

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY**

**Individual Work №: 4**

**Topic: Heap in Python**

**Subject: OP Teacher: Mamed Shahmaliyev**

**Group: 1459i Student: Ahmad Mammadov**

**Date: Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**Baku 2022**

**Heap queue (or heapq) in Python**

[Heap data structure is mainly used to represent a priority queue](https://www.geeksforgeeks.org/applications-of-heap-data-structure/). 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.

* **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()

# importing "heapq" to implement heap queue

import heapq

# 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="")

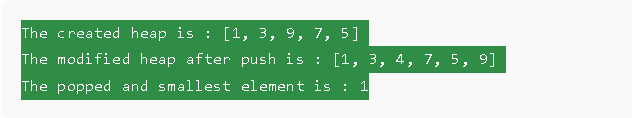
print (list(li))

# using heappop() to pop smallest element

print ("The popped and smallest element is : ",end="")

print (heapq.heappop(li))

**Output :**



* **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 heapify() to convert list into heap

heapq.heapify(li1)

heapq.heapify(li2)

# using heappushpop() to push and pop items simultaneously

# pops 2

print ("The popped item using heappushpop() is : ",end="")

print (heapq.heappushpop(li1, 2))

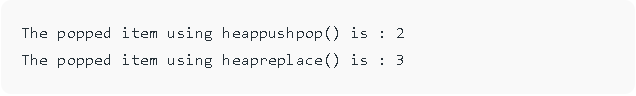
# using heapreplace() to push and pop items simultaneously

# pops 3

print ("The popped item using heapreplace() is : ",end="")

print (heapq.heapreplace(li2, 2))

**Output:**



* **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.**