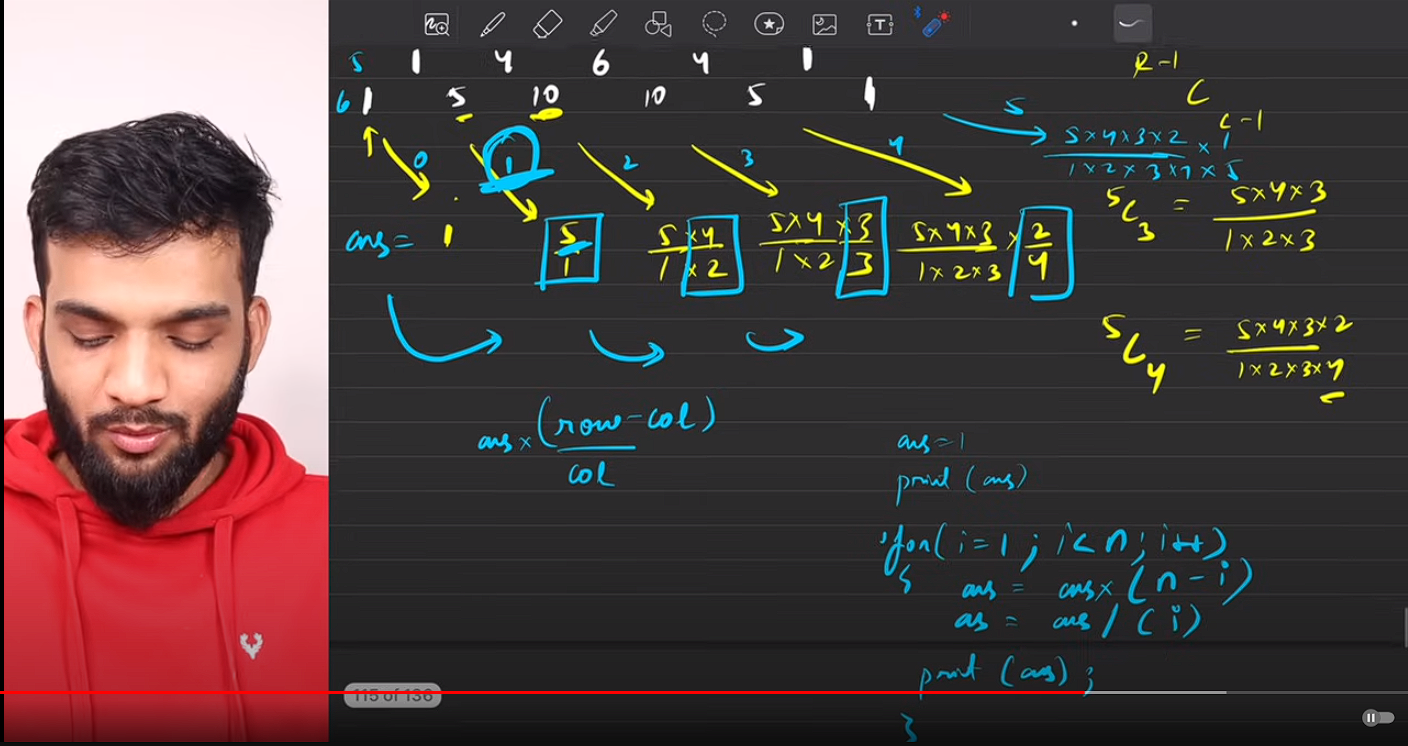
**Day 1**: Arrays

1] Set Matrix Zero:

1. Arrays have a minimum time complexity of n^2 due to traversal.
2. If time complexity exceeds n^2, consider using extra space to reduce it, as there's often a tradeoff between time and space.
3. Be mindful of dependencies, as they can impact performance.

2] Pascal Triangle

1. Element at row n and col r = (n-1)C(r-1) # row and col starting with 1
2. nCr shortcut: Ex. 10C3 = (10/1) \* (9/2) \* (8/3) #start denominator with 1
3. take *long long* data type for storing result of factorial.



1. Use above method of printing row (time complexity O(n)) to print whole triangle (time complexity O(n^2))
2. To simplify big problem, try to optimise subpart of that problem.
3. If code is getting longer, break it down into parts and put them into function for better code quality.

**Day 2**: Arrays Part-II

**Day 3**: Arrays Part-III

**Day 4**: Arrays Part-IV

**Day 5**: Linked List

**Day 6**: Linked List Part-II

**Day 7**: Linked List and Arrays

**Day 8**: Greedy Algorithm

**Day 9**: Recursion

**Day 10**: Recursion and Backtracking

**Day 11**: Binary Search

**Day 12**: Heaps

**Day 13**: Stack and Queue

**Day 14**: Stack and Queue Part-II

**Day 15**: String

**Day 16**: String Part-II

**Day 17**: Binary Tree

**Day 18**: Binary Tree part-II

**Day 19**: Binary Tree part-III

**Day 20**: Binary Search Tree

**Day 21**: Binary Search Tree Part-II

**Day 22**: Binary Trees[Miscellaneous]

**Day 23**: Graph

**Day 24**: Graph Part-II

**Day 25**: Dynamic Programming

**Day 26**: Dynamic Programming Part-II

**Day 27**: Trie

**Day 28**: Operating System

**Day 29**: DBMS

**Day 30**: Computer Networks