Python基础3：容器类型

一、列表

1.【重点】列表的定义和使用

1.1. 【重点】列表的定义和使用

```
"""
列表定义的格式:
列表变量的名字 = [元素1, 元素2, ……]

使用格式:
    列表变量[位置]
    位置:也叫下标、索引 (可以是正数或者负数)
"""

# 列表定义
name_list = ['rose', 'tom', 'mike', 'lily',
1, 2, 3]
#                0       1     2       3
4   5   6
#                                      ...
-2   -1
# 最后一个元素取出来
print(name_list[6])
print(name_list[-1])

# 访问列表元素,不要超出范围,不要越界
```

```
# print(name_list[10])    # IndexError:
list index out of range
```

1.2. 【重点】列表的嵌套定义和使用

```
school_list = [['北京大学', '清华大学'], ['中
山大学', '华南理工大学'], ['哈工大', '哈工程']]
#                    0                          1
                           2


# 取出第 2 个元素，也就是索引为1的元素
print(school_list[1])    # ['中山大学', '华南
理工大学']

# 先取出索引为1的元素，再继续索引为1的元素
print(school_list[1][1])     # 华南理工大学
print(school_list[0][1])     # 清华大学
```

2.【知道】列表常用操作

分类	关键字 / 函数 / 方法	说明
增加	列表.append(值)	在末尾追加数据
删除	列表.remove(值)	删除第一个出现的指定数据
修改	列表[索引] = 值	修改指定索引的数据，数据不存在会报错
查询	列表[索引]	根据索引取值，索引不存在会报错
	len(列表)	列表长度(元素个数)
	if 值 in 列表:	判断列表中是否包含某个值

2.1.【知道】增删操作

```
"""
列表.append(值)    在末尾追加数据
列表.remove(值)    删除第一个出现的指定数据
"""
```

```
tmp_list = [1,2,3]
print(tmp_list)

# 增加一个元素，值为4
tmp_list.append(4)
print(tmp_list)

# 删除值为2的元素
tmp_list.remove(2)
print(tmp_list)
```

2.2. 【知道】修改查询

```
"""
列表[索引] = 值   修改指定索引的数据，数据不存在会报错
列表[索引]          根据索引取值，索引不存在会报错
len(列表)       列表长度(元素个数)
if 值 in 列表: 判断列表中是否包含某个值
"""

tmp_list = [1, 2, 3, 3]
print(tmp_list)

# 最后一个元素，修改为5
```

```
tmp_list[-1] = 5
print(tmp_list)

# 访问索引为1的元素
print(tmp_list[1])
# print(tmp_list[5])  # IndexError: list
index out of range

# len(列表)      列表长度(元素个数)
_len = len(tmp_list)
print(_len)

# 判断3，是否在列表中
if 3 in tmp_list:
    print("3在列表中")
else:
    print("3不在列表中")
```

2.3. 【知道】排序

```
my_list = [10, 20, 13]

# 列表.sort() 升序排序
my_list.sort()
print(my_list)

# reverse=True, 降序排序
my_list.sort(reverse=True)
print(my_list)
```

3.【重点】列表的遍历

3.1 【重点】通过for循环遍历

```
# 遍历，从头开始找，直到结束

name_list = ['tom', 'rose', 'lily', 'yoyo',
'mike']

# 通过while实现遍历
# 1. 定义条件变量i = 0
i = 0
# 2. while i < 列表元素个数:
while i < len(name_list):
```

```
    # 3. 取出某个元素，打印
    name = name_list[i]
    print(name)

    # 4. 条件变量的修改
    i += 1

print('==========华丽分割线==========')

# for遍历循环，和上面的while效果等价
# 从头到尾 依次从 列表 中取出 每一个元素
# for 变量 in 列表:   依次从列表中取出每一个元素赋值给变量
#     print(变量)

for name in name_list:
    print(name)
```

