Stack and Queue in Python using queue Module

A simple python List can act as queue and stack as well. Queue mechanism is used widely and for many purposes in daily life. A queue follows FIFO rule(First In First Out) and is used in programming for sorting and for many more things. Python provides Class queue as a module which has to be generally created in languages such as C/C++ and Java.

1. Creating a FIFO Queue

```
// Initialize queue
Syntax: queue.Queue(maxsize)

// Insert Element
Syntax: Queue.put(data)

// Get And remove the element
Syntax: Queue.get()
```

Initializes a variable to a maximum size of maxsize. A maxsize of zero '0' means a infinite queue. This Queue follows FIFO rule. This module also has a LIFO Queue, which is basically a Stack. Data is inserted into Queue using put() and the end. get() takes data out from the front of the Queue. Note that Both put() and get() take 2 more parameters, optional flags, block and timeout.

```
import queue

# From class queue, Queue is
# created as an object Now L
# is Queue of a maximum
# capacity of 20
L = queue.Queue(maxsize=20)

# Data is inserted into Queue
# using put() Data is inserted
# at the end
L.put(5)
L.put(9)
L.put(1)
L.put(7)

# get() takes data out from
# the Queue from the head
# of the Queue
print(L.get())
print(L.get())
print(L.get())
print(L.get())
```

Output:

```
5
9
1
7
```

2. UnderFlow and OverFlow

When we try to add data into a Queue above is maxsize, it is called OverFlow(Queue Full) and when we try removing an element from an

empty, it's called Underflow. put() and get() do not give error upon Underflow and Overflow, but goes into an infinite loop.

```
import queue

L = queue.Queue(maxsize=6)

# qsize() give the maxsize
# of the Queue
print(L.qsize())

L.put(5)
L.put(9)
L.put(1)
L.put(7)

# Return Boolean for Full
# Queue
print("Full: ", L.full())

L.put(9)
L.put(10)
print("Full: ", L.full())

print(L.get())
print(L.get())
print(L.get())

# Return Boolean for Empty
# Queue
print("Empty: ", L.empty())

print(L.get())
print(L.get())
print(L.get())

print("Empty: ", L.empty())
print("Full: ", L.full())

# This would result into Infinite
# Loop as the Queue is empty.
# print(L.get())
```

Output:

```
0
Full:  False
Full:  True
5
9
1
Empty:  False
7
9
10
Empty:  True
Full:  False
```

3. Stack

This module queue also provides LIFO Queue which technically works as a Stack.

```
import queue

L = queue.LifoQueue(maxsize=6)

# qsize() give the maxsize of
# the Queue
print(L.qsize())

# Data Inserted as 5->9->1->7,
# same as Queue
L.put(5)
L.put(9)
L.put(1)
```

```
L.put(7)
L.put(9)
L.put(10)
print("Full: ", L.full())
print("Size: ", L.qsize())

# Data will be accessed in the
# reverse order Reverse of that
# of Queue
print(L.get())
print(L.get())
print(L.get())
print(L.get())
print(L.get())
print("Empty: ", L.empty())
```

Output:

```
0
Full:  True
Size: 6
10
9
7
1
9
Empty:  False
```

Reference:

https://docs.python.org/3/library/asyncio-queue.html

Recommended Posts:

Check if a queue can be sorted into another queue using a stack

Implement a stack using single queue

Check if moves in a stack or queue are possible or not

Implement Stack and Queue using Deque

How to implement stack using priority queue or heap?

Difference between Stack and Queue Data Structures

Level order traversal in spiral form | Using one stack and one queue

Priority Queue in Python

Heap queue (or heapq) in Python

Python | Queue using Doubly Linked List

Queue using Stacks

Reversing a Queue

Applications of Priority Queue

Reversing the first K elements of a Queue

Reversing a queue using recursion

piyushdoorwar
Check out this Author's contributed articles.