

CAU PL - LL(1) parser - External Document (1)

20224566 김동우

LL(1)파서 제작 과제의 External Document 이다

Dependency

anytree 설치법

pip3 install anytree

또는

pip install anytree

-v 옵션을 사용하지 않을때 실행방법 예시

python3 main.py test.txt

또는

python main.py test.txt

-v 옵션을 사용할때 실행방법 예시

python3 main.py -v test.txt

또는

python main.py -v test.txt

Error List - Lexer.py

error case 1 왼쪽 괄호가 식별자 직후에 등장하는 경우

출력되는 오류 메시지

(Error) There is left parenthesis after identifier

오류 메시지가 출력되는 상황

현재 인식된 연산자가 (이고 before_token이 TokenType.IDENT일경우

예시 입력

a := 10;

b := a(1+1)

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

a := 10;

ID: 1; CONST: 1; OP: 0;

(OK)

b := a(1+1)

ID: 2; CONST: 2; OP: 1;

(Error) There is left parenthesis after identifier

Result ==> a: 10; b: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - 오류로 인해 파싱은 종료되었지만 Lexer는 계속 동작하므로 (이후의 부분도 출력된다.

1 (IDENT)

3 (ASSIGN_OP)

2 (CONST)

4 (SEMI_COLON)

1 (IDENT)

3 (ASSIGN_OP)

1 (IDENT)

7 (LEFT_PAREN)

2 (CONST)

5 (ADD_OP)

2 (CONST)

8 (RIGHT_PAREN)

9 (END)

error case 2 대입 연산자이후 연산자가 등장하는 경우

출력되는 오류 메시지

(Error) Operator(operator or right_paren, semi_colon, assign_op) after assignment operator

예시 입력

a := + 1

오류 메시지가 출력되는 상황

:= 다음에 +/*:=같은 문자들이 나오는 경우

위의 예시 입력에 대한 -v옵션이 없을 때의 예시 출력

a := + 1

ID: 1; CONST: 1; OP: 1;

(Error) Operator(operator or right_paren, semi_colon, assign_op) after assignment operator

Result ==> a: Unknown;

위의 예시 입력에 대한 -v옵션이 있을 때의 예시 출력

1 (IDENT)

3 (ASSIGN_OP)

5 (ADD_OP)

2 (CONST)

9 (END)

error case 3-1 선언(식별자에 값이 정상적으로 할당)되지 않은 식별자가 사용된 경우

→ parser의 동작이 중지되었을때만 lexer에서 처리한다.

출력되는 오류 메시지

(Error) Using undeclared identifier(+ undeclared identifier +)

예시 입력

b := a

오류 메시지가 출력되는 상황

선언하지 않은 연산자가 사용되었거나 이전에 오류로 인해 값이 정상적으로 할당되지 않은 식별자가 사용되었을 경우

위의 예시 입력에 대한 -v옵션이 없을 때의 예시 출력

b := a

ID: 2; CONST: 0; OP: 0;

(Error) Undefined variable is referenced(a)

Result ==> b: Unknown; a: Unknown;

위의 예시 입력에 대한 -v옵션이 있을 때의 예시 출력

1 (IDENT)

3 (ASSIGN_OP)

1 (IDENT)

9 (EOF)

Warning List - Lexer.py

warning case 1 여러 개의 식별자가 연속해서 등장하는 경우

출력되는 오류 메시지

(Warning) Continuous identifiers - ignoring identifiers (+ continuous identifiers except the first one +)

오류 메시지가 출력되는 상황

next_token과 before_token이 TokenType.IDENT일경우

예시 입력

```
a := 10;  
b := 20;  
c := 30;  
d := a b c
```

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력 - b와 c가 무시되어 d의 결과가 a에 의해 10이 되었다.

```
a := 10;  
ID: 1; CONST: 1; OP: 0;  
(OK)  
b := 20;  
ID: 1; CONST: 1; OP: 0;  
(OK)  
c := 30;  
ID: 1; CONST: 1; OP: 0;  
(OK)  
d := a  
ID: 2; CONST: 0; OP: 0;  
(Warning) Continuous identifiers - ignoring identifiers(b)  
(Warning) Continuous identifiers - ignoring identifiers(c)  
Result ==> a: 10; b: 20; c: 30; d: 10;
```