3.2 【重点】if...in 和 for...in 的区别

```
# if…in：判断某个元素是否在列表中，如果在，if的条件为True
# for…in：从头到尾 依次从 列表 中取出 每一个元素，这个元素给name赋值
```

```
name_list = ['tom', 'rose', 'lily', 'yoyo',
'mike']

name = 'rose'

# if…in：判断某个元素是否在列表中，如果在，if的条
件为True
if name in name_list:
    print("%s在列表中" % name)


# for…in：从头到尾 依次从 列表 中取出 每一个元
素，这个元素给name赋值
for name in name_list:
    print(name)
```

3.3 【重点】for...else的使用

```
name_list = ['tom', 'rose', 'lily', 'yoyo',
'mike']
```

```
# for循环中没有break，for循环执行结束，执行else的代码块
for name in name_list:
    print(name)

    if name == "rose":
        break

else:
    print("for循环中没有break，for循环执行结束，执行else的代码块")
```

二、元组

1.【重点】元组的定义和使用

```
# 元组：元组变量 = (元素1, 元素2, ……)
my_tuple = ('rose', 'tom', 'lily', 'mike')
print(my_tuple)

# 元组只有一个元素的时候，格式：(元素,)
# my_tuple2 = (250)
# print(type(my_tuple2))  # <class 'int'>

my_tuple2 = (250, )
print(type(my_tuple2))  # <class 'tuple'>
```

2.【知道】元组和列表的区别

- 元组的 **元素不能修改**，列表中的元素可以修改

```
# 元组只有一个元素的时候，格式：(元素,)
# my_tuple2 = (250)
# print(type(my_tuple2))  # <class 'int'>

my_tuple2 = (250, )
print(type(my_tuple2))  # <class 'tuple'>

# 元组的元素只读，不能改
```

```
# my_tuple[0] = 'yoyo'    # TypeError:
'tuple' object does not support item
assignment

print(len(my_tuple))

if "rose" in my_tuple:
    print("rose在元组中")
# 循环遍历和列表一样
for name in my_tuple:
    print(name)
```

三、字典

1.【重点】字典的定义和使用

```
"""
字典定义格式:
    字典变量 = {k1:v1, k2:v2,……}

取出元素的值:
    字典变量[键值]
"""
```

```python
# 字典的定义
student_dict = {"name": "rose", "age": 18,
"sex": "male"}

# 取出元素的值：字典变量[键值]
print(student_dict['age'])
print(student_dict['name'])
print(student_dict['sex'])
```

2. 【知道】字典常用操作

分类	关键字 / 函数 / 方法	说明
增加	字典[键] = 值	键不存在，会添加键值对
删除	字典.pop(键)	删除指定键值对,返回被删除的值,如果键不存在,会报错
修改	字典[键] = 值	键存在，会修改键值对的值
查询	字典[键]	根据键取值，键值对不存在会报错
	字典.get(键)	根据键取值，键值对不存在返回None, 不会报错
	for key, value in 字典.items()	遍历字典, 获取所有的键值对(键, 值)

2.1. 【知道】增加删除

```
"""
字典[键] = 值        键不存在，会添加键值对
字典.pop(键)         删除指定键值对,返回被删除的值,如果键不存在,会报错
"""
```

```
# 字典定义
student_dict = {"name": "rose", "age": 18,
"sex": "male"}
print(student_dict)

# 字典[键] = 值 键不存在，会添加键值对
student_dict['class'] = 'python40期'
print(student_dict)

# 字典.pop(键)      删除指定键值对,返回被删除的
值,如果键不存在,会报错
old_value = student_dict.pop("sex")
print(student_dict)
print(old_value)

# student_dict.pop("sex")  # KeyError:
'sex'
```

2.2. 【知道】修改查询

```
"""
字典[键] = 值      键存在，会修改键值对的值
字典[键]           根据键取值，键值对不存在会报错
字典.get(键)       根据键取值，键值对不存在返回
None，不会报错
```

```
"""

student_dict = {"name": "rose", "age": 18,
"sex": "male"}
print(student_dict)

# 字典[键] = 值 键存在，会修改键值对的值
student_dict['age'] = 20
print(student_dict)

# 字典[键]          根据键取值，键值对不存在会报错
print(student_dict['name'])
# print(student_dict['class'])    #
KeyError: 'class'

print(student_dict.get('name'))
print(student_dict.get('class'))  # None


#扩展：get("key", 默认值) 如果key存在,返回
value,如果不存在,返回默认值
print(student_dict.get("class", "班级待定"))
```

