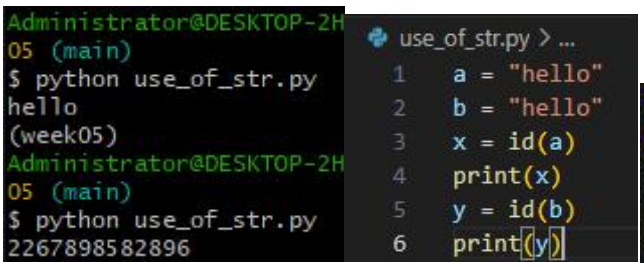
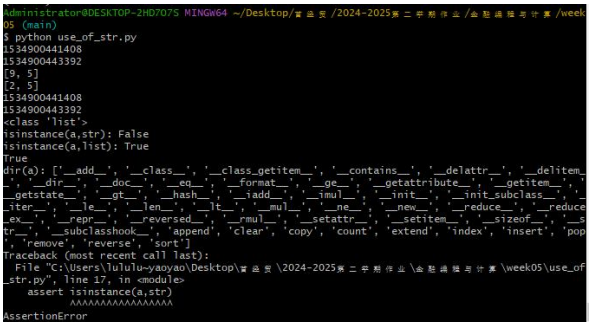
Week05 学习笔记



VScode.py 代码

```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))
print(type(a))
print('isinstance(a,str):',isinstance(a,str))
print('isinstance(a,list):',isinstance(a,list))
print(isinstance(a,(str,list)))
print('dir(a):',dir(a))
try:
    assert isinstance(a,str)
except AssertionError:
    breakpoint()
    print('type error')
print('goodbey')
```

Assert 检验语句是否为真，否则报错 assertionerror



Try 拦截

调出 pdb 调试器

```
$ python use_of_str.py
2353521889600
2353521891584
[9, 5]
[2, 5]
2353521889600
2353521891584
<class 'list'>
isinstance(a,str): False
isinstance(a,list): 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__', '__new__', '__reduce__', '__reduc
_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__
tr__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'po
', 'remove', 'reverse', 'sort']
> c:\users\lululu~yaoyao\desktop\首 经 贸 \2024-2025第 二 学 期 作 业 \金 融 编 程 与 计 算 \week05\use_of_str.
y(21)<module>()
-> print('type error')
(Pdb) p a
[9, 5]
(Pdb) p isinstance(a,str)
False
```

逐个尝试

n 为换行的意思

```
Administrator@DESKTOP-2HD7O7S MINGW64 ~/Desktop/首 经 贸 /2024-2025第 二 学 期 作 业 /金
融 编 程 与 计 算 /week05 (main)
$ python use_of_str1.py
李 国 伟
university
True
f-string
name:Tom
TAB a    b
NEW-line aaa
bbb
```

f - string 格式化

索引值从 0 开始数

字符串不可修改，可以调用返回到新的字符串

字符串不支持“减”“除”运算

```
$ python use_of_str1.py
李 国 伟
university
True
f-string
name:Tom
TAB a    b
NEW-line aaa
bbb
初 始 化

[2, 5, 8]
==========
o
HELLO
(week05)
```

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

assert str(1.1+2.2) != '3.3'

s = '='
u = s*10
print (u)

s='hello'
print(s[4])

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

字符串有独特编码规则，比大小判断

十进制 DEC	八进制 OCT	十六进制 HEX	二进制 BIN	符号 Symbol	HTML 实体编码	中文解释 Description
32	040	20	00100000		&#032;	空格
33	041	21	00100001	!	&#033;	感叹号
34	042	22	00100010	"	&#034;	双引号
35	043	23	00100011	#	&#035;	井号
36	044	24	00100100	$	&#036;	美元符
37	045	25	00100101	%	&#037;	百分号
38	046	26	00100110	&	&#038;	与
39	047	27	00100111	'	&#039;	单引号
40	050	28	00101000	(	&#040;	左括号
41	051	29	00101001	)	&#041;	右括号
42	052	2A	00101010	*	&#042;	星号
43	053	2B	00101011	+	&#043;	加号
44	054	2C	00101100	,	&#044;	逗号
45	055	2D	00101101	-	&#045;	连字号或减号
46	056	2E	00101110	.	&#046;	句点或小数点
47	057	2F	00101111	/	&#047;	斜杠
48	060	30	00110000	0	&#048;	0
49	061	31	00110001	1	&#049;	1
50	062	32	00110010	2	&#050;	2
51	063	33	00110011	3	&#051;	3
52	064	34	00110100	4	&#052;	4
53	065	35	00110101	5	&#053;	5
54	066	36	00110110	6	&#054;	6
55	067	37	00110111	7	&#055;	7
56	070	38	00111000	8	&#056;	8
57	071	39	00111001	9	&#057;	9

