6 Data Structures in Python

6.1 INTRODUCTION

Every programming language provides salient features of data structures. A data structure is more precisely defined as a way of storing, organizing, accessing and retrieving data in a computer so that it can be used most efficiently to give optimal performance. For example, string is a data structure containing a sequence of elements where each element is a character. On the other hand, list is a sequence data structure in which each element may be of different type. We can apply different operations like traversal, reversal, slicing, counting of elements, etc., on strings and lists. Hence, a data structure organizes multiple elements in such a way that certain operations on each element as well as the collective data unit could be performed easily. There are several types of data structures which are designed and used for different kinds of applications.

CTM: A Data Structure is a named group of data of different data types which is stored in a specific way and can be processed as a single unit. A data structure has well-defined operations, behaviour and properties.

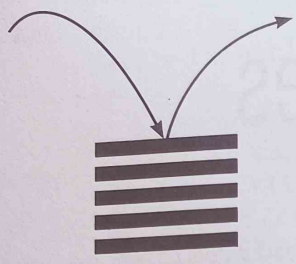
In Class XI, we have studied about an important sequential linear data structure, **list**, which is an ordered sequence of items. When we intend to refer to or access a specific value in the sequence/list, it is done using index value or subscript and is accessed from the 0^{th} position till *lastindex* – 1.

Python lists are dynamic in nature, *i.e.*, they can grow/increase and shrink/decrease in size/number of items as and when required. They are also heterogeneous, which means we can store elements of multiple data types in a single list. But there are considerable differences between data type and data structures in Python which are as follows:

- **Data Type** defines the type of values we can store and operations we can perform on those values. For example, in **int** data type we cannot store decimal values and we cannot multiply two string type values. It also specifies the amount of memory it will take.
- On the contrary, **Data Structure** is the physical implementation that clearly defines a way of storing, accessing and manipulation of data stored in data structure. Every data structure has a specific way of insertion and deletion, like Stack works on LIFO (Last In First Out), *i.e.*, all operations will take from one end, *i.e.*, TOP, whereas QUEUE works on FIFO (First In First Out), *i.e.*, item inserted first will be removed first and new item will always be added to the end of Queue.

In this chapter, we shall further discuss two data structures—Stacks and Queues—and their implementation in Python using lists.

6.2 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**. Because of this, Stack is called **LIFO (last in, first out)** data structure. LIFO means the element inserted last would be the first to be deleted.

For example, a pile of books, a Stack of coins, where you can remove only the top book or the coin placed at the top (Fig. 6.1).

Fig. 6.1: Pile of Books/Stack of Coins

Some of the applications of Stack in real life are:

- Pile of clothes in an almirah
- Multiple chairs in a vertical pile
- Bangles worn on wrist
- Pile of boxes of eatables in a pantry or on a kitchen shelf

Stack performs two major operations, *viz.* **PUSH** and **POP**.

(i) When an element is inserted/added on top of the Stack, it is called **PUSH** operation.

(ii) When an element is deleted/removed from the top of the Stack, it is called **POP** operation.

To perform the above two major operations, we need to declare a variable that points to the top of the Stack, *i.e.*, "TOP", and acts as a storage location or a container to store elements of the Stack.

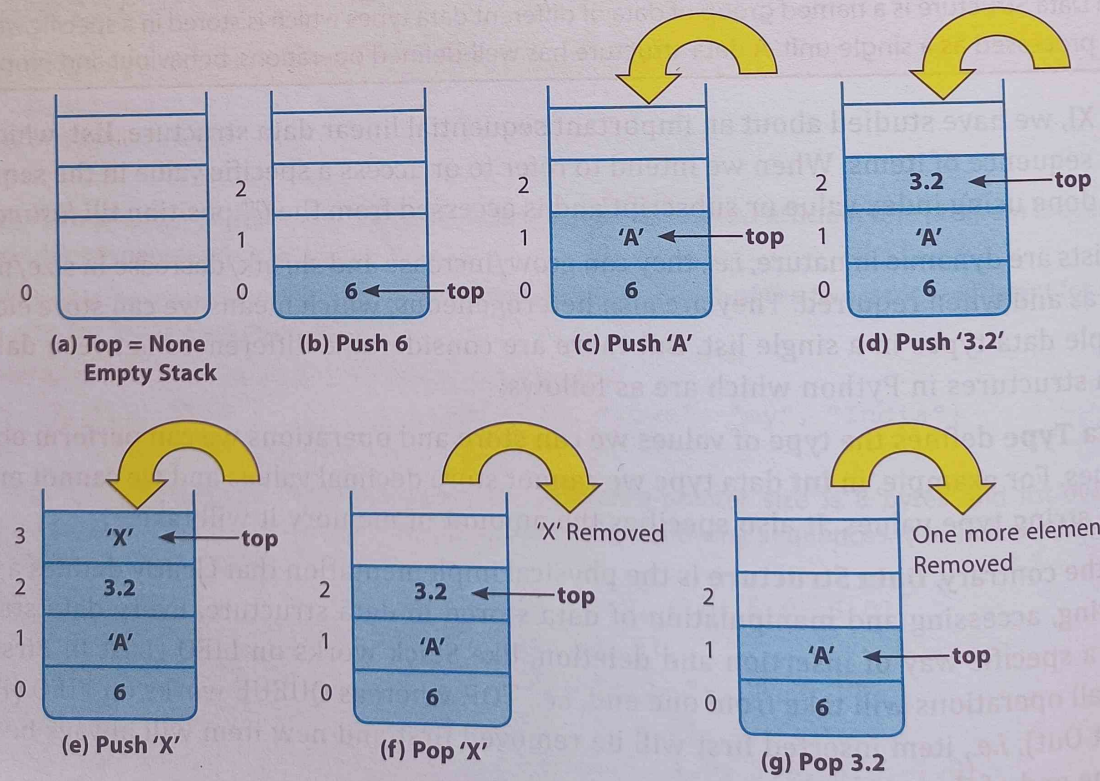


Fig. 6.2: Stack Operations—PUSH and POP

As is evident from Fig. 6.2, the two Stack operations are performed through the "top"—elements are inserted as well as deleted from the top only.

The Stack is a dynamic data structure as it grows (with an increase in the number of elements) or shrinks (with a decrease in the number of elements). A static data structure, on the other hand, is one that has a fixed size.

Stacks are fundamentally important as they can be used to reverse the order of items or elements. The order of insertion is reverse of the order of removal. Fig. 6.3 shows the Python data object Stack created in the original order and traversed/deleted in the reverse order.

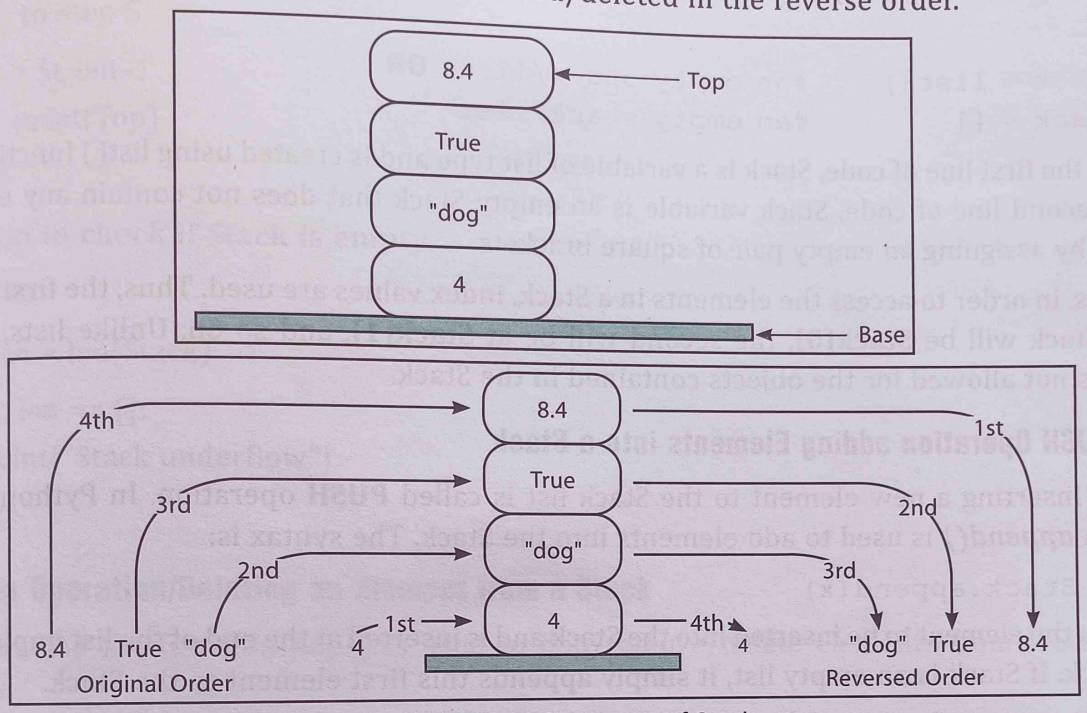


Fig. 6.3: Reversal Property of Stack

One of the most frequently used applications of Stack that we encounter in our day-to-day life is when a user browses the internet for information retrieval. While navigating a website, web page to web page, these pages are placed on a Stack or, in other words, the URLs are placed on the Stack. The current page that you are viewing is on the top and the first page you looked at is at the bottom. Now, when you click on the "Back" button, you begin to move in reverse order through the pages. Thus, this browsing process is implemented through Stack.

