Python 对象类型

一、常用的对象检视函数和语句

1.创建 week05 的项目并安装环境

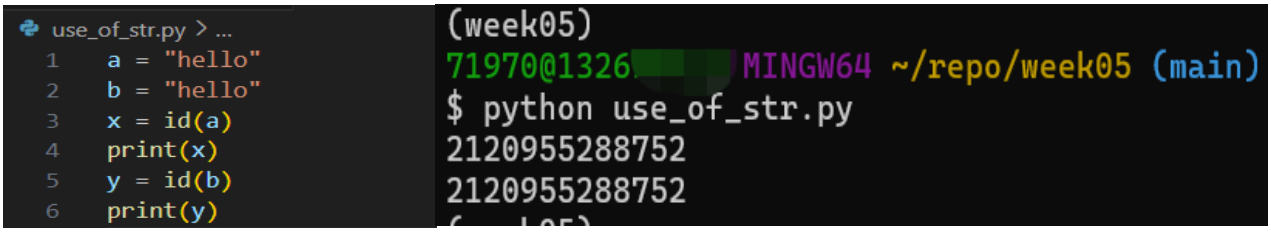
2. Python 对象类型 (type): 包括字符串 (str)、字节串 (bytes)、整数 (int)、浮点数 (float)、布尔值 (bool)、列表 (list)、字典 (dict)、元组 (tuple)、集合 (set)。[Python 解释器 内置的 (built-in)，不需要任何导入 (import)]

