File "C:\Users\ASUS\anaconda3\envs\week05\Lib\bdb.py", line 627, in run
    exec(cmd, globals, locals)
  File "<string>", line 1, in <module>
  File "C:\Users\ASUS\repo\week05\use_of_str.py", line 30, in <module>
    assert str(1.1 + 2.2) == "3.3"
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError
Uncaught exception. Entering post mortem debugging
Running 'cont' or 'step' will restart the program
> c:\users\asus\repo\week05\use_of_str.py(30)<module>()
-> assert str(1.1 + 2.2) == "3.3"
(Pdb) p str(1.1+2.2)
'3.3000000000000003'
(Pdb)
```

4. 索引值。正确的不报错，s[3]是索引从左到右的三个字符。

```
34 s = "="
35 x = id(s)
36 s = s * 20
37 y = id(s)
38 print(s)
39 assert x != y
40
41 s = "hello"
42 assert s[3] == "l"
43
```

```
41 s = "hello"
42 assert s[3] == "l"
43 assert s[:3] == "hel"
44
```

字符串的对象不能被修改。

```
(week05)
ASUS04C:~$ python use_of_str.py
字符串
university
True
f-string
name: tom
TAB a    b
great abc
def
njjint
你吃饭了吗
初始化

[5, 8, 2]
=====
HELLO
hello
(week05)
ASUS04C:~$ python use_of_str.py
44  assert s[4] == s[-1]
45
46  s = "hello"
47  u = s.upper()
48  print(u)
49  print(s)
50
```

加法和乘法可以运算。比较字符大小按顺序比较，先比较第一个字符，第一个字符相等再比较第二个，与长短无关。字符串的长度为 0 才会报错。