

컴파일러 과제 2

Syntax analyzer 구현

Team 11

소프트웨어학과 20194653 윤다인

소프트웨어학과 20194863 이승연

목차

1. CFG
2. SLR parsing table
3. lexical.py 수정
4. syntax.py 코드 설명
5. test case

1. CFG

1. CODE \rightarrow VDECL CODE | FDECL CODE | CDECL CODE | ϵ
2. VDECL \rightarrow vtype id semi | vtype ASSIGN semi
3. ASSIGN \rightarrow id assign RHS
4. RHS \rightarrow EXPR | literal | character | boolstr
5. EXPR \rightarrow FACTOR addsub EXPR | FACTOR
6. FACTOR \rightarrow TERM multdiv FACTOR | TERM
7. TERM \rightarrow lpren EXPR rpren | id | num
8. FDECL \rightarrow vtype id lpren ARG rpren lbrace BLOCK RETURN rbrace
9. ARG \rightarrow vtype id MOREARGS | ϵ
10. MOREARGS \rightarrow comma vtype id MOREARGS | ϵ
11. BLOCK \rightarrow STMT BLOCK | ϵ
12. STMT \rightarrow VDECL | ASSIGN semi | if lpren COND rpren lbrace BLOCK rbrace ELSE | while lpren COND rpren lbrace BLOCK rbrace
13. COND \rightarrow BOOL comp BOOL | BOOL
14. BOOL \rightarrow boolstr
15. ELSE \rightarrow else lbrace BLOCK rbrace | ϵ
16. RETURN \rightarrow return RHS semi
17. CDECL \rightarrow class id lbrace ODECL rbrace
18. ODECL \rightarrow VDECL ODECL | FDECL ODECL | ϵ

1. EXPR 부분을 FACTOR와 TERM을 추가해 multdiv와 addsub 간의 우선순위를 표현한다.
2. COND 부분을 BOOL을 활용해 수정한다. a conditional operation which consists of Boolean strings and an condition operator. 부분에서 알 수 있듯이, true/false의 boolean string이나 boolean string과 condition operator를 활용한 표현이 조건문에 해당한다. 따라서 COND를 BOOL을 활용해 수정한다.

2. SLR parsing table

```
FINAL -> CODE
CODE -> VDECL CODE
CODE -> FDECL CODE
CODE -> CDECL CODE
CODE -> ''
VDECL -> vtype id semi
VDECL -> vtype ASSIGN semi
ASSIGN -> id assign RHS
RHS -> EXPR
RHS -> literal
RHS -> character
RHS -> boolstr
EXPR -> FACTOR addsub EXPR
EXPR -> FACTOR
FACTOR -> TERM multdiv FACTOR
FACTOR -> TERM
TERM -> lparen EXPR rparen
TERM -> id
TERM -> num
FDECL -> vtype id lparen ARG rparen lbrace BLOCK RETURN rbrace
ARG -> vtype id MOREARGS
ARG -> ''
MOREARGS -> comma vtype id MOREARGS
MOREARGS -> ''
BLOCK -> STMT BLOCK
BLOCK -> ''
STMT -> VDECL
STMT -> ASSIGN semi
STMT -> if lparen COND rparen lbrace BLOCK rbrace ELSE
STMT -> while lparen COND rparen lbrace BLOCK rbrace
COND -> BOOL comp BOOL
COND -> BOOL
BOOL -> boolstr
ELSE -> else lbrace BLOCK rbrace
ELSE -> ''
RETURN -> return RHS semi
CDECL -> class id lbrace ODECL rbrace
ODECL -> VDECL ODECL
ODECL -> FDECL ODECL
ODECL -> ''
```

과제설명파일에 있는 사이트에 위의 cfg를 입력하여 slr table을 구하였다. (Table은 엑셀 파일 첨부 하였습니다.)

맨 처음에 FINAL을 추가한 이유는 FINAL을 추가하지 않으면 맨 처음 CODE에서 VDECL만 인식하고 CDECL을 인식하지 못하였다. 이 사이트에서는 줄바꿈이 | 의 역할을 수행하는데, FINAL을 추가하지 않으면 인식이 되지 않았다.

