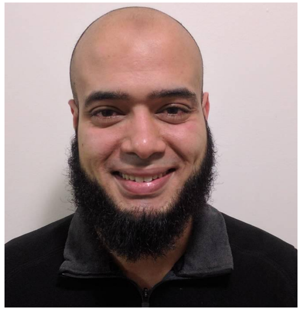
Data Structures

Array-based Stack

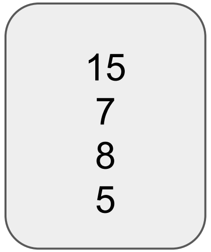
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Using an array

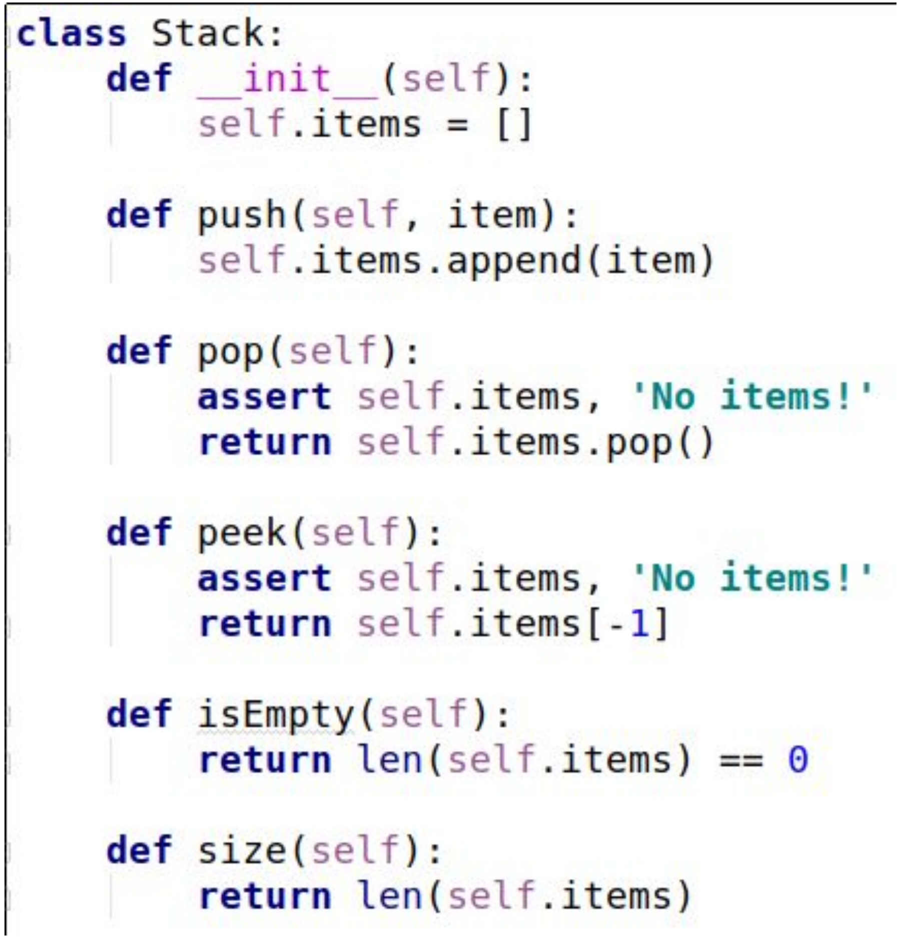
- The array is a great match for the stack!
- We push into the stack in the order: 5, 8, 7, 15 (shown on the right)
- Simply, the corresponding array is {5, 8, 7, 15}
    - The last element in the array will be its peek (15) = its **peek**
    - To add a new element, 4, we **add** to the end ⇒ {5, 8, 7, 15, 4}
    - If we want to **pop 2** elements ⇒ {5, 8, 7}



