Data Structure: Stack & Queue Using List

Data Structure:

Data Structure is collection values of different datatypes and stored in a single unit. A data structure has well defined operations, behaviors and properties. It is used to store, access and manipulate data in different ways. (Python inbuilt structures: list, tuple, dictionary, set)

Different Data Structures:

Data Structure combines various data values in single unit, but allow various processing on group as single unit. Data structure is classified in two ways:

1. **Simple Data Structure: These type of data structure stored the primitive data type values like integer, float, character and Boolean.**

   Example: and Linear List

   (i). Numpy Array (ii). Linear Lists

2. **Compound Data Structure: Compound data structure is formed by collection of simple data structure in various ways. There are two categories of compound data structure.**

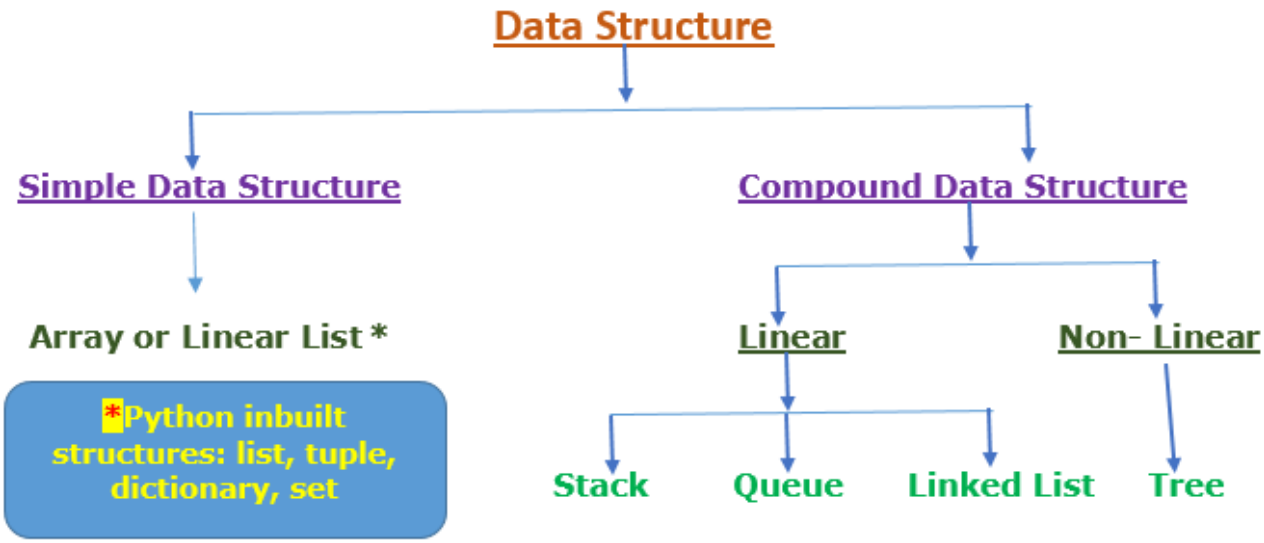
   (A) Linear Data Structure: It is a single level data structure that store the elements in a sequence.

   Example:

   (i) Stack (ii) Queue (iii) Linked List

   (B) Non-Linear Data Structure: These are Multi level data structure and data stored in multiple levels and linked together.

   Example: Tree



(Different Data Structure in Python)

Operations on Data Structures

Following are the basic operations which can performed on data structures.

1. **Insertion: Addition of new data element in data structure.**
2. **Deletion: Remove of data element from data structure.**
3. **Searching: Searching of specific element from structure.**
4. **Traversal: Processing all data element one by one.**
5. **Sorting: Arranging data element of structure either in ascending or descending order.**
6. **Merging: Combining data elements of two similar type of sorted structures and form the new data structure.**

Linear List / Array

Linear list or array is a collection of homogeneous data elements. Elements are assembled in array in form of index. Insertion and deletion can take place at any location in the array.

