Operator Functions Set 1 & Set 2
Data Types
Introduction
Arrays Set 1, Set 2
String Methods Set 1, Set 2, Set 3
String Template Class & String Formatting using %
List Methods Set 1, Set 2, Set 3
Tuples & Sets
Dictionary Methods Set 1, Set 2
ChainMap
Explore More...
Control Flow
Loops and Control Statements
Counters & Accessing Counters
Iterators & Iterator Functions Set 1, Set 2
Generators
Explore More...
Functions
Function Decorators
Returning Multiple Values
Yield instead of Return
Python Closures & Coroutine
Explore More...
Modules
Introduction

Numeric Functions & Logarithmic and Power functions
Calender Functions Set 1, Set 2
Complex Numbers Introduction & Important functions
Explore More...
Object Oriented Concepts
Class, Object and Members
Data Hiding and Object Printing
Inheritance, Subclass and super
Class method vs static method & Class or Static Variables
Explore More...
Exception Handling
Exception Handling
User-Defined Exceptions
Built-in Exceptions
Libraries and Functions
Timeit
Numpy Set 1, Set 2
Get and Post
import module & reload module
Collection Modules Deque, Namedtuple & Heap
Explore More...
Machine Learning with Python
Classifying data using Support Vector Machines(SVMs) in Python

K means Clustering
How to get synonyms/antonyms from NLTK WordNet in Python?
Explore More...
Misc
Sql using Python & MongoDB and Python
Json formatting & Python Virtual environment
Metaprogramming with Metaclasses in Python
Python Input Methods for Competitive Programming
Explore More...
Applications and Projects
Creating a proxy webserver Set 1, Set 2
Send Messsage to FB friend
Twitter Sentiment Analysis & Whatsapp using Python
Desktop Notifier & Junk File Organizer
Explore More...

Deque in Python

Deque can be implemented in python using the module "**collections**". Deque is preferred over list in the cases where we need quicker append and pop operations from both the ends of container, as deque provides an **O(1)** time complexity for append and pop operations as compared to list which provides O(n) time complexity.

Operations on deque :

1. append() :- This function is used to **insert** the value in its argument to the **right end** of deque.

2. appendleft() :- This function is used to **insert** the value in its argument to the **left end** of deque.

3. pop() :- This function is used to **delete** an argument from the **right end** of deque.

4. popleft() :- This function is used to **delete** an argument from the **left end** of deque.

```
# Python code to demonstrate working of
# append(), appendleft(), pop(), and popleft()

# importing "collections" for deque operations
import collections

# initializing deque
de = collections.deque([1,2,3])

# using append() to insert element at right end
# inserts 4 at the end of deque
de.append(4)

# printing modified deque
print ("The deque after appending at right is : ")
print (de)

# using appendleft() to insert element at right end
# inserts 6 at the beginning of deque
de.appendleft(6)

# printing modified deque
print ("The deque after appending at left is : ")
print (de)

# using pop() to delete element from right end
# deletes 4 from the right end of deque
de.pop()

# printing modified deque
print ("The deque after deleting from right is : ")
print (de)

# using popleft() to delete element from left end
# deletes 6 from the left end of deque
de.popleft()

# printing modified deque
print ("The deque after deleting from left is : ")
print (de)
```

Run on IDE

Output:

```
The deque after appending at right is :
deque([1, 2, 3, 4])
The deque after appending at left is :
deque([6, 1, 2, 3, 4])
The deque after deleting from right is :
deque([6, 1, 2, 3])
The deque after deleting from left is :
deque([1, 2, 3])
```

5. index(ele, beg, end) :- This function **returns the first index of the value** mentioned in arguments, **starting searching from beg till end** index.

6. insert(i, a) :- This function **inserts the value** mentioned in arguments(a) **at index(i)** specified in arguments.

7. remove() :- This function **removes the first occurrence** of value mentioned in arguments.

8. count() :- This function **counts the number of occurrences** of value mentioned in arguments.

```
# Python code to demonstrate working of
# insert(), index(), remove(), count()

# importing "collections" for deque operations
import collections

# initializing deque
de = collections.deque([1, 2, 3, 3, 4, 2, 4])

# using index() to print the first occurrence of 4
print ("The number 4 first occurs at a position : ")
print (de.index(4,2,5))

# using insert() to insert the value 3 at 5th position
de.insert(4,3)

# printing modified deque
print ("The deque after inserting 3 at 5th position is : ")
print (de)

# using count() to count the occurrences of 3
print ("The count of 3 in deque is : ")
print (de.count(3))

# using remove() to remove the first occurrence of 3
de.remove(3)

# printing modified deque
print ("The deque after deleting first occurrence of 3 is : ")
print (de)
```

Run on IDE

Output:

```
The number 4 first occurs at a position :
4
The deque after inserting 3 at 5th position is :
deque([1, 2, 3, 3, 3, 4, 2, 4])
The count of 3 in deque is :
3
The deque after deleting first occurrence of 3 is :
deque([1, 2, 3, 3, 4, 2, 4])
```

9. extend(iterable) :- This function is used to **add multiple values at the right end** of deque. The argument passed is an iterable.

10. extendleft(iterable) :- This function is used to **add multiple values at the left end** of deque. The argument passed is an iterable. **Order is reversed** as a result of left appends.

11. reverse() :- This function is used to **reverse order** of deque elements.

12. rotate() :- This function **rotates the deque** by the number specified in arguments. **If the number specified is negative, rotation occurs to left. Else rotation is to right.**

```
# Python code to demonstrate working of
# extend(), extendleft(), rotate(), reverse()
```

```
# importing "collections" for deque operations
import collections

# initializing deque
de = collections.deque([1, 2, 3,])

# using extend() to add numbers to right end
# adds 4,5,6 to right end
de.extend([4,5,6])

# printing modified deque
print ("The deque after extending deque at end is : ")
print (de)

# using extendleft() to add numbers to left end
# adds 7,8,9 to right end
de.extendleft([7,8,9])

# printing modified deque
print ("The deque after extending deque at beginning is : ")
print (de)

# using rotate() to rotate the deque
# rotates by 3 to left
de.rotate(-3)

# printing modified deque
print ("The deque after rotating deque is : ")
print (de)

# using reverse() to reverse the deque
de.reverse()

# printing modified deque
print ("The deque after reversing deque is : ")
print (de)
```

Run on IDE

Output :

```
The deque after extending deque at end is :
deque([1, 2, 3, 4, 5, 6])
The deque after extending deque at beginning is :
deque([9, 8, 7, 1, 2, 3, 4, 5, 6])
The deque after rotating deque is :
deque([1, 2, 3, 4, 5, 6, 9, 8, 7])
The deque after reversing deque is :
deque([7, 8, 9, 6, 5, 4, 3, 2, 1])
```

This article is contributed by **Manjeet Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

GATE CS Corner **Company Wise Coding Practice**

Python

Recommended Posts:

Namedtuple in Python
Heap queue (or heapq) in Python
getpass() and getuser() in Python (Password without echo)
enum in Python
Statistical Functions in Python | Set 1 (Averages and Measure of Central Location)

(Login to Rate and Mark)

2.5 Average Difficulty : **2.5/5.0**
Based on **2** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!