In this chapter, lists are introduced as one of the most important data structures in Python.

What Is a List?

A list in Python is a collection of elements stored in a specific order. You can create a list that contains any type of data, such as numbers, strings, or more complex objects. Lists are created using square brackets [ ], and items inside are separated by commas. Example:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)
```
Output: ['trek', 'cannondale', 'redline', 'specialized']

Accessing Elements in a List

You can access any element in a list by its index, starting from 0. For example, to access the first item in the list:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[0])  # Output: trek
```

Index Positions Start at 0, Not 1

In Python, the first element of a list is at index 0, not 1. This indexing system is common in most programming languages.

Using Individual Values from a List

You can use individual values from a list just like variables. For example, you can use an element from the list to create a personalized message:

```
message = "My first bicycle was a " + bicycles[0].title() + "."
print(message)  # Output: My first bicycle was a Trek.
```

Changing, Adding, and Removing Elements

➢ Modifying Elements in a List

To change an item in a list, you use the list name followed by the index of the element you want to modify and assign a new value to it:

```
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles[0] = 'ducati'
print(motorcycles)  # Output: ['ducati', 'yamaha', 'suzuki']
```

➢ Adding Elements to a List

The **method for adding elements** includes two **commands**: append and insert."

1- Appending Elements to the End of a List

The simplest way to add a new element to a list is to append the item to the list. When you append an item to a list, the new element is added to the end of the list. Using the same list we had in the previous example, we'll add the new element '**ducati**' to the end of the list.

Example:

```
fruits = ['apple', 'banana', 'cherry']

print("Original list:", fruits) # change the name of the list from fruits to original list

# Appending a new fruit to the list

fruits.append('orange')

print("List after appending:", fruits)
```

Output:

```
Original list: ['apple', 'banana', 'cherry']

List after appending: ['apple', 'banana', 'cherry', 'orange']
```

Appending Multiple Items

If you want to add multiple items to a list, you can use a loop to call append() multiple times:

```
# Appending multiple items

more_fruits = ['grape', 'pear', 'mango']

for fruit in more_fruits:

    fruits.append(fruit)
```

Example about **extend** commend

```
fruits = ['apple', 'banana', 'cherry']

eat = ['grape', 'pear', 'mango']

fruits.extend(eat)

print(fruits)
```

عند استخدام الحلقة for fruit in more_fruits: يتم المرور على كل عنصر في قائمة more_fruits واحدًا تلو الآخر.

في كل دورة من الحلقة، يتم تنفيذ fruits.append(fruit) أي إضافة العنصر الحالي إلى قائمةfruits .

بالتالي يتم إضافة 'grape'، ثم 'pear'، ثم 'mango' بشكل منفصل، في النهاية نحصل على قائمة موحّدة تمامًا مثل ما يحدث مع extend().

Output:

print("List after appending more fruits:", fruits)

List after appending more fruits: ['apple', 'banana', 'cherry', 'orange', 'grape', 'pear', 'mango']

2- Inserting Elements into a List

You can add a new element at any position in your list by using the insert() method. You do this by specifying the index of the new element and the value of the new item.

Suppose we have a list containing some motorcycles, and we want to add a new motorcycle at the beginning of the list. We can do this using insert() as follows:

```
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.insert(0, 'ducati')
print(motorcycles)
```

Output:
['ducati', 'honda', 'yamaha', 'suzuki']

Difference between append and insert.

- `append()`: `1-` always adds the element at the **end** of the list. `2-` You **don't** need to specify the position (index) where you want to add the element.
- `insert()`:`1-` allows you to add the element at **any** position you choose within the list, pushing the other elements to the right. `2-` You need to specify the position (index) where you want to add the element.

In this example, the code inserts **'ducati'** at **position 0** (the beginning), and the other items are pushed to the right to maintain their order. Notice that the other elements remain in the list but their positions change due to the new insertion.

