프로그래밍 과제 3

이상윤 201802529

컴퓨터전자시스템공학전공

제출 일자 : 2022.04.01

1.

(1) 문제 기술

한 줄로 입력된 괄호들 (,),{,},[,]에 대하여, 모든 괄호가 짝이 맞는 경우 1을 출력하고, 그렇지 않으면 0을 출력한다.

(2) 코드

```
class Stack():
    # Stack 을 클래스로 미리 구현하였다.
    def __init__(self):
        self.items = []

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

    def pop(self):
        try:
            return self.items.pop()
        except IndexError:
            print("Stack is empty");

    def top(self):
        try:
            return self.items[-1]
        except IndexError:
            print("Stack is empty")

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

    def empty(self):
        return not self.items
        # return not self.items


def check(string):
    # 입력받은 문자열 내 괄호들이 모두 짝이 맞는지 검사하는 함수이다.
    # 짝이 모두 맞으면 1 을, 그렇지 않으면 0 을 반환한다.

    for elem in string:
        # 여는 괄호들은 스택에 삽입한다.
        if elem == '(' or elem =='{' or elem == '[':
            s.push(elem)
```

```
        # 닫는 괄호들은 우선 스택의 맨 위 요소를 확인한다.
        elif elem == ')' or elem == '}' or elem == ']':
            if s.empty():
                # 만일 스택이 비어있어 맨 위 요소를 확인할 수 없는 경우,
                # 현재 탐색 중인 닫는 괄호의 짝이 없다는 의미이므로 error 이다.
                return 0
            else:
                # 스택에 요소가 있어 맨 위 요소를 확인할 수 있는 경우
                # 맨 위 요소가 현재 탐색 중인 닫는 괄호와 맞는 짝이면, 이를 pop 한다.
                peek_ch = s.pop()
                if (elem == ')' and peek_ch !='(') or (elem == '}' and
peek_ch != '{') or (elem == ']' and peek_ch != '['):
                    return 0
        else:
            # 괄호가 아니라면 무시
            pass

    # 문자열 탐색이 모두 끝났다.

    if s.empty():
        # 만일 스택에 남은 요소가 없다면, 모든 괄호의 짝이 맞았다는 뜻이다.
        return 1
    else:
        # 만일 스택에 요소가 남아있다면, 이들은 모두 짝이 맞지 않는 여는 괄호
(,{,[란 의미이다. (error 케이스)
        return 0


string = input()  # 문자열을 입력받는다.
s = Stack()  # 스택 객체 s 를 선언한다.

ans = check(string)  # 입력받은 문자열을 check 함수를 통해 판정한다.

# 결과값을 출력한다.
print(ans)
```

(3) 자료구조 및 알고리즘 설명

모든 괄호가 짝이 맞는지 알아보기 위하여 스택을 활용합니다.

우선 문자열을 입력 받고 앞에서부터 하나씩 탐색합니다. 탐색한 값이 여는 괄호이면 이를 스택에 넣고, 닫는 괄호이면 스택에서 요소를 pop합니다. 이 때, pop한 요소(여는 괄호)가 현재 탐색중인 닫는 괄호와 짝이 맞아야 합니다. 그렇지 않으면 괄호의 짝이 맞지 않는 것입니다.

모든 괄호의 탐색이 정상적으로 끝나면, 스택 역시 비어있어야 합니다. 만일 스택에 여는 괄호가 남아있으면, 그 여는 괄호는 짝이 없는 괄호이기 때문입니다.

(4) 느낀점

괄호 문자열 문제는 스택을 활용하는 대표적인 문제로 유명하지만, 여러 종류의 괄호가 들어있는 문제는 이번 기회에 처음으로 접했습니다. 괄호의 종류가 여러가지인 문자열을 받는 경우, 스택에서 pop시 같은 종류의 괄호인지도 추가적으로 확인해야 한다는 점에서 조금 까다로웠던 문제였습니다.

2.

(1) 문제 기술

한 줄로 입력된 괄호들 (,),{,},[,]에 대하여, 모든 괄호가 짝이 맞는 경우 1을 출력하고, 그렇지 않으면 괄호검사 알고리즘에 의하여 **처음으로 발견된 (짝을 찾지 못한 괄호) 오류에 대하여, 입력 문자열에서 이 괄호의 위치(인덱스)를 출력한 후 다음 중 하나를 출력한다**.

