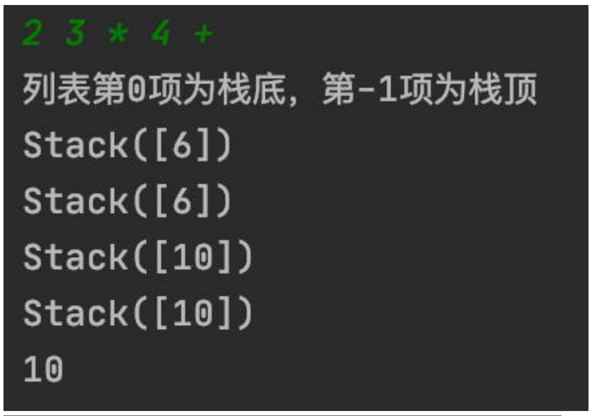
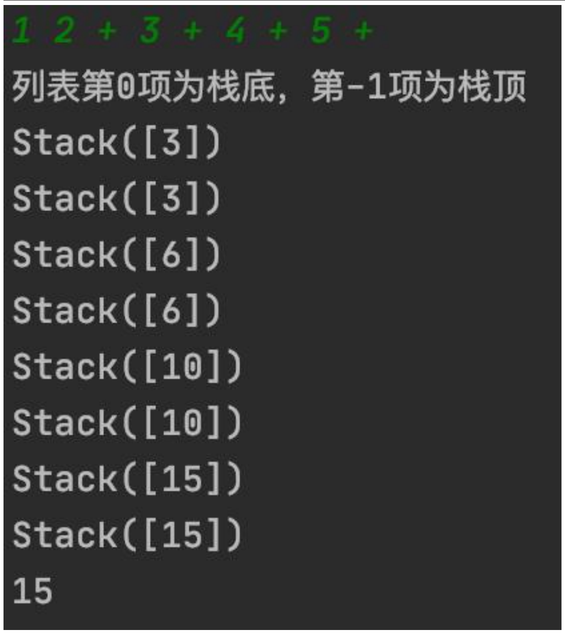
【K03】展示后缀表达式计算过程的栈变化

```
class Stack:
    def __init__(self):
        self.items = []
    def isEmpty(self):
        return self.items == []
    def push(self, item):
        self.items.append(item)
    def pop(self):
        return self.items.pop()
    def __str__(self):
        return 'Stack(%s)' % self.items
    __repr__ =    __str__


def postfixEval(postfixExpr):
    operandStack = Stack()
    tokenList = postfixExpr.split(' ')
    for token in tokenList:
        if token in '0123456789':
            operandStack.push(int(token))
        else:
            operand2 = operandStack.pop()
            operand1 = operandStack.pop()
            result = doMath(token,operand1,operand2)
            operandStack.push(result)
            print(operandStack)
            print(operandStack.__repr__())
    return operandStack.pop()

def doMath(op,op1,op2):
    if op == '*':
        return op1 * op2
    elif op == '/':
        return op1 / op2
    elif op =='+':
        return op1 + op2
    else:
        return op1 - op2

postfixString = str(input())
```

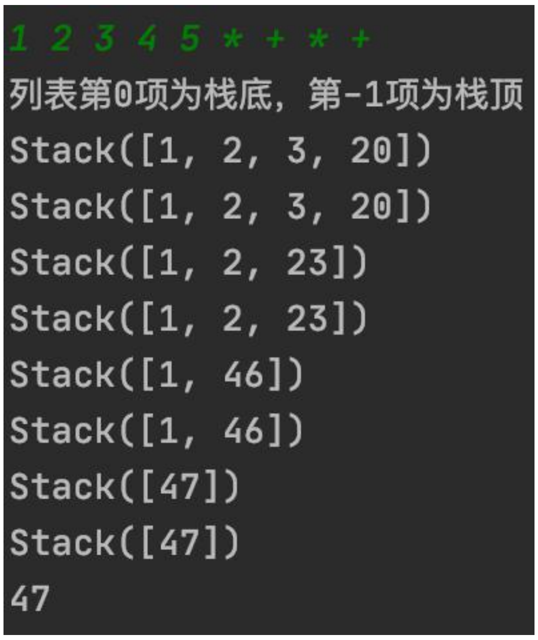
```
print('列表第 0 项为栈底，第-1 项为栈顶')
print(postfixEval(postfixString))
```



```
2 3 * 4 +
列表第0项为栈底，第-1项为栈顶
Stack([6])
Stack([6])
Stack([10])
Stack([10])
10
```



```
1 2 + 3 + 4 + 5 +
列表第0项为栈底，第-1项为栈顶
Stack([3])
Stack([3])
Stack([6])
Stack([6])
Stack([10])
Stack([10])
Stack([15])
Stack([15])
15
```



```
1 2 3 4 5 * + * +
列表第0项为栈底，第-1项为栈顶
Stack([1, 2, 3, 20])
Stack([1, 2, 3, 20])
Stack([1, 2, 23])
Stack([1, 2, 23])
Stack([1, 46])
Stack([1, 46])
Stack([47])
Stack([47])
47
```