Stack (Implementation)

In this lesson, we are going to look at how Stacks are implemented in Python and how the main functions in the stack actually work.

We'll cover the following

Introduction#

Most programming languages come with the built-in Stack data structure. In Python, you can use the pre-built Stack class by importing them into your program. However, implementing a Stack from scratch will allow you to truly master the ins and outs of the data structure.

Implementation#

Stacks can be implemented using Lists or Linked Lists in Python language. Each implementation has its own advantages and disadvantages. Here, however, we will show an implementation of *stacks* using **lists**.

As mentioned in the previous lesson, a typical Stack must contain the following functions:

- `push(element)`
- `pop()`
- `peek()`
- `is_empty()`
- `size()`

We will take a closer look at these functions individually, but, before we do, let’s construct a simple Stack class called `MyStack` and create its object with a `stack_obj` name. This class will consist of the member functions given above and a list called `stack_list` that will hold all the elements of the stack.

Stack.py

```python
class MyStack:
    def __init__(self):
        self.stack_list = []

stack_obj = MyStack()
```

Now, before adding the `push(element)` and `pop()` functions into this code, let’s implement the following functions:

- `is_empty()`
- `peek()`
- `size()`

Let’s examine the following code:

Stack.py

```
class MyStack:
    def __init__(self):
        self.stack_list = []
        self.stack_size = 0

    def is_empty(self):
        return self.stack_size == 0

    def peek(self):
        if self.is_empty():
            return None
        return self.stack_list[-1]

    def size(self):
        return self.stack_size

if __name__ == "__main__" :
    stack_obj = MyStack()
    print("is_empty(): " + str(stack_obj.is_empty()))
    print("peek(): " + str(stack_obj.peek()))
    print("size(): " + str(stack_obj.size()))
```

The `is_empty()` function returns the boolean you get from comparing the length of the array to 0. If the length of the array is 0, then the function will return `True`. Otherwise, it will return `False`.

The `peek()` function returns the last element in the list, which we are considering to be the *top* of the stack! In case of an empty `stack_list`, it would return `None`.

Lastly, the `size()` method simply returns the count of elements in the stack.

Now, study the implementation of the `push(element)` and `pop()` functions.

Stack.py

```
class MyStack:
    def __init__(self):
        self.stack_list = []
        self.stack_size = 0
    def is_empty(self):
        return self.stack_size == 0

    def peek(self):
        if self.is_empty():
            return None
        return self.stack_list[-1]

    def size(self):
        return self.stack_size

    def push(self, value):
        self.stack_size += 1
        self.stack_list.append(value)

    def pop(self):
        if self.is_empty():
            return None
        self.stack_size -= 1
        return self.stack_list.pop()

if __name__ == "__main__" :
    stack_obj = MyStack()

    print("Pushing elements into the stack")
    for i in range(5):
        print(i)
        stack_obj.push(i)

    print("is_empty(): " + str(stack_obj.is_empty()))
    print("peek(): " + str(stack_obj.peek()))
    print("size(): " + str(stack_obj.size()))

    print("Popping elements from the stack")
    for x in range(5):
        print(stack_obj.pop())

    print("is_empty(): " + str(stack_obj.is_empty()))
    print("peek(): " + str(stack_obj.peek()))
    print("size(): " + str(stack_obj.size()))
```

If you look at the output of the code, you can see that the elements popped out of the stack in the **exact reverse order** that they were pushed in. That means our Stack works perfectly. Congratulations, you have ss implemented a Stack using a Python List!

Complexities of Stack Operations#

Let’s look at the time complexity of each stack operation.

Operation	Time Complexity
`push(element)`	O(1)
`pop()`	O(1)
`peek()`	O(1)
`is_empty()`	O(1)
`size()`	O(1)

The next data structure that we are going to look at is a **Queue**.

← Back
What is a Stack?

Next →
What is a Queue?

☑ Completed

Report an Issue