3. lexical.py 수정

1. operator -> addsub/ multdiv

```
elif state in [3,4]:  
    if lex in ['+', '-']:  
        return "addsub"  
    else:  
        return "multdiv"
```

각자 token을 서로 다른 것으로 나눴다.

2. boolstr 추가

```
elif state in [173,178]: # true, false  
    return "boolstr"
```

토큰 결정하는 함수에서 true 와 false일 때 token을 boolstr로 리턴하게 수정하였다.

3. output 파일 출력하는 형식

```
for i in range(len(fin)):  
    line = ""  
    #line= line + fin[i][1] + ' , ' + fin[i][0]  
  
    out.write(fin[i][1] + " ")  
    #out.write("\n")  
out.close()
```

Token 의 값만 입력하게 코드를 고쳤다.

4. syntax.py 코드 설명

1. 실행 방법

```
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> python our_lexical.py test5.java
output file - test5_output.txt - is created
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> python our_syntax.py test5_output.txt
access
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> █
```

Java 파일을 lexical.py 를 이용하여 token 별로 parsing 한다. 그리고 그 output 파일을 syntax.py 의 입력으로 사용한다.

Accept 가 되면 access 가, error 가 발견되면 error report가 나온다.

(각각의 케이스는 뒤에서 다시 설명한다.)

2. 코드 설명

1) 변수선언

SLR parsing한 결과를 코드에 딕셔너리의 형태로 입력해준다.

before_reduce는 reduce 중 우변을 의미하고 after_reduce는 좌변을 의미한다.

예를 들어

r2: CODE -> VDECL CODE 를 표현할 때,

좌변 CODE는 after_reduce에서 r2을 키로 찾을 수 있다.

우변 VDECL CODE는 before_reduce에서 r2을 키로 찾을 수 있다.

goto_table과 action_table도 딕셔너리의 형태로 저장해준다.

2) 전반적인 코드 흐름

syntax 함수가 syntax analyzer 역할을 구현한다.

인자로 받은 token의 경우, lexical analyzer를 실행하고 난 결과를 main 에서 배열로 저장한 것을 의미한다.

전체적인 코드 흐름은 자료구조로 stack을 사용한다. stack에서 꺼내온 결과와 token 배열의 값을 활용해 action_table에서 그에 맞는 action 값(예를 들어 reduce나 shift)을 가져온다. 이때 token 배열의 값은 0부터 하나씩 증가해나간다. 즉, token 배열을 index 0부터 하나씩 증가시키며 참조하고 그때마다 stack의 원소(가장 마지막에 들어간 원소)를 참조하면서 취할 action을 찾는다.

Action을 찾기 위하여 try except 문을 사용하였고, 나올 수 있는 action의 경우는 다음과 같다.

1. action이 shift인 경우
2. action이 reduce인 경우
3. action이 acc인 경우
4. action을 찾지 못한 경우.

shift일 경우, token 값이랑 action에서 s를 제외한 숫자를 stack에 넣어준다. 예를 들어 s5일 경우, s는 제거하고 5를 stack에 넣는다.

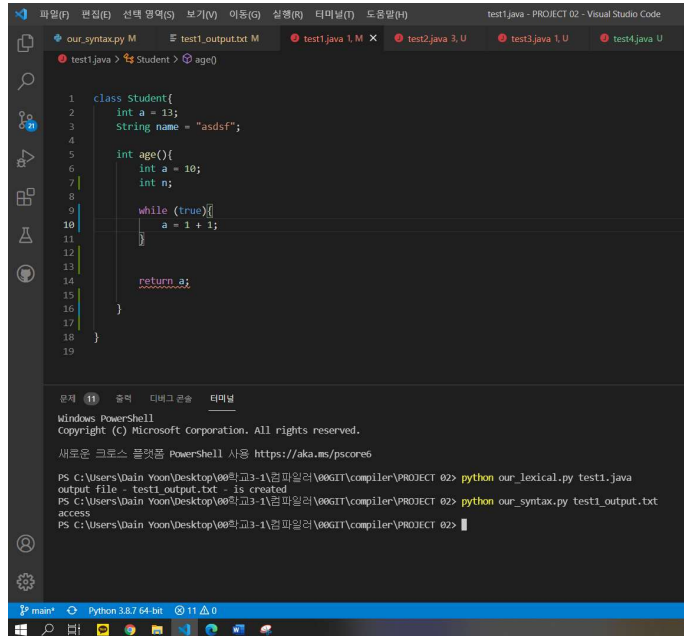
reduce의 경우, after_reduce와 before_reduce에서 각각 해당 reduce에 해당하는 action을 조회해 찾아온다. 그리고 reduce 과정을 수행한다. 다음은 reduce 된 after reduce 값을 stack에 넣어주고 goto 값도 stack에 넣어준다.

