Stack class

February 4, 2021

Stack Implementation

[ ]:
```
class Stack:
    #Constructor
    def __init__(self):
        self.stack = list() #create a empty list
        self.maxSize = 8 # set the max size for the stack
        self.top = 0  # initialise the top  to 0

    #Adds element to the Stack
    def push(self,data):
        if self.top>=self.maxSize:  # check if the stack can store any more␣
↪data.
            return ("Stack Full!")  # if not print stack full

        self.stack.append(data)     #  else add the data to stack  using append␣
↪method.
        self.top += 1               # increment the top  by 1
        return True

    #Removes element from the stack
    def pop(self):
        if self.top<=0:  # check if the stack has any data to be popped
            return ("Stack Empty!")  # if not print stack empty
        item = self.stack.pop()      # else  pop the data from stack
        self.top -= 1                # decrement the top
        return item                  # return the popped data

    #Size of the stack
    def size(self):
        return self.top               # variable top also serves to keep info␣
↪about size of stack

    # top of the stack
    def topOfStack(self): # you can implement this method
        # Todo
        pass
    # isEmpty()
    def isEmpty(self): # you can implement this method
        # Todo
        pass
```

[ ]:
```
s = Stack()
```

[ ]:
```
print(s.push(1))
print(s.push(2))
print(s.push(3))
print(s.push(4))
print(s.push(5))
print(s.push(6))
print(s.push(7))
print(s.push(8))
print(s.push(9))
```

[ ]:
```
print(s.size())
```

[ ]:
```
print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())
print(s.pop())
```

[ ]:
```
print(s.size())
```

[ ]:
```

```