



## **CC ASSIGNMENT 2**

<b>Name</b>	<b>Areeba Farooq</b>
<b>Registration Number</b>	<b>200901058</b>
<b>Batch &amp; Section</b>	<b>BSCS 01 (SECTION A)</b>
<b>Instructor's Name</b>	<b>Mam Tayyaba</b>

**Date:30-12-2022**

# **Module 1**

## **Implementation of lexical analyzer**

**Tokenization of expression (expression can be i.e  $a + (b * c)$  or  $3 + (5 * 2)$  digits, alphabets, characters )**

- **Building regex for the expression**
- **Output tags/ tokens of the expression (i.e. ['a', '+', '(', 'b', '\*', 'c', ''])]**

We will use re library for this module

### **Code:**

```
import re

#Expression

Exp = " 5* (4*3)"

alpha = re.findall(r"[\.\(\)\*\w+]", Exp)

print(alpha)

if(Exp):

    print("Expression is  tokenized")

else:

    print("Expression is not tokenized")
```

```
main.py
1 import re
2 #Expression
3 Exp = " 5* (4*3)"
4 alpha = re.findall(r"[.\\(\\)*\\w+]", Exp)
5 print(alpha)
6 if(Exp):
7     print("Expression is  tokenized")
8
9 else:
10    print("Expression is not tokenized")

input
['5', '*', '(', '4', '*', '3', ')']
Expression is  tokenized

...Program finished with exit code 0
Press ENTER to exit console.
```

## Expression:

5+ (9/7+8+Areeba)

## Code:

```
import re
```

```
#Expression
```

```
Exp = " 5+ (9/7+8+Areeba)"
```

```
x = re.findall(r"[.\\(\\)*\\w+]", Exp)
```

```
print(x)
```

```
main.py
1 import re
2 #Expression
3 Exp = " 5+ (9/7+8+Areeba)"
4 x = re.findall(r"[.\\(\\)*\\w+]", Exp)
5 print(x)
6 |

input
['5', '+', '(', '9', '7', '+', '8', '+', 'A', 'r', 'e', 'e', 'b', 'a', ')']

...Program finished with exit code 0
Press ENTER to exit console.
```

## Module 2

### Implementation of syntax tree using AST library of python

For implementation of this we will use AST library of python. AST is also known as Abstract Syntax Tree, is a python tool that is used to directly interact with python code and modify them.

**alpha=(5\*4+3)**

### Implementation

```
import ast
```

```
tree_ast = ast.parse("""
```

```
alpha=(5*4+3)
```

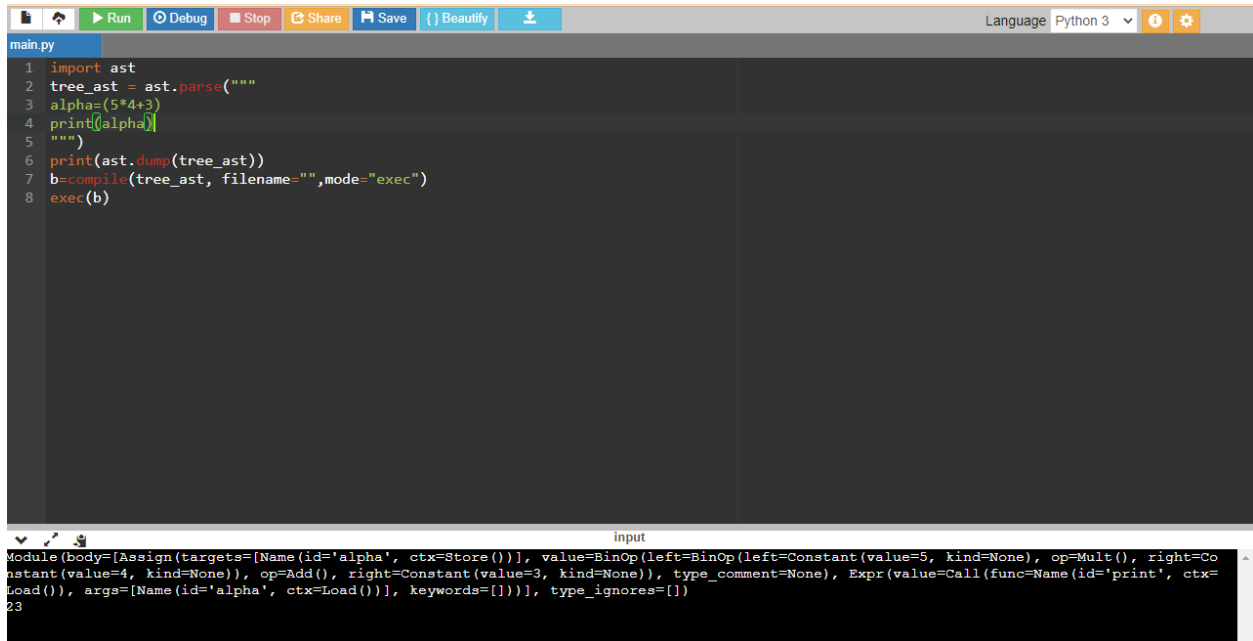
```
print(alpha)
```

```
""")
```

```
print(ast.dump(tree_ast))
```

```
b=compile(tree_ast, filename="",mode="exec")
```

```
exec(b)
```



The screenshot shows a Python IDE with a toolbar at the top containing icons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to Python 3. The main editor window displays the following code in main.py:

```
1 import ast
2 tree_ast = ast.parse("""
3 alpha=(5*4+3)
4 print(alpha)
5 """)
6 print(ast.dump(tree_ast))
7 b=compile(tree_ast, filename="",mode="exec")
8 exec(b)
```

Below the editor, the 'input' pane shows the AST dump:

```
Module(body=[Assign(targets=[Name(id='alpha', ctx=Store())], value=BinOp(left=BinOp(left=Constant(value=5, kind=None), op=Mult(), right=Co
nstant(value=4, kind=None)), op=Add(), right=Constant(value=3, kind=None)), type_comment=None), Expr(value=Call(func=Name(id='print', ctx=
Load()), args=[Name(id='alpha', ctx=Load())], keywords=[])], type_ignores=[])]
23
```

```
x=(5+7/9)
```

## Code

```
import ast
```

```
tree_ast = ast.parse("""
```

```
x=(5+7/9)
```

```
print(x)
```

```
""")
```

```
print(ast.dump(tree_ast))
```

```
b=compile(tree_ast, filename="",mode="exec")
```

```
exec(b)
```

```
main.py
1 import ast
2 tree_ast = ast.parse("""
3 x=(5+7/9)
4 print(x)
5 """)
6 print(ast.dump(tree_ast))
7 b=compile(tree_ast, filename="", mode="exec")
8 exec(b)

input
Module(body=[Assign(targets=[Name(id='x', ctx=Store())], value=BinOp(left=Constant(value=5, kind=None), op=Add(), right=BinOp(left=Constant(value=7, kind=None), op=Div(), right=Constant(value=9, kind=None))), type_comment=None), Expr(value=Call(func=Name(id='print', ctx=Load()), args=[Name(id='x', ctx=Load())], keywords=[])], type_ignores=[])], type_ignores=[])
5.777777777777778

...Program finished with exit code 0
Press ENTER to exit console.
```

## GitHub link:

<https://github.com/AREEBA-FAROOQ2001/IST>