acc일 경우, 함수가 True 값을 반환하도록 하고 대응하는 action 값이 없을 경우, False를 반환한다.

main 함수에서는 lexical analyzer의 결과값인 token이 vtype이나 class로 시작하지 않으면 CODE \rightarrow VDECL CODE | FDECL CODE | CDECL CODE | ϵ 을 만족하지 않기 때문에 해당 경우에 대해 예외 처리를 해주고 syntax 함수를 작동시킨다.

5. test case

1) 제대로 access 되는 경우



The screenshot shows the Visual Studio Code editor with a Java file named `test1.java` open. The code defines a `Student` class with an `age()` method that increments a variable `a` in a loop and returns it. The terminal shows the execution of the program using `python our_lexical.py test1.java`, which creates an output file `test1_output.txt`. Subsequent commands `python our_syntax.py test1_output.txt` and `access` are executed successfully, indicating that the access operation worked as expected.

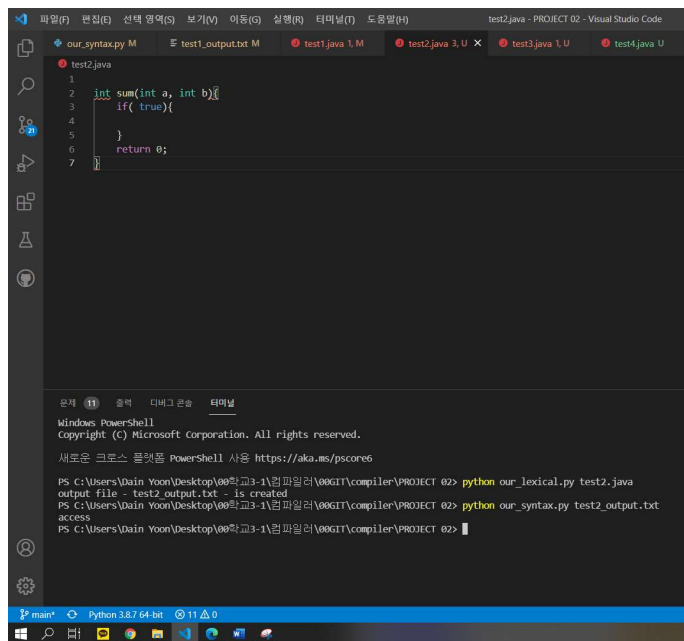
```
1 class Student{
2     int a = 13;
3     String name = "asdfs";
4
5     int age(){
6         int a = 10;
7         int n;
8
9         while (true){
10            a = 1 + 1;
11        }
12
13        return a;
14    }
15 }
16
17
18 }
```

문제 (1) 출력 디버그 콘솔 터미널

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>

PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> python our_lexical.py test1.java
output file - test1_output.txt - is created
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> python our_syntax.py test1_output.txt
access
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> |



The screenshot shows the Visual Studio Code editor with a Java file named `test2.java` open. The code defines a `sum` method that takes two integers `a` and `b` and returns their sum. The terminal shows the execution of the program using `python our_lexical.py test2.java`, which creates an output file `test2_output.txt`. Subsequent commands `python our_syntax.py test2_output.txt` and `access` are executed successfully, indicating that the access operation worked as expected.

