Heap queue (or heapq) in Python

Difficulty Level : Easy • Last Updated : 11 Sep, 2020

Heap data structure is mainly used to represent a priority queue. In Python, it is available using "**heapq**" module. The property of this data structure in Python is that each time the **smallest of heap element is popped(min heap)**. Whenever elements are pushed or popped, **heap structure in maintained**. The heap[0] element also returns the smallest element each time.

Let's see various Operations on heap :

- **heapify(iterable)** :- This function is used to **convert the iterable into a heap** data structure. i.e. in heap order.
- **heappush(heap, ele)** :- This function is used to **insert the element** mentioned in its arguments into heap. The **order is adjusted**, so as **heap structure is maintained**.
- **heappop(heap)** :- This function is used to **remove and return the smallest element** from heap. The **order is adjusted**, so as **heap structure is maintained**.

```
# Python code to demonstrate working of
# heapify(), heappush() and heappop()
# initializing list
li = [5, 7, 9, 1, 3]

# using heapify to convert list into heap
heapq.heapify(li)

# printing created heap
print ("The created heap is : ",end="")
print (list(li))

# using heappush() to push elements into heap
# pushes 4
heapq.heappush(li,4)

# printing modified heap
print ("The modified heap after push is : ",end="")
print (list(li))

# using heappop() to pop smallest element
print ("The popped and smallest element is : ",end="")
print (heapq.heappop(li))
```

Output :

```
The created heap is : [1, 3, 9, 7, 5]
The modified heap after push is : [1, 3, 4, 7, 5, 9]
The popped and smallest element is : 1
```

- **heappushpop(heap, ele)** :- This function **combines the functioning of both push and pop operations** in one statement, increasing efficiency. Heap order is maintained after this operation.
- **heapreplace(heap, ele)** :- This function also inserts and pops element in one statement, but it is different from above function. In this, **element is first popped, then the element is pushed.i.e, the value larger than the pushed value can be returned.** `heapreplace()` returns the smallest value originally in heap regardless of the pushed element as opposed to `heappushpop()`.

```
# Python code to demonstrate working of
# heappushpop() and heapreplce()

# importing "heapq" to implement heap queue
import heapq

# initializing list 1
li1 = [5, 7, 9, 4, 3]

# initializing list 2
li2 = [5, 7, 9, 4, 3]
```

```
# using heappushpop() to push and pop items simultaneously
# pops 2
print ("The popped item using heappushpop() is : ",end="")
print (heapq.heappushpop(li1, 2))

# using heapreplace() to push and pop items simultaneously
# pops 3
print ("The popped item using heapreplace() is : ",end="")
print (heapq.heapreplace(li2, 2))
```

Output :

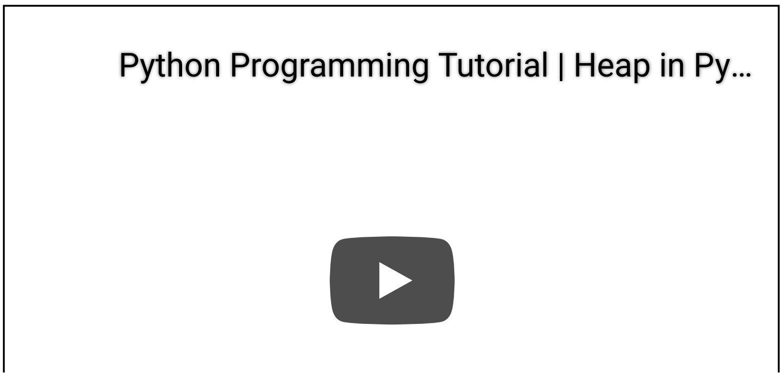
```
The popped item using heappushpop() is : 2
The popped item using heapreplace() is : 3
```

- **nlargest(k, iterable, key = fun)** :- This function is used to **return the k largest elements from the iterable specified and satisfying the key if mentioned.**
- **nsmallest(k, iterable, key = fun)** :- This function is used to **return the k smallest elements from the iterable specified and satisfying the key if mentioned.**

```
# Python code to demonstrate working of
# nlargest() and nsmallest()

# importing "heapq" to implement heap queue
import heapq

# initializing list
li1 = [6, 7, 9, 4, 3, 5, 8, 10, 1]

# using heapify() to convert list into heap
heapq.heapify(li1)

# using nlargest to print 3 largest numbers
# prints 10, 9 and 8
print("The 3 largest numbers in list are : ",end="")
print(heapq.nlargest(3, li1))

# using nsmallest to print 3 smallest numbers
# prints 1, 3 and 4
print("The 3 smallest numbers in list are : ",end="")
print(heapq.nsmallest(3, li1))
```

Output :

```
The 3 largest numbers in list are : [10, 9, 8]
The 3 smallest numbers in list are : [1, 3, 4]
```



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.

♡ **Like** 0

|< Previous

Next >|

RECOMMENDED ARTICLES

Page : 1 2 3

01 **Heap and Priority Queue using heapq module in Python**
29, Sep 20

02 **Python heapq to find K'th smallest element in a 2D array**
26, Dec 17

03 **Heapq with custom predicate in Python**
28, Sep 20

05 **heapq in Python to print all elements in sorted order from row and column wise sorted matrix**
07, Nov 17

06 **Why is Binary Heap Preferred over BST for Priority Queue?**
07, Sep 15

07 **How to implement stack using priority queue or heap?**
25, May 17

```
08
```

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Improved By : shreyashagrawal

Article Tags : priority-queue, Python

Practice Tags : priority-queue

Improve Article Report Issue

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

Load Comments

GeeksforGeeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Careers

Privacy Policy

Contact Us

Copyright Policy

Learn

Algorithms

Data Structures

Languages

CS Subjects

Video Tutorials

Practice

Courses

Company-wise

Topic-wise

How to begin?

Contribute

Write an Article

Write Interview Experience

Internships

Videos