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - b와 c가 무시되어 마지막 statement에서 assign_op 다음에 ident가 1회만 출력되었다.

```
1 (IDENT)  
3 (ASSIGN_OP)
```

2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
1 (IDENT)
9 (EOF)

warning case 2 여러 개의 상수가 연속해서 등장하는 경우

출력되는 오류 메시지

(Warning) Continuous constants - ignoring constants (+ continuous constants except the first one +)

예시 입력

a := 3.14159265358979 1 2 3 4 5 6

오류 메시지가 출력되는 상황

next_token과 before_token이 TokenType.CONST일 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력 - 1 2 3 4 5 6이 무시된 것을 확인할 수 있다.

a := 3.14159265358979

ID: 1; CONST: 1; OP: 0;

(Warning) Continuous constants - ignoring constants(1)

(Warning) Continuous constants - ignoring constants(2)

(Warning) Continuous constants - ignoring constants(3)

(Warning) Continuous constants - ignoring constants(4)

(Warning) Continuous constants - ignoring constants(5)

(Warning) Continuous constants - ignoring constants(6)

Result ==> a: 3.14159265358979;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - 1 2 3 4 5 6이 무시된 것을 확인할 수 있다.

1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
9 (EOF)

warning case 3 소수점이 여러 개인 소수가 등장하는 경우

출력되는 오류 메시지

(Warning) Multiple decimal points - ignoring decimal points and digits after the second decimal point(+ from second decimal point to end of constant +)

예시 입력

a := 3.1415.92.65.3.5.8.9.7

오류 메시지가 출력되는 상황

next_token이 TokenType.CONST이고 인식된 상수 토큰에 .이 포함(소수)이며 상수토큰 다음 문자가 .일 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력 - .92.65.3.5.8.9.7이 무시된 것을 확인할 수 있다.

a := 3.1415

ID: 1; CONST: 1; OP: 0;

(Warning) Multiple decimal points - ignoring decimal points and digits after the second decimal point(.92.65.3.5.8.9.7)

Result ==> a: 3.1415;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - .92.65.3.5.8.9.7이 무시된 것을 확인할 수 있다.

1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
9 (EOF)

warning case 4 :=가 :로 잘못 작성된 경우

출력되는 오류 메시지

(Warning) Using = instead of := ==> assuming :=

오류 메시지가 출력되는 상황

:가 Lexer에 의해 인식된 경우

예시 입력

a : 10

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력 - :가 :=로 처리되었다.

```
a := 10
ID: 1; CONST: 1; OP: 0;
(Warning) Using : instead of := ==> assuming :=
Result ==> a: 10;
```

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - :가 :=로 처리되었다.

```
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
9 (EOF)
```

warning case 5 :=가 =로 잘못 작성된 경우

출력되는 오류 메시지

```
(Warning) Using = instead of := ==> assuming :=
```

오류 메시지가 출력되는 상황

=가 Lexer에 의해 인식된 경우

예시 입력

```
a = 10
```

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력 - =가 :=로 처리되었다.

```
a := 10
ID: 1; CONST: 1; OP: 0;
(Warning) Using : instead of := ==> assuming :=
Result ==> a: 10;
```

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - =가 :=로 처리되었다.

```
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
9 (EOF)
```

warning case 6 여러 개의 연산자가 연속해서 등장하는 경우

출력되는 오류 메시지

```
(Warning) Using multiple operators(operator or left_paren) ==> ignoring multiple operators
except the first one(+ continuous operators except the first one + )
```

예시 입력

a := 1 +-* / 2

오류 메시지가 출력되는 상황

next_token과 before_token이 TokenType.ADD_OP나 TokenType.MULT_OP중 하나인 경우

위의 예시 입력에 대한 -v옵션이 없을 때의 예시 출력 - +뒤의 -*/ 가 무시된 것을 볼 수 있다.

a := 1 + 2

ID: 1; CONST: 2; OP: 1;

(Warning) Using multiple operators(operator or left_paren) ==> ignoring multiple operators except the first one(-*/)

Result ==> a: 3;

위의 예시 입력에 대한 -v옵션이 있을 때의 예시 출력 - +뒤의 -*/ 가 무시된 것을 볼 수 있다.