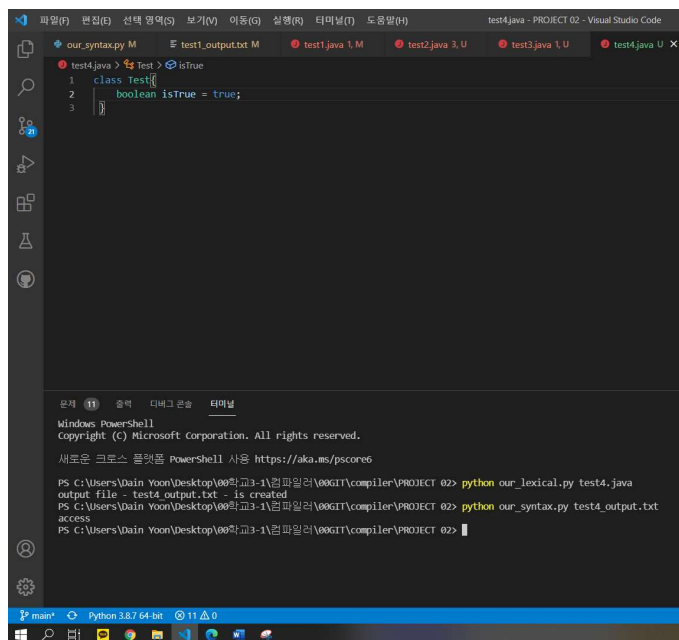
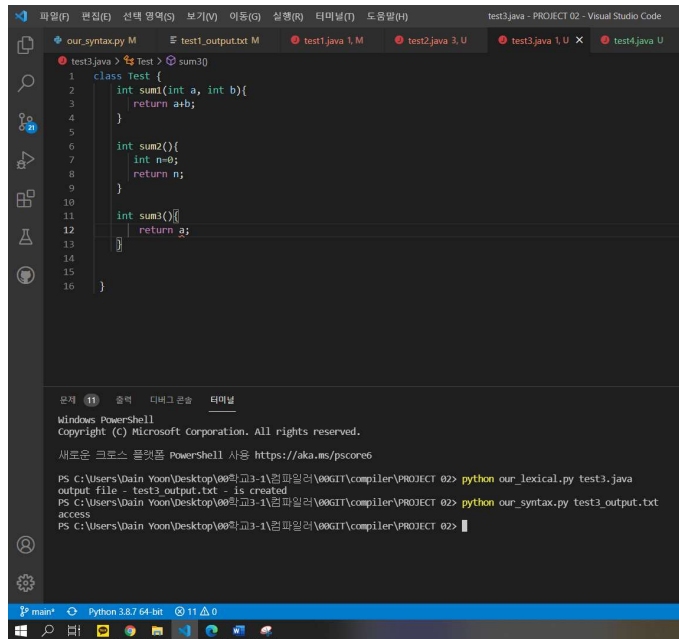
```
1
2 int sum(int a, int b){
3     if( true){
4
5     }
6     return 0;
7 }
```