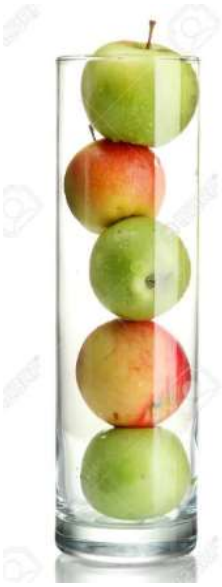
Array can be implemented using numpy array in python.

Ex. L=[2,4,6,7] Elements : 2,4,6,7 Index: 0,1,2,3

Array can be 1-D, 2-D or Multi- dimensional

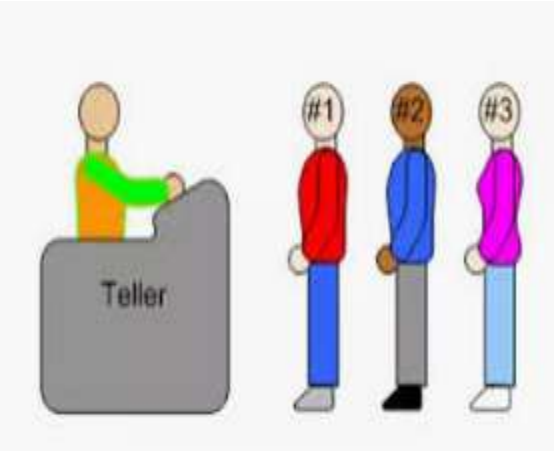
Stack

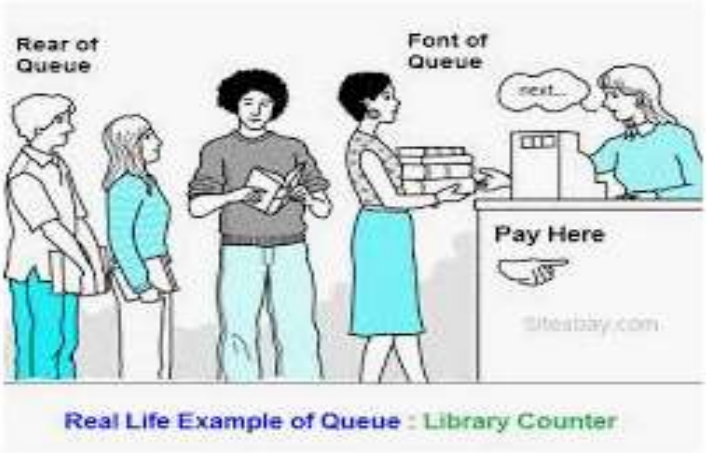
Stack structure stores the list elements in such a way that LIFO (Last In First Out) technique followed. In stack the Insertion and Deletion operation take place at one end that is called top.



Queue

Queue data structure is works on FIFO (First In First Out) technique. In Queue the Insertion of element take place at "rear" end where Deletion at "front" end.



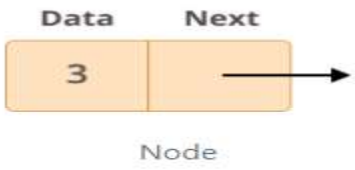


Linked List

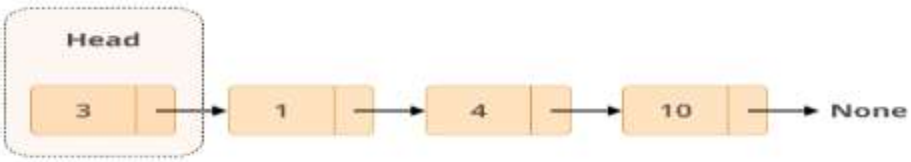
Linked list is special type of data structure in which elements linked to one another. Logically the first element pointing to second element and so on.

Each element is called node that has two parts.

(A) Data / Info : it store the data / element value.
(B) Pointer / Next: it makes a reference that store reference of next node.