)에 대응하는 (가 없을 경우: error1

}에 대응하는 {가 없을 경우: error2

]에 대응하는 [가 없을 경우: error3

(에 대응하는)가 없을 경우: error4

{에 대응하는 }가 없을 경우: error5

[에 대응하는]가 없을 경우: error6

(2) 코드

```
class Stack():
    # Stack 을 클래스로 미리 구현하였다.
    def __init__(self):
        self.items = []

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

    def pop(self):
        try:
```

```
            return self.items.pop()
        except IndexError:
            print("Stack is empty");

    def top(self):
        try:
            return self.items[-1]
        except IndexError:
            print("Stack is empty")

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

    def empty(self):
        return not self.items
        # return not self.items


def check(string):
    # 입력받은 문자열 내 괄호들이 모두 짝이 맞는지 검사하는 함수이다.
    # error_idx 값과 error_num 값, 총 2 개의 반환값을 갖는다.

    global error_idx, error_num
    # error_idx : 에러가 처음 발생한 괄호의 인덱스

    # error_num : error 번호
    # error_num == 0 : 모든 괄호가 짝이 맞음 (error 아님)
    # error_num == 1 : )에 대응하는 (가 없을 경우
    # error_num == 2 : }에 대응하는 {가 없을 경우
    # error_num == 3 : ]에 대응하는 [가 없을 경우
    # error_num == 4 : (에 대응하는 )가 없을 경우
    # error_num == 5 : {에 대응하는 }가 없을 경우
    # error_num == 6 : [에 대응하는 ]가 없을 경우


    # 입력받은 문자열을 순차적으로 탐색한다.
    for i, elem in enumerate(string):
        # 여는 괄호들을 스택에 삽입한다.
        # 이 때 그 괄호가 어느 위치의 괄호인지 알 수 있도록, 인덱스 값과 함께 튜플로
묶어 스택에 삽입한다.
        if elem == '(' or elem =='{' or elem == '[':
            s.push((i, elem))

        # 닫는 괄호들은 우선 스택의 맨 위 요소를 확인한다.
        elif elem == ')' or elem == '}' or elem == ']':
            # 만일 스택이 비어있어 맨 위 요소를 확인할 수 없는 경우,
            # 현재 탐색 중인 닫는 괄호의 짝이 없다는 의미이므로 error 이다.
            if s.empty():
```

```
                if elem == ')':
                    error_num = 1
                    return i, error_num
                if elem == '}':
                    error_num = 2
                    return i, error_num
                if elem == ']':
                    error_num = 3
                    return i, error_num

            # 스택에 요소가 있어 맨 위 요소를 확인할 수 있는 경우
            else:
                # 맨 위 요소가 현재 탐색 중인 닫는 괄호와 맞는 짝이면, 이를 pop 한다.
                _, peek_ch = s.pop()

                # 그렇지 않으면 error 이다.
                if (elem == ')' and peek_ch !='('):
                    error_num = 1
                    return i, error_num
                if (elem == '}' and peek_ch != '{'):
                    error_num = 2
                    return i, error_num
                if (elem == ']' and peek_ch != '['):
                    error_num = 3
                    return i, error_num
                # 위의 세 경우 모두 아니라면 정상적으로 pop 수행

        else:
            # 탐색 중인 요소가 괄호가 아니라 공백 or 다른 문자일 경우 그냥 무시한다.
            pass

    # 문자열 탐색이 모두 끝났다.

    # 만일 스택에 남은 요소가 없다면, 모든 괄호의 짝이 맞았다는 뜻이다.
    if s.empty():
        return None, 0

    # 만일 스택에 요소가 남아있다면, 이들은 모두 짝이 맞지 않는 여는 괄호 (,{,[ 란
의미이다. (error 케이스)
    else:
        # 이 중 스택의 가장 위에 있는 요소가 제일 먼저 발견된, 짝 없는 괄호이다.
        idx, peek_ch = s.pop()

        if peek_ch == '(':
            error_num = 4
            return idx, error_num
        elif peek_ch == '{':
            error_num = 5
```

```
            return idx, error_num
        elif peek_ch == '[':
            error_num = 6
            return idx, error_num
        else: pass


string = input()  # 문자열을 입력받는다.
s = Stack()  # 스택 객체 s 를 선언한다.
error_idx = 0; error_num = 0

error_idx, error_num = check(string)  # 입력받은 문자열을 check 함수를 통해
판정한다.

# error_num == 0 : error 발생 X, 모든 괄호의 짝이 맞음, 1 을 출력한다.
if error_num == 0:
    print(1)

# error_num == 1~6 : error 발생, error_idx 값과 error_num 값을 출력한다.
else:
    print(f"{error_idx}  error{error_num}")
```

