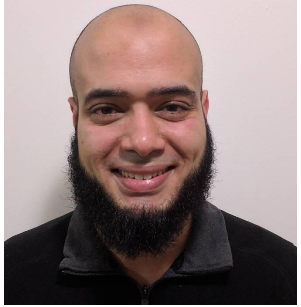
Data Structures

Heap Insertion Code

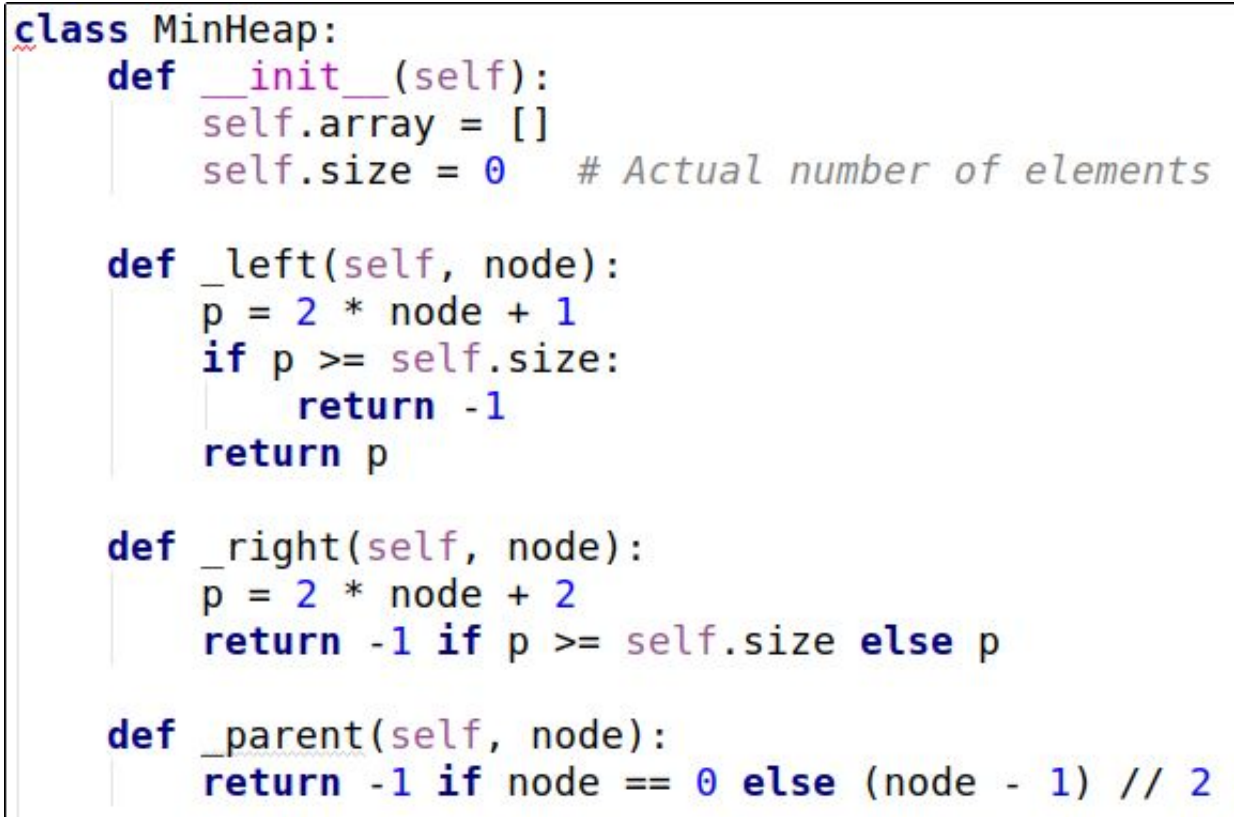
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Min Heap Class