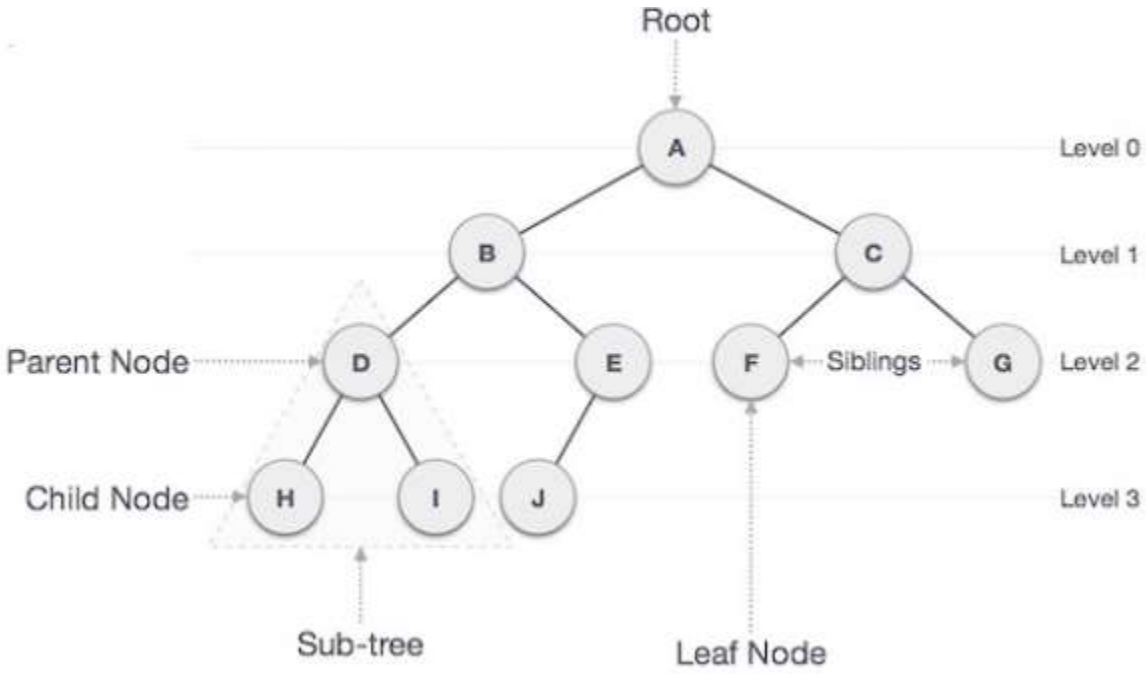


The first node is called the `head`, and it's used as the starting point for any iteration through the list. The last node must have its `next` reference pointing to `None` to determine the end of the list. Here's how it looks:



Tree

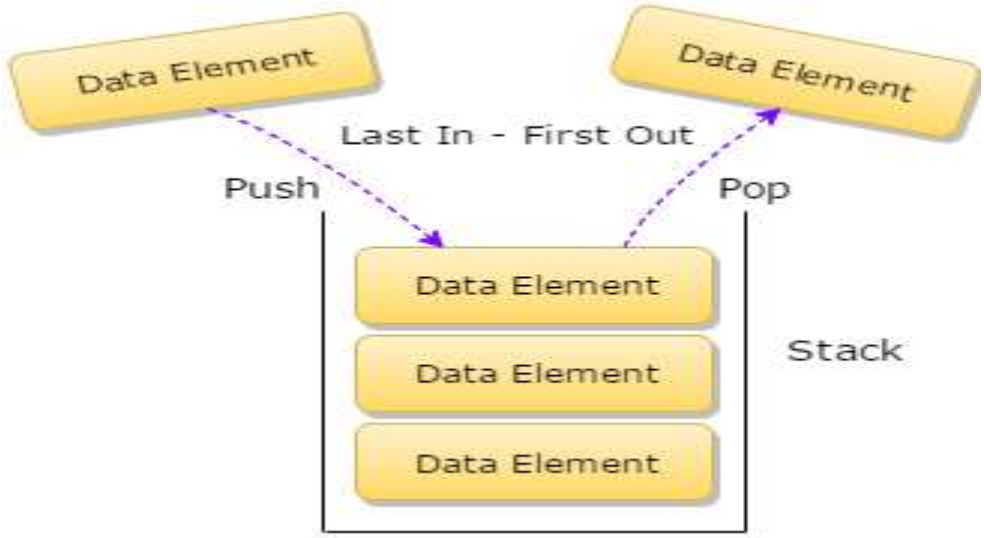
Tree is Multi Level data structure. It has hierarchical relationship among the nodes (elements). Each of node has its reference pointer that points to the node below it.



