### A New Approach to AI using Python

Many of us programmers want to learn new things and innovations using programming. With Artificial Intelligence as the new boom topic in the market, how about we try a new approach to it making the programming fun on the way. In this series, we will make an interesting game but I will make sure that we also learn some amazing things on the way including the concepts from Data structures, Algorithm design, Problem Solving and Problem Analysis.

### Why Python?

I would not mention the old irrelevant reasons of using Python in this article here. I am using Python as a programming language not because it is easy to use or it goes best when comes to AI. In fact, we can use any programming language we like for this journey. The main thing would always be to understand the concepts and implement it in any language. Are you getting my point?

…….

Ok then. Lets begin.

### What we will build?

In this series, we will build an interesting AI based bot that will learn how to pass any obstacles in its way and always find a path to its destination. It keeps on trying until it does not find the way. But, I promise you one thing we will not use any library and do almost everything on our own.

Prerequisites: Basic Knowledge of Python 3, Classes in Python, Object Oriented Programming.

Let us see the main topic we will be learning in this series:

### Creating a QUEUE class in Python

First, we will create our own Queue class in Python. A Queue is a data structure based on the FIFO — First In First Out approach. Analogous to the daily queues we see at ticketing counters, ATM Machines, you can only join the queue from the end (unless someone is not a “cutsy”) and will only be attended when you are at the front of the line. Similarly, we have a queue data structure in which you can always add new data to end and remove the data from the beginning for processing it.

Queue Data Structure

A Queue class must be able to imitate this special behaviour of Queues. Thus, in a Queue class, we will have two protected data members — “queue”. The “queue” member will simply be a list in python (an array) to hold the data elements. Also, we will add some particular functions in this class in order for it to be used specifically as Queues. Let us understand those.

1. enqueue(item):- It is used to add an element to the end of the queue.

2. dequeue():- It is used to remove an element from the beginning of the queue and return it as well.

3. \_\_str\_\_():- This is a special function in python classes which is used to get the string representation of Python object. In case you are familiar with Java, it is similar to toString().

4. size property:- It is used to get the length of the queue.

Since we have made the “queue” member as protected, you can not use it outside the class unless it is a derived class, hence you need these functions to access it and operate on it.

Here is the code for making Queue class. You can save it as MyQueue.py

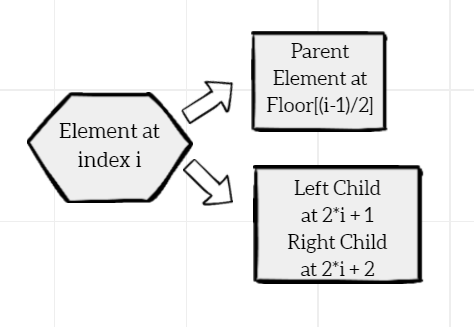
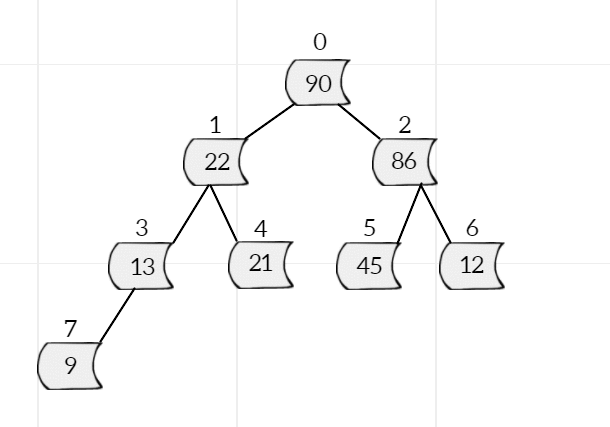
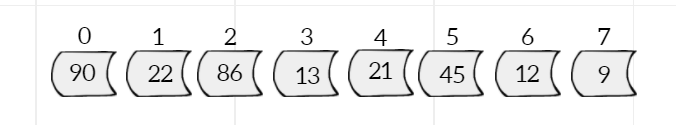
MyQueue.py

### Creating a Priority Queue using our Queue class

Now, what is a Priority Queue? Well, it is a queue that simply organizes itself according to some priority condition whenever a new element is added or an element is removed from the queue. We will make a priority queue using a queue and one other data structure known as Binary Heap. Let us understand what is a Binary Heap first.

A Binary Heap is a hierarchy based data structure created using complete binary trees ( in which all the levels are filled completely except the last one due to which they can be stored as array — our first connection of heaps to queues). A Binary Heap can be either Min Heap / Max Heap and must hold the following property always — The element at the root is minimum/ maximum of all its successors, same property holding for the root of all subtrees in the heap.

Another amazing thing about Heaps ( or complete binary trees) is that when you store them simply as a array, you can access the children or parent of an element (if exists) by a very basic formula

Children and Parent of an Element in a complete binary tree stored as arrayExample of a HeapHeap stored as Array

Now, lets come back to the priority queue. So a priority queue is a combination of a queue and a heap. Basically we can always add at the end of the priority queue and remove from the beginning. Also, after the addition or deletion of that element, the priority queue organizes itself based on the priority condition (that is associated with min/max heap) using a process known as the heapify process.

Here is a code for PriorityQueue class using Python which is derived from our Queue class and use the basic heapify functions — topDownHeapify (that organizes the heap from top to bottom) and bottomUpHeapify (that organizes the heap from bottom to top). You can also refer this article to know more about the heapify process and priority queues.

[Priority Queues with Binary Heaps  
One important variation of the queue is the priority queue. A priority queue acts like a queue in that items remain in…bradfieldcs.com](https://bradfieldcs.com/algos/trees/priority-queues-with-binary-heaps/" \o "https://bradfieldcs.com/algos/trees/priority-queues-with-binary-heaps/)

Here is the code for PriorityQueue class using Python which inherits our Queue class and also provide other functions for heapify process. MyPriorityQueue.py