VII. Stack & Queue in Python

```
import array

arr = array.array('i')   # syntax of empty array
arr = array.array('i', [1,2,3])   # this entire is printed.

arr.append(10)   # adding 10 to the array
print (arr)
arr.pop ()   # it pop's the last element in the array
print (arr)
arr.pop(0)   # to pop the element at exact location
print (arr [0:2]   # array slicing

for i in range (0,2):
    arr.pop()
    print(arr)
```

* If you find the length of the stack you will find the top

STACK (LIFO) last in first out.

* operation to insert an element is 'PUSH'
* operation to remove an element is 'POP'

```
import array
class Stack:              object
    def-init-(self):              # constructor
        self.my-stack = array.array('i', [])
        self.top = -1
    def push(self, element):
        self.my-stack.append(element)

    def pop(self):
        self.my-stack.pop() →

    def is-empty(self):
        if len(self.my-stack):
            return False
        else:
            return True
```

```
def ~~get~~top(self):
    ~~return~~(len(self.
        my-stack)-1)
```

```
if self.is-empty():
    print("Stack is empty cant pop")
else:
    self.my-stack.pop()
```

This is the entry point. main function

```
If _name_ == "_main_" :
    stack = Stack() # This is how we create
    variable   class name   # object in python

    stack.push(1)
    stack.push(2)
    stack.push(3)

    print(stack.my_stack)
    stack.top()}
     stack.pop()
    print(stack.my_stack)
```

Queue FIFO (First in First Out)

```
import array

class Queue:
    def _init_(self):
        self.my_[redacted] queue = array.array('i', [])

    def enqueue(self, element):
        self.myqueue.append(element)

*   def dequeue(self):                    # what if
        self.my_queue.pop(0)              # empty.

*

     def dequeue(self):
        if self.is_empty():
            print(" Queue is empty ")
        else:
            self.my_queue.pop(0)
```

```
if _name_ == "_main_":
    queue = Queue()
    print(queue.my_queue)
    queue.enqueue (1)
    queue.enqueue (2)
    queue.enqueue (3)
    print(queue.my_queue)
    queue.dequeue()
    print(queu.my_queue)
```