1for 循环迭代

```
b
o
o
k
```

字符串调用常见方法

```
s = "Hello, World! "

print(s.upper())
print(s.lower())
print(s.find("or"))
print(s.replace("World", "China"))

text_with_spaces = "   Hello, World!   "
print(text_with_spaces.strip())
print( text_with_spaces.lstrip())
print(text_with_spaces.rstrip())

words = s.split()
print( words)
print(", ".join(words))
print( len(s))
print( s.startswith("Hello"))
print(s.endswith("ld!"))
```

```
HELLO, WORLD!
hello, world!
8
Hello, China!
Hello, World!
Hello, World!
   Hello, World!
['Hello,', 'World!']
Hello,, World!
14
True
False
```

字节符（bytes）

通过 b'...' 字面量语法创建，例如 b = b'hello'；还可以将字符串编码为字节串，如 s = "你

好"; b = s.encode('utf - 8')　decode 解码

```python
# 1. 字节串的创建
# 使用字面量创建
byte_str1 = b'Hello, World!'
print( byte_str1)

# 使用 bytes() 函数创建空字节串
byte_str2 = bytes()
print(byte_str2)

# 使用 bytes() 函数从可迭代对象创建字节串
byte_str3 = bytes([72, 101, 108, 108, 111])  # ASCII 码对应的字符为 'Hello'
print(byte_str3)

# 2. 字符串与字节串的转换
# 字符串编码为字节串
text = "你好，世界！"
byte_str4 = text.encode('utf-8')
print(byte_str4)

# 字节串解码为字符串
decoded_text = byte_str4.decode('utf-8')
print(decoded_text)

# 3. 字节串的操作
# 索引和切片
print( byte_str1[0])  # 返回 ASCII 码值
print( byte_str1[1:5])

# 拼接字节串
byte_str5 = b'Python '
byte_str6 = b'is awesome'
combined_byte_str = byte_str5 + byte_str6
print( combined_byte_str)

# 查找子字节串
index = combined_byte_str.find(b'Python')
```

```
Administrator@DESKTOP-2HD7O7S MINGW64 ~/Desktop/[illegible]/2024-2025[illegible]/
[illegible]/week05 (main)
$ python use_of_bytes.py
b'Hello, World!'
b''
b'Hello'
b'\xe4\xbd\xa0\xe5\xa5\xbd\xef\xbc\x8c\xe4\xb8\x96\xe7\x95\x8c\xef\xbc\x81'
你好，世界！
72
b'ello'
b'Python is awesome'
0
b'Java is awesome'
```

整数（int）

整数表示没有小数部分的数字

使用 int() 函数将其他类型转换为整数，例如 int('123') 会将字符串 '123' 转换为整数 123；int(3.14) 会将浮点数 3.14 转换为整数 3

```
$ python use_of_int.py
10
<class 'int'>
-20
<class 'int'>
0
<class 'int'>
8
2
15
1.6666666666666667
1
2
125
整数 10 转换为浮点数后是 10.0，类型为 <class 'float'>
```

use_of_int.py > ...

```python
num1 = 10  # 赋值一个正整数
print(num1)
print(type(num1))

num2 = -20  # 赋值一个负整数
print(num2)
print(type(num2))

num3 = 0  # 赋值为0
print(num3)
print(type(num3))


a = 5
b = 3
print(a + b)  # 加法运算，输出8
print(a - b)  # 减法运算，输出2
print(a * b)  # 乘法运算，输出15
print(a / b)  # 除法运算，输出1.6666666666666667 ，结果为浮点数
print(a // b)  # 整除运算，输出1
print(a % b)  # 取余运算，输出2
print(a ** b)  # 幂运算，输出125

# 定义一个整数
num_int = 10
# 将整数转换为浮点数
num_float = float(num_int)
print(f"整数 {num_int} 转换为浮点数后是 {num_float}，类型为 {type(num_float)}")
```

浮点数（float）

浮点数表示带有小数部分的数字，采用 IEEE 754 标准表示。

0.1 + 0.2 并不精确等于 0.3。

使用 float() 函数将其他类型转换为浮点数，如 float('3.14') 会将字符串 '3.14' 转换为浮点数 3.14，float(3) 会将整数 3 转换为浮点数 3.0。

use_of_float.py > ...