Even the compilers that are conferred with the job of debugging and translation also use Stack for evaluating expressions, generating machine code and during function calls.

6.3 IMPLEMENTATION OF STACK USING LIST

The implementation of Stack using list in Python is the easiest of all programming languages. It offers a convenient set of methods to operate lists as Stacks.

The basic operations performed on the Stack are:

1. Creating a Stack
2. PUSH Operation/Adding Elements to a Stack
3. Pop Operation/Deleting Elements from a Stack
4. Traversal/Displaying a Stack.

Checking the Status of Stack

Before implementing these operations, it is important to check Stack status.

6.3.1 Creating a Stack

The built-in data structure list available in Python makes it flexible and easy to implement Stack. In Python, when you declare/create a list, it creates an address in memory and can hold any number of heterogeneous elements. Thus, in order to create an empty Stack, we just need to use inbuilt function list() or [] as per the syntax given below:

Syntax:

```
Stack = list()      #an empty stack/list  OR
Stack = []          #an empty stack/list
```

Here, in the first line of code, Stack is a variable of list type and is created using list() function while in the second line of code, Stack variable is an empty Stack that does not contain any elements, created by assigning an empty pair of square brackets.

Like lists, in order to access the elements in a Stack, index values are used. Thus, the first element in the Stack will be **Stack[0]**, the second will be at **Stack[1]**, and so on. Unlike lists, random access is not allowed for the objects contained in the Stack.

6.3.2 PUSH Operation adding Elements into a Stack

Adding/Inserting a new element to the Stack list is called **PUSH** operation. In Python, the list function ***append()*** is used to add elements into the Stack. The syntax is:

Syntax: `Stack.append(x)`

Here, x is the element to be inserted into the Stack and is inserted at the end of the list implemented as a Stack. If Stack is an empty list, it simply appends this first element to the Stack.

Note: Attempts to insert an element into a full Stack, *i.e.*, (Top=n–1), where n is the size of the Stack, is called **Stack Overflow**.

Algorithm for PUSH Operation

The steps to be performed for carrying out the PUSH operation on the Stack are as follows:

1. START
2. Stack=list() or Stack=[] #Initialize a Stack using list
3. element = input("Enter the value to be added in the stack :")
4. Stack.append(element) #Adding element into list
5. END

While we perform PUSH operation into the Stack, we push the 'element' onto the Stack using append() function.

To use Stack efficiently, we need to check the status of Stack as well. For the same purpose, the following functionality is added to Stack:

(i) peek()—To get the most recent value of stack, *i.e.*, top element of the Stack without removing it or value at the top. It will throw an exception if the Stack is empty or null.

(ii) isFull()(Overflow)—To check if Stack is full which leads to overflow while trying to insert a new element in the Stack. In Python (for stack and queue implemented as list), since list can grow, no overflow condition will arise until all memory is exhausted.

(iii) isEmpty() (Underflow)—To check if Stack is empty which leads to underflow while trying to delete an element from the empty Stack.

Algorithm to display top element without deleting it, *i.e.*, peek()

1. START
2. St_len = len(Stack) #Count the total number of elements in a Stack
3. if St_len == []:
   print("Underflow")
   go to step 5
4. Top = St_len−1
   print(Top)
5. END

Algorithm to check if Stack is empty

1. START
2. St_len = len(Stack)
3. if St_len == []:
   print("Stack underflow")
4. END

6.3.3 Pop Operation/Deleting an Element from a Stack

In Python, the list function pop() is used to pop/remove/delete elements from a Stack. The syntax is:

```
Stack.pop()
```

Here,

- The function pop() removes the last element which is on the top of the Stack.
- It also returns the popped/deleted elements from the list.

Therefore, in order to delete an element from the Stack, no subscript/index value is required to be passed to the function pop(). If a user tries to delete an element from the empty Stack, it is called underflow of Stack.

Algorithm to delete an element from a Stack

Steps to be followed are:

1. START
2. St_len = len(Stack) #Count the total number of elements in the Stack
3. if St_len == []: #Checks whether the Stack is empty or not
   print("Stack is empty")
   go to step 5
4. element = Stack.pop() #Removing last element from top of the Stack
5. print(element)
6. END

The above algorithm can also be written in an alternative manner, which is as follows:

1. START
2. St_len = len(Stack) #Count the total number of elements in the Stack
3. if not Stack: #Checks whether any element is present inside the Stack or not
   print("Stack is empty")
   go to step 5
4. element = Stack.pop() #Removing last element from the Stack
5. print(element)
6. END

Alternatively,

1. START
2. if not len(Stack): #Checks whether any element is present inside the Stack or not
   print("Stack is empty")
   go to step 4
3. element = Stack.pop() #Removing last element from the Stack
4. print(element)
5. END

- In the above algorithms, the if condition shall check whether the Stack is empty or not. In case of non-empty Stack only, the element shall be removed from the Stack using the built-in function pop() that removes last element.
- Once an element is popped, the Stack automatically reduces one index position.

6.3.4 Traversing/Displaying Stack Elements

By now we have understood that a Stack is a data type that serves as a collection of elements with two principal operations: push(), which adds an element to the collection, and pop(), which removes the most recently added element that was not yet removed. In order to check the status of elements on the Stack whenever PUSH and POP operations are performed, traversal of Stack elements becomes mandatory. In traversal process, the elements from the top position, *i.e.*, last inserted element, get displayed and processed in the reverse order.

Algorithm for traversing Stack Elements:

1. START
2. l = len(Stack)
3. for i in range(l–1,–1,–1):
   print(Stack[i])
4. End

Practical Implementation–1

Write a Python program to implement all basic operations of a Stack, such as adding element (PUSH operation), removing element (POP operation) and displaying the Stack elements (Traversal operation) using lists.

prog_st1.py - C:/Users/preeti/AppData/Local/Programs/Python/Python37-32/prog_st1.py (3.7.0)

File Edit Format Run Options Window Help

```
#Implementation of List as stack
s=[]
c="y"
while (c=="y"):
    print ("1. PUSH")
    print ("2. POP ")
    print ("3. Display")
    choice=int(input("Enter your choice: "))
    if (choice==1):
        a=input("Enter any number :")
        s.append(a)
    elif (choice==2):
        if (s==[]):
            print ("Stack Empty")
        else:
            print ("Deleted element is : ",s.pop())
    elif (choice==3):
        l=len(s)
        for i in range(l-1,-1,-1):  #To display elements from last element to first
            print (s[i])
    else:
        print("Wrong Input")
    c=input("Do you want to continue or not? ")
```

Ln: 27 Col: 4

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
 RESTART: C:\Users\preeti\AppData\Local\Programs\Pyt
hon\Python37-32\prog_st1.py
1. PUSH
2. POP
3. Display
Enter your choice: 1
Enter any number :5
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 1
Enter any number :t
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 1
Enter any number :66
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 1
Enter any number :88
```

Ln: 56 Col: 33

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 3
88
66
t
5
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 2
Deleted element is :  88
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 2
Deleted element is :  66
Do you want to continue or not? y
1. PUSH
2. POP
3. Display
Enter your choice: 3
t
5
Do you want to continue or not? n
```

Ln: 56 Col: 33

Practical Implementation–2

Write a program to display unique vowels present in the given word using Stack.

```
#Write a program to display unique vowels present in the given word
#using Stack

vowels =['a','e','i','o','u']
word = input("Enter the word to search for vowels :")
Stack = []
for letter in word:
    if letter in vowels:
        if letter not in Stack:
            Stack.append(letter)
print(Stack)
print("The number of different vowels present in",word,"is",len(Stack))
```

Explanation:

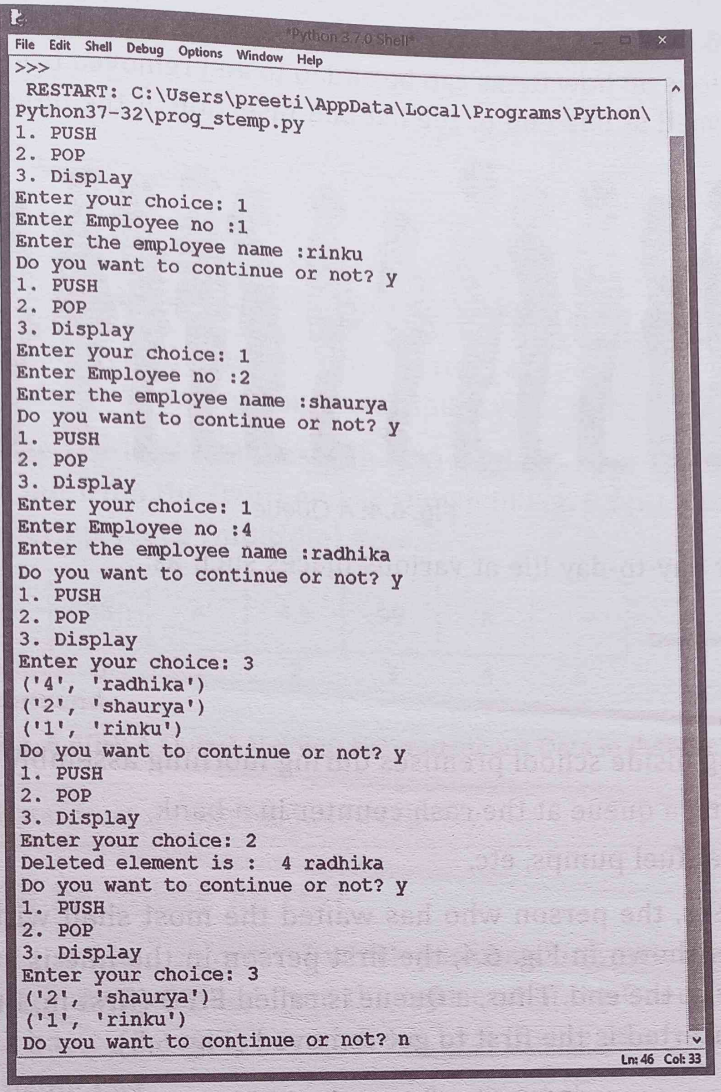
In the above program, we have to find the number of non-repetitive vowels present in the inputted word. This is done by comparing each letter of the word with the vowels list and, when found, the vowels are added to the Stack (only unique vowels). Thus, Stack in the above program acts as a container to hold all the vowels which are present in the given word and finally displayed using len(Stack) function with print() statement and, hence, the output shown below is obtained.

```
Type "copyright", "credits" or "license()" for more informat
ion.
>>>
 RESTART: C:/Users/preeti/AppData/Local/Programs/Python/Pyth
on37-32/prog_st3.py
Enter the word to search for vowels :HelloPython
['e', 'o']
The number of different vowels present in HelloPython is 2
>>>
```

