Chapter --> 3

[STACK]

Why do we use data Structure?

A data structure is a way of storing organizing, accessing, and retrieving data in a computer.

Ex -> A string is a structure containing a sequence of elements where each element is a character. On the other hand, a list is a sequence data structure in which each element may be of different types.

Note -> Python lists are dynamic in nature i.e. they can grow/increase and shrink/ decreases in size/ number of items.

Data Type:- It defines the type of values we store

and the operation we can perform on those values.
Ex:- In (int) data type we cannot store decimal values.

Data structure:-

A data structure in which elements are organized in a sequence is called a linear data structure.

Every data structure has a specific way of insertion and deletion of data.

Stack:-

A stack is a linear sequence structure or a list of elements in which insertion and deletion can take place only at one end i.e. stack's top. The stack is called LIFO(Last In First Out) which means the

element inserted last would be the first to be deleted.

Ex -> 1. Can of a Tennis ball

2. Bangles are worn on the wrist

--> Features of Stack:-

Stack provides only one end for entry and exit data.

No element can be accessed from the middle.

Note:- The end from which elements are added or deleted is called Top of the Stack.

--> Stack performs two major operations.

1. PUSH:- When an element is inserted/added on top of the stack, it is called PUSH operation.

2. POP:- When an element is deleted/ removed from the top of the stack, it is called POP operation.

--> The basic operations performed on the stack are;

1. Creating a stack
2. PUSH operation/ Adding ELements to a stack
3. Checking the status of the stack
4. Pop operation/ Deleting elements form a stack
5. Traversal/ Displaying a stack.

TO KNOW HOW TO DO THIS SEE THE CODE OF THIS CHAPTER :-)

Infix Notation:

Infix notation is the most commonly used notation for writing mathematical expressions, where operators

are placed between the operands. For example, the expression "3 + 4" is written in infix notation. In infix notation, parentheses are often used to indicate the order of operations, such as "(3 + 4) * 2".

Prefix Notation:

Prefix notation, also known as Polish notation, reverses the order of operands and operators. In this notation, the operator is placed before the operands. For example, the expression "3 + 4" in prefix notation is written as "+ 3 4". Prefix notation eliminates the need for parentheses as the order of operations is determined by the position of the operators.

Postfix Notation:

Postfix notation, also known as Reverse Polish

notation (RPN), places the operator after the operands. In this notation, the expression "3 + 4" is written as "3 4 +". Like prefix notation, postfix notation eliminates the need for parentheses as the order of operations is unambiguous.

Infix: 3 + 4

Prefix: + 3 4

Postfix: 3 4 +

Note that infix notation is more human-readable and commonly used in everyday mathematical expressions, while prefix and postfix notations are more commonly used in computer science and programming languages for parsing and evaluating mathematical expressions.