

Day 7 Learning Notes

1. Aptitude – Averages

Goal: Improve speed and accuracy in numerical aptitude questions.

Key Concepts

- **Definition:** An average (Arithmetic Mean) is the sum of all observations divided by the total number of observations.
 - Formula:

$$\text{Average} = \frac{\text{Sum of Observations}}{\text{Number of Observations}}$$

- **Sum Formula:**

$$\text{Sum} = \text{Average} \times \text{Number of Observations}$$

Quick Calculation Techniques

Instead of summing large numbers, use the **Deviation Method**:

1. Assume a "Working Mean" (an approximate average close to the data points).
2. Find the deviation of each number from this Working Mean.
3. Calculate the average of these deviations.
4. Add this result to the Working Mean to get the actual Average.

Example: Find the average of 42, 45, 48, 50, 55.

- **Traditional:** $(42 + 45 + 48 + 50 + 55)/5 = 240/5 = 48$.
- **Deviation Method:** Assume Working Mean = 50.
 - Deviations: -8, -5, -2, 0, +5
 - Sum of deviations: -10
 - Avg of deviations: $-10/5 = -2$
 - **Final Average:** $50 + (-2) = 48$.

◆ 2. Programming – Largest of Three Numbers

Goal: Strengthen conditional logic (`if-else`) skills.

Logic

To find the largest of three variables (a, b, c):

1. Check if $a \geq b$ AND $a \geq c$. If true, a is the largest.
2. Else, check if $b \geq a$ AND $b \geq c$. If true, b is the largest.
3. Otherwise, c is the largest.

Implementation Logic (Pseudocode)

```

IF a >= b AND a >= c:
    PRINT "a is largest"
ELSE IF b >= a AND b >= c:
    PRINT "b is largest"
ELSE:
    PRINT "c is largest"

```

3. Concept – Python Lists & Operations

Goal: Understand dynamic data structures in Python.

What is a List?

A **List** in Python is an ordered, mutable collection of items. Unlike arrays in C, lists can hold elements of different data types.

Common Operations

- **Creation:** `my_list = [1, 2, "hello", 3.5]`
- **Accessing:** 0-based indexing. `my_list[0]` is 1. Negative indexing `my_list[-1]` gets the last item.
- **Adding Elements:**
 - `list.append(item)` : Adds to the end.
 - `list.insert(index, item)` : Adds at a specific position.
- **Removing Elements:**
 - `list.remove(item)` : Removes the first occurrence of the value.
 - `list.pop(index)` : Removes and returns element at index (default last).
- **Iterating:**

```

fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)

```

4. C/C++ Concept – Arrays

Goal: Understand low-level memory management for collections.

Key Characteristics

1. **Contiguous Memory:** Array elements are stored in adjacent memory locations.
2. **Fixed Size:** In C/C++, the size of a static array must be known at compile time (mostly).
3. **Homogeneous:** All elements must be of the same data type (e.g., all `int`).

Memory Visualization

If `int arr[3] = {10, 20, 30};` is stored starting at memory address 1000 (assuming 4 bytes per `int`):

- `arr[0]` -> Address 1000 (Value: 10)
- `arr[1]` -> Address 1004 (Value: 20)
- `arr[2]` -> Address 1008 (Value: 30)

Code Example

```
int numbers[5] = {1, 2, 3, 4, 5};
// Accessing the 3rd element
cout << numbers[2]; // Outputs 3
```

5. SQL – ORDER BY & LIMIT

Goal: Retrieve specific, ranked data from a database.

ORDER BY

Sorts the result set of a query by one or more columns.

- **ASC:** Ascending order (default).
- **DESC:** Descending order.

LIMIT

Restricts the number of rows returned by the query. Crucial for pagination or finding "Top N" results.

Syntax & Example

Scenario: Find the top 3 students with the highest marks.

```
SELECT student_name, marks
FROM students
ORDER BY marks DESC
LIMIT 3;
```

- `ORDER BY marks DESC` : Puts the highest marks at the top.

- I T M T T - 2 : Cut the list off after the first 2 rows.