# Workshop #3: Stacks

## GitHUb: abhijit-baruah

Implement the following for a Stack data structure:

- `__len__`
- `__bool__`
- `__repr__`
- `__str__`
- `__contains__`

In [1]:

```
class Stack:
    def __init__(self):
        self.items = []  # or, use 'list()'

    def is_empty(self):
        return self.items == []

    def size(self):
        return len(self.items)

    def peek(self):
        # look at last item in our underlying list
        if len(self.items): # This 'if' condition automatically executes for len(self.items
            return self.items[-1]
        else:
            return None

    def push(self, item):
        self.items.append(item)

    def pop(self):
        return self.items.pop() # list already has inbuilt 'pop' which acts the same way as

    def __len__(self):
        return len(self.items)
        # or return self.size()

    def __bool__(self):
        return self.items != []
        # or return not self.is_empty()


    def __repr__(self): # this should give the pythonic representation of what the object i
                        # this object again by copy-paste of the output and assigning to a
        return "Stack()"


    def __str__(self):
        v = "Stacks({0})".format(self.items)
        return v
    '''
    or, to have a more customized representation of stack

    def __str__(self):
        returnVal = "<Stack:"
        for item in self.items:
            returnVal += str(item) + "; "
        returnVal += ">"
        return returnVal

    '''


    def __contains__(self, item):
        return item in self.items

    '''
    or
    def __contains__(self, item):
```

```
        if item in self.items:
            return True
        else:
            return False
    '''
```

In [2]:

```
s = Stack()
s.push(1)
s.push(2)
s.push(3)
s.push(4)
s.push(5)
```

In [3]:

```
# check __repr__
s
```

Out[3]:

```
Stack()
```

In [4]:

```
# check __str__
print(s)
```

```
Stacks([1, 2, 3, 4, 5])
```

In [5]:

```
# check __len__
len(s)
```

Out[5]:

```
5
```

In [6]:

```
# check __contains__
6 in s
```

Out[6]:

```
False
```

In [7]:

```
# check __bool__
bool(s)
```

Out[7]:

```
True
```