Stack Data Structure

Stack is linear structure implemented in LIFO (Last in First Out) manner where insertions and deletions are take place at one end- TOP. It follow the following operations..

1. **POP Operation: Data can only be removed from top. That mean element at top will remove and this is called POP operation.**
2. **PUSH Operation: A new data element can only be added at the top of the stack and this is called PUSH operation.**



3. **Peek / Inspection: To inspect the element at the top of stack without removing it is referred as Peek.**
4. **Overflow: In the static Stack, when it is already full then we can't increase the size of stack. When user tries to PUSH new element in already full stack then an error occurred and this situation known as stack overflow.**
5. **Underflow: It is situation when one tries to POP element from an empty stack.**

For Program of Stack, Visit the blog.

https://ictswami.wordpress.com/python-projects/

Application of Stack

In Python there are three types of notations of expression.

1. Infix notation 2. Postfix notation 3. Prefix notation

Infix Notation	Prefix Notation	Postfix Notation
A+B	+AB	AB+
(A-C)*B	*-ACB	AC-B*
A+(B*C)	+A*BC	ABC*+

There are two Stack Application used in Python.

1. **Infix to postfix Conversion using Stack**
2. **Evaluation of Expression using Stack**

1. Infix to postfix Conversion using Stack

To convert infix to postfix notation by using stack, there are some priority rules for evaluation. These rules are

(i) Bracket or Parenthesis
(ii) Exponentiation
(iii) Multiplication or Division
(iv) Addition or Subtraction

General Rules:

1. **If input is operand/ Variable then shift in output.**
2. **If input is operator then must be PUSH in stack as per following cases.**
   - **(i) If Stack operators having equal or higher priority than input operator, in this case first POP all Equal and Higher priority operators from stack one by one and after it PUSH input operator in stack.**
   - **(ii) Other than above case mentioned in part (i), the input operator will directly PUSH in stack.**
3. **(or { or [will be remove only with their respective) or } or]**
4. **At last POP all operators one by one from stack and shift in output.**
5. **Stack position at the end of conversion should be Empty.**

Example: Convert (A+B)* C/D into postfix notation.

Step	Input	Action	Stack Position	Output
1	(	PUSH in Stack	(	----------
2	A	Shift to Output	(	A
3	+	PUSH in Stack	(+	A
4	B	Shift to Output	(+	AB
5	)	POP + Remove (	Empty	AB+
6	*	PUSH in stack	*	AB+
7	C	Shift to Output	*	AB+C
8	/	POP * PUSH / in Stack	/	AB+C*
9	D	Shift to Output	/	AB+C*D
10	---------	POP all Operators from Stack	Empty	AB+C*D/

2. Evaluation of Expression using Stack

Rules for evaluation:

1. **If input is operand (Value/True/False) then the operand will PUSH in stack.**
2. **If input is Unary operator then POP one value from stack to evaluate with operator and PUSH the result in stack.**
3. **If input is Binary operator then POP Two values from stack to evaluate with operator and PUSH the result in stack.**
4. **Intermediate Evaluation should be as…**
   Result= (Second POP Value) Operator (First POP Value)
5. **At the end of evaluation, POP all Values from stack one by one to find final result. Make sure that stack should empty.**

Example-1: Evaluate the expression 5 6 2 + * 12 4 / - by using stack.