Practical Implementation–3

Write a program to create a Stack called Employee, to perform the basic operations on Stack using list. The list contains the two values—employee number and employee name. The program should include the options for addition, deletion and display of employee details.

```
#Program to add, delete and display the records of an Employee using list
#implementation through Stack
Employee=[]
c="y"
while (c=="y"):
    print ("1. PUSH")
    print ("2. POP ")
    print ("3. Display")
    choice=int(input("Enter your choice: "))
    if (choice==1):
        e_id=input("Enter Employee no :")
        ename=input("Enter the employee name :")
        emp = (e_id,ename)
        Employee.append(emp)
    elif (choice==2):
        if (Employee==[]):
            print ("Stack Empty")
        else:
            e_id,ename =Employee.pop()
            print ("Deleted element is : ",e_id,ename)
    elif (choice==3):
        i=len(Employee)
        while i> 0:  #To display elements from last element to first
            print(Employee[i-1])
            i = i - 1
    else:
        print("Wrong Input")
    c=input("Do you want to continue or not? ")
```



Applications of Stacks

Some of the important applications of Stacks include:

1. **Reversing a Word/Line:** A simple example of Stack application is reversal of a given line. This can be accomplished by pushing each character on to a Stack as it is read. When the line is finished, characters are then popped off the Stack and they will come off in the reverse order.
2. The compilers use Stacks to store the previous state of a program when a function is called or during recursion.
3. Another important Stack application is backtracking. (Backtracking is a form of recursion. But it involves choosing only one option out of the possibilities.) Backtracking is used in a large number of puzzles like Sudoku and in optimization problems such as knapsack.
4. **Undo Mechanism in Text Editors:** This operation is accomplished by keeping all text changes in a Stack.

6.4 QUEUE

In the previous topics, we have discussed the implementation of Stack. Now, we will learn the implementation of Queue with Python list.

Queues are similar to Stacks in that a Queue also consists of a sequence of items (a linear list), but there are restrictions on how items can be added to and removed from the list. Queue permits deletion to be performed at one end of the list and insertion at the other.

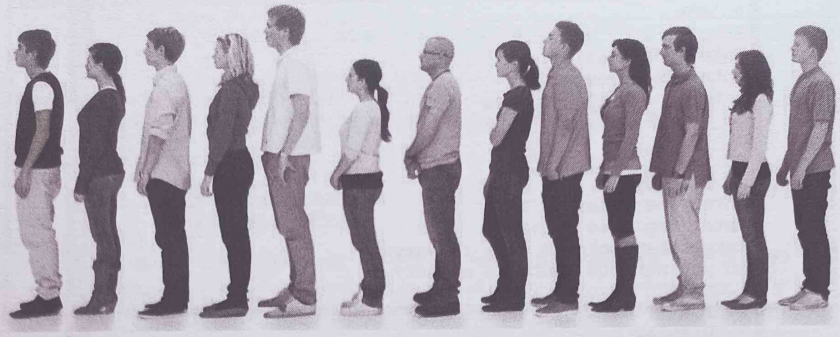


Fig. 6.4: A Queue

We see Queues in our day-to-day life at various places such as:

- Bank ATM,
- Fee counter,
- Shopping centre,
- Students standing inside school premises during morning assembly,
- Customers forming a queue at the cash counter in a bank,
- Vehicles queued at fuel pumps, etc.

In all of the above cases, the person who has waited the most shall withdraw the money first or pay the fee first. As shown in Fig. 6.4, the first person in the line is served first, and others can enter the line only at the end. Thus, a Queue is called FIFO (first in, first out) data structure, where the item first inserted is the first to get removed (Fig. 6.5).

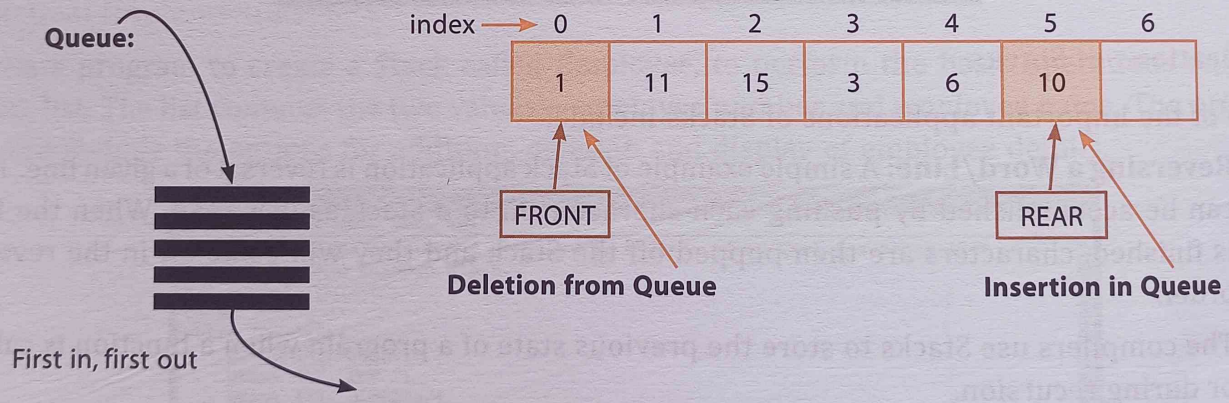


Fig. 6.5: A Queue in Python

Thus, Queue is also known as a First Come First Served (FCFS) approach.

A Queue consists of two open ends—**Front** and **Rear** (Fig. 6.5).

1. Data can only be removed from the front end, *i.e.*, the element at the front end of the Queue.
2. A new data element can only be added to the rear end of the Queue.

In Python, an empty list is created by using list() function. This empty list can be manipulated as a Queue.

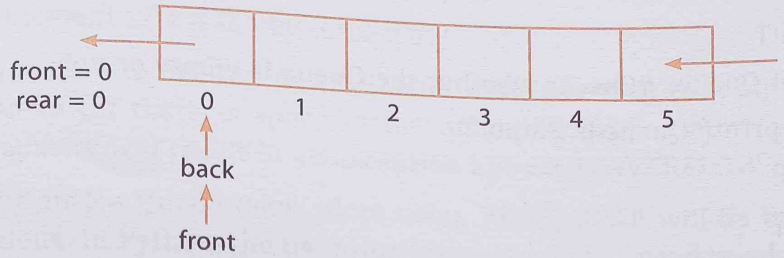


Fig. 6.6(a): A Queue with Index Values in Python

When we implement default list as Queue, the rear and front are placed at 0^{th} position of the Queue as shown in Fig. 6.6(a). It shows an empty Queue with index value starting from 0 with no elements.

If an element is to be inserted in this Queue, it has to be done from the **rear** end and deletion of an element is carried out from the **front** end as shown in Fig. 6.6(b). Like a Stack, Queue also holds data objects of heterogeneous (multiple) type.

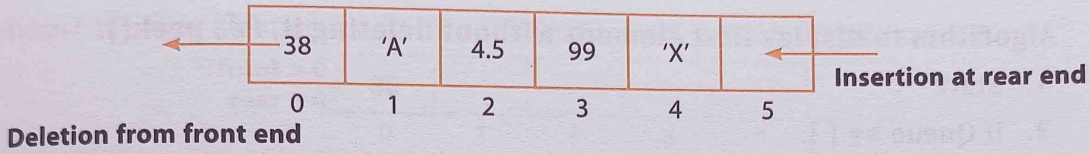


Fig. 6.6(b): A Queue Holding Heterogeneous Data in Python

CTM: Queue is a linear data structure that follows FIFO policy.

6.5 IMPLEMENTATION OF QUEUE AS LIST

A Queue is a specialized list (data structure) which performs only two operations, *i.e.*, insertion and deletion. Elements are always added from the **rear end** of the Queue and removed from the **front** of the Queue. The basic operations carried out by a Queue as a list are:

(a) Creating a Queue using List
(b) Inserting an element in a Queue (Enqueue)
(c) Deleting Elements from a Queue (Dequeue)
(d) Traversal/Displaying Queue items

Before implementing these operations, it is important to check the queue status.