3. 创建 use_of_{name}.py 文件，其中 {name} 替换为要求掌握的对象类型，例如 use_of_str.py

4.在 global scope（全局作用域，不做 def）验证概念（PoC）

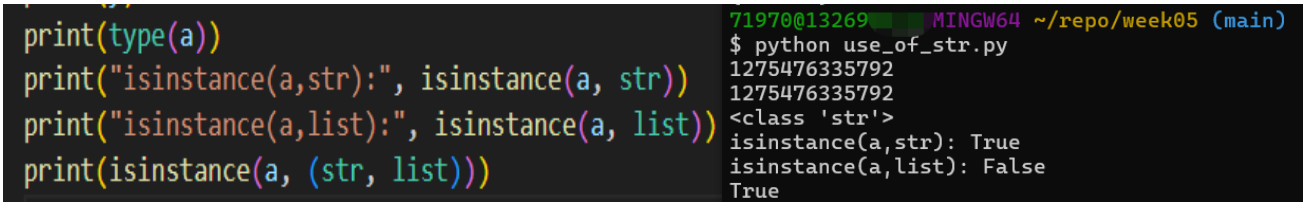
5.用内置函数(built-in function)检视(inspect)对象

①id( )：返回对象在虚拟内存中的地址(每次运行会更新)，如果 id(a) == id(b)，那么 a is b

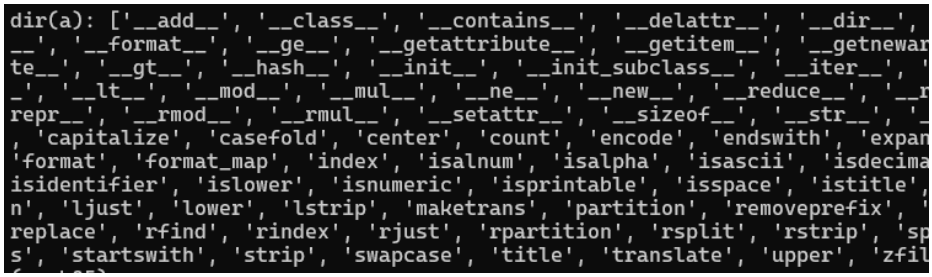


②type( )：返回对象类型

③isinstance( )：判断对象是否属于**某个/些**类型



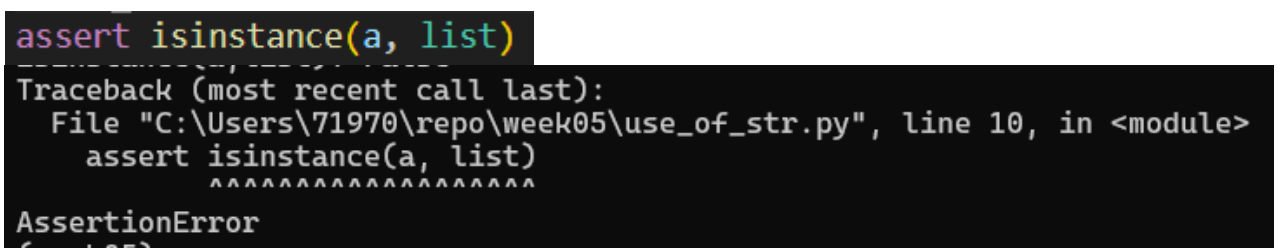
④dir( )：返回对象所支持的属性 (attributes) 的名称列表



⑤str( )：返回对象 print 时要显示在终端的字符串

6.print( )：将表达式 (expression) 输出到终端，查看结果是否符合预期

7.assert 语句：查验某个表达式 (expression) 为真（直接通过），否则报错 (AssertionError) 退出



8.try 语句：拦截报错，避免退出，将流程 (flow) 转入 except 语句

```python
try:
    assert isinstance(a, list)
except AssertionError:
    print("type error")
```

```
71970@132      MINGW64 ~/repo/week05 (main)
$ python use_of_str.py
2550364527792
2550364527792
<class 'str'>
isinstance(a,str): True
isinstance(a,list): False
type error
```

9. breakpoint( )： 在插入处暂停程序运行，进入 pdb 模式

PS：相较于以上介绍的内容，wat 检视则更倾向于交互

二、字符串 str

1.获得 str 类型实例

①字面值（literal）

```python
# 通过字面值获得:
print("通过字面值获得: ")
s = "hello"
print(s)
assert type(s) is str

# try f-string
x = "Blaze"
s = f"name:{x}"
print(s)

# 制表符
s = "a\tb"
print(s)
# 换行符
s = "aaa\nbbb"
print(s)

#三引号允许字符串跨越多行
#且保留格式：字符串中的换行、缩进、空格都会原样输出
s = """xyz
abc
    eee
aaa"""
print(s)
```

```
$ python use_of_str.py
通过字面值获得:
hello
name:Blaze
a       b
aaa
bbb
xyz
abc
    eee
aaa
```

②初始化(init)

```
# 通过初始化获得:
print("通过初始化获得: ")
s = str()  # 空字符串
print(s)

s = str([5, 8, 2])
print(s)
assert str([5, 8, 2]) == "[5, 8, 2]"
assert str(1.1 + 2.2) != "3.3"  # 二进制，不等
```

```
通过初始化获得:

[5, 8, 2]
```

③运算符(operator)

```
# 通过运算值获得:
print("通过运算值获得: ")
s = "="
x = id(s)
s = s * 20  # 此时s指向的对象内存地址变化了
y = id(s)
print(s)
assert x != y
```

```
通过运算值获得:
====================
```

④提取值(subscription)

```
# 通过提取值获得:
print("通过提取值获得: ")
s = "hello"
assert s[3] == "l"
assert s[:3] == "hel"
print(s[-1])
try:
    s[5]
except IndexError as e:
    print(e)
```

```
通过提取值获得:
o
string index out of range
```

⑤返回值(return value of function/method call)

```
# 通过返回值获得:
print("通过返回值获得: ")
s = "hello"
u = s.upper()
print(u)
print(s)

t = "name:{},age:{}"
print(t)
T = t.format("Blaze", 21)  # 并没有修改t
print(T)
```

```
通过返回值获得:
HELLO
hello
name:{},age:{}
name:Blaze,age:21
```

2.str 其他属性和操作

```python
# 1. 数学运算符
# 支持 + (拼接) 和 * (重复)
s1 = "Hello"
s2 = "World"
print(s1 + s2)
print(s1 * 3)
# 不支持 - / // % ** @

# 2. 比较运算
print("abc" == "abc")
print("abc" != "ABC")
print("apple" < "banana")  # 字典序 (排在前的小)
print("Cat" < "cat")  # 基于ASCII

# 3. 布尔值
# 空字符串为False, 非空为True
print(bool(""))  # False
print(bool(" "))  # True
print(bool("0"))  # True

# 4. 可迭代性
# 可用for循环遍历
s = "book"
print(iter(s))

for c in s:
    print(c)


# 5. 长度和索引
# 支持len()和下标访问
s = "Python"
print(len(s))  # 6
print(s[0])  # P
print(s[-1])  # n
print(s[1:4])  # yth

# 6. 常用方法
# 大小写转换
print("Hello".lower())  # hello
print("hello".upper())  # HELLO
```