1 (IDENT)

3 (ASSIGN_OP)

2 (CONST)

5 (ADD_OP)

2 (CONST)

9 (EOF)

warning case 7-1 프로그램의 끝이 ;인 경우

→오류로 인해 Parser가 아닌 Lexer만 작동하는 상황에서만 Lexer.py에서 작동됨

출력되는 오류 메시지

```
(Warning) There is semicolon at the end of the program ==> ignoring semicolon
```

예시 입력

b := a + (1 + 1);

오류 메시지가 출력되는 상황

오류로 인해 Lexer만 작동중이며 program의 끝이 ;인 경우

위의 예시 입력에 대한 -v옵션이 없을 때의 예시 출력

b := a + (1 + 1)

ID: 2; CONST: 2; OP: 2;

(Error) Undefined variable is referenced(a)

(Warning) There is semicolon at the end of the program ==> ignoring semicolon

Result ==> b: Unknown; a: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - ;가 출력되지 않은 것을 볼 수 있다.

1 (IDENT)
3 (ASSIGN_OP)
1 (IDENT)
5 (ADD_OP)
7 (LEFT_PAREN)
2 (CONST)
5 (ADD_OP)
2 (CONST)
8 (RIGHT_PAREN)
4 (SEMI_COLON)
9 (EOF)

Error List - Parser.py

error case 4 입력받은 소스코드가 공백으로만 이루어진 경우

→ 주어진 문법은 공백을 생성할 수 없다. <statement>는 소스코드에서 1개 이상 등장할 수 밖에 없으며 <statement> → <ident><assignment_op><expression> 에서 <statement>는 정상적인 상황이라면 공백으로만 이루어질 수 없다.

출력되는 오류 메시지

(Error) Grammer of this LL(1) parser cannot generate empty source code

예시 입력

오류 메시지가 출력되는 상황

소스코드로 공백이 입력된 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

ID: 0; CONST: 0; OP: 0;

(Error) Grammer of this LL(1) parser cannot generate empty source code

Result ==> There is no identifier

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력

9 (EOF)

error case 5 Syntax Error - 나머지 에러들에 해당하지 않는 애러

출력되는 오류 메시지

(Error) Syntax error - invalid token or invalid token sequence or missing token

예시 입력

a := 1 +

오류 메시지가 출력되는 상황

잘못된 토큰이 사용되거나 토큰들의 순서가 잘못되었거나 문법상 등장하여야하는 종류의 토큰이 등장하지 않았을때

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

a := 1 +

ID: 1; CONST: 1; OP: 1;

(Error) Syntax error - invalid token or invalid token sequence or missing token

Result ==> a: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - lexer가 끝을 인식하는 것이 parser가 + 뒤에 상수가 없다는 것을 인식하는 것 보다 빨라서 EOF가 UNKNOWN보다 먼저 출력된다.

1 (IDENT)

3 (ASSIGN_OP)

2 (CONST)

5 (ADD_OP)

9 (EOF)

0 (UNKNOWN)

error case 3-2 선언(식별자에 값이 정상적으로 할당)되지 않은 식별자가 사용된 경우

→ 다른 에러들과 달리 이 에러는 syntax error가 아니라 semantic error이므로 Parser가 계속 동작한다.

