Stack is a data structure which have two types of operation i.e., ***push and pop*** with a time complexity of *O(1)*. It follows the principle of **last-in first-out**.

Real Life Examples of Stack:

- Mechanism of undo and redo in text editors
- Recursion in our code where the call of next function pile up onto the stack
- Internet web browsers store the addresses of recently visited sites

Array Stack: Array Stack is nothing but it has a static size from which the data will not be overflowed.
In python, there is no any concept of array but we can create Array Stack using *Adaptor Design Pattern.*

Adaptor Design Pattern means to customize/modify the existing class in order to get our functionalities of the data structure we are working for. Here we can create Array stack using list which has a data member of size that will declare array with sentinel values up to the parametric size. Whenever user tries to push their data that will be greater than size it will trigger the warning with a text *"Stack is Full"*.

Below is the implementation of ADT (Stack):

```
#Stack Implementation using cpp array (static sized array)
class ArrayStack:
    def __init__(self, size):
        self.data = ["$"] * size
        self.size = 0

    def push(self, data):
        if not self.is_Full():
            self.data[self.size] = data
            self.size += 1
        else:
            raise "Stack is Full"

    def pop(self):
        if not self.is_Empty():
            temp = self.data[self.size - 1]
            self.data[self.size - 1] = "$"
            self.size -= 1
            return temp
        else:
            raise "Stack is Empty"

    def peek(self):
        if not self.is_Empty():
            return self.data[self.size - 1]
        raise ("Stack is Empty")

    def length(self):
        return self.size + 1

    def is_Full(self):
        return True if self.size == len(self.data) else False
    def is_Empty(self):
        return True if self.size == 0 else False

    def copy(self, size):
        stack = ArrayStack(size)
        for i in range(self.size):
            stack.push(self.data[i])

        return stack

    def __str__(self) -> str:
        s = ""
        for i in range(self.size):
            s += f"{self.data[i]} "
        s += "\n"
        return s

# Driver's Code for stack practicality
if __name__ == "__main__":
    stack = ArrayStack(3)
    stack.push(1)
    stack.push(2)
    stack.push(3)

    print ("Length of Stack:",stack.length())
    print ("Popping from the stack:", stack.pop())
```

~~~
print ("Peek data of the stack:", stack.peek())
print ("Stack is Full:", stack.is_Full())
print ("Stack is Empty:", stack.is_Empty())

```
~~~