3. 【知道】字典遍历

```
"""
# 遍历字典，获取所有的键值对 (键，值)
for k, v in 字典变量.items():
    print(k, v)
"""

student_dict = {"name": "rose", "age": 18,
"sex": "male"}

# 直接使用for变量，依次获取字典的key
for key in student_dict:
    print("key: %s " % key)
    print("value: ", student_dict[key])

# 遍历字典，获取所有的键值对 (键，值)
# for k, v in 字典变量.items():
#     print(k, v)
for key, value in student_dict.items():
    print("key: ", key, " value: ", value)
```

4. 【应用】案例：登录注册系统

4.1. 【理解】登录注册系统-思路分析

```
"""
需求：1.用户注册 2.用户登录
用户1：{'name'：'张三', 'pwd'：123456}
用户2：{'name'：'李四', 'pwd'：123456}
.....

用户管理是列表：
[{'name'：'张三', 'pwd'：123456}, {'name'：
'李四', 'pwd'：123456}]
"""
"""
# 1. 注册功能，新增加一个用户
# 1.1 用户信息
#    1.1.2. 判断某个用户，是否在列表中
#    1.1.3. 找到用户,说明用户已经注册了,不允许再次
注册
# 1.2 创建一个字典
# 1.3 追加字典到列表中
"""

"""
# 2. 判断某个用户，是否在列表中
```

```
# 2.1 需要找的用户
# 2.2 通过for遍历列表，取出的每个元素是字典
    # 2.3 字典['name']和reg_name比较是否相等
        # 2.4 如果相等，打印提示名字在列表中
        # 2.5 跳出循环
# 2.6 for循环的else，循环里面没有执行到break，则会执行else
    # 2.7 打印，名字不在列表中
"""

# 3. 登陆功能：和上面流程差不多，同时判断用户名和密码是否相等
```

4.2. 【应用】登录注册系统-主界面逻辑

```
"""
登录注册系统需求：1.用户注册/ 2.用户登录/ 3.退出程序

# 1. 死循环 while True:
    # 2. 输入数字指令
    # 3. 判断指令，选择分支
"""
```

```
# 1. 死循环 while True:
while True:
    # 2. 输入数字指令
    num = int(input("登录注册系统需求：1.用户注册/ 2.用户登录/ 3.退出程序："))
    # 3. 判断指令，选择分支
    if num == 1:
        pass
    elif num == 2:
        pass
    elif num == 3:
        print("结束程序")
        break
    else:
        print("输入错误,请重新输入")
```

4.3. 【应用】登录注册系统-用户注册

```
"""
登录注册系统需求：1.用户注册/ 2.用户登录/ 3.退出程序

# 0. 定义一个列表，用于存储用户字典
```

```
# 1. 死循环 while True:
    # 2. 输入数字指令
    # 3. 判断指令，选择分支
    # 4. 用户注册功能
        # 4.1 输入注册的用户名
        # 4.2 通过for遍历列表，取出的每个元素是字典
            # 4.3 字典['name']和输入注册的用户名比较是否相等
                # 4.4 如果相等，打印提示：名字在列表中，不允许注册
                # 4.5 跳出循环
        # 4.6 for循环的else，循环里面没有执行到break，则会执行else
            # 4.7 输入注册的密码
            # 4.8 创建一个字典
            # 4.9 字典追加到列表中
            # 4.10 打印：注册成功
"""

# 0. 定义一个列表，用于存储用户字典
user_list = [{"name": "rose", "password": "123"}, {"name": "mike", "password": "456"}]
```

```
while True:
    num = int(input("请输入操作： 1.用户注册/ 2.用户登录/ 3.退出程序 ： "))
    if num == 1:
    # 4. 用户注册功能
        # 4.1 输入注册的用户名
        user_name = input("请输入需要注册的用户名： ")
        # 4.2 通过for遍历列表，取出的每个元素是字典
        for user_dict in user_list:
            # 4.3 字典['name']和输入注册的用户名比较是否相等
            if user_dict['name'] == user_name:
                # 4.4 如果相等，打印提示：名字在列表中，不允许注册
                print("名字在列表中，不允许注册")
                # 4.5 跳出循环
                break
        # 4.6 for循环的else，循环里面没有执行到break，则会执行else
        else:
            # 4.7 输入注册的密码
```

```
            user_password = input("请求输入注册密码: ")
            # 4.8 创建一个字典
            new_dict = {"name": user_name, "password": user_password}
            # 4.9 字典追加到列表中
            user_list.append(new_dict)
            # 4.10 打印: 注册成功
            print("注册成功")
            # 测试是否注册成功:
            # print(user_list)
    elif num == 2:
        pass
    elif num == 3:
        print("退出程序")
        break
    else:
        print("输入错误,重新输入")
```

