TAMILNADU ADVANCED TECHNICAL TRAINING INSTITUTE

PYTHON

Stack

A stack is a linear data structure in which the insertion of a new element and removal of an existing element takes place at the same end represented as the top of the stack.

Basic operations

- Push: Add an element to the top of a stack
- Pop: Remove an element from the top of a stack
- IsEmpty: Check if the stack is empty
- IsFull: Check if the stack is full
- Peek: Get the value of the top element without removing it
- Top:Returns the top element of the stack.

Types

- Fixed Size Stack
- Dynamic Size Stack
- Infix to Postfix Stack
- Expression Evaluation Stack
- Recursion Stack
- Memory Management Stack
- Balanced Parenthesis Stack
- Undo-Redo Stack

Implementation

We can implement a stack in Python in the following ways.

- Arrays
- List
- collections.dequeu
- queue.LifoQueue

Programs

```
stack = []

# append() function to push
stack.append('a')
stack.append('b')
stack.append('c')

print('Initial stack')
print(stack)
```

Programs

```
stack = []
stack.append('a')
stack.append('b')
stack.append('c')
print(stack)

# pop() function to pop
print(stack.pop())
print(stack.pop())
print('\nStack after elements are popped:')
print(stack)
```

Programs

```
from collections import deque
stack = deque()
stack.append('a')
stack.append('b')
stack.append('c')
  print(stack)

print(stack.pop())
print(stack.pop())
print(stack.pop())
print(stack)
```

Programs

```
from queue import LifoQueue
  stack = LifoQueue(maxsize=3)
  print(stack.qsize())
stack.put('a')
stack.put('b')
stack.put('c')
print("Full: ", stack.full())
print("Size: ", stack.qsize())
print(stack.get())
print(stack.get())
print("\nEmpty: ", stack.empty())
```