Hands on-5

⇨ While implementing min heap data structure, there are several function such as parent and left/right using bit manipulation operator shown below:

```
⇨def parent(self, index):
        return (index - 1) >> 1  # Bitwise equivalent to (index - 1) //
    2

    def left(self, index):
        return (index << 1) + 1  # Bitwise equivalent to 2 * index + 1

    def right(self, index):
        return (index << 1) + 2  # Bitwise equivalent to 2 * index + 2
```

⇨ Examples of heap working is shown below including all functionality I have implemented: I have used random data (64, 4, 82, 1, 33, 63,81) for example.

```
⇨# Demonstration of functionality
    if __name__ == "__main__":
        data = [64, 4, 82, 1, 33, 63, 81]
        heap = MinHeap(data)
        print("Initial min heap:", heap)

        heap.push(2)
        print("Heap after inserting 2:", heap)

        min_element = heap.pop()
        print("Extracted min element:", min_element)
        print("Heap after extracting min:", heap)

        heap.push(0)
        print("Heap after inserting 0:", heap)

        while heap.heap:
            print("Extracting:", heap.pop(), "| Heap now:", heap)
```