# print("我是小小柴犬")

# s = "我是中国�?"

# print(s)

# print("f-string")

# x = "tom"

# s = f"name:{x}"

# print(s)

# 1. use\_of\_str.py (字符串类�?)

# """字符串类型的基本操作验证"""

# 创建方式

# -\*- coding: utf-8 -\*-

import io

import sys

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding="utf-8")

s1 = "hello"  # 字面�?

s2 = str(3.14)  # 初始�?

s3 = f"PI={s2}"  # f-string

s4 = "world".upper()  # 方法返回�?

# 基本检�?

assert id(s1) == id(s1)  # 相同对象

assert isinstance(s1, str)  # 类型检�?

assert "upper" in dir(s1)  # 方法存在性检�?

# 运算符验�?

assert "py" + "thon" == "python"

assert "ha" \* 3 == "hahaha"

assert "a" < "b"  # 比较运算

# 索引和迭�?

assert s1[0] == "h"  # 索引

assert [c for c in s1] == ["h", "e", "l", "l", "o"]  # 迭代

# 常用方法

assert "HELLO".lower() == "hello"

assert "  spaces  ".strip() == "spaces"

# 2.use\_of\_list.py (列表类型)

# python

# """列表类型的基本操作验�?"""

# 创建方式

lst1 = [1, 2, 3]  # 字面�?

lst2 = list("abc")  # 初始�?

lst3 = [x \* 2 for x in lst1]  # 推导�?

lst4 = lst1  # 方法返回�?

# 类型检�?

assert type(lst1) is list

assert isinstance(lst1, (list, object))

# 运算符验�?

assert [1] + [2] == [1, 2]

assert [0] \* 3 == [0, 0, 0]

assert lst1[1:] == [2, 3]  # 切片操作

# 迭代和长�?

assert len(lst1) == 3

assert sum(lst1) == 6  # 可迭代求�?

# 方法测试

lst1.append(4)

assert lst1 == [1, 2, 3, 4]

lst1.reverse()

assert lst1 == [4, 3, 2, 1]

# use\_of\_dict.py (字典类型)

# python

# """字典类型的基本操作验�?"""

# 创建方式

d1 = {"a": 1, "b": 2}  # 字面�?

d2 = dict(a=1, b=2)  # 初始�?

d3 = {k: v \* 2 for k, v in d1.items()}  # 推导�?

# 检视操�?

assert "a" in d1  # 键存在检�?

assert list(d1.keys()) == ["a", "b"]

assert str(d1) == "{'a': 1, 'b': 2}"  # str()输出

# 索引操作

try:

    print(d1["c"])  # 捕获KeyError

except KeyError:

    print("Key不存在")

# 更新操作

d1.update({"c": 3})

assert len(d1) == 3

assert d1.get("d", 0) == 0  # 安全访问

import operator  # noqa: E402

from collections.abc import Iterable  # noqa: E402

# ===== 实例创建 =====

# [此处放置该类型的多种创建方式示例]

# ===== 核心验证 =====

def validate\_type(obj):

    """执行流程"""

    print(f"\n=== 验证 {type(obj).\_\_name\_\_} 行为 ===")

    # 1. 运算符支持验�?

    print("\n[运算符支持]")

    for op in ["+", "-", "\*", "/", "//", "%", "@"]:

        try:

            func = getattr(operator, op)

            result = func(obj, obj) if op != "@" else func(obj, ...)  # @运算符特殊处�?

            print(f"{op} 运算结果: {result}")

        except (TypeError, AttributeError):

            print(f"不支�? {op} 运算�?")

    # 2. 相等判断

    print("\n[相等判断]")

    copy = eval(repr(obj))  # 通过repr重建对象

    print(f"== 判断: {obj == copy} (预期True)")

    print(f"is 判断: {obj is copy} (通常False)")

    # 3. 比较运算

    print("\n[比较运算]")

    for op in [">", "<", ">=", "<="]:

        try:

            func = getattr(operator, op)

            print(f"{op} 比较: {func(obj, copy)}")

        except TypeError:

            print(f"不支�? {op} 比较")

    # 4. 布尔转换

    print("\n[布尔转换]")

    print(f"bool() 结果: {bool(obj)}")

    print(f"not 运算: {not obj}")

    # 5. 可迭代�?

    print("\n[可迭代性]")

    print(f"是否可迭�?: {isinstance(obj, Iterable)}")

    if isinstance(obj, Iterable):

        print(f"迭代结果: {[item for item in obj]}")

    # 6. 长度支持

    print("\n[长度支持]")

    try:

        print(f"长度: {len(obj)}")

    except TypeError:

        print("不支�? len()")

    # 8. 常用方法

    print("\n[方法列表]")

    methods = [m for m in dir(obj) if not m.startswith("\_")]

    print(f"�?5个方�?: {methods[:5]}...")

# 比较大小

print("abc" > "ABC")

s = "look in my eyes"

for a in s:

    print(a)

print(len(s))

print("好")

x = 3

y = 8

assert y // x == 2

y // x

a = [14, 36, 9]

b = [i\*\*2 for i in a]

print(b)