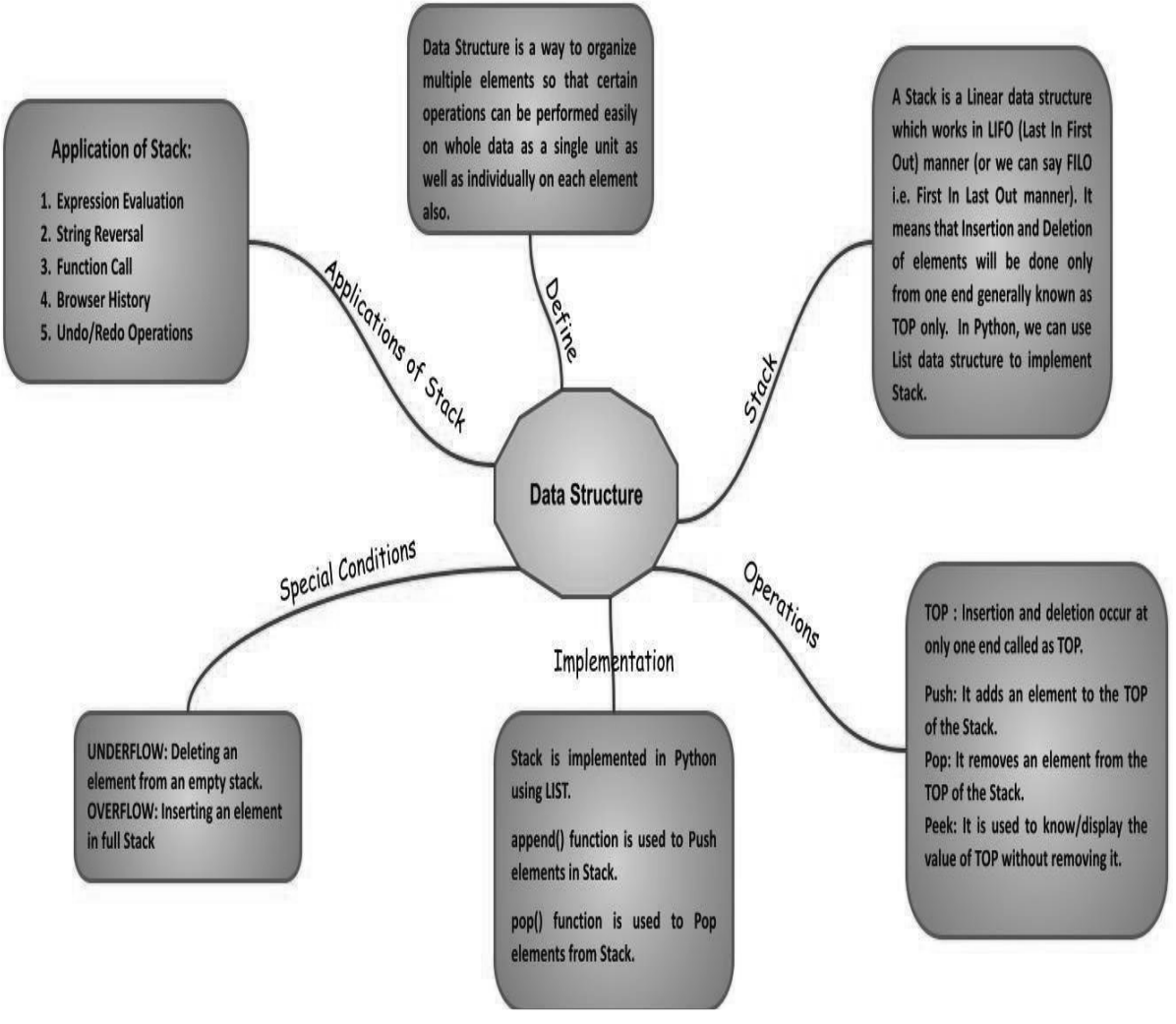
Data Structures

Topic To Be Covered:

- Stack
- Operation On Stack (Push, Pop, Peek, Display)
- Implementation Of Stack Using List
- Applications Of Stack

