课程：列表

目标

- 列表的应用场景
- 列表的格式
- 列表的常用操作
- 列表的循环遍历
- 列表的嵌套使用

一. 列表的应用场景

思考：有一个人的姓名(TOM)怎么书写存储程序?

答：变量。

思考：如果一个班级100位学生，每个人的姓名都要存储，应该如何书写程序？声明100个变量吗?

答：列表即可， 列表一次性可以存储多个数据。

二. 列表的格式

```
[数据1，数据2，数据3，数据4......]
```

列表可以一次性存储多个数据，且可以为不同数据类型。

三. 列表的常用操作

列表的作用是一次性存储多个数据，程序员可以对这些数据进行的操作有：增、删、改、查。

3.1 查找

3.1.1 下标

```
name_list = ['Tom', 'Lily', 'Rose']

print(name_list[0])  # Tom
print(name_list[1])  # Lily
print(name_list[2])  # Rose
```

3.1.2 函数

- index()：返回指定数据所在位置的下标 。

1. 语法

```
列表序列.index(数据，开始位置下标，结束位置下标)
```

2. 快速体验

```
name_list = ['Tom', 'Lily', 'Rose']

print(name_list.index('Lily', 0, 2))  # 1
```

> 注意：如果查找的数据不存在则报错。

- count()：统计指定数据在当前列表中出现的次数。

```
name_list = ['Tom', 'Lily', 'Rose']

print(name_list.count('Lily'))  # 1
```

- len()：访问列表长度，即列表中数据的个数。

```
name_list = ['Tom', 'Lily', 'Rose']

print(len(name_list))  # 3
```

3.1.3 判断是否存在

- in：判断指定数据在某个列表序列，如果在返回True，否则返回False

```
name_list = ['Tom', 'Lily', 'Rose']

# 结果：True
print('Lily' in name_list)

# 结果：False
print('Lilys' in name_list)
```

- not in：判断指定数据不在某个列表序列，如果不在返回True，否则返回False

```python
name_list = ['Tom', 'Lily', 'Rose']

# 结果：False
print('Lily' not in name_list)

# 结果：True
print('Lilys' not in name_list)
```

- 体验案例

需求：查找用户输入的名字是否已经存在。

```python
name_list = ['Tom', 'Lily', 'Rose']

name = input('请输入您要搜索的名字：')

if name in name_list:
    print(f'您输入的名字是{name}，名字已经存在')
else:
    print(f'您输入的名字是{name}，名字不存在')
```

3.2 增加

作用：增加指定数据到列表中。

- append()：列表结尾追加数据。

1. 语法

```
列表序列.append(数据)
```

2. 体验

```python
name_list = ['Tom', 'Lily', 'Rose']

name_list.append('xiaoming')

# 结果：['Tom', 'Lily', 'Rose', 'xiaoming']
print(name_list)
```

```
C:\Users\黑马程序员\AppData\Local\Programs\Python\Python37\python3.exe
['Tom', 'Lily', 'Rose', 'xiaoming']

Process finished with exit code 0
```

> 列表追加数据的时候，直接在原列表里面追加了指定数据，即修改了原列表，故列表为可变类型数据。

3. 注意点

如果append()追加的数据是一个序列，则追加整个序列到列表

```
name_list = ['Tom', 'Lily', 'Rose']

name_list.append(['xiaoming', 'xiaohong'])

# 结果：['Tom', 'Lily', 'Rose', ['xiaoming', 'xiaohong']]
print(name_list)
```

- extend()：列表结尾追加数据，如果数据是一个序列，则将这个序列的数据逐一添加到列表。

1. 语法

```
列表序列.extend(数据)
```

2. 快速体验

2.1 单个数据

```
name_list = ['Tom', 'Lily', 'Rose']

name_list.extend('xiaoming')

# 结果：['Tom', 'Lily', 'Rose', 'x', 'i', 'a', 'o', 'm', 'i', 'n', 'g']
print(name_list)
```

2.2 序列数据

```
name_list = ['Tom', 'Lily', 'Rose']

name_list.extend(['xiaoming', 'xiaohong'])

# 结果：['Tom', 'Lily', 'Rose', 'xiaoming', 'xiaohong']
print(name_list)
```

