**Python Lists — Complete Notes**

**✅ What is a List?**

A **list** is a **mutable, ordered collection** of items in Python. It can hold elements of **different data types** (int, str, float, etc.).

my\_list = [1, 2, 3, "hello", 4.5, True]

**📌 Key Features of Lists**

| **Property** | **Value** |
| --- | --- |
| Ordered | ✅ Yes |
| Mutable | ✅ Yes (Changeable) |
| Duplicates allowed | ✅ Yes |
| Heterogeneous data | ✅ Yes (Mixed types) |
| Indexing | ✅ Yes (0-based) |
| Slicing | ✅ Yes |

**🛠️ Creating Lists**

# Empty list

a = []

# List of numbers

nums = [1, 2, 3, 4]

# Mixed data types

mixed = [1, "apple", 3.14, True]

# Using list() constructor

lst = list(("apple", "banana", "cherry"))

**📥 Accessing List Elements**

colors = ["red", "green", "blue"]

print(colors[0])      # Output: red

print(colors[-1])     # Output: blue (negative index)

**🔁 Looping Through a List**

for color in colors:

    print(color)

# Using index

for i in range(len(colors)):

    print(i, colors[i])

**✂️ List Slicing**

a = [10, 20, 30, 40, 50]

print(a[1:4])   # [20, 30, 40]

print(a[:3])    # [10, 20, 30]

print(a[-2:])   # [40, 50]

**⚙️ Modifying Lists (Mutability)**

a = [1, 2, 3]

a[0] = 100

print(a)        # [100, 2, 3]

**➕ Adding Elements**

# append() – adds to end

a.append(4)

# insert(index, value)

a.insert(1, 50)

# extend() – merge another list

a.extend([5, 6])

a.remove(2)     # removes value 2

a.pop()         # removes last item

a.pop(0)        # removes index 0

del a[1]        # deletes index 1

a.clear()       # empties the list

**🔍 Searching in List**

a = [10, 20, 30, 40]

print(20 in a)         # True

print(a.index(30))     # 2

**📏 Length & Count**

len(a)             # Number of elements

a.count(10)        # Occurrences of 10

**🔄 Sorting & Reversing**

nums = [3, 1, 4, 2]

nums.sort()        # Ascending

nums.sort(reverse=True)  # Descending

nums.reverse()     # Reverse the list

**📋 Copying Lists**

# Shallow copy

new\_list = old\_list.copy()

# Or

new\_list = old\_list[:]

**🧪 List Comprehension**

squares = [x\*x for x in range(5)]   # [0, 1, 4, 9, 16]

**🧰 Built-in Functions on Lists**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| len() | Length of list | len([1,2,3]) → 3 |
| max() | Maximum element | max([1,5,2]) → 5 |
| min() | Minimum element | min([1,5,2]) → 1 |
| sum() | Sum of elements | sum([1,2,3]) → 6 |
| sorted() | Returns a new sorted list | sorted([3,1,2]) → [1,2,3] |
| list() | Converts iterable to list | list("abc") → ['a','b','c'] |

**🔗 Common List Methods (With Example)**

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

fruits.append('orange') # ['apple', 'banana', 'cherry', 'orange']

fruits.insert(1, 'mango') # ['apple', 'mango', 'banana', ...]

fruits.remove('banana') # Removes 'banana'

fruits.pop() # Removes last item

fruits.index('apple') # Returns index

fruits.count('apple') # Counts how many times 'apple'

fruits.sort() # Sorts list alphabetically

fruits.reverse() # Reverses the list

fruits.copy() # Copies list

fruits.clear() # Empties list

**🧠 Why Use Lists?**

* When you need to **store a sequence of values** (numbers, strings, etc.)
* Useful in **loops**, **data processing**, and **dynamic collections**
* Supports **mutability**, making it ideal for changing data

**✅ Summary Table**

| **Feature** | **Supported** |
| --- | --- |
| Mutable | ✅ Yes |
| Indexed | ✅ Yes |
| Slicing | ✅ Yes |
| Heterogeneous | ✅ Yes |
| Duplicate items | ✅ Yes |

Addational things :

