4장 스택 과제

GBT학부
201904385 우인경

1. Stack 클래스 버전 구현 및 실행

<코드 화면>

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

    def isEmpty(self):
        return len(self.top) == 0
    def size(self):
        return len(self.top)
    def clear(self):
        self.top = []

    def push(self, item):
        self.top.append(item)
    def pop(self):
        if not self.isEmpty():
            return self.top.pop()
    def peek(self):
        if not self.isEmpty():
            return self.top[-1]

    def __str__(self):
        return str(self.top[::])
```

```python
from stack import *

odd = Stack()
even = Stack()

for i in range(10):
    if i % 2 == 0:
        even.push(i)
    else:
        odd.push(i)

print("=" * 30)
print("even : ", even)
print("odd : ", odd)

print("even peek : ", even.peek())
print("odd peek: ", odd.peek())
print()

for i in range(2):
    even.pop()
for i in range(3):
    odd.pop()

print("=" * 30)
print("even : ", even)
print("odd : ", odd)
```

<실행 화면>

```
================ RESTART: /Users/inkyung/Documents/stack_test.py
==============================
even :  [0, 2, 4, 6, 8]
odd :  [1, 3, 5, 7, 9]
even peek :  8
odd peek:  9

==============================
even :  [0, 2, 4]
odd :  [1, 3]
>>>
```

2. 중위 표기 수식을 후기 표기로 변환

(evalPostfix 프로그램 코드는 3번에 있습니다)

```
from evalPostfix import *

def precedence(op):
    if op == '(' or op == ')':
        return 0
    elif op == '+' or op == '-':
        return 1
    elif op == '*' or op == '/':
        return 2
    else:
        return -1


def Infix2Postfix(expr):
    s = Stack()
    output = []

    for term in expr:
        if term in '(':
            s.push('(')
        elif term in ')':
            while not s.isEmpty():

                op = s.pop()
                if op == '(': break;
                else:
                    output.append(op)
        elif term in "+-*/":
            while not s.isEmpty():
                op = s.peek()
                if(precedence(term) <= precedence(op)):
                    output.append(op)
                    s.pop()
                else: break
            s.push(term)
        else:
            output.append(term)

    while not s.isEmpty():
        output.append(s.pop())

    return output

infix1 = ['8', '/', '2', '-', '3', '+', '(', '3', '*', '2', ')']
infix2 = ['1', '/', '2', '*', '4', '*', '(', '1', '/', '4', ')']
print()

postfix1 = Infix2Postfix(infix1)
postfix2 = Infix2Postfix(infix2)

print('중위표기: ', infix1)
print('후위표기: ', postfix1, end='\n\n')
print('중위표기: ', infix2)
print('후위표기: ', postfix2)
```

```
>>>
=============== RESTART: /Users/inkyung/Documents/infixpostfix.py ==============
['8', '2', '/', '3', '-', '3', '2', '*', '+']  계산결과 -->  7.0
['1', '2', '/', '4', '*', '1', '4', '/', '*']  계산결과 -->  0.5

중위표기:  ['8', '/', '2', '-', '3', '+', '(', '3', '*', '2', ')']
후위표기:  ['8', '2', '/', '3', '-', '3', '2', '*', '+']

중위표기:  ['1', '/', '2', '*', '4', '*', '(', '1', '/', '4', ')']
후위표기:  ['1', '2', '/', '4', '*', '1', '4', '/', '*']
>>>
```

3. 후위 표기 수식을 stack을 이용하여 계산 구현 및 실행

