# [heapq module](https://www.youtube.com/watch?v=HqPJF2L5h9U)

Heap queue is a special tree structure in which each parent node is less than or equal to its child node.

In python it is implemented using the heapq module. It is very useful is implementing priority queues where the queue item with higher weight is given more priority in processing.

**Operations on heap :**

**1. heapify(iterable)** :- This function is used to**convert the iterable into a heap** data structure. i.e. in heap order.

**2. 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**.

**3. 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**.

# 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)

print(list(li))

# [1, 3, 9, 7, 5]

# using heappush() to push elements into heap

# pushes 4

heapq.heappush(li, 4)

print(list(li))

# [1, 3, 4, 7, 5, 9]

# using heappop() to pop smallest element

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

print(heapq.heappop(li))

# The popped and smallest element is : 1

**4. 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.

**5. 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 element is pushed.i.e, the value larger than the pushed value can be returned.**

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 (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))

**6. 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.**

**7. 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.**

import heapq

nums = [1, 8, 9, 2, 23, 7, -4, 42, 37, 2]

print(heapq.nlargest(2, nums))  # [42, 37]

print(heapq.nsmallest(2, nums))  # [-4, 1]

# Second Example

portfolio = [

    {'name': 'IBM', 'shares': 100, 'price': 91.1},

    {'name': 'AAPL', 'shares': 50, 'price': 543.22},

    {'name': 'FB', 'shares': 200, 'price': 21.09},

    {'name': 'HPQ', 'shares': 35, 'price': 31.75},

    {'name': 'YHOO', 'shares': 45, 'price': 16.35},

    {'name': 'ACME', 'shares': 75, 'price': 115.65}

]

cheap = heapq.nsmallest(2, portfolio, key=lambda s: s['price'])

expensive = heapq.nlargest(2, portfolio, key=lambda s: s['price'])

print(cheap)

# [{'name': 'YHOO', 'shares': 45, 'price': 16.35}, {'name': 'FB',

# 'shares': 200, 'price': 21.09}]

print(expensive)

# [{'name': 'AAPL', 'shares': 50, 'price': 543.22},

# {'name': 'ACME', 'shares': 75, 'price': 115.65}]