(3) 자료구조 및 알고리즘 설명

문제를 해결하는 핵심 알고리즘은 위 문제와 동일하나, 괄호 짝이 맞지 않는 에러가 발생한 경우 (여기서 '에러'는 콘솔에 등장하는 에러의 의미가 아닙니다), 이가 어떤 종류의 에러인지, 그리고 어느 위치에서 에러가 발생했는지 추가적으로 확인하는 작업이 필요합니다. 그러기 위해, 괄호 짝이 맞지 않는 에러가 발생했을 때 조건문을 통해 케이스를 나누고, 어떤 괄호의 짝이 맞지 않는지 확인 후 그 결과를 함께 출력합니다. 더불어 문자열을 탐색할 때도 for문에 enumerate 함수를 사용하여 탐색 문자의 인덱스 값을 파악하고 이를 활용함으로써 에러가 발생한 위치를 구합니다. 그리고 결과 출력 시 이 값을 함께 출력합니다.

(4) 느낀점

괄호 짝이 맞지 않을 경우 어떤 괄호의 짝이 맞지 않는지 일일이 조건을 나눠서 코드를 구현하는 작업이 생각보다 헷갈렸지만, 덕분에 이 문제를 푸는 알고리즘을 한층 더 깊이 이해할 수 있게 되었습니다.

3.

(1) 문제 기술

여러 줄로 입력된 괄호들 (,),{,},[,]에 대하여 괄호가 짝이 맞는 경우 1을 출력하고, 그렇지 않으면 괄호검사 알고리즘에 의하여 처음으로 발견된 (짝을 찾지 못한 괄호) 오류에 대하여, 다음 중 하나를 출력한다. (다음에서 xx와 yy는 양의 정수임)

)에 대응하는 (가 없을 경우 error 1:) at position yy in line xx

}에 대응하는 {가 없을 경우 error 2: } at position yy in line xx

]에 대응하는 [가 없을 경우 error 3:] at position yy in line xx

(에 대응하는)가 없을 경우 error 4: (at position yy in line xx

{에 대응하는 }가 없을 경우 error 5: { at position yy in line xx

[에 대응하는]가 없을 경우 error 6: (at position yy in line xx

(2) 코드

```
import sys


class Stack():
    # Stack 을 클래스로 미리 구현하였다.
    def __init__(self):
        self.items = []

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

    def pop(self):
        try:
            return self.items.pop()
        except IndexError:
            print("Stack is empty");

    def top(self):
        try:
            return self.items[-1]
        except IndexError:
            print("Stack is empty")
```

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

    def empty(self):
        return not self.items
        # return not self.items


def check(string):
    # 입력받은 문자열 내 괄호들이 모두 짝이 맞는지 검사하는 함수이다.
    # error_idx 값과 error_num 값, 총 2 개의 반환값을 갖는다.

    global error_idx, error_num
    # error_idx : 에러가 처음 발생한 괄호의 인덱스

    # error_num : error 번호
    # error_num == 0 : 모든 괄호가 짝이 맞음 (error 아님)
    # error_num == 1 : )에 대응하는 (가 없을 경우
    # error_num == 2 : }에 대응하는 {가 없을 경우
    # error_num == 3 : ]에 대응하는 [가 없을 경우
    # error_num == 4 : (에 대응하는 )가 없을 경우
    # error_num == 5 : {에 대응하는 }가 없을 경우
    # error_num == 6 : [에 대응하는 ]가 없을 경우


    # 입력받은 문자열을 순차적으로 탐색한다.
    for i, elem in enumerate(string):
        # 여는 괄호들을 스택에 삽입한다.
        # 이 때 그 괄호가 어느 위치의 괄호인지 알 수 있도록, 인덱스 값과 함께 튜플로
묶어 스택에 삽입한다.
        if elem == '(' or elem =='{' or elem == '[':
            s.push((i, elem))

        # 닫는 괄호들은 우선 스택의 맨 위 요소를 확인한다.
        elif elem == ')' or elem == '}' or elem == ']':
            # 만일 스택이 비어있어 맨 위 요소를 확인할 수 없는 경우,
            # 현재 탐색 중인 닫는 괄호의 짝이 없다는 의미이므로 error 이다.
            if s.empty():
                if elem == ')':
                    error_num = 1
                    return i, error_num, ')'

                if elem == '}':
                    error_num = 2
                    return i, error_num, '}'

                if elem == ']':
```

```
                    error_num = 3
                    return i, error_num, ']'

            # 스택에 요소가 있어 맨 위 요소를 확인할 수 있는 경우
            else:
                # 맨 위 요소가 현재 탐색 중인 닫는 괄호와 맞는 짝이면, 이를 pop 한다.
                _, peek_ch = s.pop()

