第3章 列表简介

在本章和下一章中，我们将介绍列表以及如何使用列表元素。 列表能够在一个地方存储成组的信息，其中可以只包含几个元素，也可以包含数百万个元素。 列表是新手可直接使用的最强大的Python功能之一，它融合了众多重要的编程概念。

3.1 列表是什么

列表由一系列按特定顺序排列的元素组成。 你可以创建包含字母表中所有字母、数字0～9或所有家庭成员姓名的列表；也可以将任何东西加 入列表中，其中的元素之间可以没有任何关系。 列表通常包含多个元素，因此给列表指定一个表示复数的名称（如letters 、digits或names ）是个不错的主意。 在Python中，用方括号（ [] ）表示列表，并用逗号分隔其中的元素。 下面是一个简单的列表示例，其中包含几种车：

In [2]:
```
cars = ['Volvo',1, 'BYD', 'Wey']
print(cars)
```

```
['Volvo', 1, 'BYD', 'Wey']
```

3.1.1 访问列表元素

列表是**有序集合**，因此要访问列表的任意元素，只需将该元素的位置（索引 ）告诉Python即可。 要访问列表元素，可指出列表的名称，再指出元素的索引，并将后者放在方括号内。 例如，下面的代码从列表cars 中提取第一款车：

In [4]:
```
print(cars)
print(cars[1])
```

```
['Volvo', 'Lync&Co', 'BYD', 'Wey']
Lync&Co
```

3.1.2 索引从0而不是1开始

在Python中，第一个列表元素的索引为0，而不是1。 多数编程语言是如此规定的，这与列表操作的底层实现相关。

In [ ]:
```
print(cars[1])
print(cars[3])
```

Python为访问最后一个列表元素提供了一种特殊语法。通过将索引指定 为-1 ，可让Python返回最后一个列表元素：

In [3]:
```
print(cars)
print(cars[-2])
```

```
['Volvo', 1, 'BYD', 'Wey']
BYD
```

这种语法很有用，因为你经常需要在不知道列表长度的情况下访问最后的元素。 这种约定也适用于其他负数索引。 例如，索引-2 返回倒数第二个列表元素，索引-3 返回倒数第三个列表元素，依此类推。

3.1.3 使用列表中的各个值

可以像使用其他变量一样使用列表中的各个值。 例如，可以使用f字符串根据列表中的值来创建消息。 下面尝试从列表中提取第一款车，并使用这个值创建一条消息:

In [ ]:
```
cars = ['Volvo', 'Lync&Co', 'BYD', 'Wey']
message = f"My first car was a {cars[0].title()}."
print(message)
```

3.2 修改、添加和删除元素

创建的大多数列表将是**动态**的，这意味着列表创建后，将随着程序的运行**增删元素**。 例如，你创建一个游戏，要求玩家射杀从天而降的外星人。 为此，可在开始时将一些外星人存储在列表中，然后每当有外星人被射杀时，都将其从列表中删除，而每次有新的外星人出现在屏幕上时，都将其添加到列表中。 在整个游戏运行期间，外星人列表的长度将不断变化。

3.2.1 修改列表元素

修改列表元素的语法与访问列表元素的语法类似。 要修改列表元素，可指定列表名和要修改的元素的索引，再指定该元素的新值。 例如，假设有一个摩托车列表，其中的第一个元素为'honda' ，如何修改它的值呢？

In [5]:
```
motorcycles = ['honda', 'yamaha', 'suzuki','BMW']
print(motorcycles)
motorcycles[0] = 'ducati'
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'BMW']
['ducati', 'yamaha', 'suzuki', 'BMW']
```

3.2.2 在列表中添加元素

你可能出于众多原因要在列表中添加新元素。 例如，你可能希望游戏中出现新的外星人、添加可视化数据或给网站添加新注册的用户。 Python提供了多种在既有列表中添加新数据的方式。

In [6]:
```
# 01. 在列表末尾添加元素

motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)

motorcycles.append('ducati')
print(motorcycles)

# 方法append() 将元素'ducati' 添加到列表末尾，而不影响列表中的其他所有元素.
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha', 'suzuki', 'ducati']
```

