

Modifying a List

Name		Date	
Class / Sec		Roll no.	

1 · AIM

To write a Python program that **modifies** a list — adding items with append and insert, removing them with remove and pop, changing an item by index, and sorting the result.

2 · SOFTWARE AND LIBRARIES USED

TOOL	WHAT IT IS FOR
Python 3	The programming language the program is written in
Jupyter Notebook	The environment used to write and run the code (IDLE or Colab also acceptable)
No extra library	List methods are built into Python — nothing to import and nothing to install

3 · THEORY

Lists are **mutable** — they can be changed after they are created. Python provides a method for each kind of change, and the differences between them are exactly what gets examined.

METHOD	WHAT IT DOES	WATCH OUT FOR
append(x)	Adds x at the end of the list	Only ever adds at the end — no position given
insert(i, x)	Adds x at index i	Everything after i shifts along by one
remove(x)	Removes the first item equal to x	Takes a value . Error if x is not there
pop()	Removes and returns the last item	Takes an index , not a value. pop(0) takes the first
list[i] = x	Replaces the item at index i	Length does not change — nothing added or removed
sort()	Reorders the list in place	Alphabetical for text, numerical for numbers

remove() vs pop() — the classic confusion: **remove("Science")** takes the **value** you want gone. **pop(2)** takes the **index**. Give remove an index and it will look for that number as a value — and usually raise an error.

4 · PROGRAM

```
# Create the starting list
subjects = ["Maths", "Science", "English"]
print("Start      :", subjects)

# Add an item at the end
subjects.append("Hindi")
print("After append  :", subjects)

# Insert an item at a chosen position
subjects.insert(1, "AI")
print("After insert   :", subjects)

# Remove an item by its value
```

```

subjects.remove("Science")
print("After remove   :", subjects)

# Remove the last item, and keep what was removed
removed = subjects.pop()
print("After pop      :", subjects, "| removed:", removed)

# Replace an item using its index
subjects[0] = "Mathematics"
print("After replacing:", subjects)

# Sort the list alphabetically
subjects.sort()
print("After sort     :", subjects)

```

These methods change the list itself: `append`, `insert`, `remove`, `pop` and `sort` all modify **subjects** directly. They do not return a new list — so writing **subjects = subjects.sort()** is wrong and gives you `None`.

5 • EXPECTED OUTPUT

```

Start           : ['Maths', 'Science', 'English']
After append    : ['Maths', 'Science', 'English', 'Hindi']
After insert    : ['Maths', 'AI', 'Science', 'English', 'Hindi']
After remove    : ['Maths', 'AI', 'English', 'Hindi']
After pop       : ['Maths', 'AI', 'English'] | removed: Hindi
After replacing: ['Mathematics', 'AI', 'English']
After sort      : ['AI', 'English', 'Mathematics']

```

[PASTE SCREENSHOT HERE]

Paste a screenshot of YOUR program running, with the code and output visible together

6 • FOLLOWING THE LIST THROUGH

Track the **length** at each step. It tells you at a glance what each method actually did.

AFTER	LENGTH	WHAT CHANGED
Start	3	The original three subjects
append	4	Hindi added at the end — position 3
insert(1, AI)	5	AI placed at index 1, and Science shifted from index 1 to 2
remove(Science)	4	Found by value and deleted
pop()	3	Last item removed and returned — Hindi is stored in the variable
subjects[0] = ...	3	Unchanged length — Maths was replaced, not added or removed
sort()	3	Same items, reordered alphabetically

The length tells the story: Adding methods raise it, removing methods lower it, and replacing or sorting leave it alone. If your length changes when you expected it not to, you used the wrong method.

7 • OBSERVATION AND RESULT

Two or three sentences in your own words. Say which methods added, which removed, and which left the length unchanged.

Example observation: Lists are mutable, so the same list was changed seven times without creating a new one. `append` and `insert` increased the length, while `remove` and `pop` decreased it — `pop` also returned the item it removed. Replacing an item by index and sorting both left the length at 3, because nothing was added or taken away.

Worth knowing: `sort()` changes the list itself and returns nothing. If you want a sorted **copy** while keeping the original, use `sorted(subjects)` instead — that returns a new list.

8 · VIVA QUESTIONS FOR THIS PROGRAM

LIKELY QUESTION	SHAPE OF A STRONG ANSWER
What is the difference between <code>append</code> and <code>insert</code> ?	append always adds at the end. insert adds at a chosen index and shifts the rest along
What is the difference between <code>remove</code> and <code>pop</code> ?	remove takes a value; pop takes an index. <code>pop</code> also returns the removed item
Why did the length stay at 3 when you replaced an item?	Replacing swaps one value for another. Nothing is added or removed
What happens if you remove something not in the list?	Python raises a ValueError — the value could not be found
What does <code>sort()</code> return?	Nothing. It reorders the list in place. Use <code>sorted()</code> if you want a new list back

9 · HOW THIS ENTRY IS MARKED

FULL MARKS

The list printed **after every change**, not just at the end · all six operations demonstrated · commented code · a screenshot of **your own** output · an observation explaining **which methods changed the length**

LOSES MARKS

Only the final list printed, so no change can be seen · confuses **remove** with **pop** · writes **subjects = subjects.sort()** · no screenshot, or someone else's · an observation that just lists the methods used

Before you submit: Use **your own** list — your subjects, or your favourite films. **Print the list after every single change**, because a file that only shows the final list proves nothing about what each method did.