                # 그렇지 않으면 error 이다.
                if (elem == ')' and peek_ch !='('):
                    error_num = 1
                    return i, error_num, ')'

                if (elem == '}' and peek_ch != '{'):
                    error_num = 2
                    return i, error_num, '}'

                if (elem == ']' and peek_ch != '['):
                    error_num = 3
                    return i, error_num, ']'
                # 위의 세 경우 모두 아니라면 정상적으로 pop 수행

        else:
            # 탐색 중인 요소가 괄호가 아니라 공백 or 다른 문자일 경우 그냥 무시한다.
            pass

    # 문자열 탐색이 모두 끝났다.

    # 만일 스택에 남은 요소가 없다면, 모든 괄호의 짝이 맞았다는 뜻이다.
    if s.empty():
        return None, 0, 0

    # 만일 스택에 요소가 남아있다면, 이들은 모두 짝이 맞지 않는 여는 괄호 (,{,[ 란
의미이다. (error 케이스)
    else:
        # 이 중 스택의 가장 위에 있는 요소가 제일 먼저 발견된, 짝 없는 괄호이다.
        idx, peek_ch = s.pop()

        if peek_ch == '(':
            error_num = 4
            return idx, error_num, '('
        elif peek_ch == '{':
            error_num = 5
            return idx, error_num, '{'
        elif peek_ch == '[':
            error_num = 6
            return idx, error_num, '['
        else: pass
```

```
str_len = []  # 입력 받은 여러 줄의 문자열의 각 줄 길이를 저장하는 리스트이다.
long_str = ""  # 입력 받은 여러 줄의 문자열을 하나의 긴 문자열로 합쳐 저장할 변수이다.
line = 1  # 문자열을 탐색할 때 몇 번째 줄인지 확인하기 위한 용도의 변수이다.


# 빈 문자열이 입력될 때까지 문자열을 줄 단위로 입력 받아 저장한다.
while True:
    string = sys.stdin.readline().strip()  # 문자열을 입력받는다. (공백 제거)

    if not string:  # 빈 문자열을 입력 받으면 그만 입력 받는다.
        break

    str_len.append(len(string))  # 각 줄의 문자열의 길이를 리스트로 저장한다.
    long_str += string  # 입력 받은 여러 줄의 문자열을 순서대로 합친다.


s = Stack()  # 스택 객체 s 를 선언한다.
error_idx = 0; error_num = 0

error_idx, error_num, error_ch = check(long_str)  # 입력받은 문자열을 check 함수를 통해 판정한다.

# error_num == 0 : error 발생 X, 모든 괄호의 짝이 맞음, 1 을 출력한다.
if error_num == 0:
    print(1)

# error_num == 1~6 : error 발생, error_idx 값과 error_num 값을 출력한다.
# 이 때,
else:
    error_idx += 1  # 인덱스와 실제 문자열의 위치가 1 만큼 다르므로, position 결괏값에 1 을 더해준다.

    # 에러가 발생한 위치를 계산한다.
    # error_idx 값은 여러 문자열을 하나로 합쳤을 때의 에러 발생 위치이므로,
    # 이를 line 번째 줄, position 번째 위치로 변환해야 한다.
    for elem in str_len:
        if elem >= error_idx:
            break
        error_idx -= elem
        line += 1

    # 계산한 결과값을 출력한다.
    print(f"error {error_num}: {error_ch} at position {error_idx} in line {line}")
```

(3) 자료구조 및 알고리즘 설명

문제 해결의 핵심 알고리즘은 역시 동일하지만 여러 줄의 입력을 처리해야 하므로, 입력 받은 각 줄의 문자열을 하나의 새 문자열로 합친 후 괄호 짝이 맞는지 확인합니다. 결과를 출력할 땐, 에러가 발생한 위치를 n번째 줄 m번째 글자로 출력해야 하므로, 입력 받은 문자열의 각 줄의 길이를 에러가 발생한 위치 값에서 순서대로 빼며 에러가 발생한 위치를 찾습니다.

(4) 느낀점

처리해야 할 문자열이 여러 줄에 나누어 들어온다는 점과, 에러가 발생한 위치를 이전과 다르게 나타내야 한다는 점, 이 두 가지 고려사항이 추가되었을 뿐 임에도 이유를 알 수 없는 오답으로 인해 결국 일부 테스트케이스를 통과하지 못한 채 과제를 제출하게 되었습니다. 이로 인해 큰 아쉬움을 느꼈고, 제 프로그래밍 실력이 많이 부족함을 다시 한 번 깨달았습니다. 그래도 VS code를 통해 제가 작성한 코드를 계속 디버깅 해보면서, 코드가 어떤 논리로 동작하는 지 좀 더 자세히 알 수 있었습니다.