Removing Elements from a List

This section explains the different ways to remove items from a list in Python. You can remove elements using the **del** statement, the **pop()** method, or the **remove()** method. Below is a detailed explanation of each approach:

1- Removing an Item Using the del Statement

If you know the index of the item you want to remove, you can use the **del** statement. For example, to remove the first item from the list of motorcycles:

Example 1:

```
motorcycles = ['honda', 'yamaha', 'suzuki']
del motorcycles[0]
print(motorcycles)#  Output:  ['yamaha', 'suzuki']
```

Example 2:

```
fruits = ['apple', 'banana', 'cherry', 'orange']
del fruits[1:3]
print(fruits)
```

Output: ['apple', 'orange']

You can remove any item from the list using **del**, as long as you know the index of the item.

2- Removing an Item Using the pop() Method

Sometimes, you want to remove an item from the list but still use it **later**. The pop() method is useful for this, as it removes the **last item** from the list but allows you to keep working with the item. For example:

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

1- تعريف القائمة: القائمة motorcycles تحتوي على العناصر ['honda', 'yamaha', 'suzuki'].

2- الدالة :()pop تقوم بإزالة آخر عنصر من القائمة، وفي هذه الحالة هو 'suzuki'، وتخزينه في المتغير popped_motorcycle.

3- طباعة القائمة: بعد تنفيذ()pop ، سيتم طباعة القائمة motorcycles بدون العنصر الأخير، وبالتالي ستصبح['honda', 'yamaha'] .

4- طباعة العنصر المزال: العنصر الذي تمت إزالته popped_motorcycle سيتم طباعته، وهو'suzuki' .

Output:

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

In this example, the last item, **'suzuki'**, is removed from the list and stored in the variable **popped_motorcycle**.

Popping Items from Any Position in the List

You can use pop() to remove an item at any position by specifying the index of the item you want to remove. For instance, to remove the first item:

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

first_owned = motorcycles.pop(0)
print('The first motorcycle I owned was a ' + first_owned.title() + '.')
```

Output: The first motorcycle I owned was a Honda.

In this example, the first item in the list (at index 0) is removed and used in a message.

Removing an Item by Value Using the remove() Method

If you know the value of the item you want to remove but not its position in the list, you can use the remove() method. For example, to remove 'ducati' from the list of motorcycles:

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

Output:

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

Difference between del, pop, and remove:

del: يُستخدم لحذف عنصر أو مجموعة من العناصر من القائمة بناءً على موقعها (الفهرس فقط). بعد استخدام del، لا يمكنك الوصول إلى العنصر المحذوف لأنه يتم حذفه بالكامل.
pop: يُستخدم لحذف عنصر من القائمة مع القدرة على استخدامه لاحقًا اي له القدرة على تخزينه. يحدف العنصر الذي تحدده و إذا لم تحدد فهرسًا، سيتم حذف العنصر الأخير.
Remove: لا يستخدم فهارس بل يُستخدم لحذف عنصر يطابق القيمة التي تحددها . و كما في del لا يمكنك الوصول إلى العنصر المحذوف لأنه يتم حذفه بالكامل.

Note: The remove() method only removes the first occurrence of the value you specify. If the value appears more than once in the list, you will need to use a loop to ensure all occurrences are removed.

Example :

```
motorcycles = ['ducati', 'yamaha', 'suzuki', 'ducati']

motorcycles.remove('ducati')
```

مثال للاطلاع فقط:

```
motorcycles = ['ducati', 'yamaha', 'suzuki', 'ducati']
while 'ducati' in motorcycles:
    motorcycles.remove('ducati')
print(motorcycles)
```

Output = ['yamaha', 'suzuki', 'ducati']# removed the first occurrence (ducati).

Organizing a List

- **Sorting a List Permanently with sort()**

You can sort a list in alphabetical order using the sort() method:

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

في هذا المثال، تم ترتيب قائمة السيارات أبجديًا باستخدام sort().

ملاحظة: القائمة **الأصلية** تتغير، ولا يمكنك العودة إلى الترتيب القديم لان تغيير الترتيب هنا دائم permanent.

- **Sorting A List in Permanent Reverse Alphabetical Order With: Sort(Reverse=True)**

In Python, you can sort the elements of a list in reverse alphabetical or numerical order (from largest to smallest or from the last letter to the first) using the sort() function with the argument reverse=True.

Example(alphabetical) :

sort(): يقوم بترتيب العناصر ترتيبًا تصاعديًا (أبجديًا أو رقميًا).
sort(reverse=True): يقوم بترتيب العناصر ترتيبًا تنازليًا (عكسيًا) و بشكل دائم كما هو الحال مع sort().

