Arrays

What is an Array?

An array is a collection of element stored in contiguous memory and accessed using an index.

In Python, we mostly use:

list → behave like a dynamic Array.

example -

arr = [10, 20, 30, 40]

Index	value
0	10
1	20
2	30
3	40

Key Properties

- Indexed
- mutable
- Ordered
- can store mixed type

```
arr = [10, "Hello", 3.5] # allowed, but
                         bad Practice for DSA.
```

Task 1

1. create an array with values: 5, 10, 15, 20
2. Print : First element
   last element
3. change 15 to 17
4. Print updated Array

Solution

```
arr = [5, 10, 15, 20]
Print (arr[0])
Print (arr[len(arr)-1])
arr[2] = 17
Print (arr)
```

Time complexity

Operation	Time
Access	O(1)
Update	O(1)
Search	O(n)

Insertion & Deletion

Concept 1 : Append vs Insert

append()

- Add element at end
- Fast → O(1)

```
arr = [1, 2, 3]
arr.append(4)
```

output

```
[1, 2, 3, 4]
```

Insert(Index, Value)

- Insert at a specific index
- slow → O(n) (because shifting happen)

```
arr.insert(1, 99)
```

output [1, 99, 2, 3, 4]

Why slow?

Element after the index must shift right.

Task 2 [10, 20, 30, 40]

1. Add 50 at end
2. Add 15 at index 1
3. Print Array

```
arr.= [10, 20, 30, 40]
arr.append(50)
arr.insert(1, 15)
Print(arr)
```

output ⇒ [10, 15, 20, 30, 40, 50]

- Prefer append() in DSA Problem.

Concept 2: Deletion in Array

Method 1: pop()

- Remove by index
- Return removed element

```
arr.pop()    # remove last
arr.pop(2)   # removes index 2
```

Method 2 : remove (value)

- Remove first occurence
- slow → O(n)

```
arr.remove(20)
```

Task 3

```
arr = [5, 10, 15, 10, 20]
```

1. Remove element at index 2
2. Remove value 10
3. Predict array

```
arr = [5, 10, 15, 10, 20]
arr.pop(2)
Print(arr)
# [5, 10, 10, 20]
arr.remove(10)
Print(arr) # [5, 10, 20]
```

Prefer Index based deletion when control matters.

Array Traversal

Traversal = visiting each element of the array in a specific order.

Most array problem = "How you traverse" + "What you do while traversing"

Pattern 1: Simple Forward Traversal

Visit element from left → right

```
arr = [10, 20, 30]
For i in range(len(arr)):
    Print(arr[i])
```

Used when:

- ① sum
- ② max/min
- ③ counting
- ④ checking condition

Task 4

```
arr = [3, 7, 1, 9, 4]
```

① Traverse array
② Print only even number

```
arr = [3, 7, 1, 9, 4]
For i in range(len(arr)):
    if arr[i] % 2 == 0:
        Print(arr[i])
```

Time complexity = O(n)

Pattern 2: Direct value Traversal

```
For val in arr:
    Print(val).
```

But: ① you don't get index
② can't modify array properly

Pattern 3: Reverse Traversal

Method 1: Using Index

```
For i in range(len(arr)-1, -1, -1):
    Print (arr[i])
```

Method 2: Python Shortcut

```
For x in arr[::-1]:
    Print (x)
```

Task 5

```
arr = [10, 5, 20, 8]
```

① Traverse from right to left
② Find maximum element

```
arr = [10, 5, 20, 8]
max_val = arr[0]
For i in range (len (arr)-1, -1, -1):
    if arr[i] > max_val :
        max_val = arr[i]
Print (max_val)
```

Pattern 4: Index + value together

```
For i, val in enumerate (arr):
    Print (i, val)
```

Useful when:
- modify array
- Track Position
- Prefix sum logic

Task 6

arr = [1, 2, 3, 4]

Create a new array where:
each element = original value × index

output = [0, 2, 6, 12]

```
arr = [1, 2, 3, 4]
new_arr = [val * i for i, val in enumerate
                                (arr)]
Print (new_arr)
```

Alternative: Using for loop

```
arr = [1, 2, 3, 4]
new_arr = []
for i, val in enumerate(arr):
    new_arr.append(val * i)
```

SEARCHIN IN ARRAYS

Searching = Finding whether an element exist or Finding its index / count / condition

In Array Foundation is:

Linear search

- Check elements one by one.
- Work on unsorted Array
- Time : O(n)

Basic Linear search Template

```
For i in range(len(arr)):
    if arr[i] == target:
        # Found
```

Task 7

```
arr = [4, 9, 2, 7, 1]
target = 7
```

1. check if 7 exist
2. Print it's index
3. if not Found, Print "Not Found"

```
arr = [4, 9, 2, 7, 1]
target = 7
Found = False
For i in range(len(arr)):
    if arr[i] == target:
        Print(i)
        Found = True
        break
if not Found:
    Print("Not Found")
```

Linear search

Best case = O(1) (element at start)

Worst case = O(n) element at end / absent

Search Variation

① Find all occurrences

```
For i in range(len(arr)):
    if arr[i] == target:
        Print(i)
```

② Find count occurrences

```
Count = 0
For x in arr:
    if x == target:
        count += 1
```

Task 8

```
arr = [1, 2, 3, 2, 2, 4]
target = 2
```

① Count how many time 2 appears

② Print all its indices

```
indices = [ ]
Count = 0
For i, val in enumerate(arr):
    if val == target:
        indices.append(i)
        Count += 1
```

Concept 4: Searching with condition

First even number

```
For x in arr:
    if x % 2 == 0:
        Print(x)
        break
```

Task 9

arr = [1, 3, 5, 8, 9]

① Find First even number
② If none exist, Print "No even number"

```
arr = [1, 3, 5, 8, 9]
For val in arr:
    if val % 2 == 0:
        Print(F"First even number:{val}")
        break
else:
    Print("No even number")
```

Searching in Sorted vs Unsorted Arrays

Array Type	Best Method
Unsorted	Linear Search
Sorted	Binary Search

Searching Patterns

Existence check
Index Find
Count
Condition-based search
Early stopping

Keyword in Interview to listen

- First occurrence
- any occurrence
- Count
- minimum index