4.4. 【应用】登录注册系统-用户登录

```
"""
登录注册系统需求：1.用户注册/ 2.用户登录/ 3.退出程序

# 0. 定义一个列表，用于存储用户字典

# 1. 死循环 while True:
    # 2. 输入数字指令
    # 3. 判断指令，选择分支
    # 4. 用户注册功能
        # 4.1 输入注册的用户名
        # 4.2 通过for遍历列表，取出的每个元素是字典
            # 4.3 字典['name']和输入注册的用户名比较是否相等
                # 4.4 如果相等，打印提示：名字在列表中，不允许注册
                # 4.5 跳出循环
        # 4.6 for循环的else，循环里面没有执行到break，则会执行else
            # 4.7 输入注册的密码
            # 4.8 创建一个字典
            # 4.9 字典追加到列表中
            # 4.10 打印：注册成功
```

```
    # 5. 用户登陆功能
        # 5.1 输入登陆的用户名和密码
        # 5.2 通过for遍历列表，取出的每个元素是字典
            # 5.3 字典['name']和登陆用户名比较 and 字典['pwd']和登陆密码比较
                # 5.4 如果都相等，打印提示：登陆成功
                # 5.5 跳出循环
        # 5.6 for循环的else，循环里面没有执行到break，则会执行else
            # 5.7 打印：用户名或密码错误，请重新登陆
"""

# 0. 定义一个列表，用于存储用户字典
user_list = [{"name": "rose", "password": "123"}, {"name": "mike", "password": "456"}]

# 1. 死循环 while True:
while True:
    # 2. 输入数字指令
    num = int(input("请输入操作：1.用户注册/ 2.用户登录/ 3.退出程序 ："))
    # 3. 判断指令，选择分支
```

```
    if num == 1:
    # 4. 用户注册功能
        # 4.1 输入注册的用户名
        user_name = input("请输入需要注册的用户名: ")
        # 4.2 通过for遍历列表，取出的每个元素是字典
        for user_dict in user_list:
            # 4.3 字典['name']和输入注册的用户名比较是否相等
            if user_dict['name'] == user_name:
                # 4.4 如果相等，打印提示：名字在列表中，不允许注册
                print("名字在列表中，不允许注册")
                # 4.5 跳出循环
                break
        # 4.6 for循环的else，循环里面没有执行到break，则会执行else
        else:
            # 4.7 输入注册的密码
            user_password = input("请求输入注册密码: ")
            # 4.8 创建一个字典
```

```
            new_dict = {"name": user_name,
"password": user_password}
            # 4.9 字典追加到列表中
            user_list.append(new_dict)
            # 4.10 打印：注册成功
            print("注册成功")
            # 测试是否注册成功：
            # print(user_list)
    elif num == 2:
    # 5. 用户登陆功能
        # 5.1 输入登陆的用户名和密码
        user_name = input("请输入登录的用户名：
")
        user_password = input("请输入登录的密
码： ")
        # 5.2 通过for遍历列表，取出的每个元素是
字典
        for user_dict in user_list:
            # 5.3 字典['name']和登陆用户名比较
and 字典['pwd']和登陆密码比较
            if user_dict['name'] ==
user_name and user_dict['password'] ==
user_password:
                # 5.4 如果都相等，打印提示：登
陆成功
                print("登陆成功")
```

```
                    # 5.5 跳出循环
                    break
            # 5.6 for循环的else，循环里面没有执行到break，则会执行else
            else:
                # 5.7 打印：用户名或密码错误，请重新登陆
                print("用户名或密码错误，请重新登陆")

    elif num == 3:
        print("退出程序")
        break
    else:
        print("输入错误,重新输入")
```