☞ **Checking the Queue Status:** The following constructs are used to check the status of queue while adding or deleting elements from it.

(i) isEmpty()(Underflow): It is used to check whether the queue has any element or not so as to avoid underflow exception while performing dequeue operation.

Algorithm to Check an Empty Queue

The steps to be followed are as follows:

1. START
2. if Queue == []: # Checks whether the Queue is empty or not
   print("Queue is Empty")

STOP

Or

1. START
2. if not Queue: #Checks whether the Queue is empty or not
   print("Queue is Empty")
3. STOP

Or

1. START
2. if not len(Queue): #Checks whether the Queue is empty or not
   print("Queue is Empty")
3. STOP

(ii) peek(): It is used to view elements at the front end of the queue, without removing them from the queue.

Algorithm to display first element without deleting it, *i.e.*, peek():

1. Start
2. if Queue == [ ]:
   print("Queue is Empty")
   Go to step 4
3. print(Queue[0])
4. End

(iii) isFull() (Overflow): It is used to check whether any more element can be added to the queue or not to avoid overflow exceptions while performing enqueue operation.

POINT TO REMEMBER

We don't need to implement isFull as Python, being a dynamic language, does not ask for creation of list having a fixed size. Hence, we will never encounter a situation when the queue is full.

Let us now discuss the above operations in detail.

6.5.1 Creating a Queue using List

Whenever we create a list in Python, it creates an address in memory and can store any number of heterogeneous elements. Like a Stack is created using the Python inbuilt function list(), in the same manner Queue is created. The Queue created using list() method is shown below and can be implemented as linear structure of a Queue. The syntax is:

```
Queue = list()    OR
Queue = []
```

Here, in the first line of code, an empty Queue is created as a variable of list type and in the second line of code, Queue variable is an empty Queue as it does not contain any element.

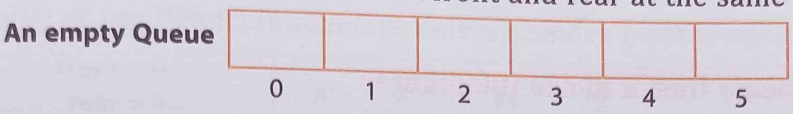
The first element added to the Queue will be at **Queue[0]**, where both front and rear position is 0, followed by the second element at **Queue[1]**, where front will be at 0 and rear shall increment by 1 and shall point to index 1 and so on.

6.5.2 Inserting an Element into a Queue (ENQUEUE)

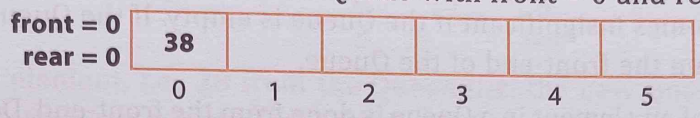
Enqueue operation is used to insert a new element to the queue at the rear end. We can insert elements in the queue till there is space in the queue for adding more elements. Inserting elements beyond capacity will result in an exception known as OVERFLOW.

Inserting an element into a Queue takes place using REAR; REAR will be incremented to hold the new value in Queue. In Python, the list function `append()` is used to add elements to a list. Suppose we want to insert 5 elements, 38, 'A', 4.5, 99, 'X', in a Queue. Then the following steps show the insertion of elements in a Queue using a list. Also, the position of rear gets changed at every step of insertion.

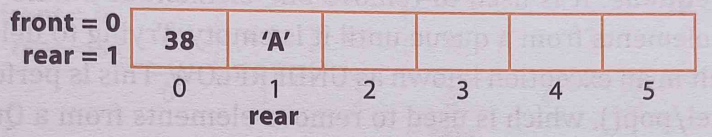
Step 1: To begin with an empty Queue, with both front and rear at the same position.



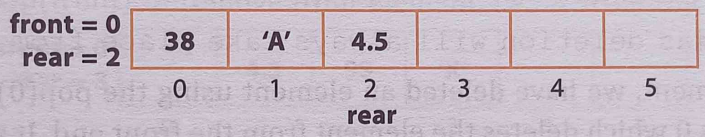
Step 2: Insert the first element as 38 into the Queue with front = 0 and rear = 0.



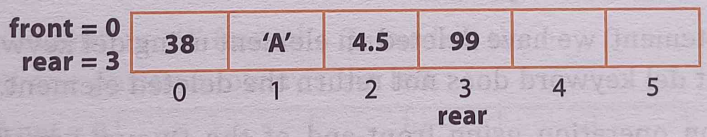
Step 3: Next element 'A' is inserted into the Queue with front = 0 and rear = 1.



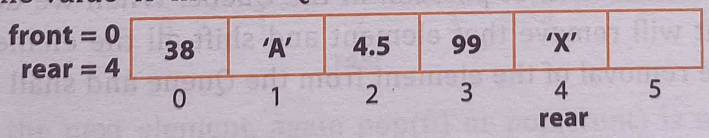
Step 4: Insert another element 4.5 into the Queue with front = 0 and rear = 2.



Step 5: Insert 99 as the next element in the Queue with front = 0 and rear =3.



Step 6: Finally, insert the value 'X' into the Queue with front = 0 and rear = 4.



As you must have observed in all the above six steps to insert elements in a Queue, we have inserted 5 elements at one end (rear) and, hence, the index value for rear has been changed every time a new item has been inserted, but the value of front remains the same at 0. So, with the insertion of the last element, front remains at 0^{th} position of the Queue and the rear reaches the 4^{th} position of the Queue. Inserting an element in a Queue is called **Enqueue**. In Python, list function `append()` is used to add elements to the Queue. The syntax is:

Syntax:

```
Queue.append(x)
```

Here, x is the element to be inserted at the rear end position in the Queue. If Queue is an empty list, it simply appends elements where the position is front=0 and rear=0.

Algorithm to insert an element into a list Queue

The built-in list function `append()` is used to insert elements at the rear end in the Queue. Thus, the algorithm to append/insert an element in a Queue is as follows:

1. START
2. Queue = list() or Queue = [] #Initialize a Queue using list
3. element = input("Enter the element to be added to the Queue:")
4. Queue.append(element) #Adding element into the list Queue
5. STOP

Here, we insert an element into a list Queue from the rear end using the built-in list function append().

6.5.3 Deleting Elements from a Queue (DEQUEUE)

Before deleting an element from a Queue, it becomes mandatory to check whether the Queue is empty or not as it becomes insignificant if the Queue is empty. If the Queue is non-empty, we can remove element(s) from the front-end of the Queue.

The deletion/removal of an element in a Queue is done from the front-end. Deleting an element from the Queue is called **Dequeue**. It is used to remove one element at a time from the FRONT of the queue. We can delete elements from a queue until it is empty. Trying to delete an element from an empty queue will result in an exception known as UNDERFLOW. This is performed using the Python built-in list function del/pop(), which is used to remove elements from a Queue. The syntax is:

```
Queue.pop(0)
del Queue[0] #as deletion will always take place from 0 index position
```

- In the first statement, we have deleted an element using the pop(0), the position or index value of the list is 0 which deletes the element from the front end. It also allows us to return the deleted element from the Queue.
- In the second statement, we have deleted an element using del keyword where the position of the list is 0 but del keyword does not return the deleted element.

To begin with deletion operation using front-end of the Queue, we should use the `pop(0)` function. The 0^{th} position is the first position in the Queue. When we delete an element using the pop(0) function, it will remove that element and shift all the elements after it to "fill the gaps" created after the removal of the element from the Queue and shall return the value which was removed.

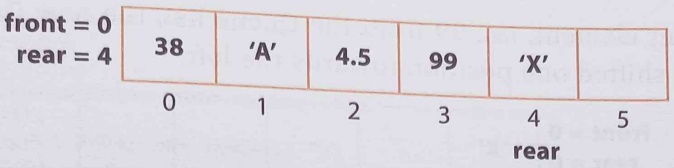
Algorithm to delete an element from a list Queue

1. START
2. if Queue == [ ]: # Checks whether the Queue is empty or not
   print("Queue is Empty")
   go to step 4
3. element = Queue.pop(0) # Deleting an element
   OR del Queue[0]
4. print(element)
5. END

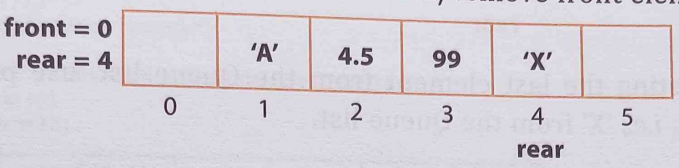
Learning Tip: A Queue is represented as a list or collection with the restriction that insertion can be performed at one end (rear) and deletion can be performed at the other end (front).

Let us understand this process using the same example of a list Queue which we took while performing insertion of elements into the Queue. The list Queue, which we created in the previous section, holds 5 elements, *viz.*, **38, 'A', 4.5, 99, 'X'**. To remove these elements one by one from the Queue front, the following steps are to be taken along with the value changed for front and rear position:

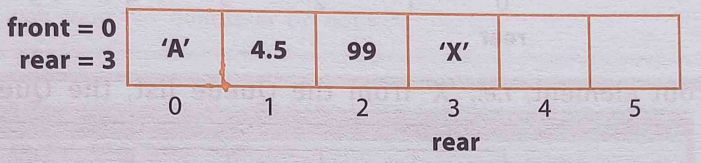
Step 1: A Queue consists of five elements as shown:



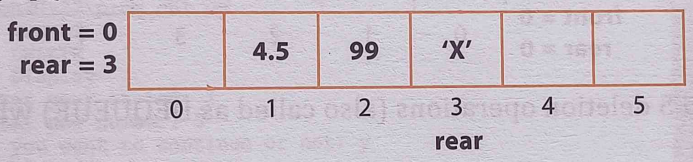
Step 2: Execute pop(0) or pop(front) function to delete/remove front element, which is 38.