문제 (1) 출력 디버그 콘솔 터미널

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>

PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> python our_lexical.py test2.java
output file - test2_output.txt - is created
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> python our_syntax.py test2_output.txt
access
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> |



```
test3.java - PROJECT 02 - Visual Studio Code
our_syntax.py M test1_output.txt M test1.java 1.M test2.java 3.U test3.java 1.U test4.java U
1 test3.java > % Test > sum3()
2 String b="1234";
3 char c='a';
4
5 boolean isTrue=true;
6
7 class Test {
8     int sum1(int a, int b){
9         return a+b;
10    }
11
12    int sum2(){
13        int n=0;
14        return n;
15    }
16
17    int sum3(){}
18    return 0;
19 }
20
21 }

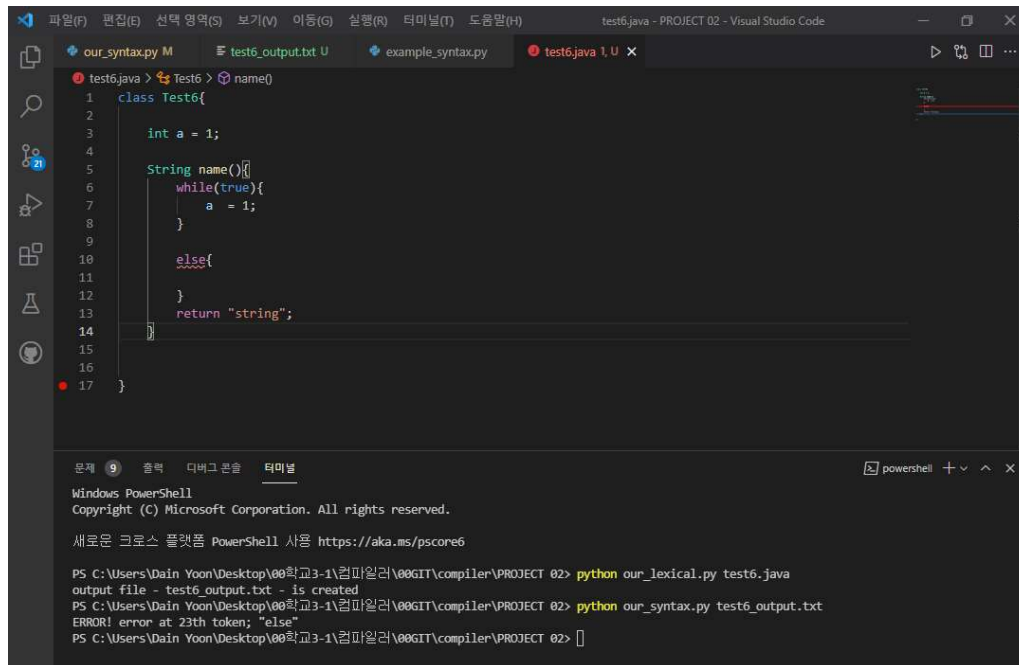
문제 ⑤ 출력 디버그 콘솔 터미널
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 https://aka.ms/powershell

PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> python our_lexical.py test5.java
output file - test5_output.txt - is created
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02> python our_syntax.py test5_output.txt
access
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\PROJECT 02>
```

2) error뜨는 경우

에러가 뜨면 다음과 같이 어떤 토큰 위치에서 에러가 발생했는지 알려준다.



```
our_syntax.py M test6_output.txt U example_syntax.py test6.java 1, U X
test6.java > Test6 > name()
1 class Test6{
2
3     int a = 1;
4
5     String name(){
6         while(true){
7             a = 1;
8         }
9
10        else{
11        }
12    }
13    return "string";
14
15
16
17 }
```

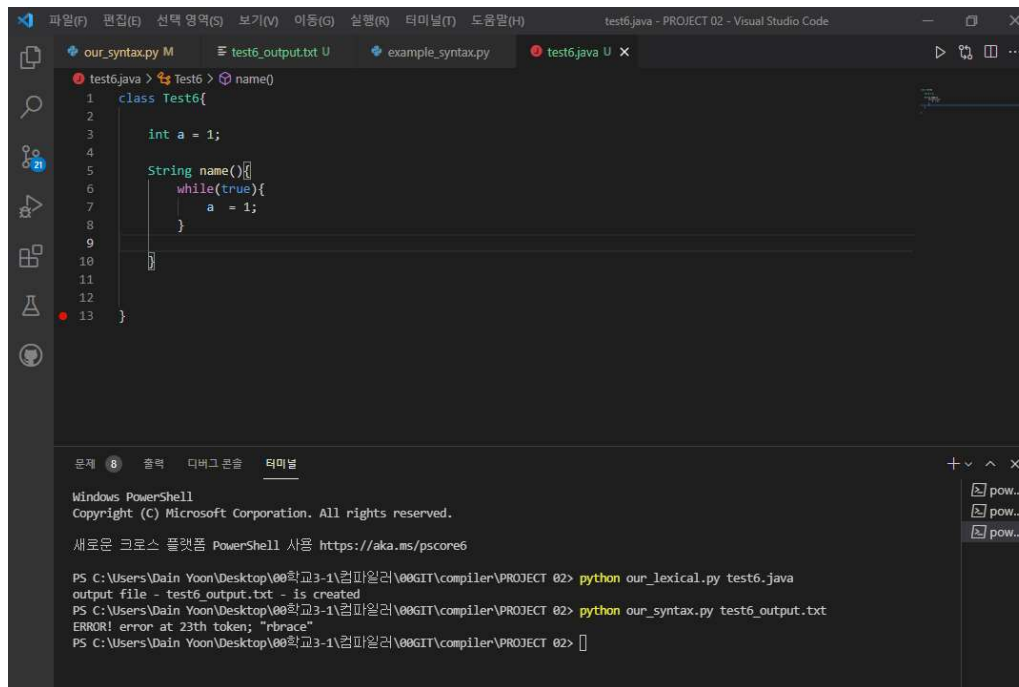
문제 9 출력 디버그 콘솔 터미널 powershell + ^ x

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>

PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> python our_lexical.py test6.java
output file - test6_output.txt - is created
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> python our_syntax.py test6_output.txt
ERROR! error at 23th token; "else"
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02>