```
players = ['michael', 'kobe', 'lebron', 'shaq']

players.sort(reverse=True)

print(players) # Result: ['shaq', 'michael', 'lebron', 'kobe']
```

Example(numerical) :

```
numbers = [3, 6, 1, 9, 4]

numbers.sort(reverse=True)

print(numbers)  # Result: [9, 6, 4, 3, 1]
```

بشكل مختصر : يتضمن موضوع ترتيب القوائم Organizing a List ثلاث اوامر و هما **sort()** و sort(reverse=True) و **sorted()**.

- **Sorting a List Temporarily with sorted()**

If you want to sort a list without modifying the original order, you can use the sorted() function:

```
print(sorted(cars))  # ['audi', 'bmw', 'subaru', 'toyota']
print(cars)  # ['bmw', 'audi', 'toyota', 'subaru']  # Original list remains unchanged
```

- **Finding the Length of a List**

You can quickly find the length of a list by using the len() function. The list in this example has four items, so its length is 4:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
 len(cars) #
Output: 4
```

تقوم دالة **len()** بإرجاع عدد العناصر الموجودة في القائمة.

فوائد استخدام len():

إحصاء عدد العناصر: يساعدك len() في معرفة عدد العناصر الموجودة في قائمة دون الحاجة إلى عدّها يدويًا.

- **Slicing**

The general syntax for slicing:
list_name[start:stop:step]

- start: The index of the element you want to start slicing from (this element is included).
- stop: The index of the element where you want the slicing to stop (this element is not included).
- step: The number of steps between each element (optional, the default is 1).

Example:

```
cars = ['bmw', 'audi', 'toyota', 'subaru', 'ford', 'chevrolet', 'honda']

# Slice elements from index 1 to index 4 (the element at index 4 is not included)

subset_of_cars = cars[1:4]

print(subset_of_cars)  # Output: ['audi', 'toyota', 'subaru']
```

You can specify the step to select elements at intervals.

يمكنك تحديد الخطوة (step) لاختيار العناصر بشكل متقطع.
اي التوقف عند العنصر السادس اقتطاع كل عنصر ثاني في النطاق من الفهرس 0 إلى 6. في هذا المثال يبدأ العد من العنصر ذات الترتيب صفر و الاقتطاع يكون لكل عنصر ثاني, على سبيل المثال فأن audi هي عنصر ثاني بالنسبة لbmw و ان Subaruهي عنصر ثاني بالنسبة ل Toyota و هكذا لغاية العنصر السادس.

Example:

```
every_second_car = cars[0:6:2]

print(every_second_car)  # Output: ['bmw', 'toyota', 'ford']
```

Slicing from the beginning to a specific index:

You can leave the start value empty to begin from the first element in the list.

يمكنك ترك قيمة البداية فارغة لتبدأ من أول عنصر في القائمة.

اقتطاع من البداية إلى الفهرس 3 (العنصر عند الفهرس 3 غير مشمول)

```
from_start = cars[:3]

print(from_start)# Output: ['bmw', 'audi', 'toyota']
```

Slicing from a specific index to the end:

You can leave the stop value empty to slice from a specific index to the end of the list.

يمكنك ترك قيمة النهاية فارغة لتقتطع من فهرس معين حتى نهاية القائمة.
باختصار فهو يظهر البيانات التي في المساحة الفارغة التي تكون قبل او بعد النقطتين.

Example:

```
# Slice from index 2 to the end

from_index_to_end = cars[2:]

print(from_index_to_end)  # Output: ['toyota', 'subaru', 'ford', 'chevrolet', 'honda']
```

Slicing with negative indices:

You can use negative indices to count from the end of the list.

التقطيع السالب هو كما في الموجب لكن يبدا العد من النهاية الى الى البداية , فالبيانات التي قبل النقطتين تبقى و التي بعدها تحذف.

```
# Slice the last 3 elements in the list

last_three_cars = cars[-3:]

print(last_three_cars)  # Output: ['ford', 'chevrolet', 'honda']
```

Avoiding Index Errors

If you try to access an index that doesn't exist, you will get an "IndexError". To avoid this, ensure the index is within the range of the list.