四、字符串

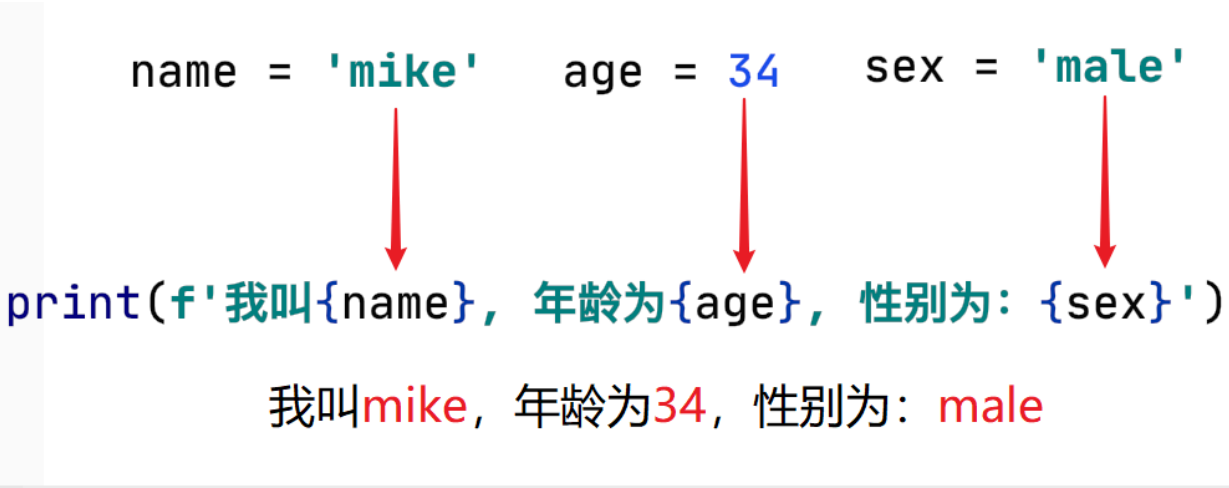
1.【重点】字符串的定义和使用

```
"""
字符串变量 = '字符串内容'
说明：可以是单引号、双引号、三引号
```

```
"""

my_str = "python"
#          012345
print(my_str)

# 说明：可以是单引号、双引号、三引号
my_str2 = 'python' \
          '.com'
print(my_str2)

my_str3 = '''
python
itcast
itheima
....
'''
print(my_str3)


# 取某个元素，和列表一样
print(my_str[0])

# 遍历取所有元素
for c in my_str:
    print(c)
```

```
# 注意：嵌套使用单引号和双引号
my_str4 = "字符串为'python'"
print(my_str4)
```

2.【了解】f-strings 字符串格式化




```
name = "tom"
age = 20
sex = "male"

# 格式化字符串
print("姓名: %s 年龄: %d 性别: %s" % (name,
age, sex))

# fstrings的格式化字符串
```

```
print(f"姓名：{name} 年龄：{age} 性别：
{sex}")
print(F"姓名：{name} 年龄：{age} 性别：
{sex}")


# 扩展：format的格式化字符串
print("姓名：{} 年龄：{} 性别：
{}".format(name, age, sex))
```

3. 【知道】字符串常用操作

分类	关键字 / 函数 / 方法	说明
查找	字符串.find(目标字符串, 开始索引, 结束索引)	在指定范围内, 查询目标字符串的索引, 不存在返回-1
替换	字符串.replace(原内容, 新内容, 替换次数)	返回一个替换了原内容的新字符串，可以指定替换次数
分割	字符串.split(分割符)	以分割符拆分字符串, 返回列表
拼接	字符串 + 字符串	拼接两个字符串
	字符串.join(字符串列表)	以字符串来连接字符串列表中每个元素，合并为一个新的字符串

