🔹 Feature	🧠 Description
✅ Mutable	You can change, add, or delete items after creation
📐 Indexed	Supports indexing and slicing: `list[0]`, `list[-1]`, etc.
🔄 Ordered	Preserves the order of insertion
🧪 Heterogeneous	Can store mixed data types — ints, strings, floats, objects
♻️ Dynamic	Size can grow or shrink at runtime
🧰 Supports Nesting	Lists within lists (multi-dimensional) are allowed
🎯 Versatile Methods	Comes with useful functions like `append()`, `extend()`, `pop()`, etc.

```
my_list = [1, 2, 3, 4, 5]
print(my_list[0])  # First element
print(my_list[-1]) # Last element
my_list.append(6)        # Add to end
my_list.insert(2, 99)     # Insert at index 2
my_list.remove(10)        # Removes element with value 10
del my_list[0]           # Deletes first element
popped = my_list.pop()    # Removes and returns last element

print(99 in my_list)
print(len(my_list))       # Number of elements
print(my_list[1:4])       # Sublist from index 1 to 3
my_list.sort()           # Sorts the list
my_list.reverse()        # Reverses the list
for item in my_list:
    print(item)
squares = [x**2 for x in range(10)]
copy_list = my_list.copy()
copy_list = my_list[:]
my_list.clear()
my_list.extend([7, 8, 9])
a, *rest = [1, 2, 3, 4]
print(a)
print(rest)
names = ['Alice', 'Bob']
```

Tuple inside the list

```
scores = [85, 90]

for n, s in zip(names, scores):

    print(f"{n}: {s}")
```

[(alice,85),(bob,90)}

22.

```
names = ['Alice', 'Bob']
scores = [85, 90]
# Combine into list of tuples
paired = list(zip(names, scores))
paired.sort(key=lambda x: x[1])
for n, s in paired:

    print(f"{n}: {s}")
```

Access second element of a tuple inside the list

```
students = [('Alice', 85), ('Bob', 90), ('Charlie', 80)]

lowest = min(students, key=lambda x: x[1])

print(lowest)

# if you try same code with sum, it wont work.
```

23.

```
numbers = [1, 2, 3, 4, 5, 6]
print(numbers[::2])  # [1, 3, 5]
print(numbers[::-1]) # [6, 5, 4, 3, 2, 1]
```

24.

```
nums = [10, 50, 30]
print(sum(nums))     # 90
print(max(nums))     # 50
print(min(nums))     # 10
```

25.

```
nums = [1, 2, 3]
squared = list(map(lambda x: x**2, nums))
```

Questions :

Give one example for each function using lambda.

Common Built-in Functions Where `lambda` Shines

🦴 Function	💡 Purpose	✅ Lambda Use
`sorted()`	Sort items based on custom logic	✅ Yes
`min()` / `max()`	Find items with custom comparison	✅ Yes
`map()`	Transform each item	✅ Yes
`filter()`	Select items based on condition	✅ Yes
`reduce()`	Accumulate result across sequence *(from* `functools` *)*	✅ Yes
`any()` / `all()`	Logical checks	✅ Works with `map()` or generator + lambda

 Basic Construction:

- *How can you create a list of the first 10 square numbers using a loop and also with list comprehension?*

2. **List Transformation:**
   - *Write a Python function that removes all duplicate values from a list while preserving the order.*
3. **Slicing Logic:**
   - *Explain how slicing can be used to reverse a list or extract every third element.*
4. **Matrix Representation:**
   - *Create a 3x3 matrix using lists and access its diagonal elements.*
5. **Sorting Challenge:**
   - *Given two lists — names and scores — how can you sort them by scores in descending order and display the names accordingly?*
6. **Unpacking Practice:**
   - *Use unpacking to separate the first element from the rest in a list. How is this helpful in real-world data processing?*
7. **Nested Lists:**
   - *How would you flatten a nested list like [[1,2],[3,4]] using list comprehension?*
8. **List vs Set:**
   - *Demonstrate how converting a list to a set helps in removing duplicates — but what is lost in the process?*
9. **Memory Management:**
   - *What is the difference between list1 = list2 vs list1 = list2.copy()? How does this affect memory and modifications?*
10. **AI Applications:**

- *In NLP, how can lists be used to tokenize and filter out stopwords from a sentence?*