-إذا حاولت الوصول إلى فهرس غير موجود في القائمة، ستحصل على خطأ يسمى **IndexError.**
- يحدث هذا الخطأ عندما تطلب عنصرًا في موقع (فهرس) يتجاوز حجم القائمة.

Example:

```
fruits = ['apple', 'banana', 'cherry']

print(fruits[3])  # IndexError: list index out of range
```

كيفية تجنب الخطأ:

تأكد دائمًا أن الفهرس الذي تحاول الوصول إليه **داخل نطاق القائمة**. يمكنك استخدام len() للتحقق من عدد العناصر في القائمة وتحديد إذا كان الفهرس الذي تريد استخدامه صحيحًا.

List Programming in Python

List commands allow you to create a new list in a single line of code. A list command combines a for loop with the creation of elements in one line and automatically adds each new element. Number manually, making it easier and more efficient.

• Using the range() Function:

Example:

ياخذ هذا الايعاز الارقام من 1 الى 11 و يقوم بتربيعها واحد بعد الاخر باعتبارها تابعة للمتغير value.

```
squares = [value**2 for value in range(1, 11)]

print(squares)

print(squares)#      Output: [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Making Numerical Lists

There are many reasons to maintain a collection of numbers. You may need to keep track of the position of each character in a game, or monitor high scores.

Example:

```
for value in range(1, 5):
    print(value)
```

الكود هذا يستخدم حلقة for مع دالة range() لإنشاء قائمة من الأرقام وطباعتها.

Output:

1

2

3

4

مثال لطباعة امر دعوة على مناسبة معينة:

```
guests = ["Ahmed", "Mona", "Hassan"]
for guest in guests:
    print(f"مرحباً {guest}, أدعوك إلى العشاء الليلة.")
```

Output:

مرحباً Ahmed, أدعوك إلى العشاء الليلة.

مرحباً Mona, أدعوك إلى العشاء الليلة.

مرحباً Hassan, أدعوك إلى العشاء الليلة.

Using range() to Make a List of Numbers

```
numbers = list(range(1, 6))
print(numbers)
```

Output:

[1, 2, 3, 4, 5]

فائدة f" : يسمح لك بإدخال القيم مباشرة داخل النص بدون الحاجة إلى عمليات ربط (+) أو تنسيق معقد.

```
name = "Alaa"
age = 25
print(f"My name is {name} and I am {age} years old.")
```

Output:

My name is Alaa and I am 25 years old.

- **Obtaining Odd Numbers.**

```
odd_numbers = list(range(1, 21, 2))
print(odd_numbers)
```

Output:

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

Simple Statistics with a List of Numbers

You can use functions like **min()**, **max()**, and **sum()** to find the minimum and maximum values and the sum of a list of numbers.

```
digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```

print(min(digits)) # Result : 0

print(max(digits)) # Result :9

print(sum(digits)) # Result :45

Tuples

Tuples are a special type of list that cannot be changed after they are created.
tuples are defined using regular parentheses ().

dimensions = (200, 50)

print(dimensions[0])# 200

print(dimensions[1])# 200

Output: 200

Output: 50

Tuple = قائمة ثابتة، أسرع وأكثر أمانًا عندما لا تحتاج لتعديل القيم.

- هنا يتم إنشاء tuple باسم `dimensions` يحتوي على عنصرين، وهما `200`و `50`. هذا يمكن أن يمثل أبعادًا مثل الطول والعرض.

- **`print(dimensions[0])`**:

- يتم طباعة العنصر الأول من tuple، والذي هو `200`. في بايثون، يبدأ العد من الصفر، لذا `dimensions[0]` يشير إلى أول عنصر.

- **`print(dimensions[1])`**:

- يتم طباعة العنصر الثاني من tuple، والذي هو `50`.

Exercises (Try It Yourself)

The book provides many exercises to practice these concepts, such as:

- **Exercise 3-1**: Create a list of names of your friends and print them.
- **Exercise 3-4**: Create a guest list of people you'd invite to dinner.

This chapter teaches how to work with lists, modify them, and organize them in Python efficiently.