3.1. 【知道】查询

```
"""
字符串.find(目标字符串，开始索引，结束索引) 在指定范围内，查询目标字符串的索引，不存在返回-1
"""
```

```
str1 = "hello abc python"
#       012345678911
#                   01

# 在str1字符串中，查找'abc'字符串，返回找到字符串所对应的索引
index = str1.find("abc")
print(index)

# 如果没有找到，返回-1
index = str1.find("88")
print(index)

# 指定区间查找子串，从索引2到索引11-1为止
index = str1.find("py", 2, 11)  # [2,11) 包含开始索引,不包含结尾索引
print(index)
```

3.2. 【知道】替换

```
"""
字符串.replace(原内容，新内容，替换次数)  返回一个替换了原内容的新字符串，可以指定替换次数
"""
```

```
str1 = "hello python2.5 python3.4 python3.5 python3.6 python3.8"

# 源字符串里的py，替换为Py，返回值才是替换后的内容
new_str = str1.replace("py", "Py")
print(str1)
print(new_str)

new_str = str1.replace("py", "Py", 2)
print(str1)
print(new_str)
```

3.3. 【知道】分割

```
# 字符串.split(分割符)  以分割符拆分字符串，返回列表

str1 = "hello python hello world"

# str1中字符串内容以' '分隔，返回字符串列表 ['hello', 'python', 'hello', 'world']
new_list = str1.split(" ")
print(str1)
print(new_list)
```

```
# 扩展1: \t: tab键 \n: 回车键
str2 = "hello python\thello\nworld"
new_list = str2.split()  # split() 不指定分割字符,会以空白字符作为分割, 空白字符:空格,tab键,回车换行键.
print(new_list)


# 扩展2: split("分隔符", maxsplit=最大分割次数)
str3 = "hello python hello world"
new_list = str3.split(" ", maxsplit=2)
print(new_list)


# 注意:光标移动到查询的函数, ctrl+b, 可以进入代码帮助
```

3.4. 【知道】拼接

```
"""
字符串 + 字符串                拼接两个字符串
字符串.join(字符串列表) 以字符串来连接字符串列表中每个元素, 合并为一个新的字符串, 返回新的字符串
"""
```

```
# 多个字符串连接再一起

# 字符串 + 字符串            拼接两个字符串
str1 = "hello "
str2 = "world "
str3 = "python"

new_str = str1 + str2 + str3
print(new_str)

# 字符串.join(字符串列表) 以字符串来连接字符串列表中每个元素，合并为一个新的字符串，返回新的字符串

str4 = ","
str_list = ['rose', 'tom', 'lily', 'mike']

new_str = str4.join(str_list)
print(new_str)
```

4.【重点】字符串的切片操作

- 字符串[开始位置:结束位置:步长]
  - 步长默认为1，步长理解为走几步
  - **字符串[开始位置:结束位置]**： 开始位置 ~（结束位置-1）

- **字符串[开始位置:]**： 开始位置 ~ 结束位置，末尾位置不写，默认能取到末尾那个位置
- **字符串[:结束位置]**： 0 ~（结束位置-1），开始位置不写，默认从第0个元素开始

```
"""
切片适用于：字符串,元组,列表
字符串[开始索引:结束索引:步长]
0．步长默认为1，步长理解为走几步，正数从左往右，负数从右往左
1．字符串[开始索引:结束索引]    开始索引 ~（结束索引-1）
2．字符串[开始索引： ]          开始索引 ~ 结束索引，末尾索引不写，默认能取到末尾那个索引
3．字符串[ :结束索引]           0 ~（结束索引-1），开始索引不写，默认从第0个元素开始
"""

my_str = "123456789"
#         012345678
#                -2-1

# 截取从 2 ~ 5 位置 的字符串
print(my_str[1:5])  # 取出下标1~4的所有字符
# 截取从 2 ~ 末尾 的字符串
print(my_str[1:])
```

```
# 截取从 开始 ~ 5 位置 的字符串
print(my_str[:5])    # 取出下标0-4的所有字符
# 从开始位置，每隔一个字符截取字符串，也就是说走2步
print(my_str[::2])
# 截取字符串末尾两个字符
print(my_str[-2:])
# 字符串的逆序（面试题）
print(my_str[::-1])
```