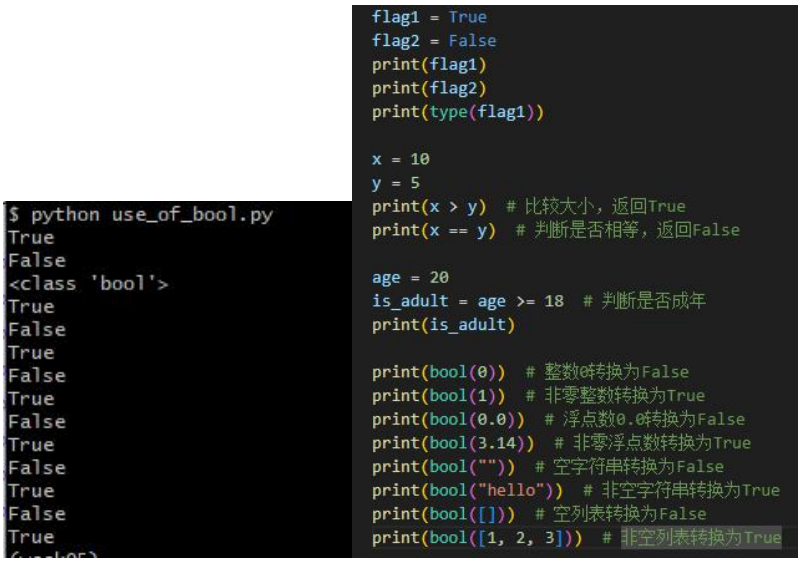
```python
num1 = 3.14  # 普通小数形式
print(num1)
print(type(num1))

num2 = -2.5  # 负浮点数
print(num2)
print(type(num2))

num3 = 0.0  # 浮点数0
print(num3)
print(type(num3))

small_num = 5.6e-8  # 表示0.000000056
print(small_num)

big_num = 2.5e4  # 表示25000
print(big_num)

a = 0.1
b = 0.2
print(a + b)  # 输出结果可能接近0.3，但不一定精确等
```

```
$ python use_of_float.py
3.14
<class 'float'>
-2.5
<class 'float'>
0.0
<class 'float'>
5.6e-08
25000.0
0.30000000000000004
```

布尔值（bool）

布尔值只有两个取值：True（真）和 False（假），用于表示逻辑判断的结果。

转换规则：在条件判断中，0、0.0、空字符串 ""、空列表 []、空字典 {}、空元组 ()、None 等会被转换为 False，其他值（非零数字、非空容器等）会被转换为 True。例如，bool(0) 为 False，bool(1) 为 True，bool("hello") 为 True。



```
flag1 = True
flag2 = False
print(flag1)
print(flag2)
print(type(flag1))

x = 10
y = 5
print(x > y)  # 比较大小，返回True
print(x == y)  # 判断是否相等，返回False

age = 20
is_adult = age >= 18  # 判断是否成年
print(is_adult)

print(bool(0))  # 整数0转换为False
print(bool(1))  # 非零整数转换为True
print(bool(0.0))  # 浮点数0.0转换为False
print(bool(3.14))  # 非零浮点数转换为True
print(bool(""))  # 空字符串转换为False
print(bool("hello"))  # 非空字符串转换为True
print(bool([]))  # 空列表转换为False
print(bool([1, 2, 3]))  # 非空列表转换为True
```

```
$ python use_of_bool.py
True
False
<class 'bool'>
True
False
True
False
True
False
True
False
True
False
True
```

四大容器

列表（list）

列表是一种有序、可变的数据集合，可以存储不同类型的元素。

特点：有序（元素有先后顺序）、可变（可以添加、删除、修改元素）。

常见操作：

索引和切片：与字符串类似，可通过索引访问元素，通过切片获取子列表。例如，lst = [1, 2, 3]; lst[0] 会返回 1，lst[1:3] 会返回 [2, 3]。

增删改：append() 方法用于在列表末尾添加元素；insert() 方法用于在指定位置插入元素；remove() 方法用于删除指定元素；pop() 方法用于弹出指定位置的元素（默认弹出最后一个元素）；可以通过索引直接修改元素，如 lst[0] = 10。

其他方法：extend() 用于合并两个列表，sort() 用于对列表进行排序，reverse() 用于反转列表等。

```
# 列表 (list)
# 创建列表
my_list = [1, 2, 3, 'apple', 'banana']
print("创建的列表:", my_list)

# 访问列表元素
print("列表的第一个元素:", my_list[0])
print("列表的最后一个元素:", my_list[-1])

# 修改列表元素
my_list[1] = 10
print("修改后的列表:", my_list)

# 添加元素到列表末尾
my_list.append('cherry')
print("添加元素后的列表:", my_list)

# 从列表中删除元素
del my_list[3]
print("删除元素后的列表:", my_list)

# 列表切片
print("列表的前三个元素:", my_list[:3])
```

```
$ python use_of_list.py
创建的列表： [1, 2, 3, 'apple', 'banana']
列表的第一个元素： 1
列表的最后一个元素： banana
修改后的列表： [1, 10, 3, 'apple', 'banana']
添加元素后的列表： [1, 10, 3, 'apple', 'banana', 'cherry']
删除元素后的列表： [1, 10, 3, 'banana', 'cherry']
列表的前三个元素： [1, 10, 3]
```