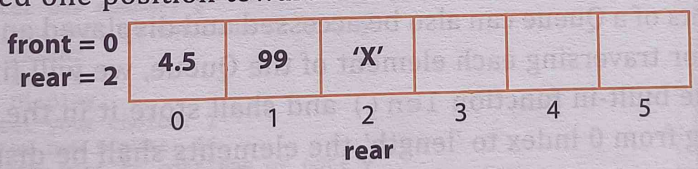
After deleting the front element, *i.e.*, 38 from the Queue list, the new Queue automatically fills the gap and the list is shifted one position towards the left.



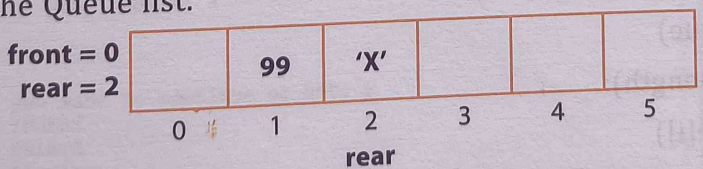
Step 3: Use pop(0) or pop(front) to remove front element, *i.e.*, 'A' from the Queue list.



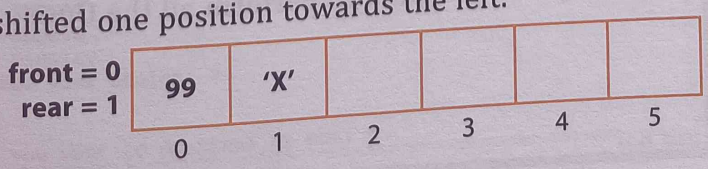
After deleting the front element, *i.e.*, 'A' from the Queue list, the new Queue automatically fills the gap and the list is shifted one position towards the left.



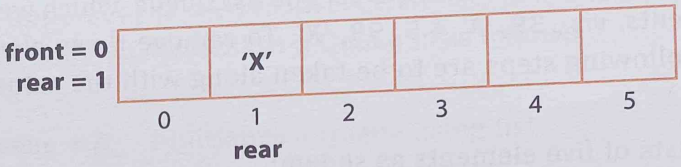
Step 4: For deletion of the next element, again pop(0) or pop(front) is used to remove front element, *i.e.*, 4.5 from the Queue list.



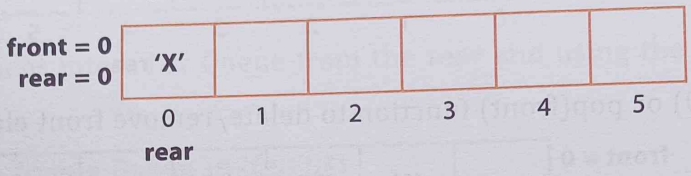
After deleting the front element, *i.e.*, 4.5 from the Queue list, the new Queue automatically fills the gap and the list is shifted one position towards the left.



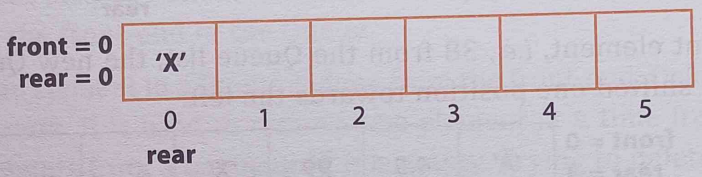
Step 5: Again, pop(0) or pop(front) is used to remove front element, *i.e.*, 99 from the Queue list.



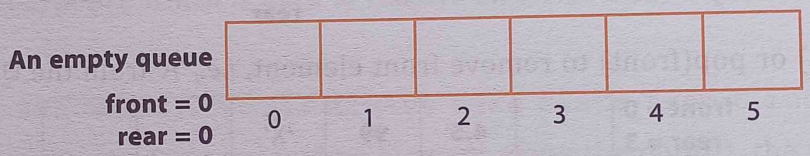
After deleting the front element, *i.e.*, 99 from the Queue list, the new Queue automatically fills the gap and the list is shifted one position towards the left.



Step 6: Now, for deleting the last element from the Queue list, use pop(0) or pop(front) to remove front element, *i.e.*, 'X' from the Queue list.



After deleting the front element, *i.e.*, 'X' from the Queue list, the Queue becomes an empty Queue.



So, we have performed 5 deletion operations (also called as DEQUEUE) which results in an empty Queue.

6.5.4 Traversal of Queue Elements

Like Stack, the elements of a Queue can also be accessed and displayed and the process is termed as Queue Traversal. For traversing each element of the Queue, we will first calculate the length of the Queue using the built-in function `len()` and shall store it in the variable 'length'. Then, using **for** loop starting from 0 index to 'length', the elements shall be displayed from the Queue.

Algorithm for traversing Queue Elements:

1. START
2. length = len(Queue)
3. for i in range(0,length):
   print(Queue[i])
4. End

We will now write a complete Python program to perform all the Queue operations.

Practical Implementation–4

Write a program in Python to add, delete and display elements from a Queue using list.

prog_qu1.py - C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_qu1.py (3.7.0)

File Edit Format Run Options Window Help

```
#Implementing List as a Queue - using function pop()
a=[]
c='y'
while (c=='y'):
    print ("1. INSERT")
    print ("2. DELETE ")
    print ("3. Display")
    choice=int(input("Enter your choice: "))
    if (choice==1):
        b=int(input("Enter new number: "))
        a.append(b)
    elif (choice==2):
        if (a==[]):
            print("Queue Empty")
        else:
            print ("Deleted element is:",a[0])
            a.pop(0)
    elif (choice==3):
        l=len(a)
        for i in range(0,l):
            print (a[i])
    else:
        print("wrong input")
    c=input("Do you want to continue or not: ")
```

Ln: 26 Col: 4

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
>>>
 RESTART: C:\Users\preeti\AppData\Local\Programs\Python\Py
thon37-32\prog_qu1.py
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter new number: 5
Do you want to continue or not: y
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter new number: 8
Do you want to continue or not: y
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter new number: 66
Do you want to continue or not: y
1. INSERT
2. DELETE
3. Display
Enter your choice: 3
5
8
66
Do you want to continue or not: y
1. INSERT
2. DELETE
3. Display
Enter your choice: 2
Deleted element is: 5
Do you want to continue or not: y
1. INSERT
2. DELETE
3. Display
Enter your choice: 3
8
66
Do you want to continue or not: n
```

Ln: 43 Col: 33

Practical Implementation–5

Consider the details of the books where they entered in a Queue. Book details like book id and book name are added to the Queue. Write a program to perform the basic operations of a Queue using list, where the basic operations include inserting, deleting, and showing the Queue elements.

prog_qu2.py - C:\Users\preeti\AppData\Local\Programs\Python\Python37-32\prog_qu2.py (...

File Edit Format Run Options Window Help

```
#Program to add, remove and display the book details using list
#implementation as a Queue
Library=[]
c='y'
while (c=='y'):
    print ("1. INSERT")
    print ("2. DELETE ")
    print ("3. Display")
    choice=int(input("Enter your choice: "))
    if (choice==1):
        book_id=input("Enter book_id : ")
        bname = input("Enter the book name :")
        lib = (book_id,bname)    #tuple created for a new book
        Library.append(lib)    #new book added to the list/queue
    elif (choice==2):
        if (Library==[]):
            print("Queue Empty")
        else:
            print("Deleted element is:",Library.pop(0))
    elif (choice==3):
        l=len(Library)
        for i in range(0,l):
            print (Library[i])
    else:
        print("wrong input")
    c=input("Do you want to continue or not : ")
```

Ln: 30 Col: 4

Python 3.7.0 Shell

File Edit Shell Debug Options Window Help