**🧾 1. Quick Revision — Python Lists**

| **Concept** | **Summary** |
| --- | --- |
| **Definition** | Lists are mutable, ordered collections that can store heterogeneous elements. |
| **Syntax** | my\_list = [1, 2, 3, "hi", 4.5, True] |
| **Mutability** | You can modify, add, or remove elements after creation. |
| **Indexing** | 0-based indexing, supports both positive & negative indices. |
| **Slicing** | a[start:end:step] — creates a new list (shallow copy). |
| **Duplicates** | Allowed. |
| **Nesting** | Lists can contain other lists. |
| **Dynamic** | Length can change at runtime. |

**🧩 2. Missing / Less Common Things to Add**

Here are **important but often-missed** details:

**🧱 List Nesting**

matrix = [[1,2,3], [4,5,6], [7,8,9]]

print(matrix[1][2]) # 6

**🧮 List + Operators**

# Concatenation

a = [1, 2]; b = [3, 4]

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

# Repetition

print(a \* 3) # [1, 2, 1, 2, 1, 2]

**🧍 Copy vs Assignment**

a = [1, 2, 3]

b = a # same reference

b[0] = 100

print(a) # [100, 2, 3] (same object)

c = a.copy() # new object

c[0] = 999

print(a) # [100, 2, 3] (unchanged)

**🧠 Shallow vs Deep Copy**

import copy

a = [[1,2], [3,4]]

b = copy.deepcopy(a)

b[0][0] = 99

print(a) # [[1,2], [3,4]] - unaffected

**🧰 Built-in Functions (Extra)**

| **Function** | **Description** | **Example** |
| --- | --- | --- |
| all() | True if all elements are True | all([1, True, 5]) → True |
| any() | True if any element is True | any([0, False, 3]) → True |
| enumerate() | Returns index & value | for i,v in enumerate(lst) |
| zip() | Combine lists element-wise | list(zip([1,2],[3,4])) → [(1,3),(2,4)] |
| map() | Apply function to each element | list(map(str.upper, fruits)) |
| filter() | Filter based on condition | list(filter(lambda x:x>2,[1,2,3])) |
| reversed() | Reverse iterator | list(reversed([1,2,3])) → [3,2,1] |

**🧩 3. Complete List of List Methods (with Examples)**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| append(x) | Add single element to end | a.append(10) |
| extend(iterable) | Add multiple elements | a.extend([4,5]) |
| insert(i, x) | Insert at position | a.insert(1, 'hi') |
| remove(x) | Remove first occurrence | a.remove(10) |
| pop([i]) | Remove & return element | a.pop(2) |
| clear() | Remove all items | a.clear() |
| index(x[, start[, end]]) | Return index of x | a.index('apple') |
| count(x) | Count occurrences | a.count(3) |
| sort(reverse=False, key=None) | Sort in place | a.sort(reverse=True) |
| reverse() | Reverse list in place | a.reverse() |
| copy() | Return shallow copy | b = a.copy() |

**🧠 4. Useful Tips & Gotchas**

| **Concept** | **Tip** |
| --- | --- |
| **Slicing Copy** | b = a[:] → makes a shallow copy |
| **List of lists trap** | [[0]\*3]\*3 creates 3 *references* to the same list |
| **Comparison** | Lists are compared element-wise |
| **Sorting with key** | a.sort(key=len) sorts by string length |
| **Unpacking** | x, y, \*rest = [1,2,3,4] |
| **Joining Strings** | ' '.join(['hello', 'world']) → 'hello world' |

**🧮 5. List Comprehension (Advanced Examples)**

# Basic

squares = [x\*\*2 for x in range(6)]

# Conditional

evens = [x for x in range(10) if x % 2 == 0]

# Nested

pairs = [(x,y) for x in [1,2] for y in [3,4]]

# Flatten a 2D list

flat = [num for sub in [[1,2,3],[4,5]] for num in sub]

**💬 6. 20+ Python List Interview Questions**

**🧩 Basic Level**

1. What is the difference between a list and a tuple?
2. How do you access the last element of a list?
3. How can you reverse a list without using the reverse() method?
4. How to check if an element exists in a list?
5. What is list slicing and how does it work?
6. What happens if you modify a list while iterating over it?