Step	Input	Action	Stack Position	Intermediate Output
1	5	PUSH in Stack	5	---------
2	6	PUSH in Stack	5, 6	---------
3	2	PUSH in Stack	5, 6, 2	---------
4	+	POP 2 elements and PUSH(6+2=8)	5, 8	6+2=8
5	*	POP 2 elements and PUSH(8*5=40)	40	8*5=40
6	12	PUSH in Stack	40, 12	
7	4	PUSH in Stack	40, 12, 4	
8	/	POP 2 elements and PUSH(12/4=3)	40, 3	12/4=3
9	-	POP 2 elements and PUSH(40-3=37)	37	40-3=37
10	------	POP All elements	Empty	37 (Result)

Example-2: Evaluate the expression by using stack.

True, False, True, NOT, False, True, OR, NOT, AND, OR, AND

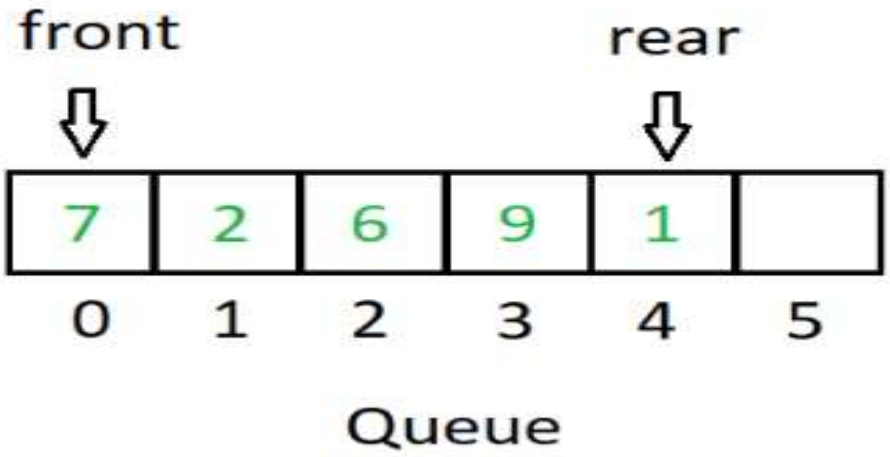
Step	Input	Action	Stack Position	Intermediate Output
1	True	PUSH in Stack	True,	---------
2	False	PUSH in Stack	True,False	---------
3	True	PUSH in Stack	True,False,True	---------
4	NOT	POP 1 elements and PUSH NOT(True)=False	True,False,False	NOT(True)=False
5	False	PUSH in Stack	True,False,False,False	---------
6	True	PUSH in Stack	True,False,False,False,True	---------
7	OR	POP 2 elements and PUSH False OR True=True	True,False,False,True	False OR True=True
8	NOT	POP 1 elements and PUSH NOT(True)=False	True,False,False,False	NOT(True)=False
9	AND	POP 2 elements and PUSH False AND False=False	True,False,False	False AND False=False
10	OR	POP 2 elements and PUSH False OR False=False	True,False	False OR False=False
11	AND	POP 2 elements and PUSH True AND False=False	False	True AND False=False
12	------	POP All from Stack	Empty	False (Result)

Queue Data Structure

Similar to stack, the queue is also consist of sequence of elements in linear list. The queue works on FIFO (First In First Out) Mechanism. The queue structure has two ends.

Front end: This is the end where the Deletion operation of queue take place. That mean the element will delete at the front end of queue. Removal of element from queue is called DEQUEUE operation

Rear end: This is the end where the Insertion operation of queue take place. That mean the new element will insert at the rear end of queue. Insertion of element into queue is called ENQUEUE operation



Queue

1. **Peek / Inspection:** To inspect the element at front of queue without removing it is referred as Peek.
2. **Overflow:** In the static Queue, when it is already full then we can't increase the size of Queue. When user tries to ENQUEUE new element in already full Queue then an error occurred and this situation known as Queue overflow.
3. **Underflow:** It is situation when one tries to DEQUEUE an element from an empty Queue.

For Program of Stack, Visit the blog.

https://ictswami.wordpress.com/python-projects/