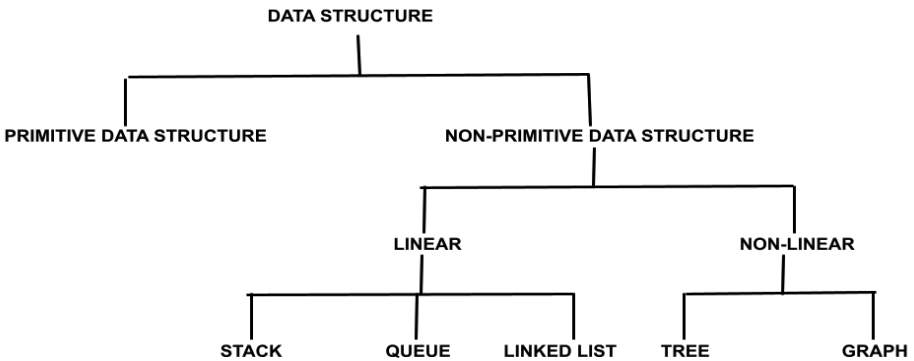

Data Structure
Define
Data Structure is a way to organize multiple elements so that certain operations can be performed easily on whole data as a single unit as well as individually on each element also.
Applications of Stack
Application of Stack:
1. Expression Evaluation
2. String Reversal
3. Function Call
4. Browser History
5. Undo/Redo Operations
Stack
A Stack is a Linear data structure which works in LIFO (Last In First Out) manner (or we can say FILO i.e. First In Last Out manner). It means that Insertion and Deletion of elements will be done only from one end generally known as TOP only. In Python, we can use List data structure to implement Stack.
Special Conditions
UNDERFLOW: Deleting an element from an empty stack.
OVERFLOW: Inserting an element in full Stack
Implementation
Stack is implemented in Python using LIST.
append() function is used to Push elements in Stack.
pop() function is used to Pop elements from Stack.
Operations
TOP : Insertion and deletion occur at only one end called as TOP.
Push: It adds an element to the TOP of the Stack.
Pop: It removes an element from the TOP of the Stack.
Peek: It is used to know/display the value of TOP without removing it.

DATA STRUCTURE

Data structure can be defined as a set of rules and operations to organize and store data efficiently. We can also say that it is a way to store data in a structured way. We can apply different operations like reversal, slicing, counting etc. of different data structures. Hence, Data Structure is a way to organize multiple elements so that certain operations can be performed easily on whole data as a single unit as well as individually on each element.

In Python, Users are allowed to create their own Data Structures which enable them to define the functionality of created data structures. Examples of User-defined data structures in Python are Stack, Queue, Tree, Linked List etc. There are some built-in data structures also available in Python like List, Tuple, Dictionary and Set.

Types of Data Structure:



Primitive Data Structures

Primitive Data Structures contain simplified data values and are directly operated by machine-level instructions. For example, integer, real, character etc. are primitive data structures.

Non-Primitive Data Structures

Non-Primitive Data Structures are derived from the primitive data structures. There are two types of non – Primitive Data Structures:

Linear data structures are single-level data structures having their elements in a sequence like a stack, queue and linked list.

Non-linear data structures are multilevel data structures like tree and graph.

STACK:

A Stack is a Linear data structure which works in a LIFO (Last In First Out) manner (or we can say FILO i.e. First In Last Out manner). It means that Insertion and Deletion of elements will be done only from one end generally known as TOP only. In Python, we can use a List data structure to implement Stack.

Application of Stack:

