CSC148 Worksheet 10 Solution

Hyungmo Gu

April 22, 2020

Question 1

a. The following code must be changed.

- *self._items.append(item)* in *push()* method must be changed to *self._items.insert(0,item)*
- *self._items.pop()* in *pop()* method must be changed to *self._items.pop(0)*.

b.

```
class Stack:
    """A last-in-first-out (LIFO) stack of items.
    Stores data in first-in, last-out order. When removing an item from the
    stack, the most recently-added item is the one that is removed.
    """
    # === Private Attributes ===
    # _items:
    # The items stored in the stack. The end of the list represents
    # the top of the stack.
    _items: List

    def __init__(self) -> None:
        """Initialize a new empty stack.
        """
        self._items = []

    def is_empty(self) -> bool:
        """Return whether this stack contains no items.
        >>> s = Stack()
        >>> s.is_empty()
        True
        >>> s.push(
        hello
        )
        >>> s.is_empty()
        False
        """

        return self._items == []

    def push(self, item: Any) -> None:
```

```
32              """Add a new element to the top of this stack.
33              """
34              # ========== Solution (Question 1.b) ==========
35              self._items.insert(0,item)
36              # =============================================
37
38          def pop(self) -> Any:
39              """Remove and return the element at the top of this stack.
40              >>> s = Stack()
41              >>> s.push(
42              hello
43              )
44              >>> s.push(
45              goodbye
46              )
47              >>> s.pop()
48
49              goodbye
50
51              """
52              # ========== Solution (Question 1.b) ==========
53              self._items.pop(0)
54              # =============================================
55
```

Listing 1: worksheet_10_q1b_solution.py

Question 2

- The following changes in docstring must be made.

    1. The line 'The items stored in the stack. The end of the list represents the top of the stack.' under the description of *_items* in private attribute should be changed to 'The items stored in the stack. The end of the list represents the *bottom* of the stack.'
    2. The line 'Add a new element to the top of this stack.' in *push()* method must be changed to 'Add a new element to the *bottom* of this stack.'
    3. The line 'Remove and return the element at the top of this stack.' in *pop()* method must be changed to 'Remove and return the element at the *bottom* of this stack.'

Correct Solution:

- The following changes in docstring must be made.

    1. The line 'The items stored in the stack. The end of the list represents the top of the stack.' under the description of *_items* in private attribute should be

changed to 'The items stored in the stack. The front of the list represents the top of the stack.'

- This is because stack is LIFO. Last element in is the first to come out.
- Last element added and removed are now at the beginning of the list.

Notes:

- Learned that the **top of the stack** means where the push and pop occurs.
- 형모야. 쪼금만이라도 무너지지 않고 여보에게 더 빨리 갈 수 만있다면...
- 형모야. 내 여보 있어
- 형모야. 괜찮아
- 형모야. 차분히...

Question 3

- None of the code should be changed.

  The below is the code used in last lecture

```
        def is_balanced(line: str) -> bool:
        """Return whether <line> contains balanced parentheses.

        >>> is_balanced('abc')
        True
        >>> is_balanced('(a * (3 + b))')
        True
        >>> is_balanced('(a * (3 + b]]')
        False
        >>> is_balanced('(a * [3 + b])')
        True
        >>> is_balanced('1 + 2(x-y)}')
        False
        >>> is_balanced('{3 + [2 * 4(x-y)]}')
        True
        >>> is_balanced('3 - (x')
        False
        """
        brackets_stack = Stack()

        for character in line:
            # If the character is one of '[', '{'. or '(',
            if (character == '(' or
                character == '[' or
                character == '{'):
```

```
                # Store it in stack
                brackets_stack.push(character)
            # If the character is one of ']', '}', or ')',
            elif (character == ')' or
                  character == ']' or
                  character == '}'):
                # Check for the non-emptiness of stack.
                if brackets_stack.is_empty():
                    # if empty, return false.
                    return False

                # If the list is not empty, then pop an element form stack.
                left_bracket = brackets_stack.pop()

                # If popped bracket doesn't match, then return false
                if ((left_bracket == '(' and character != ')') or
                    (left_bracket == '[' and character != ']') or
                    (left_bracket == '{' and character != '}')):

                    return False

        # Check parenthesis are balanced by checking stack is empty.
        if not brackets_stack.is_empty():
            return False

        return True

```

Listing 2: worksheet_10_q1b_solution.py

Because we know the code is built with the thought of *Stack* functioning as LIFO, and because we know from question 2 that *Stack* still behaves the same after the changes, we can conclude no new changes are required.

Question 4

- The original stack class is the better choice.

  The criteria used for the conclusion is the performance of adding element to list.

  The new stack class uses *.insert()* method to add elements.

  Because we know each element added using this method requires a new list to be created and all elements need to be transferred from one list to another, we can conclude the new stack class is not efficient.

Correct Solution:

The original stack class is the better choice.

The criteria used for the conclusion is the performance of adding element to list.

The new stack class uses *.insert()* method to add elements.

Because we know each element added using this method requires a new list to be created and all elements need to be transferred from one list to another, we can conclude the new stack class is not efficient.

In terms of readability, both are the great choice.