字典（dict）

字典是一种以键值对（key - value pairs）形式存储数据的可变容器，键必须是唯一且可哈希的（如字符串、数字、元组等），值可以是任意类型。

特点：无序（Python 3.7 及以上版本会记住插入顺序，但在逻辑上仍视为无序）、可变。

常见操作：

访问元素：通过键来访问对应的值，例如 d = {'name': 'Alice', 'age': 30}; d['name'] 会返回 'Alice'。

增删改：可以通过 d[key] = value 的形式添加或修改键值对；使用 del d[key] 删除指定键值对；pop(key) 方法也可用于删除并返回指定键对应的值。

其他方法：keys() 方法返回所有键的视图，values() 方法返回所有值的视图，items() 方法返回所有键值对的视图。

```python
# 字典 (dict)
# 创建字典
my_dict = {'name': 'Alice', 'age': 25, 'city': 'New York'}
print("创建的字典:", my_dict)

# 访问字典元素
print("字典中 'name' 对应的值:", my_dict['name'])

# 修改字典元素
my_dict['age'] = 26
print("修改后的字典:", my_dict)

# 添加新的键值对
my_dict['job'] = 'Engineer'
print("添加键值对后的字典:", my_dict)

# 删除字典中的键值对
del my_dict['city']
print("删除键值对后的字典:", my_dict)
```

```
$ python use_of_dict.py
创建的字典： {'name': 'Alice', 'age': 25, 'city': 'New York'}
字典中 'name' 对应的值： Alice
修改后的字典： {'name': 'Alice', 'age': 26, 'city': 'New York'}
添加键值对后的字典： {'name': 'Alice', 'age': 26, 'city': 'New York', 'job': 'Engineer'}
删除键值对后的字典： {'name': 'Alice', 'age': 26, 'job': 'Engineer'}
(week05)
```

元组（tuple）

元组是一种有序、不可变的数据集合，与列表类似，但一旦创建就不能修改其中的元素。

特点：有序、不可变。

创建方式：使用圆括号 () 或直接用逗号分隔元素，例如 t = (1, 2, 3) 或 t = 1, 2, 3（省略圆括号）。

应用场景：常用于存储一组相关但不会改变的数据，比如函数的返回多个值时，通常以元组形式返回。

```python
# 元组 (tuple)
# 创建元组
my_tuple = (1, 2, 'apple', 'banana')
print("创建的元组:", my_tuple)

# 访问元组元素
print("元组的第二个元素:", my_tuple[1])

# 元组不可修改，以下代码会报错
my_tuple[0] = 10

# 元组切片
print("元组的后两个元素:", my_tuple[2:])
```

```
$ python use_of_tuple.py
创建的元组: (1, 2, 'apple', 'banana')
元组的第二个元素: 2
Traceback (most recent call last):
  File "C:\Users\lululu~yaoyao\Desktop\首经贸\2024-2025第二学期作业\金融编程与计
f_tuple.py", line 10, in <module>
    my_tuple[0] = 10
    ~~~~~~~~~~^^^
TypeError: 'tuple' object does not support item assignment
```

集合（set）

集合是一种无序、可变的数据集合，其中的元素是唯一的（不允许重复）。

特点：无序、可变、元素唯一。

常见操作：

添加和删除元素：add() 方法用于添加元素，remove() 方法用于删除指定元素（元素不存在时会报错），discard() 方法也可删除元素（元素不存在时不会报错）。

集合运算：支持交集 &、并集 |、差集 -、对称差集 ^ 等运算，例如 s1 = {1, 2, 3}; s2 = {2, 3, 4}; s1 & s2 会返回 {2, 3}（交集）。

应用场景：常用于去重、判断元素是否存在、集合间的关系运算等。

```python
# 集合 (set)
# 创建集合
my_set = {1, 2, 3, 'apple', 'banana'}
print("创建的集合:", my_set)

# 添加元素到集合
my_set.add('cherry')
print("添加元素后的集合:", my_set)

# 从集合中移除元素
my_set.remove('apple')
print("移除元素后的集合:", my_set)

# 集合的交集操作
set1 = {1, 2, 3}
set2 = {3, 4, 5}
intersection = set1.intersection(set2)
print("集合 set1 和 set2 的交集:", intersection)
```

```
$ python use_of_set.py
创建的集合：{1, 2, 3, 'banana', 'apple'}
添加元素后的集合：{1, 2, 3, 'cherry', 'banana', 'apple'}
移除元素后的集合：{1, 2, 3, 'cherry', 'banana'}
集合 set1 和 set2 的交集：{3}
```