**⚙️ Intermediate Level**

1. What is the difference between append() and extend()?
2. What does list \* 3 do?
3. Explain the difference between copy() and assignment (=).
4. How to remove duplicates from a list while maintaining order?
5. Explain shallow vs deep copy of lists.
6. How to flatten a nested list in Python?
7. How to find the second largest element in a list?
8. How to sort a list of tuples by the second value?

**🧠 Advanced Level**

1. What is the time complexity of append(), insert(), and remove()?
2. Explain why [[0]\*3]\*3 can lead to unexpected results.
3. How can you convert a list of strings into a single string?
4. What is the output of:
5. a = [1, 2, 3]
6. b = a
7. b.append(4)
8. print(a)
9. How to get the intersection of two lists (common elements)?
10. How to remove all occurrences of a specific element without using a loop?
11. What’s the difference between sorted(a) and a.sort()?
12. How do you transpose a matrix using list comprehension?
13. How do you count unique elements in a list?
14. How do you create a list from a string and vice versa?
15. What are some alternatives to lists in Python for efficiency? (e.g., array, deque, numpy.array)

Interview questions :

**1️⃣ What is the difference between a list and a tuple?**

**Answer:**

* **List** → Mutable, defined using []
* **Tuple** → Immutable, defined using ()

lst = [1, 2, 3]

tpl = (1, 2, 3)

lst[0] = 100 # Works

# tpl[0] = 100 ❌ Error: tuple object is not mutable

✅ **Use list** when data changes; **tuple** for fixed data.

**2️⃣ How do you access the last element of a list?**

**Answer:**  
Use **negative indexing**:

a = [10, 20, 30, 40]

print(a[-1]) # 40

**3️⃣ How can you reverse a list without using reverse()?**

**Answer:**  
There are two ways:

a = [1, 2, 3, 4]

print(a[::-1]) # Using slicing

print(list(reversed(a))) # Using reversed() built-in

**4️⃣ How to check if an element exists in a list?**

**Answer:**  
Use the **in operator**:

colors = ["red", "green", "blue"]

print("green" in colors) # True

**5️⃣ What is list slicing and how does it work?**

**Answer:**  
Slicing returns a **sublist**: list[start:end:step]

a = [10, 20, 30, 40, 50]

print(a[1:4]) # [20, 30, 40]

print(a[:3]) # [10, 20, 30]

print(a[::2]) # [10, 30, 50]

**6️⃣ What happens if you modify a list while iterating over it?**

**Answer:**  
It can cause **unpredictable behavior** or skip elements.

✅ Use a **copy** instead:

a = [1, 2, 3, 4]

for x in a[:]:

if x == 2:

a.remove(x)

print(a) # [1, 3, 4]

**7️⃣ What is the difference between append() and extend()?**

**Answer:**

* append() adds **one element**.
* extend() adds **multiple elements**.

a = [1, 2]

a.append([3, 4]) # [1, 2, [3, 4]]

a.extend([3, 4]) # [1, 2, 3, 4]

**8️⃣ What does list \* 3 do?**

**Answer:**  
Repeats the list 3 times.

a = [1, 2]

print(a \* 3) # [1, 2, 1, 2, 1, 2]

**9️⃣ Explain the difference between copy() and assignment (=).**

**Answer:**  
= assigns **reference**, while .copy() makes a **shallow copy**.

a = [1, 2, 3]

b = a

c = a.copy()

b[0] = 99

print(a) # [99, 2, 3]

print(c) # [1, 2, 3]

**🔟 How to remove duplicates from a list while maintaining order?**

**Answer:**  
Use a loop or dict.fromkeys():

a = [1, 2, 3, 1, 2, 4]

res = list(dict.fromkeys(a))

print(res) # [1, 2, 3, 4]

**11️⃣ Explain shallow vs deep copy of lists.**

**Answer:**

* **Shallow copy** → copies outer list, inner objects shared.
* **Deep copy** → copies everything recursively.