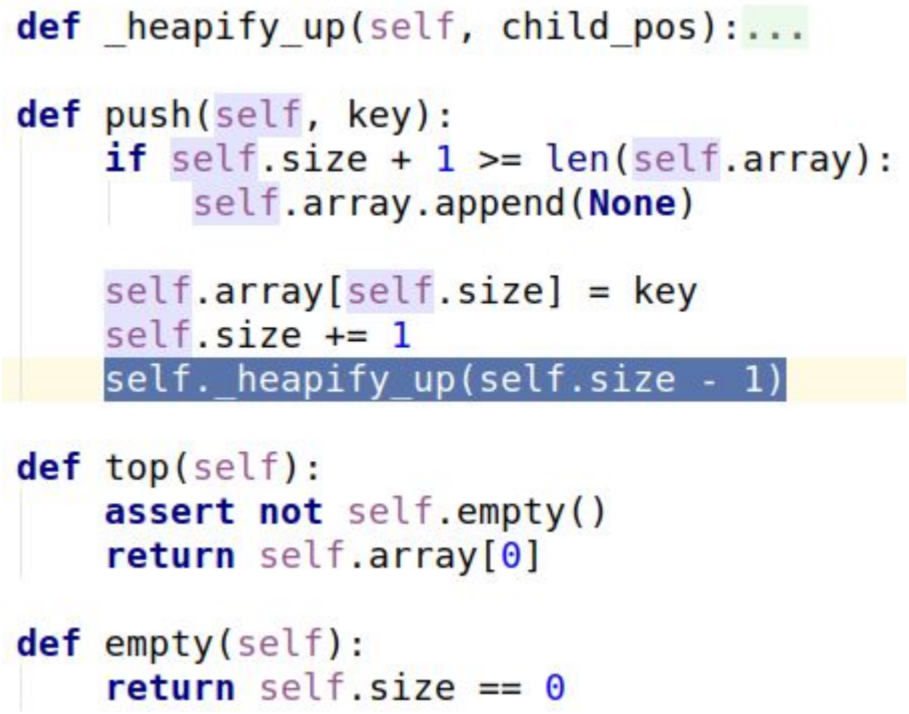
- Let’s create a class based on list (array)
- We use size to express the actual number of elements
    - Later, we delete values