- insert()：指定位置新增数据。

1. 语法

```
列表序列.insert(位置下标，数据)
```

2. 快速体验

```python
name_list = ['Tom', 'Lily', 'Rose']

name_list.insert(1, 'xiaoming')

# 结果：['Tom', 'xiaoming', 'Lily', 'Rose']
print(name_list)
```

3.3 删除

- del

1. 语法

```
del 目标
```

2. 快速体验

   2.1 删除列表

```python
name_list = ['Tom', 'Lily', 'Rose']

# 结果：报错提示：name 'name_list' is not defined
del name_list
print(name_list)
```

2.2 删除指定数据

```python
name_list = ['Tom', 'Lily', 'Rose']

del name_list[0]

# 结果：['Lily', 'Rose']
print(name_list)
```

- pop()：删除指定下标的数据(默认为最后一个)，并返回该数据。

1. 语法

```
列表序列.pop(下标)
```

2. 快速体验

```python
name_list = ['Tom', 'Lily', 'Rose']

del_name = name_list.pop(1)

# 结果：Lily
print(del_name)

# 结果：['Tom', 'Rose']
print(name_list)
```

- remove()：移除列表中某个数据的第一个匹配项。

1. 语法

```
列表序列.remove(数据)
```

2. 快速体验

```python
name_list = ['Tom', 'Lily', 'Rose']

name_list.remove('Rose')

# 结果：['Tom', 'Lily']
print(name_list)
```

- clear()：清空列表

```python
name_list = ['Tom', 'Lily', 'Rose']

name_list.clear()
print(name_list) # 结果： []
```

3.4 修改

- 修改指定下标数据

```python
name_list = ['Tom', 'Lily', 'Rose']

name_list[0] = 'aaa'

# 结果：['aaa', 'Lily', 'Rose']
print(name_list)
```

- 逆置：reverse()

```python
num_list = [1, 5, 2, 3, 6, 8]

num_list.reverse()

# 结果：[8, 6, 3, 2, 5, 1]
print(num_list)
```

- 排序：sort()

1. 语法

```
列表序列.sort( key=None, reverse=False)
```

> 注意：reverse表示排序规则，**reverse = True** 降序， **reverse = False** 升序（默认）

2. 快速体验

```python
num_list = [1, 5, 2, 3, 6, 8]

num_list.sort()

# 结果：[1, 2, 3, 5, 6, 8]
print(num_list)
```

3.5 复制

函数：copy()

```python
name_list = ['Tom', 'Lily', 'Rose']

name_li2 = name_list.copy()

# 结果：['Tom', 'Lily', 'Rose']
print(name_li2)
```

四. 列表的循环遍历

需求：依次打印列表中的各个数据。

4.1 while

- 代码

```python
name_list = ['Tom', 'Lily', 'Rose']

i = 0
while i < len(name_list):
    print(name_list[i])
    i += 1
```

- 执行结果

```
C:\Users\黑马程序员\AppData\Local\Programs\Python\Python37\python3.exe
Tom
Lily
Rose

Process finished with exit code 0
```

4.2 for

- 代码

```python
name_list = ['Tom', 'Lily', 'Rose']

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

- 执行结果

```
C:\Users\黑马程序员\AppData\Local\Programs\Python\Python37\python3.exe
Tom
Lily
Rose

Process finished with exit code 0
```

五. 列表嵌套

所谓列表嵌套指的就是一个列表里面包含了其他的子列表。

应用场景：要存储班级一、二、三三个班级学生姓名，且每个班级的学生姓名在一个列表。

```
name_list = [['小明', '小红', '小绿'], ['Tom', 'Lily', 'Rose'], ['张三', '李四', '王五']]
```

> 思考： 如何查找到数据"李四"?

```
# 第一步：按下标查找到李四所在的列表
print(name_list[2])

# 第二步：从李四所在的列表里面，再按下标找到数据李四
print(name_list[2][1])
```

六. 综合应用 -- 随机分配办公室

需求：有三个办公室，8位老师，8位老师随机分配到3个办公室

七. 总结

- 列表的格式

```
[数据1, 数据2, 数据3]
```

- 常用操作方法
  - index()
  - len()
  - append()
  - pop()
  - remove()
- 列表嵌套

```
name_list = [['小明', '小红', '小绿'], ['Tom', 'Lily', 'Rose'], ['张三', '李四', '王五']]
name_list[2][1]
```