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():

*# Stack을 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 1을, 그렇지 않으면 0을 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 error이다.*

*return* 0

*else*:

*# 스택에 요소가 있어 맨 위 요소를 확인할 수 있는 경우*

*# 맨 위 요소가 현재 탐색 중인 닫는 괄호와 맞는 짝이면, 이를 pop한다.*

                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 s를 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. 여는 괄호면 이를 스택에 넣고, 닫는 괄호면 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 괄호 인지도 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():

*# Stack을 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\_idx값과 error\_num값, 총 2개의 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 떄 그 괄호가 어느 위치의 괄호인지 알 수 있도록, 인덱스 값과 함꼐 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 error이다.*

*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 pop한다.*

                \_, peek\_ch = s.pop()

*# 그렇지 않으면 error이다.*

*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 아나리면 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 s를 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, 1을 출력한다.*

*if* error\_num == 0:

    print(1)

*# error\_num == 1~6 : error 발생, error\_idx값과 error\_num값을 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, 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 일일이 확인 후 그 결과를 함께 출력합니다. 더불어 문자열을 탐색할 때도 for문에 enumerate 함수를 사용하여 문자의 인덱스 값을 활용함으로써 에러의 발생 위치 정보를 저장하고 함께 결과로 출력해줍니다.

(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 한 층 더 깊이 이해할 수 있었습니다.

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

(3) 자료구조 및 알고리즘 설명

(4) 느낀점