1. Expression Evaluation
2. String Reversal
3. Function Call
4. Browser History
5. Undo/Redo Operations

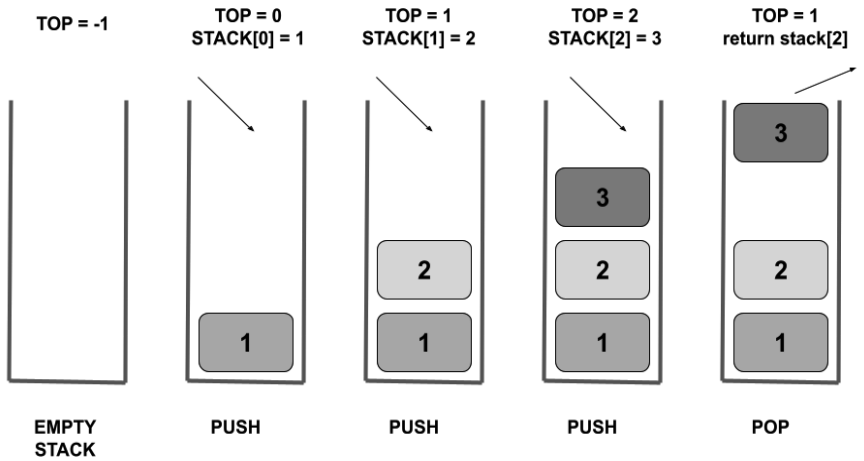
Operations of Stack:

The Stack supports the following operations:

1. **Push:** It adds an element to the TOP of the Stack.
2. **Pop:** It removes an element from the TOP of the Stack.
3. **Peek:** It is used to know/display the value of TOP without removing it.
4. **isEmpty:** It is used to check whether the Stack is empty.

OVERFLOW: It refers to the condition in which we try to PUSH an item in a Stack which is already FULL.

UNDERFLOW: It refers to the condition in which we are trying to POP an item from an empty Stack.



Implementation of Stack:

In Python, the List data structure is used to implement Stack. For the PUSH operation, we use **the append()** method of List while for the POP operation, we use the **pop()** method of List.

PROGRAM: To illustrate the basic operations of Stack:

```
def Push(emp,el): #method to add an element at TOP of stack
    top = len(emp) - 1
    emp.append(el)
    print(top)
    print("Data inserted successfully")
    return top
def Pop(emp):  #method to delete last element (TOP) of stack
    if isEmpty(emp):
        print("Stack is empty... Underflow Case")
    else:
        print("Deleted data is: ",emp.pop())
def Peek(emp): #method to display element at TOP
    if isEmpty(emp):
        print("Stack is empty...Nothing to Display")
    else:
        top = len(emp) - 1
        print("The last data added is: ",emp[top])

def Display(emp): #methjod to display stack
    if isEmpty(emp):
        print("Stack is empty... Nothing to Display")
    else:
        top = len(emp)
        d=emp[::-1]   #reversal of list done to display the TOP element first
        print("Data in stack is as follows: ")
        for i in d:
            print(i)

def isEmpty(emp):  #method to check stack is empty or not.
    if len(emp)==0:
        return True
    else:
        return False
```

```
emp=[]
top=None
while True:
    print("Stack operations")
    print("1. Add employee data.")
    print("2. Delete employee data.")
    print("3. Display employee data.")
    print("4. Display last added data.")
    print("5. Exit")
    ch=int(input("Choose operation on stack:"))
    if ch==1:
        e_no=int(input("Enter employee number: "))
        e_name=input("Enter employee name: ")
        data=[e_no,e_name]
        Push(emp,data) #method calling
    elif ch==2:
        Pop(emp)
    elif ch==3:
        Display(emp)
    elif ch==4:
        Peek(emp)
    elif ch==5:
        break
    else:
        print("Invalid choice")
```

2. A list contains the following record of customers: [CBSE EXAM 2022-23]

[Customer_name, Room_type]

Write the following user-defined functions to perform given operations on the stack named "Hotel":

i. Push_Cust() - Push customer's names of those customers who are staying in the 'Delux' Room Type.

ii. Pop_Cust() - To Pop the names of Customers from the stack and display them. Also, display "Underflow" when there are no customers in the stack.

For example: If the list with customer details is as follows:

["Siddarth", "Delux"]

["Rahul", "Standard"]

["Jerry", "Delux"]

The stack should contain:

Jerry

Siddharth

The output should be:

Jerry

Siddharth

Underflow

```
Hotel = []
Customer = [['Siddarth', 'Delux'], ['Rahul', 'Standard'], ['Jerry', 'Delux']]
def Push_Cust():
    for rec in Customer:
        if rec[1] == 'Delux':
            Hotel.append(rec[0])

def Pop_Cust():
    while len(Hotel) > 0:
        print(Hotel.pop())
    else:
        print("Underflow")

Push_Cust()
Pop_Cust()
```