```
from stack import *

def evalPostfix(expr):
    s = Stack()

    for token in expr:

        if token in "+-*/":
            val2 = s.pop()
            val1 = s.pop()

            if (token == '+'): s.push(val1 + val2)
            elif (token == '-'): s.push(val1 - val2)
            elif (token == '*'): s.push(val1 * val2)
            elif (token == '/'): s.push(val1 / val2)
        else:
            s.push(float(token))

    return s.pop()

expr1 = ['8', '2', '/', '3', '-', '3', '2', '*', '+']
expr2 = ['1', '2', '/', '4', '*', '1', '4', '/', '*']

print(expr1, ' 계산결과 --> ', evalPostfix(expr1))
print(expr2, ' 계산결과 --> ', evalPostfix(expr2))
```

```
>>>
=============== RESTART: /Users/inkyung/Documents/evalPostfix.py ===
['8', '2', '/', '3', '-', '3', '2', '*', '+']  계산결과 -->  7.0
['1', '2', '/', '4', '*', '1', '4', '/', '*']  계산결과 -->  0.5
>>>
```

4. Stack을 이용한 미로 탐색 구현 및 실행

```
from stack import *

def isValidPos(x,y):
    if x<0 or y<0 or x>=MAX_SIZE or y>=MAX_SIZE:
        return False
    else:
        return map[x][y]=='0' or map[x][y]=='x'


def DFS():
    stack = Stack()
    stack.push((1,0))
    print('DFS: ')

    while not stack.isEmpty():
        here = stack.pop()
        print(here, end='->')
        (x,y) = here

        if map[x][y] == 'x':
            return True
        else:
            map[x][y] = ','
            if isValidPos(x-1, y): stack.push((x-1,y))
            if isValidPos(x+1, y): stack.push((x+1,y))
            if isValidPos(x, y-1): stack.push((x,y-1))
            if isValidPos(x, y+1): stack.push((x,y+1))

        print("현재 스택 : ", stack)

    return False


map = [ ['1','1','1','1','1','1'],
        ['0','0','0','0','0','1'],
        ['1','0','1','0','1','1'],
        ['1','1','1','0','0','x'],
        ['1','1','1','0','1','1'],
        ['1','1','1','1','1','1'] ]

MAX_SIZE = 6

result = DFS()

if result:
    print(" --> 미로탐색 성공")
else:
    print(" --> 미로탐색 실패")
```

```
================== RESTART: /Users/inkyung/D
DFS:
(1, 0)->현재 스택 :  [(1, 1)]
(1, 1)->현재 스택 :  [(2, 1), (1, 2)]
(1, 2)->현재 스택 :  [(2, 1), (1, 3)]
(1, 3)->현재 스택 :  [(2, 1), (2, 3), (1, 4)]
(1, 4)->현재 스택 :  [(2, 1), (2, 3)]
(2, 3)->현재 스택 :  [(2, 1), (3, 3)]
(3, 3)->현재 스택 :  [(2, 1), (4, 3), (3, 4)]
(3, 4)->현재 스택 :  [(2, 1), (4, 3), (3, 5)]
(3, 5)-> --> 미로탐색 성공
>>>
```

5. 회문 체크 구현 및 실행

```
from stack import *

def palindrome(s):

    print("회문 체크용 문자열 : " + s)

    str1 = []
    str2 = []

    for x in s:
        if x.isalpha():
            str1.append(x.lower())
            str2.append(x.lower())

    while str1:
        if str1.pop(0) != str2.pop():
            print(s + " : 회문아님!!")
            print("=" * 30)
            return False
    print(s + " : 회문임!!")
    print("=" * 30)
    return True


s = "madam, I'm Adam"
palindrome(s)

s = "eye"
palindrome(s)

s = "race car"
palindrome(s)

s = "my name is Hong"
palindrome(s)
```

```
>>>
=============== RESTART: /Users/inkyung/Do
회문 체크용 문자열 : madam, I'm Adam
madam, I'm Adam : 회문임!!
==============================
회문 체크용 문자열 : eye
eye : 회문임!!
==============================
회문 체크용 문자열 : race car
race car : 회문임!!
==============================
회문 체크용 문자열 : my name is Hong
my name is Hong : 회문아님!!
==============================
>>>
```