五、集合

1.【知道】通过set对list中的元素去重

- 通过集合完成对列表中元素的去重功能：**set(列表)**

```
# 集合：元素不会重复，{元素1，元素2……}
# 定义集合
# 通过集合完成对列表去重功能

my_set = {1, 2, 1, 2, 1, 2}
print(my_set)
print(type(my_set))  # <class 'set'>
```

```python
# 注意: 空集合定义不能使用{}, {}定义的是字典.空集合使用set()
# my_dict = {}  # <class 'dict'>
# print(type(my_dict))

# my_list = []
# print(type(my_list))    # <class 'list'>
#
# my_tuple = ()
# print(type(my_tuple))   # <class 'tuple'>

my_set = set()
print(type(my_set))  # <class 'set'>

print("==========华丽分隔符==========")

my_list = [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
print(my_list)

my_set = set(my_list)
print(my_set)

my_list = list(my_set)
print(my_list)
```

2.【知道】list、tuple、set之间类型转换

函数	说明
list(x)	将 x 转换为列表类型
tuple(x)	将 x 转换为元组类型
set(x)	将 x 转换为集合类型

```
"""
列表类型名：list
元组类型名：tuple
集合类型名：set
list(x):    x转换为列表类型
tuple(x):   x转换为元组类型
set(x):     x转换为集合类型
"""

# 列表转元组、集合 类型
my_list = [1, 2, 3, 5, 3, 5]  # 列表可以修改
# 列表转换为元组类型
my_tuple = tuple(my_list)  # 元组不能修改
print(my_tuple)
# 列表转换为集合类型
my_set = set(my_list)  # 集合不会出现重复
```

```python
print(my_set)

print('============华丽分割线============')

# 元组转列表、集合 类型
my_tuple = (1, 2, 3, 5, 3, 5)
# 元组转换为列表 类型
my_list = list(my_tuple)
print(my_list)
# 元组转换为集合 类型
my_set = set(my_tuple)
print(my_set)

print('============华丽分割线============')

# 集合转元组、列表 类型
my_set = {1, 2, 3, 5}
# 集合转换为列表 类型
my_list = list(my_set)
print(my_list)
# 集合转换为元组 类型
my_tuple = tuple(my_set)
print(my_tuple)

# 扩展：字符串转换：
print('============华丽分割线============')
```

```
my_str = "hello"
print(list(my_str))
print(tuple(my_str))
print(set(my_str))

my_list = ['h', 'e', 'l', 'l', 'o']
my_str = "".join(my_list)
print(my_str)
```

六、公共语法

1.【知道】容器公共语法：内置函数、切片、运算符

1. 内置函数
    - len(容器变量)： 计算容器中元素个数
2. 切片
    - 字符串、列表、元组都支持切片类型
3. 运算符
    - +：合并
    - *：复制

```
# 内建（内置）函数
```

```python
# 通过len获取容器类型的元素个数
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
my_set = {1, 2, 3}
my_dict = {"k1": 1, "k2": 2, "k3": 3}
my_str = "123"

print(len(my_list), len(my_tuple),
len(my_set), len(my_dict), len(my_str))

print(max(my_list), max(my_str),
max(my_dict))

# 字符串、列表、元组都支持切片操作
print(my_list[::-1])
print(my_tuple[::-1])

# 运算符
# +：拼接，同类型的容器
# 字符串拼接合并
str1 = "hello "
str2 = "world"
print(str1 + str2)

# 列表元素拼接合并
list1 = [1, 2]
```

```
list2 = [3, 4]

new_list = list1 + list2
print(new_list)

# *: 复制
print("="*50, "华丽分割线", "="*50)
```