import copy

a = [[1,2],[3,4]]

b = copy.copy(a)

c = copy.deepcopy(a)

b[0][0] = 99

print(a) # [[99,2],[3,4]] → affected

print(c) # [[1,2],[3,4]] → unaffected

**12️⃣ How to flatten a nested list in Python?**

**Answer:**  
Use **list comprehension**:

nested = [[1, 2, 3], [4, 5], [6]]

flat = [x for sub in nested for x in sub]

print(flat) # [1, 2, 3, 4, 5, 6]

**13️⃣ How to find the second largest element in a list?**

**Answer:**

a = [10, 20, 4, 45, 99]

a.sort()

print(a[-2]) # 45

Or without sorting:

unique = list(set(a))

unique.remove(max(unique))

print(max(unique))

**14️⃣ How to sort a list of tuples by the second value?**

**Answer:**  
Use key argument:

data = [(1, 3), (2, 1), (4, 2)]

data.sort(key=lambda x: x[1])

print(data) # [(2, 1), (4, 2), (1, 3)]

**15️⃣ What is the time complexity of append(), insert(), and remove()?**

**Answer:**

| **Method** | **Time Complexity** | **Note** |
| --- | --- | --- |
| append() | O(1) | Amortized constant |
| insert() | O(n) | Shifts elements |
| remove() | O(n) | Searches for element |

**16️⃣ Explain why [[0]\*3]\*3 can lead to unexpected results.**

**Answer:**  
It creates **three references to the same list**.

a = [[0]\*3]\*3

a[0][0] = 9

print(a) # [[9,0,0],[9,0,0],[9,0,0]]

✅ Correct way:

a = [[0]\*3 for \_ in range(3)]

**17️⃣ How can you convert a list of strings into a single string?**

**Answer:**  
Use .join():

words = ["I", "love", "Python"]

print(" ".join(words)) # "I love Python"

**18️⃣ What is the output of this code?**

a = [1, 2, 3]

b = a

b.append(4)

print(a)

**Answer:**  
[1, 2, 3, 4] — because a and b refer to the **same object**.

**19️⃣ How to get the intersection of two lists (common elements)?**

**Answer:**

a = [1, 2, 3, 4]

b = [3, 4, 5]

print(list(set(a) & set(b))) # [3, 4]

**20️⃣ How to remove all occurrences of a specific element without using a loop?**

**Answer:**  
Use list comprehension:

a = [1, 2, 3, 2, 4]

a = [x for x in a if x != 2]

print(a) # [1, 3, 4]

**21️⃣ What’s the difference between sorted(a) and a.sort()?**

**Answer:**

| **Function** | **Returns** | **Modifies original** |
| --- | --- | --- |
| sorted(a) | New sorted list | ❌ No |
| a.sort() | None | ✅ Yes |

nums = [3, 1, 2]

print(sorted(nums)) # [1, 2, 3]

print(nums.sort()) # None

**22️⃣ How do you transpose a matrix using list comprehension?**

**Answer:**

matrix = [[1,2,3], [4,5,6], [7,8,9]]

transpose = [[row[i] for row in matrix] for i in range(3)]

print(transpose)

# [[1,4,7], [2,5,8], [3,6,9]]

**23️⃣ How do you count unique elements in a list?**

**Answer:**  
Use set():

a = [1, 2, 2, 3, 3, 3]

unique = set(a)

print(len(unique)) # 3

**24️⃣ How do you create a list from a string and vice versa?**

**Answer:**

s = "python"

lst = list(s)

print(lst) # ['p', 'y', 't', 'h', 'o', 'n']

new\_s = "".join(lst)

print(new\_s) # python

**25️⃣ What are alternatives to lists for better efficiency?**

**Answer:**

| **Alternative** | **Description** |
| --- | --- |
| array (from array module) | For numeric data (faster, less memory) |
| deque (from collections) | Faster appends/pops from both ends |
| numpy.array | For large numerical operations |
| set | For unique unordered items |

**✅ Bonus — Common Trick Questions**

**Q:** What is the output?

a = [1, 2, 3]

print(a \* 0)

**A:** [] — empty list, because repetition with 0 gives no elements.

**Q:** How to get both index and value when looping?

for i, val in enumerate(['a','b','c']):

print(i, val)

✅ Output:

0 a

1 b

2 c