方法append() 让动态地创建列表易如反掌。

例如，你可以先创建一个空列表，再使用一系列函数调用append() 来添加元素。

下面来创建一个空列表，再在其中添加元素'honda' 、'yamaha'和'suzuki' ：

In [ ]:
```
motorcycles = []
motorcycles.append('honda')
motorcycles.append('yamaha')
motorcycles.append('suzuki')
print(motorcycles)
```

这种创建列表的方式极其常见，因为经常要等程序运行后，你才知道用户要在程序中存储哪些数据。 为控制用户，可首先创建一个空列表，用于存储用户将要输入的值，然后将用户提供的每个新值附加到列表中。

在列表中插入元素

In [7]:
```
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.insert(2, 'ducati')
print(motorcycles)
```

```
['honda', 'yamaha', 'ducati', 'suzuki']
```

在这个示例中，值'ducati' 被插入到了列表开头。方法insert() 在索引0 处添加空间，并将值'ducati' 存储到这个地方。 这种操作将列表中既有的每个元素都右移一个位置

3.2.3 从列表中删除元素

你经常需要从列表中删除一个或多个元素。 例如，玩家将空中的一个外星人射杀后，你很可能要将其从存活的外星人列表中删除； 当用户在你创建的Web应用中注销账户时，你就需要将该用户从活动用户列表中删除。 你可以根据位置或值来删除列表中的元素。

使用del语句删除元素

如果知道要删除的元素在列表中的位置，可使用del语句。

In [9]:
```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
del motorcycles[2]
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha']
```

使用del 可删除任意位置处的列表元素，条件是知道其索引。 例如，下面演示了如何删除前述列表中的第二个元素'yamaha' ：

In [ ]:
```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
del motorcycles[1]
print(motorcycles)
```

在这两个示例中，使用del 语句将值从列表中删除后，你就无法再访问它了。

使用方法pop() 删除元素

有时候，你要将元素从列表中删除，并接着使用它的值。 例如，你可能需要获取刚被射杀的外星人的x坐标和y坐标，以便在相应的位置显示爆炸效果； 在Web应用程序中，你可能要将用户从活跃成员列表中删除，并将其加入到非活跃成员列表中。

方法pop() 删除列表末尾的元素，并让你能够接着使用它。 术语弹出（pop）源自这样的类比：列表就像一个栈，而删除列表末尾的元素相当于弹出栈顶元素。

In [ ]:
```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
popped_motorcycle = motorcycles.pop()
print(motorcycles)
print(popped_motorcycle)
```

弹出列表中任何位置处的元素

实际上，可以使用pop() 来删除列表中任意位置的元素，只需在圆 括号中指定要删除元素的索引即可。

In [ ]:
```
motorcycles = ['honda', 'yamaha', 'suzuki']
first_owned = motorcycles.pop(1)
print(f"The first motorcycle I owned was a {first_owned.title()}.")
print(motorcycles)
```

每当你使用pop() 时，被弹出的元素就不再在列表中了。 如果你不确定该使用del 语句还是pop() 方法，下面是一个简单的判断标准：如果你要从列表中删除一个元素，且不再以任何方式使用它，就使用del 语句；如果你要在删除元素后还能继续使用它，就使用方法pop() 。

根据值删除元素

有时候，你不知道要从列表中删除的值所处的位置。 如果只知道要删除的元素的值，可使用方法remove() 。例如，假设要从列表motorcycles 中删除值'ducati'。

In [10]:
```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati', 'ducati']
print(motorcycles)
motorcycles.remove('ducati')
print(motorcycles)
```

```
['honda', 'yamaha', 'suzuki', 'ducati', 'ducati']
['honda', 'yamaha', 'suzuki', 'ducati']
```

使用remove() 从列表中删除元素时，也可接着使用它的值。 下面删除值'ducati' 并打印一条消息，指出要将其从列表中删除的原因：

In [ ]:
```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati','ducati','ducati']
print(motorcycles)
too_expensive = 'ducati'
motorcycles.remove(too_expensive)
print(motorcycles)
print(f"\nA {too_expensive.title()} is too expensive for me.")
```

