📘 Chapter 10: Lists– Mastering Sequences.

> *"Lists are flexible, . List is powerful, and now you'll master it."*

🎯 What You Will Learn

- What lists and tuples are
- Indexing, slicing, and looping techniques
- How to modify, copy, sort, and manage data in lists
- What makes tuples immutable
- Use cases, advanced patterns, and logic
- A real-world mini project with practical logic

🧠 Why Lists and Tuples?

- **Lists** are like notepads: flexible, editable, reorderable.
- **Tuples** are like certificates: fixed, secure, quick to access.

🔢 Indexing and Slicing (Visual)

```
List:     [ "a",  "b",  "c",  "d",  "e",  "f" ]
Index:      0     1     2     3     4     5
Reverse:   -6    -5    -4    -3    -2    -1
```

```python
letters = ["a", "b", "c", "d", "e", "f"]
print(letters[0])     # a
print(letters[-1])    # f
print(letters[1:4])   # ['b', 'c', 'd']
print(letters[::-1])  # ['f', 'e', 'd', 'c', 'b', 'a']
```

🔁 15 Must-Know List Techniques – With Descriptions & Outputs

#	Feature	Description	Example Output
1	Create	Make a list	`["apple", "banana"]`
2	Index	Access by position	`fruits[0] → apple`
3	Negative Index	Access from the end	`fruits[-1] → banana`
4	Modify Item	Change element by index	`fruits[1] = "orange"`

#	Feature	Description	Example Output
5	Looping	Traverse with `for` loop	Prints all items
6	Membership Check	`"apple" in fruits`	`True`
7	Append	Add to end	`fruits.append("grape")`
8	Insert	Add at specific index	`fruits.insert(1, "kiwi")`
9	Remove (value)	Delete specific item	`fruits.remove("apple")`
10	Pop (by index)	Remove and return item	`fruits.pop(0)`
11	Delete	Remove by `del`	`del fruits[0]`
12	Length	`len(fruits)` returns count	`3`
13	Slice	Extract part of list	`fruits[1:3]` → `["kiwi", "grape"]`
14	Sort	Sort in-place	`[1, 2, 3]`
15	Copy	Clone list safely	`copy = fruits.copy()`

🔼 Advanced List Techniques – With Output

✅ List Comprehension

```
squares = [x*x for x in range(1, 6)]
print(squares)  # [1, 4, 9, 16, 25]
```

✅ Enumerate in Loops

```
for i, fruit in enumerate(["apple", "banana"]):
    print(i, fruit)
# Output:
# 0 apple
# 1 banana
```

✅ Merge Lists

```
a = [1, 2]; b = [3, 4]
print(a + b)  # [1, 2, 3, 4]
```

✅ Extend List

```
a.extend([5, 6])
```

✅ Nested Lists

```
grid = [[1, 2], [3, 4]]
print(grid[1][0])  # 3
```

✅ List as Stack (LIFO)

```
stack = []
stack.append("data")
stack.pop()  # "data"
```

🧪 Mini Quiz

1. What's the difference between append() and extend()?
2. How do you make a one-item tuple?
3. Can you modify x = (1, 2)?
4. What does fruits[-1] return in ["a", "b", "c"]?
5. What's the output?

```
nums = [5, 3, 1]
nums.sort()
print(nums)
```

🧩 Basic Practice Problems (15)

- Create a list of 5 names and print the second one
- Check if "Python" is in a list of languages
- Replace the 3rd item with "new value"
- Add "orange" to the end of a fruit list
- Insert "kiwi" at the second position
- Remove the last item using pop()
- Use a loop to print all items in a list
- Use slicing to get the middle 3 items from a list of 5
- Sort a list of numbers in descending order
- Create a list of squares from 1 to 10 using list comprehension
- Count the number of items in a list

- Create a nested list and print an inner value
- Merge two lists
- Use enumerate() to print index + item
- Copy a list and modify the copy without changing the original

---

🚀 Intermediate Problems (5)

- Build a menu where users can add and remove items from a list
- Take a list of scores, remove the lowest, and print the average
- Track a list as a shopping cart (add/remove/view items)
- Flatten a nested list of integers into a single list
- From a list of names, print only the ones starting with vowels

---

🔧 Debug Challenges (5)

Fix:

```
fruits = ["apple", "banana"]
fruits[2] = "kiwi"
```

Fix:

```
x = [1, 2, 3]
y = x
y.append(4)
print(x)
```

What is the output?

```
x = [1, 2, 3]
y = x.copy()
y.append(4)
print(x)
```

Fix:

```
lst = [1, 2, 3]
print(lst[3])
```

Bonus Tip: Explore List Features Yourself

```
print(dir(list))
help(list.append)
```

🧠 This shows you everything lists can do — even advanced operations.🏁 You've Mastered Lists and Tuples

---

🏁 You've Mastered Lists and

- ✅ You can now build, sort, slice, and copy lists
- ✅ You understand when and why to use lists
- ✅ You can apply advanced logic like stacks, enumerations, and comprehensions
- ✅ You can solve real-world problems like managing inventory or user input cleanly

> Lists are for structure. Tuples are for safety. You now own both.