출력되는 오류 메시지

(Error) Using undeclared identifier(+ undeclared identifier +)

예시 입력

b := a

오류 메시지가 출력되는 상황

선언하지 않은 연산자가 사용되었거나 이전에 오류로 인해 값이 정상적으로 할당되지 않은 식별자가 사용되었을 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

b := a

ID: 2; CONST: 0; OP: 0;

(Error) Undefined variable is referenced(a)

Result ==> b: Unknown; a: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력

1 (IDENT)

3 (ASSIGN_OP)

1 (IDENT)

9 (EOF)

error case 6 assignment operator가 statement에 없는 경우

출력되는 오류 메시지

(Error) Missing assignment operator

예시 입력

a 10

오류 메시지가 출력되는 상황

assignment operator가 statement에 없는 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

a 10

ID: 1; CONST: 1; OP: 0;

(Error) Missing assignment operator

Result ==> a: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력

1 (IDENT)

2 (CONST)

9 (EOF)

error case 7 왼쪽 괄호가 없이 오른쪽 괄호가 나온 경우

출력되는 오류 메시지

(Error) Missing left parenthesis

예시 입력

b := 1 + 1)

오류 메시지가 출력되는 상황

왼쪽 괄호가 빠진 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

b := 1 + 1)
ID: 1; CONST: 2; OP: 1;
(Error) Missing left parenthesis

Result ==> b: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력

1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
5 (ADD_OP)
2 (CONST)
8 (RIGHT_PAREN)
9 (EOF)

error case 8 0으로 나눈 경우

출력되는 오류 메시지

(Error) Invalid expression - division by zero

예시 입력

a := 2/0;
b := 0;
c : 1/b

오류 메시지가 출력되는 상황

제수가 0 또는 0이 할당된 식별자일 때

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

a := 2 /0;
ID: 1; CONST: 2; OP: 1;
(Error) Invalid expression - division by zero

b := 0;
ID: 1; CONST: 1; OP: 0;
(OK)

c := 1/b
ID: 2; CONST: 1; OP: 1;
(Error) Invalid expression - division by zero

Result ==> a: Unknown; b: 0; c: Unknown;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력

1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
6 (MULT_OP)
2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
6 (MULT_OP)
1 (IDENT)
9 (EOF)

Warning List - Parser.py

warning case 8 괄호가 닫히지 않은 경우

→ 오류로 인해 Parser가 아닌 Lexer만 작동하는 상황에서는 출력되지 않음

출력되는 오류 메시지

```
(Warning) Missing right parenthesis ==> assuming right parenthesis at the end of statement
```

예시 입력

a := (1 + 1

오류 메시지가 출력되는 상황

왼쪽 괄호가 열린다음 닫히지 않은 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

a := (1 + 1)

ID: 1; CONST: 2; OP: 1;

(Warning) Missing right parenthesis ==> assuming right parenthesis at the end of statement

Result ==> a: 2;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - EOF가 인식된 시점에서)가 빠진것이 인식된다. 따라서 next_token은 다시)로 바꿨다가 EOF로 바뀐다.

1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
6 (MULT_OP)
2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
4 (SEMI_COLON)
1 (IDENT)
3 (ASSIGN_OP)
2 (CONST)
6 (MULT_OP)
1 (IDENT)
9 (EOF)

warning case 7-2 프로그램의 끝이 ;인 경우

출력되는 오류 메시지

(Warning) There is semicolon at the end of the program ==> ignoring semicolon

예시 입력

b := (1 + 1);

오류 메시지가 출력되는 상황

program의 끝이 ;인 경우

위의 예시 입력에 대한 -v 옵션이 없을 때의 예시 출력

b := (1 + 1)

ID: 1; CONST: 2; OP: 1;

(Warning) There is semicolon at the end of the program ==> ignoring semicolon

Result ==> b: 2;

위의 예시 입력에 대한 -v 옵션이 있을 때의 예시 출력 - ;가 출력되지 않은 것을 볼 수 있다.

1 (IDENT)
3 (ASSIGN_OP)
7 (LEFT_PAREN)
2 (CONST)
5 (ADD_OP)
2 (CONST)

8 (RIGHT_PAREN)

4 (SEMI_COLON)

9 (EOF)