```
>>>
 RESTART: C:\Users\preeti\AppData\Local\Programs\Pyth
on\Python37-32\prog_qu2.py
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter book_id : 2
Enter the book name :Python
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter book_id : 44
Enter the book name :Java
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter book_id : 5
Enter the book name :C++
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
Enter your choice: 1
Enter book_id : 6
Enter the book name :Networking
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
```

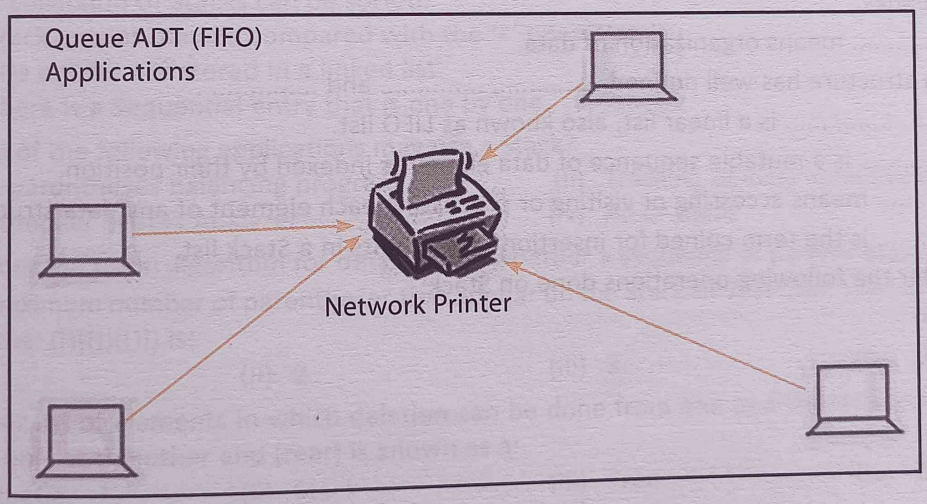
```
Enter your choice: 3
('2', 'Python')
('44', 'Java')
('5', 'C++')
('6', 'Networking')
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
Enter your choice: 2
Deleted element is: ('2', 'Python')
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
Enter your choice: 2
Deleted element is: ('44', 'Java')
Do you want to continue or not : y
1. INSERT
2. DELETE
3. Display
Enter your choice: 3
('5', 'C++')
('6', 'Networking')
Do you want to continue or not :
```

Ln: 60 Col: 33

Applications of Queues

Like Stacks, Queues are also an important data structure in Python. The varied applications where Queues are used are as follows:

1. **Printer Queue:** Queues are very useful in multi-user environment, such as in the case of a Network Printer where there are multiple requests for print operation generated by several users on the network. If the printer is busy, then successive requests for printing are spooled to the hard disk where they wait in a Queue until the printer becomes available.



2. **Single-User Operations:** Queues are also used in single processor systems where only one user can be allowed to work on the system. Rest of the users requesting to work on the system are placed in a Queue.
3. **Handling of Interrupts in Real-time System:** In a real-time system, interrupts may occur (usually through a mouse click or wireless connection) which are necessary to be handled immediately before executing the current task. If these interrupts are to be handled in the same order as they arrive, then a FIFO Queue is the most appropriate data structure to handle this.

4. Suppose there is a web-server hosting a website to declare result(s). This server can handle a maximum of 50 concurrent requests to view result(s). So, to serve thousands of user requests, a Queue would be the most appropriate data structure to use.
5. **Simulating Wait:** Call centre phone systems use a Queue to hold people in line until a customer representative gets free.
6. **Playlist Ordering:** Buffers on MP3 players and portable CD players, iPod playlists, etc., add songs to the end and play from the front of the list.

MEMORY BYTES

- A list is a collection of elements which are stored in a sequence.
- A Stack is a linear structure implemented in LIFO (last in, first out) manner where insertions and deletions take place only at one end, *i.e.*, the Stack's top.
- An insertion in a Stack is called pushing and a deletion from a Stack is called popping.
- A Queue is a linear structure implemented in FIFO (first in, first out) manner where insertions can occur only at the rear-end and deletions can occur only at the front-end.
- Traversal of a list or Stack or a Queue means visiting each element of the list.
- A Queue is a container of objects that are inserted and deleted according to the FIFO principle.
- Inserting an element in a Queue is called Enqueue.
- Deleting an element from the Queue is called Dequeue.
- When a Queue is implemented with the help of a list, it is termed as a Linear Queue.

OBJECTIVE TYPE QUESTIONS

1. Fill in the blanks.

(a) ..... means organization of data.

(b) A data structure has well defined ....., ... and ..... .

(c) A ..... is a linear list, also known as LIFO list.

(d) A ..... is a mutable sequence of data elements indexed by their position.

(e) ..... means accessing or visiting or processing each element of any data structure.

(f) ..... is the term coined for insertion of elements in a Stack list.

(g) Consider the following operations done on Stack:

```
push(5)
push(8)
pop
push(2)
push(5)
pop
pop
pop
push(1)
pop
```

The output of the above snippet is ..... .

(h) In the Stack, if a user tries to remove an element from the empty Stack, it is called ..... .

(i) If the elements "A", "B", "C" and "D" are placed in a queue and are deleted one at a time, the order in which they will be removed will be ....., ..., ... and ..... .

(j) The process of inserting an element in a queue is called ..... while deleting an element from a queue is called ..... .

2. State whether the following statements are True or False.
   (a) Single-ended linear structure is a type of queue data structure.
   (b) Reversing a word/line is an application of Stack.
   (c) The insertion and deletion from a Stack takes place only from the 'TOP'.
   (d) A queue behaves on the basis of LIFO principle.
   (e) When a queue is implemented with the help of a list, it is termed as a linear queue.
   (f) CPU uses queues to implement round-robin scheduling among waiting processes.
   (g) Deletion of an element from the queue is termed as pop operation.
   (h) An element from a queue is added from the front end.
   (i) Handling the processes of a network printer is the most important application of queues.
   (j) The append() function inserts an element at the rear end in a queue.
   (k) Stack is a linear data structure.
   (l) PUSH operation may result in underflow condition.

3. Multiple Choice Questions (MCQs)
   (a) The process of inserting an element in Stack is called:
   (i) Create (ii) Push (iii) Evaluation (iv) Pop
   (b) The process of removing an element from Stack is called:
   (i) Create (ii) Push (iii) Evaluation (iv) Pop
   (c) In a Stack, if a user tries to remove an element from an empty Stack, the situation is called:
   (i) Underflow (ii) Empty collection
   (iii) Overflow (iv) Garbage collection
   (d) Pushing an element into a Stack already having five elements and a Stack of size 5, then the Stack becomes:
   (i) User flow (ii) Crash (iii) Underflow (iv) Overflow
   (e) Entries in a Stack are "ordered". What is the meaning of this statement?
   (i) A collection of Stacks can be sorted.
   (ii) Stack entries may be compared with the '<' operation.
   (iii) The entries are stored in a linked list.
   (iv) There is a Sequential entry that is one by one.
   (f) Which of the following applications may use a Stack?
   (i) A parentheses balancing program (ii) Tracking of local variables at run time
   (iii) Compiler Syntax Analyzer (iv) All of these
   (g) Consider the usual algorithm for determining whether a sequence of parentheses is balanced. The maximum number of parentheses that appear on the Stack AT ANY ONE TIME when the algorithm analyzes: (()(()(()))) is:
   (i) 1 (ii) 2 (iii) 3 (iv) 4 or more
   (h) A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as a:
   (i) Queue (ii) Stack (iii) Tree (iv) Linked list
   (i) A queue is a:
   (i) FIFO (First In First Out) list (ii) LIFO (Last In First Out) list
   (iii) Ordered array (iv) Linear tree

SOLVED QUESTIONS

1. What is data structure?

Ans. A data structure is a logical way of organizing data in memory that considers not only the items stored but also the relationship between the items so as to give efficient and optimal performance. In other words, it is a way of organizing data such that it can be used efficiently.

2. What is a Stack? What basic operations can be performed on them?

Ans. Stack is a basic data structure where insertion and deletion of data takes place at one end called the top of the Stack, *i.e.*, it follows the last in, first out (LIFO) principle.

In Python, a Stack is generally implemented with lists.

Following basic operations can be performed on Stacks.

(i) PUSH, *i.e.*, insertion of element in the Stack
(ii) POP, *i.e.*, deletion of an element from the Stack
(iii) Traversal, *i.e.*, displaying all the elements of the Stack.

3. Enlist some applications of Stacks.

Ans. Because of LIFO property of Stacks, they are used in various applications like:

- Reversal of a sequence.
- The compilers use Stacks to store the previous state of a program when a function is called or during recursion.
- Another important Stack application is backtracking. (Backtracking is a form of recursion. But it involves choosing only option out of any possibilities.) Backtracking is used in a large number of puzzles like Sudoku, etc., and in optimization problems such as the knapsack problem.
- **"Undo"** mechanism in text editors: This operation is accomplished by keeping all text changes in a Stack.

4. What are Queues? What all operations can be performed on Queues?

Ans. Queues are the data structures where data is entered into the Queue at one end—the rear-end—and deleted from the other end—the front, *i.e.*, these follow first in, first out (FIFO) principle. Following basic operations can be performed on Queues:

(i) Insertion of an element in the Queue
(ii) Deletion of an element from the Queue
(iii) Display of/viewing all the elements in the Queue

5. Enlist some applications of Queues.

Ans. Applications of Queues include the situations where FIFO property is exploited. Some common applications of Queues include:

(i) Sharing of one resource among multiple users or seekers such as shared printer among multiple computers; call centre executive's response to waiting callers, etc.
(ii) Airport authorities make use of Queues in situations of sharing of a single runway for both landing and take-off of flights.
(iii) CPU uses Queues to implement round-robin scheduling among waiting processes.
(iv) Queues are used in many computer algorithms also.

6. Consider the following Stack of characters implemented as an array of 4 elements:

```
STACK = ["A","J", "P", "N"]
```

Describe the Stack as the following operations take place:

(a) `STACK.pop()`
(b) `STACK.append(K)`
(c) `STACK.append(S)`
(d) `STACK.pop()`
(e) `STACK.append(G)`

Ans.
(a) STACK : [A, J, P] as N is popped
(b) STACK: [A, J, P, K] as K is added
(c) STACK: [A, J, P, K, S] as S is added
(d) STACK: [A, J, P, K] as S is Popped
(e) STACK: [A, J, P, K, G] as G is added

7. Find the output of the following code:

(a)
```
result=0
numberList=[10,20,30]
numberList.append(40)
result=result+numberList.pop()
result=result+numberList.pop()
print(result)
print(numberList)
```

Ans.
```
70
[10, 20]
```

(b)
```
answer=[]; output=''
answer.append('T')
answer.append('A')
answer.append('M')
ch=answer.pop()
output=output+ch
ch=answer.pop()
output=output+ch
ch=answer.pop()
output=output+ch
print("Result=",output)
```

Ans. Result= MAT

8. Write a program to implement a Stack for these book-details (book no, book name). That is, now each item node of the Stack contains two types of information — a book no. and its name. Just implement PUSH and display operations.

Ans.
```
"""
    Stack: Implemented as a list
    Top : integer having position of topmost element in Stack
"""
def isEmpty(stk):
    if stk == [ ]:
        return True
    else :
        return False
def Push(stk, item):
    stk.append(item)
    top = len(stk) - 1
def Display(stk):
    if isEmpty(stk):
        print("Stack empty")
    else:
        top = len(stk) - 1
        print(stk[top], "top")
        for a in range (top-1, -1, -1 ):
            print(stk[a])
# ----- main -----
Stack = [ ]              # Initially stack is empty
top = None
while True:
    print("STACK OPERATIONS")
    print("1. Push")
    print("2. Display stack")
    print("3. Exit")
    ch = int(input("Enter your choice (1-3): "))
    if ch == 1:
        bno = int(input("Enter Book no. to be inserted: "))
        bname= input("Enter Book name to be inserted :")
        Item = [bno, bname] #creating a list from the input items.
        Push(Stack, item)
        input()
```

```
    elif ch == 2:
        Display(Stack)
        input()
    elif ch == 3:
        break
    else:
        print("Invalid choice!")
        input()
```

9. Write a function in Python PUSH(Arr), where Arr is a list of numbers. From this list, push all the numbers divisible by 5 into a Stack implemented by using a list. Display the Stack if it has at least one element, otherwise display appropriate error message. [CBSE Sample Question Paper 2020]

Ans.
```
def PUSH(Arr):
    s=[]
    for x in range(0,len(Arr)):
        if Arr[x]%5==0:
            s.append(Arr[x])
            if len(s)==0:
                print("Empty Stack")
            else:
                print(s)
```

10. Write a function in Python POP(Arr), where Arr is a Stack implemented by a list of numbers. The function returns the value deleted from the Stack. [CBSE Sample Question Paper 2020]

Ans.
```
def POP(Arr) :
    # If stack is empty
    if len(Arr)==0:
        print("Underflow")
    else:
        L = len(Arr)
        val=Arr.pop(L-1)
        print(val)
```

11. Write a program to perform insert and delete operations on a Queue containing Members' details as given in the following definition of item node:

Member No.	integer
MemberName	String
Age	integer

Ans.
```
"""
Queue: implemented as a list
Front: integer having position of first (frontmost) element in queue
Rear: integer having position of last element in queue
"""
def isEmpty(Qu):
    if Qu == []:
        return True
    else:
        return False
def Enqueue (Qu, item):
    Qu.append(item)
    if len(Qu) == 1:
        front = rear = 0
    else:
        rear = len(Qu) - 1
```

```
def Dequeue(Qu):
    if isEmpty(Qu):
        return "Underflow"
    else:
        item = Qu.pop(0)
    if len(Qu) == 0:                    # If it was single-element queue
        front = rear = None
    return item
def Display(Qu):
    if isEmpty(Qu):
        print("Queue Empty!")
    elif len(Qu) == 1:
        print(Qu[0], "<front, rear>")
    else:
        front = 0
        rear = len(Qu) - 1
        print(Qu[front], "<-front")
        for a in range(1, rear):
            print(Qu[a])
        print(Qu[rear], "<-rear")
# ---- main ----
Queue = []                              # Initially queue is empty
while True:
    print("QUEUE OPERATIONS")
    print("1. Enqueue")
    print("2. Dequeue")
    print("3. Display queue")
    print("4. Exit")
    ch = int(input("Enter your choice (1-4) : "))
    if ch == 1:
        print("For the new member, enter details below:")
        memberNo = int(input("Enter member no :"))
        memberName = input("Enter member name:")
        age = int(input("Enter member's age:"))
        item = [memberNo, memberName, age]
        Enqueue(Queue, item)
        input("Press Enter to continue...")
    elif ch == 2 :
        item = Dequeue(Queue)
        if item == "Underflow":
            print("Underflow! Queue is empty!")
        else :
            print("Dequeued item is", item)
        input("Press Enter to continue...")
    elif ch == 3 :
        Display(Queue)
        input("Press Enter to continue...")
    elif ch == 4:
        break
    else:
        print("Invalid choice!")
        input("Press Enter to continue...")
```

12. Write functions in Python, do_Push(Num) and do_Pop(Num), to add a new number and delete a number from a List of Numbers, considering them to act as push and pop operations of the Stack data structure.

Ans.
```
def do_Push(Num):
    a=int(input("Enter number : "))
    Num.append(a)
def do_Pop(Num):
    if (Num==[]):
        print("Stack empty")
    else:
        print ("Deleted element",Num.pop())
```

13. Write a function in Python, INSERT_QUEUE(List1), that accepts data from the user and performs insertion and DELETE_QUEUE(List1) to perform deletion operation in a Queue. List1 is the list used for implementing queue and data is the value to be inserted.

Ans.
```
def INSERT_QUEUE(List1):
    data=int(input("Enter data to be inserted: "))
    List1.append(data)
def DELETE_QUEUE(List1):
    if (List1==[]):
        print("Queue empty")
    else:
        print("Deleted element is: ", List1 [0])
        del(List1 [0])
```

14. Describe the similarities and differences between Queues and Stacks.

Ans. Similarities:

1. Both Queues and Stacks are special cases of linear lists.
2. Both can be implemented as arrays or linked lists.

Differences:

1. A Stack is a LIFO list, a Queue is a FIFO list.
2. There are no variations of Stack; a Queue, however, may be circular or deque (double ended queue).

15. Propose a data structure which supports the Stack PUSH and POP operations and a third operation FindMin, which returns the smallest element in the data structure.

Ans. Use two Stacks say S1 and S2 — S1 into which the elements are pushed and S2 into which only the current minimum is pushed.

1. When first element is pushed in Stack S1, push it to Stack S2 too, considering that this is the minimum element.
2. When one needs to insert an element, say Ele:
   (a) We first push Ele on to S1, and
   (b) Then access the top element, say T of S2, which is the minimum before the insertion of Ele.
   (c) Compare Ele with T; if only Ele is less than T, we push Ele on to S2. Hence, the current minimum will always be on top of S2.
3. When one needs to pop an element,
   (a) Pop the top element of S1, and
   (b) If this element is also equal to the one on top of S2, then pop from S2 as well.

16. Give the necessary declaration of a list implemented Stack containing numeric type data. Also write user-defined function to pop a number from the Stack.

Ans.
```
#Declaration for list implemented stack
Stack = list()
#Function body to pop a number from the stack
def POP(Stack, top):
    if not Stack:
        print("stack is empty")
    else:
        Num = Stack.pop() # Removing top elements from the stack.
        top = top - 1
        print("Value deleted from stack is", Num)
    return top
```

17. Write PUSH(Names) and POP(Names) methods in Python to add Names and remove Names considering them to act as PUSH and POP operations in Stack. [AI 2015]

Ans. The methods are:
```
#Adding element into a stack.
def PUSH(Names):
    Stack.append(Names)      #Name added into Stack
    print(Names, "added.")
#Removing stack elements
def POP():
    if not len(Stack):       #Checks if there is no element in stack
        print("Stack is empty")
    else:
        Names = Stack.pop()  #Removing top elements from stack
        print("Value deleted from stack is", Names)
```

18. What is the situation called when an insertion is attempted in a full Stack? [HOTS]

Ans. Overflow

19. What is the situation called when read/deletion is attempted in an empty Queue? [HOTS]

Ans. Underflow

20. What is the difference between an array and a Stack housed in an array? Why is Stack called a LIFO data structure? Explain how push and pop operations are implemented on a Stack.

Ans. An array is a group of homogeneous elements stored in contiguous memory locations. The elements in an array can be processed from anywhere in the array.

A Stack implementation as an array also has elements stored in contiguous memory locations. But a Stack is always processed in a LIFO manner, *i.e.*, Last In First Out manner wherein the elements can be added or removed from a top, *i.e.*, end of the Stack. That is why a Stack is also called a LIFO data structure.

An addition to the Stack is known as PUSH. The new element is added at the top and the top (variable or pointer) is made to refer to the new element.

The removal of an element from a Stack is known as POP. The element (being pointed to by top) is removed and the top is made to point to the element down the Stack.

21. An algorithm requires two Stacks of sizes M and N that are to be maintained in the memory. Illustrate with an example how you will adjust two Stacks in one-dimensional array with M + N memory locations so that the overflow condition is minimized.

Ans. Let us say that Stack A is with size M and Stack B is with size N.

If the Stack A is stored in locations O to M–1 and the Stack B is stored in locations M to M+N-1, the separate areas for the two are ensured. In the beginning, the TOPs of both Stacks are at opposite ends. When TOP A reaches at M-1, any further insertion will lead to overflow in Stack A and when TOP B reaches at M, any further insertion in Stack B will lead to overflow.

22. Write a menu-based program to add, delete and display the record of hostel using list as stack data structure in Python. Record of hostel contains the fields: Hostel number, Total Students and Total Rooms.

Ans.
```
def push(host):
    hn=int(input("Enter hostel number"))
    ts=int(input("Enter Total students"))
    tr=int(input("Enter total rooms"))
    temp=[hn,ts,tr]
    host.append(temp)