注意 方法remove() 只删除第一个指定的值。 如果要删除的值可能在列表中出现多次，就需要使用循环来确保将每个值都删除。

3.3 组织列表

在你创建的列表中，元素的排列顺序常常是无法预测的，因为你并非总能控制用户提供数据的顺序。这虽然在大多数情况下是不可避 免的，但你经常需要以特定的顺序呈现信息。 有时候，你希望保留列表元素最初的排列顺序，而有时候又需要调整排列顺序。 Python提供了很多组织列表的方式，可根据具体情况选用。

3.3.1 使用方法sort() 对列表永久排序

Python方法sort() 让你能够较为轻松地对列表进行**排序**。 假设你有一个汽车列表，并要让其中的汽车按字母顺序排列。 为简化这项任务，假设该列表中的所有值都是小写的。

In [ ]:
```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort()
print(cars)
```

方法sort() 永久性地修改列表元素的排列顺序。 现在，汽车是按字母顺序排列的，再也无法恢复到原来的排列顺序：

还可以按与字母顺序相反的顺序排列列表元素，只需向sort() 方法传递参数reverse=True 即可。下面的示例将汽车列表按与字母顺序相反的顺序排列：

In [ ]:
```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort(reverse=True)
print(cars)
```

3.3.2 使用函数sorted() 对列表临时排序

要保留列表元素原来的排列顺序，同时以特定的顺序呈现它们，可使用函数sorted() 。 函数sorted() 让你能够按**特定顺序**显示列表元素，同时不影响它们在列表中的**原始排列顺序**。 下面尝试来对汽车列表调用这个函数。

In [ ]:
```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print("Here is the original list:")
print(cars)
print("\nHere is the sorted list:")

car_sorted = sorted(cars)
print(car_sorted)


#print(sorted(cars,reverse=True))
print("\nHere is the original list again:")
print(cars)
```

3.3.3 倒着打印列表

要反转列表元素的排列顺序，可使用方法reverse() 。 假设汽车列表是按购买时间排列的，可轻松地按相反的顺序排列其中的汽车：

In [ ]:
```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.reverse()
print(cars)
```

In [ ]:
```
input('test')
```

3.3.4 确定列表的长度

使用函数len() 可快速获悉列表的长度。 在下面的示例中，列表包含四个元素，因此其长度为4 ：

In [ ]:
```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(len(cars))
```

注意 Python计算列表元素数时从1开始，因此确定列表长度时，你应该不会遇到差一错误。

3.4 使用列表时避免索引错误

刚开始使用列表时，经常会遇到一种错误。假设你有一个包含三个 元素的列表，却要求获取第四个元素：

In [ ]:
```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles[len(motorcycles)])
```

3.6 列表的底层实现

Python中的列表(list)底层实现是基于数组(array)实现的，每个列表元素在内存中都是连续存放的，也就是说，Python的列表是一个动态数组。

Python列表底层存储原理

In [4]:
```
lst = [1, 2, 3, 4, 5]
for i in lst:
    print(id(i))
```

```
2344623302960
2344623302992
2344623303024
2344623303056
2344623303088
```

动态数组

Python中的列表通过动态数组来实现，即当我们向列表中添加元素的时候，如果列表的当前长度无法容纳新元素，Python就会自动为其分配新的空间。

In [14]:
```
import numpy as np
a = []
print(id(a))
a.append(1)
print(id(a))
```

```
2499243622656
2499243622656
```

列表的切片操作

Python中的列表支持使用索引和切片来访问其中的元素。当我们使用切片操作时，Python实际上创建了一个新的列表对象，该列表对象与原列表对象共享一部分或全部数据。

In [5]:
```
a = [1, 2, 3, 4, 5]
b = a[:]
print(id(a))
print(id(b))
print(a is b)
```

```
2344708836096
2344708835392
False
```

3.5 小结

在本章中，我们学习了：列表是什么以及如何使用其中的元素； 如何定义列表以及如何增删元素； 如何对列表进行永久性排序，以及如何为展示列表而进行临时排序； 如何确定列表的长度，以及在使用列表时如何避免索引错误。 在第4章，你将学习如何以更高效的方式处理列表元素。 通过使用为数不多的几行代码来遍历列表元素，就能高效地处理它们，即便列表包含数千乃至数百万个元素。

In [ ]:
```
#The end
```