python_stack_practice

May 16, 2020

0.1 Building a Stack in Python

Before we start let us reiterate they key components of a stack. A stack is a data structure that consists of two main operations: push and pop. A push is when you add an element to the **top of the stack** and a pop is when you remove an element from **the top of the stack**. Python 3.x conviently allows us to demonstate this functionality with a list. When you have a list such as [2,4,5,6] you can decide which end of the list is the bottom and the top of the stack respectivley. Once you decide that, you can use the append, pop or insert function to simulate a stack. We will choose the first element to be the bottom of our stack and therefore be using the append and pop functions to simulate it. Give it a try by implementing the function below!

Try Building a Stack

```
In [8]: class Stack:
            def __init__(self):
                 # TODO: Initialize the Stack
                self.items = []

            def size(self):
                 # TODO: Check the size of the Stack
                return len(self.items)

            def push(self, item):
                 # TODO: Push item onto Stack
                self.items.append(item)

            def pop(self):
                if self.size() == 0:
                    return None
                 # TODO: Pop item off of the Stack
                return self.items.pop()
```

Test the Stack

```
In [9]: MyStack = Stack()

        MyStack.push("Web Page 1")
        MyStack.push("Web Page 2")
```

```
        MyStack.push("Web Page 3")

        print (MyStack.items)

        MyStack.pop()
        MyStack.pop()

        print ("Pass" if (MyStack.items[0] == 'Web Page 1') else "Fail")

        MyStack.pop()

        print ("Pass" if (MyStack.pop() == None) else "Fail")
```

```
['Web Page 1', 'Web Page 2', 'Web Page 3']
Pass
Pass
```

Hide Solution

```
In [ ]: # Solution

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

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

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

            def pop(self):
                if self.size()==0:
                    return None
                else:
                    return self.items.pop()

        MyStack = Stack()

        MyStack.push("Web Page 1")
        MyStack.push("Web Page 2")
        MyStack.push("Web Page 3")

        print (MyStack.items)

        MyStack.pop()
        MyStack.pop()
```

```
print ("Pass" if (MyStack.items[0] == 'Web Page 1') else "Fail")

MyStack.pop()

print ("Pass" if (MyStack.pop() == None) else "Fail")# Solution

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

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

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

    def pop(self):
        if self.size()==0:
            return None
        else:
            return self.items.pop()

MyStack = Stack()

MyStack.push("Web Page 1")
MyStack.push("Web Page 2")
MyStack.push("Web Page 3")

print (MyStack.items)

MyStack.pop()
MyStack.pop()

print ("Pass" if (MyStack.items[0] == 'Web Page 1') else "Fail")

MyStack.pop()

print ("Pass" if (MyStack.pop() == None) else "Fail")# Solution

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

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

    def push(self, item):
```

```
        self.items.append(item)

    def pop(self):
        if self.size()==0:
            return None
        else:
            return self.items.pop()

MyStack = Stack()

MyStack.push("Web Page 1")
MyStack.push("Web Page 2")
MyStack.push("Web Page 3")

print (MyStack.items)

MyStack.pop()
MyStack.pop()

print ("Pass" if (MyStack.items[0] == 'Web Page 1') else "Fail")

MyStack.pop()

print ("Pass" if (MyStack.pop() == None) else "Fail")
```