deque

- The **deque**, short for double-ended queue, is a Python data structure that efficiently adds and removes elements from both ends. It is a component of the collections module and serves as an alternative to the list for scenarios where frequent insertions and deletions occur at both ends. Deques are notably advantageous when a queue is needed to enable fast appends and pops from both ends or when a stack is required to support the same operations efficiently.

Syntax

```
from collections import deque
d = deque([iterable[, maxlen]])
```

- **Iterable** : An optional parameter representing an iterable object (like a list, tuple, or string) used to initialize the deque. If no iterable is provided, an empty deque is created.
- **Maxlen** : An optional parameter that specifies the maximum length of the deque.

Example

The following example demonstrates the usage of deque:

```
from collections import deque

# Create a deque using a tuple of integers
a = deque((8, 7, 9, 6))
print(a)

# Create a deque using a list of integers
b = deque([45, 845, 65])
print(b)
```

- # Create a deque using a range of integers from 5 to 9
- c = deque(range(5, 10))
- print(c)

- # Create a deque using a string, which will be split into individual characters
- d = deque("wxyz")
- print(d)

- # Create a dictionary with some key-value pairs
- numbers = {"firstname": "John", "age":25}

- # Create a deque containing the keys of the dictionary
- e = deque(numbers.keys())
- print(e)

- # Create a deque containing the values of the dictionary
- f = deque(numbers.values())
- print(f)

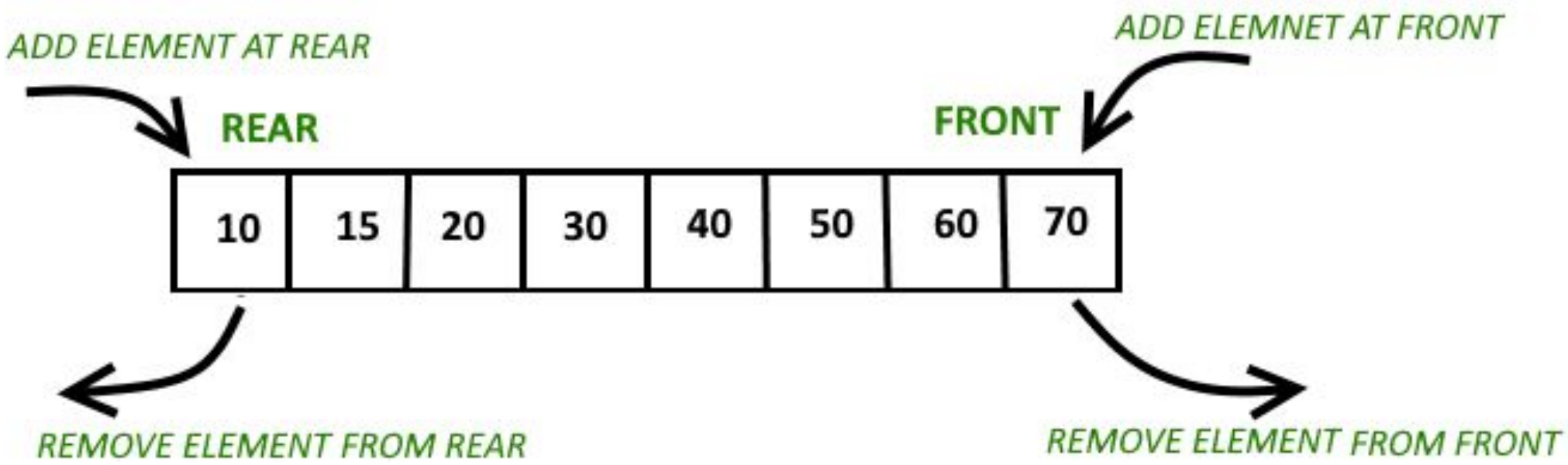
- # Create a deque containing the items (key-value pairs) of the dictionary
- g = deque(numbers.items())
- print(g)

The above code produces the following output:

- deque([8, 7, 9, 6])
- deque([45, 845, 65])
- deque([5, 6, 7, 8, 9])
- deque(['w', 'x', 'y', 'z'])
- deque(['firstname', 'age'])
- deque(['John', 25])
- deque([('firstname', 'John'), ('age', 25)])

Equivalent Methods for Stacks and Queues

- Deques can be used to implement both stacks and queues efficiently:
- **Stacks:** A stack operates on the Last In, First Out (LIFO) principle, where pushing (adding an item) is done with .append() to the right end, and popping (removing the most recent item) is performed using .pop() from the right end.
- **Queues:** A queue, based on the First In, First Out (FIFO) principle, utilizes .append() at the right end for adding (enqueueing) and .popleft() from the left end for removing (dequeuing).


ADD ELEMENT AT REAR
ADD ELEMNET AT FRONT
REAR
FRONT
10
15
20
30
40
50
60
70
REMOVE ELEMENT FROM REAR
REMOVE ELEMENT FROM FRONT

Types of Restricted Deque Input

- **Input Restricted Deque:** Input is limited at one end while deletion is permitted at both ends.
- **Output Restricted Deque:** output is limited at one end but insertion is permitted at both ends.

Python code to demonstrate deque

```
from collections import deque

# Declaring deque
queue = deque(['name','age','DOB'])

print(queue)
```

output

deque(['name', 'age', 'DOB'])

Operations on deque

Example 1: Appending Items Efficiently

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

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

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

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

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

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

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

# printing modified deque
print("\nThe deque after appending at left is : ")
print(de)
```

output

deque: deque([1, 2, 3])

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])

Example 2: Popping Items Efficiently

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

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

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

# initializing deque
de = collections.deque([6, 1, 2, 3, 4])
print("deque: ", de)

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

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

output

- deque: 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])

Example 3: Accessing Items in a deque

- index(ele, beg, end):- This function returns the first index of the value mentioned in arguments, starting searching from beg till end index.
- insert(i, a) :- This function inserts the value mentioned in arguments(a) at index(i) specified in arguments.
- remove():- This function removes the first occurrence of the value mentioned in arguments.
- 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)
```

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])
```