Else가 단독으로 나온 경우



```
our_syntax.py M test6_output.txt U example_syntax.py test6.java U X
test6.java > Test6 > name()
1 class Test6{
2
3     int a = 1;
4
5     String name(){
6         while(true){
7             a = 1;
8         }
9
10    }
11
12
13 }
```

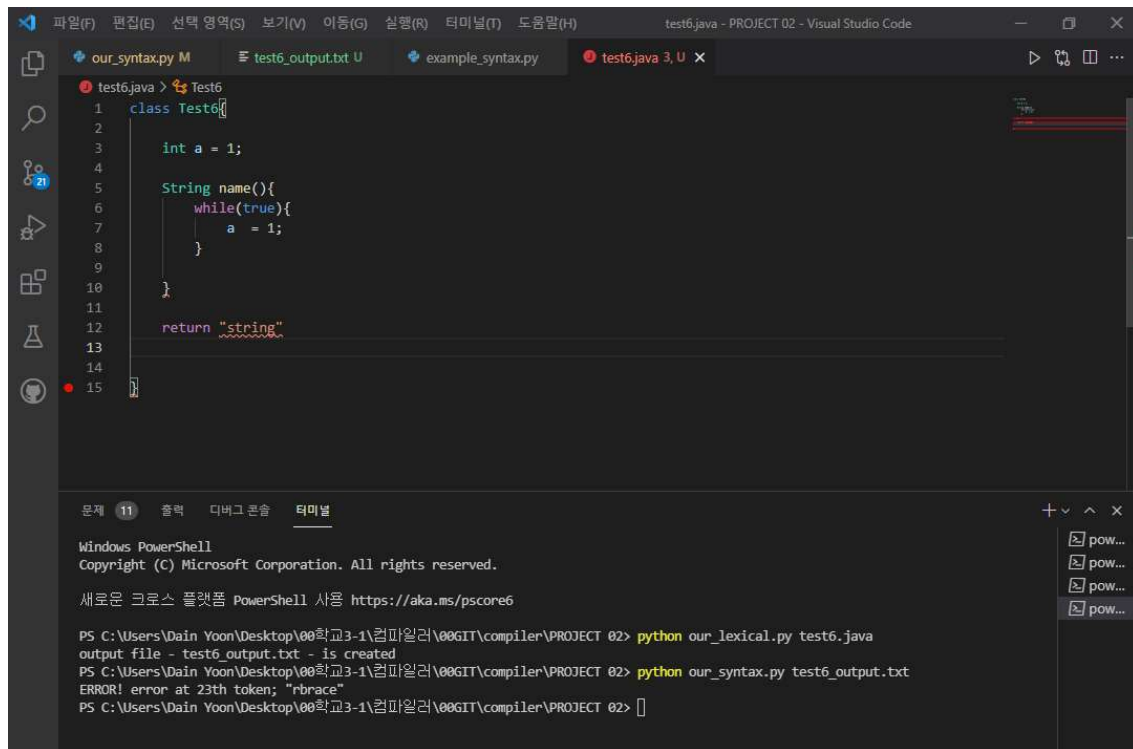
문제 8 출력 디버그 콘솔 터미널 pow... pow... pow...

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

새로운 크로스 플랫폼 PowerShell 사용 <https://aka.ms/pscore6>

PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> python our_lexical.py test6.java
output file - test6_output.txt - is created
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02> python our_syntax.py test6_output.txt
ERROR! error at 23th token; "rbrace"
PS C:\Users\Dain Yoon\Desktop\00학교3-1\컴파일러\00GIT\compiler\PROJECT 02>

함수에 리턴이 없는 경우



세미콜론이 없는 경우