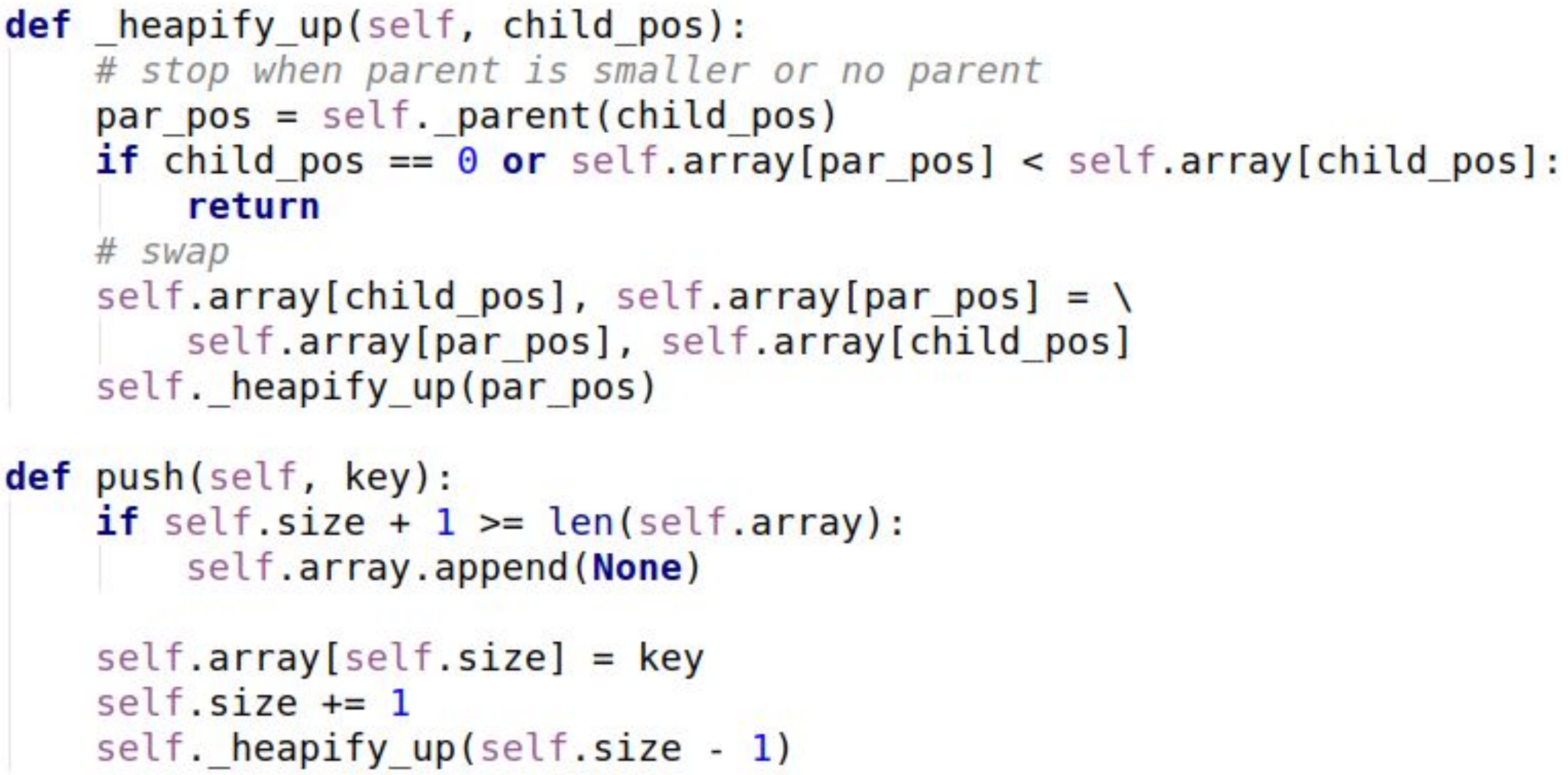
```
class MinHeap:
    def __init__(self):
        self.array = []
        self.size = 0   # Actual number of elements

    def _left(self, node):
        p = 2 * node + 1
        if p >= self.size:
            return -1
        return p

    def _right(self, node):
        p = 2 * node + 2
        return -1 if p >= self.size else p

    def _parent(self, node):
        return -1 if node == 0 else (node - 1) // 2
```

Insertion Implementation

- Add the element to the end of the array, and then adjust the position as necessary
- As we have a complete tree, its height is O(log(n))
- Given n elements to insert in a heap, we need O(nlog(n))



```
def _heapify_up(self, child_pos):...

def push(self, key):
    if self.size + 1 >= len(self.array):
        self.array.append(None)

    self.array[self.size] = key
    self.size += 1
    self._heapify_up(self.size - 1)

def top(self):
    assert not self.empty()
    return self.array[0]

def empty(self):
    return self.size == 0
```



```python
def _heapify_up(self, child_pos):
    # stop when parent is smaller or no parent
    par_pos = self._parent(child_pos)
    if child_pos == 0 or self.array[par_pos] < self.array[child_pos]:
        return
    # swap
    self.array[child_pos], self.array[par_pos] = \
        self.array[par_pos], self.array[child_pos]
    self._heapify_up(par_pos)

def push(self, key):
    if self.size + 1 >= len(self.array):
        self.array.append(None)

    self.array[self.size] = key
    self.size += 1
    self._heapify_up(self.size - 1)
```

You turn

- What if we want to remove the smallest element?
- In our min heap, the smallest element is found at the root!
- How can we fix the tree? Think in terms of a top-down procedure, similar to before

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”