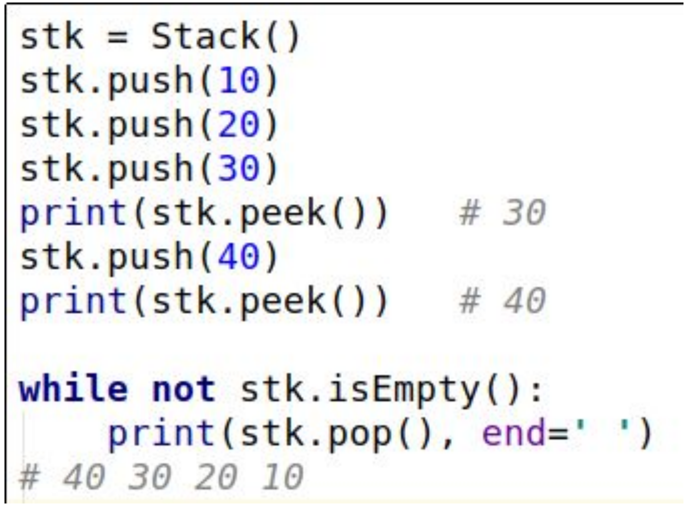
Index	0	1	2	3
Value	5	8	7	15

Design

- Similar to the Array section, We need an **array** internally to grow up **dynamically** the memory
- But, if the **List** is already implemented for us, we can just make use of it
- So, we will write a simple stack based on wrapping Python List
- Given that Python List is flexible, we don't need to implement **isFull** functionality. That is, the stack can grow up as much as we need



```python
class Stack:
    def __init__(self):
        self.items = []

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

    def pop(self):
        assert self.items, 'No items!'
        return self.items.pop()

    def peek(self):
        assert self.items, 'No items!'
        return self.items[-1]

    def isEmpty(self):
        return len(self.items) == 0

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

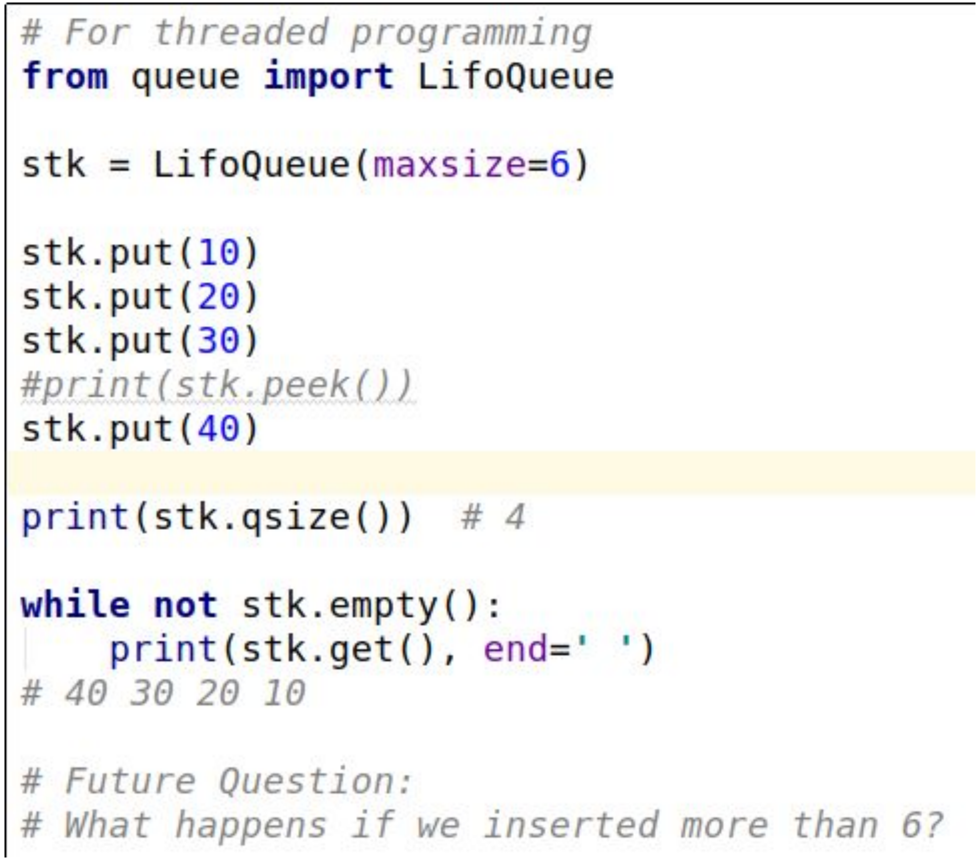


```python
stk = Stack()
stk.push(10)
stk.push(20)
stk.push(30)
print(stk.peek())   # 30
stk.push(40)
print(stk.peek())   # 40

while not stk.isEmpty():
    print(stk.pop(), end=' ')
# 40 30 20 10
```

- Tip: In practice, we may raise an **exception**, rather than asserting

Built-in

- There is also
  - `from collections import deque`
  - `stk = deque()`
- Not sure of their **internal** implementation



```python
# For threaded programming
from queue import LifoQueue

stk = LifoQueue(maxsize=6)

stk.put(10)
stk.put(20)
stk.put(30)
#print(stk.peek())
stk.put(40)

print(stk.qsize())  # 4

while not stk.empty():
    print(stk.get(), end=' ')
# 40 30 20 10

# Future Question:
# What happens if we inserted more than 6?
```

Misc

- Complexity: All other operations are O(1) time, depending on list implementation
- In practice, we usually just use the **built-in list** to simulate a stack. It doesn’t make sense to create such simple wrapper
    - In homework, use directly the list

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."