PYTHON FUNDAMENTALS

Index Positions Start at 0, Not 1

➢ *Lists:*

❖ *Adding Elements to a List:*

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.

Example:

Input:

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

Output:

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

- The append() method makes it easy to build lists dynamically. For example, you can start with an empty list and then add items to the list using a series of append() calls.

Input:

```
motorcycles = []

motorcycles.append('honda')

motorcycles.append('yamaha')

motorcycles.append('suzuki')

print(motorcycles)
```

Output:

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

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.

Input:

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


motorcycles.insert(0, 'ducati')

print(motorcycles)
```

Output:

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

- In this example, the code inserts the value **'ducati'** at the beginning of the list. The insert() method opens a space at position **0** and stores the value **'ducati'** at that location. This operation shifts every other value in the list one position to the right.

❖ ***Removing Elements from a List***

1. *Removing an Item Using the **del** Statement:*

- If you know the position of the item you want to remove from a list, you can use the del statement.

Example:

Input:

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

del motorcycles[0]
print(motorcycles)
```

Output:

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

- You can remove an item from any position in a list using the del statement if you know its index. For example, here's how to remove the second item, 'yamaha', in the list:

Input:

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

del motorcycles[1]
print(motorcycles)
```

Output:

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

! **In both examples, you can no longer access the value that was removed from the list after the *del* statement is used.**

2. *Removing an Item Using the pop() Method:*

- The pop() method removes the last item in a list, but it lets you work with that item after removing it. The term pop comes from thinking of a list as a stack of items and popping one item off the top of the stack. In this analogy, the top of a stack corresponds to the end of a list.

Input:

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

```
popped_motorcycle = motorcycles.pop()
print(popped_motorcycle)
```

Output

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

- ***The output shows that the value 'suzuki' was removed from the end of the list and is now assigned to the variable popped_motorcycle:***

! **How might this pop() method be useful? Imagine that the motorcycles in the list are stored in chronological order according to when we owned them. If this is the case, we can use the pop() method to print a statement about the last motorcycle we bought:**

Example:

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

last_owned = motorcycles.pop()
print(f"The last motorcycle I owned was a {last_owned.title()}.")
```

Output:

```
The last motorcycle I owned was a Suzuki.
```

2.1 Popping Items from any Position in a List:

- You can use **pop()** to remove an item from any position in a list by including the **index** of the item you want to remove in parentheses.

Example*:*

Input:

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


first_owned = motorcycles.pop(0)

print(f"The first motorcycle I owned was a {first_owned.title()}.")
```

Output:

```
The first motorcycle I owned was a Honda.
```

! Remember that each time you use **pop()**, the item you work with is no longer stored **in the list**!

! If you’re unsure whether to use the **del** statement or the **pop()** method, here’s a simple way to decide: when you want to delete an item from a list and **not use that item in any way**, use the **del** statement; if you want to use an item as you remove it, use the **pop()** method.

3. Removing an Item by Value

- Sometimes you won't know the position of the value you want to remove from a list. If you only know the value of the item you want to remove, you can use the **remove()** method

Example*:*

Input:

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

motorcycles.remove('ducati')
print(motorcycles)
```

Output:

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

- You can also use the **remove()** method to work with a value that's being removed from a list

Input:

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

too_expensive = 'ducati'
motorcycles.remove(too_expensive)
print(motorcycles)
print(f"A {too_expensive.title()} is too expensive for me.")
```

Output:

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

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

A Ducati is too expensive for me.
```

You can also use the string method .title() on any element in this list. For example, you can capitalize the element 'ducati' using the title() method.

❖ ***Organizing a List:***

- **Often, your lists will be created in an unpredictable order, because you can't always control the order in which your users provide their data. Although this is unavoidable in most circumstances, you'll frequently want to present your information in a particular order. Sometimes you'll want to preserve the original order of your list, and other times you'll want to change the original order. Python provides a number of different ways to organize your lists, depending on the situation.**

1. Sorting a List Permanently with the sort() Method

- Python’s **sort()** method makes it relatively easy to sort a list. Imagine we have a list of cars and want to change the order of the list to store them alphabetically.

Example:

Input:

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


print(cars)
```

Output:

```
['audi', 'bmw', 'subaru', 'toyota']
```

! You can also sort this list in **reverse** alphabetical order by passing the argument **reverse=True** to the **sort()** method.

Input:

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


print(cars)
```

Output:

```
['toyota', 'subaru', 'bmw', 'audi']
```

2. Sorting a List Temporarily with the sorted() Function.

- To maintain the original order of a list but present it in a sorted order, you can use the **sorted()** function. The **sorted()** function lets you display your list in a particular order but doesn't affect the actual order of the list.

Example:

Input:

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

print("Here is the sorted list:")
print(sorted(cars))

print("Here is the original list again:")
print(cars)
```

Output:

```
Here is the original list:
['bmw', 'audi', 'toyota', 'subaru']

Here is the sorted list:
 ['audi', 'bmw', 'subaru', 'toyota']

Here is the original list again:
['bmw', 'audi', 'toyota', 'subaru']
```

- We first print the list in its original order and then in alphabetical order. After the list is displayed in the new order, we show that the list is still stored in its original order.

! Notice that the list still exists in its original order after the **sorted()** function has been used. The **sorted()** function can also accept a **reverse=True** argument if you want to display a list in reverse alphabetical order.

3. Printing a List in Reverse Order

- To reverse the original order of a list, you can use the **reverse()** method. If we originally stored the list of cars in chronological order according to when we owned them, we could easily rearrange the list into **reverse** chronological order:

Example:

Input:

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


cars.reverse()
print(cars)
```

Output:

```
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
```

! Notice that **reverse()** doesn’t sort backward alphabetically; it simply reverses the order of the list:

- The **reverse()** method changes the order of a list permanently, but you can revert to the original order anytime by applying **reverse()** to the same list a second time

4.Finding the Length of a List

You can quickly find the length of a list by using the **len()** function.

Example:

Input:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']


len(cars)
```

Output:

```
4
```