def pop(host):
    if(host==[]):
        print("No Record")
    else:
        print("Deleted Record is :",host.pop())

def display(host):
    l=len(host)
    print("Hostel Number\tTotal Students\tTotal Rooms")
    for i in range(l-1,-1,-1):
        print(host[i][0],"\t\t",host[i][1],"\t\t",host[i][2])

host=[]
ch='y'
ch=input("Do you want to enter more(Y/N)")

while(ch=='y' or ch=='Y'):
    print("1. Add Record\n")
    print("2. Delete Record\n")
    print("3. Display Record\n")
    print("4. Exit")
    op=int(input("Enter the Choice"))
    if(op==1):
        push(host)
    elif(op==2):
        pop(host)
    elif(op==3):
        display(host)
    elif(op==4):
        break
```

23. Write a function in Python, **INSERTQ(Arr,data)** and **DELETEQ(Arr)**, for performing insertion and deletion operations in a Queue. **Arr** is the list used for implementing queue and **data** is the value to be inserted.

Ans.
```
def INSERTQ(Arr, data):
    Arr.append(data)
def DELETEQ(Arr):
    if (Arr==[]):
        print("Queue empty")
    else:
        print("Deleted element is: ",Arr[0])
        del(Arr[0])
```

24. Write a function in Python, **MakePush(Package)** and **MakePop(Package)**, to add a new Package and delete a Package from a List of Package Description, considering them to act as push and pop operations of the Stack data structure.

Ans.
```
def MakePush(Package):
    a=int(input("enter package title:"))
    Package.append(a)

def MakePop(Package):
    if (Package==[]):
        print("Stack empty")
    else:
        print("Deleted element:",Package.pop())
```

UNSOLVED QUESTIONS

1. What is Stack? Why is it called LIFO data structure?
2. List two ways to implement Stack.
3. Write applications of Stack.
4. Can a Stack be used to convert a decimal number into a binary number?
5. Write an algorithm to push an element into the Stack.
6. Write an algorithm to pop an element from the Stack.
7. Write an interactive menu-driven program implementing Stack using list. The list is storing numeric data.
8. Write an interactive menu-driven program to implement Stack using list. The list contains the names of students.
9. How does FIFO describe queue?
10. Write a menu-driven Python program using queue to implement movement of shuttlecock in its box.
11. Give the necessary declaration of a list implemented Stack containing float type numbers. Also, write a user-defined function to pop a number from this Stack.
12. A linear Stack called Directory contains the following information as contacts:
    - Pin code of city
    - Name of city

    Write add(Directory) and delete(Directory) methods in Python to add and remove contacts using append() and pop() operations in Stack.
13. Write add(Books) and delete(Books) methods in Python to add Books and Remove Books considering them to act as append() and pop() operations in Stack. [Delhi 2015]
14. Write AddClient(Client) and DeleteClient(Client) methods in Python to add a new client and delete a client from a list client name, considering them to act as insert and delete operations of the Queue data structure. [CBSE 2018]
15. Write Addscore(Game) and Delscore(Game) methods in Python to add new Score in the list of score in a game and remove a score from a list of score of a game considering these methods to act as PUSH and POP operation of data structure Stack. [CBSE 2017]
16. Write a Python program to sort a Stack in ascending order without using an additional Stack.

17. A Stack STK and Queue QUE is being maintained. Their maximum size is the same:
    (i) check whether they have the same size, *i.e.*, have the same number of elements.
    (ii) check for equality of Stack and Queue, *i.e.*,
        (a) Retrieve an element from the Stack and one element from the Queue.
        (b) Compare the two.
        (c) Stop and report if elements are different.
18. Write an algorithm to insert element into (i) Stack as a list (ii) Queue as a list
19. Write a program to create a Stack for storing only odd numbers out of all the numbers entered by the user. Display the content of the Stack along with the largest odd number in the Stack.

    (**Hint:** Keep popping out the elements from Stack and maintain the largest element retrieved so far in a variable. Repeat till Stack is empty.)
20. Write a program that, depending upon the user's choice, either adds or removes an element from a Stack.
21. Write a program that, depending upon the user's choice, either pushes or pops an element in a Stack. The elements are shifted towards the right so that the top always remains at 0^{th} (zeroth) index.
22. Write a program to insert or delete an element from a Queue depending upon the user's choice. The elements are not shifted after insertion or deletion.
23. Write a program to perform insert and delete operations on a Queue containing Students' details as given in the following definition of item node:

AdmNo	integer
Name	String
Age	integer
Class	String

```
def isEmpty(Qu):
    if Qu == []:
        return True
    else:
        return False

def Enqueue(Qu, item):
# Write the code to insert student details using Queue.

def Dequeue(Qu):
# Write the code to delete a student using Queue.
```

24. Write AddCustomer(Customer) and DeleteCustomer(Customer) methods in Python to add a new Customer and delete a Customer from a List of CustomerNames, considering them to act as push and pop operations of the Stack data structure.

CASE-BASED/SOURCE-BASED INTEGRATED QUESTIONS

1. Infopedia is an online encyclopaedia which stores detailed information about various countries. In order to provide efficient processing for retrieval and display of names of countries, it has to be digitized, which requires handling additions and deletions through data structures in Python, primarily Stacks and queues.

   Write a program constituting methods in Python to add, display and remove a name from a given Stack of names of countries.

Ans.

prog_encyclopedia_stack.py - C:/Python382/prog_encyclopedia_stack.py (3.8.2)

File Edit Format Run Options Window Help

```
#Python program to perform basic operations using stack
#for adding, removing and displaying countries names in Encyclopedia

def PUSH(country_name):
    cname =input("Enter the name of the country to be added :")
    country_name.append(cname)

def POP(country_name):
    if country_name != []:
        print(country_name.pop())
    else:
        print("Empty stack of Encyclopedia")

def SHOW(country_name):
    print(country_name)

STACK =[]
Choice =''
while Choice != 'Q':
    print("P:PUSH O:POP S:SHOW Q:QUIT")
    Choice =input("Enter your choice")
    if Choice == "P":
        PUSH(STACK)
    elif Choice == "O":
        POP(STACK)
    elif Choice == "S":
        SHOW(STACK)
    elif Choice == "Q":
        break
```

Ln: 34 Col: 4

```
>>>
========== RESTART: C:/Python382/prog_encyclopedia_stack.py ==========
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceP
Enter the name of the country to be added :INDIA
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceP
Enter the name of the country to be added :USA
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceS
['INDIA', 'USA']
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceP
Enter the name of the country to be added :SPAIN
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceS
['INDIA', 'USA', 'SPAIN']
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceO
SPAIN
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceS
['INDIA', 'USA']
P:PUSH O:POP S:SHOW Q:QUIT
Enter your choiceQ
```

2. Flyair Airlines has grown big and touches more than 140 destinations worldwide, offering excellent service to its passengers. To provide latest information to its customers at the click of a button, the company offers computerized processing of passenger details.

Write a Python program comprising methods/functions to add or delete a passenger's name from the list of passengers, considering them as insert and delete operations of the Queue data structure.

Ans.

prog_airline_queue.py - C:/Python382/prog_airline_queue.py (3.8.2)

File Edit Format Run Options Window Help

```
#Python program to perform basic operations using queue
#for adding, removing and displaying passengers of FLYAIR airlines

def INSERT(Q):
    pname =input("Enter the name of the passenger to be added :")
    Q.append(pname)

def DEL(Q):
    if Q != []:
        pname= Q.pop(0)
        print("Passenger data deleted")
    else:
        print("Empty queue. No Passenger details")

def SHOW(Q):
    print(Q)

Q =[]

while True:
    Option =input("I:INSERT D:DEL S:SHOW E:EXIT")
    if Option == "I":
        INSERT(Q)
    elif Option == "D":
        DEL(Q)
    elif Option == "S":
        SHOW(Q)
    elif Option == "E":
        print("Happy and Safe Journey")
        break
    else:
        print("Wrong Option")
```

Ln: 38 Col: 8

```
>>>
================== RESTART: C:/Python382/prog_airline_queue.py ========
I:INSERT D:DEL S:SHOW E:EXITI
Enter the name of the passenger to be added :Ranveer Malik
I:INSERT D:DEL S:SHOW E:EXITI
Enter the name of the passenger to be added :Deep Malik
I:INSERT D:DEL S:SHOW E:EXITI
Enter the name of the passenger to be added :Arun Trivedi
I:INSERT D:DEL S:SHOW E:EXITI
Enter the name of the passenger to be added :Shauryan Sharma
I:INSERT D:DEL S:SHOW E:EXITS
['Ranveer Malik', 'Deep Malik', 'Arun Trivedi', 'Shauryan Sharma']
I:INSERT D:DEL S:SHOW E:EXITD
Passenger data deleted
I:INSERT D:DEL S:SHOW E:EXITS
['Deep Malik', 'Arun Trivedi', 'Shauryan Sharma']
I:INSERT D:DEL S:SHOW E:EXITE
Happy and Safe Journey
```