

LEXICAL ANALYZER

MAIN CODE

```
import re

f = open("code.txt", "r")

count = 0

code = []

for i in f.read().split("\n"):

    if i:

        count += 1

        code.append(i);

# print(code)

tokens = []

keywords = [

    'abstract', 'continue', 'for', 'new', 'switch',

    'assert', 'default', 'goto', 'package', 'synchronized',

    'boolean', 'do', 'if', 'private', 'this',

    'break', 'implements', 'protected', 'throw',

    'byte', 'else', 'import', 'public', 'throws',

    'case', 'enum', 'instanceof', 'return', 'transient',

    'catch', 'extends', 'try',

    'final', 'interface', 'static', 'void',

    'class', 'finally', 'strictfp', 'volatile',

    'const', 'native', 'super', 'while', 'System']

datatypes = ['double', 'int', 'short', 'char', 'long', 'float', 'bool', 'String']
```

```
lineno = 0
```

```
for line in code:
```

```
    source_code = str(line).split()
```

```
    lineno += 1
```

```
    # Loop through each source code word
```

```
    for word in source_code:
```

```
        if word in keywords:
```

```
            tokens.append([lineno, 'KEYWORD', word])
```

```
        elif word in datatypes:
```

```
            tokens.append([lineno, 'DATATYPE', word])
```

```
        elif re.match("[a-z]", word) or re.match("[A-Z]", word):
```

```
            tokens.append([lineno, 'IDENTIFIER', word])
```

```
        elif word in '-/+%=*':
```

```
            tokens.append([lineno, 'OPERATOR', word])
```

```
        elif word in '(){}[]";':
```

```
            tokens.append([lineno, 'DELIMITER', word])
```

```
        elif re.match("[0-9]", word):
```

```
            if word[len(word) - 1] == ';':
```

```
                tokens.append([lineno, "INTEGER", word[:-1]])
```

```
            else:
```

```
tokens.append([lineno,"INTEGER", word])
```

```
print('\n')
```

```
print('line no  type\t\t name')
```

```
for token in tokens:
```

```
    print(f'{token[0]} \t {token[1]} \t {token[2]}')
```

CODE TEXT INPUT

```
class Code {
```

```
    public static void main ( String args [ ] ) {
```

```
        System.out.println ( " Hello Java " ) ;
```

```
        for(i=0; i<10; i++)
```

```
        {
```

```
            System.out.println ( " Experiment 1 " )
```

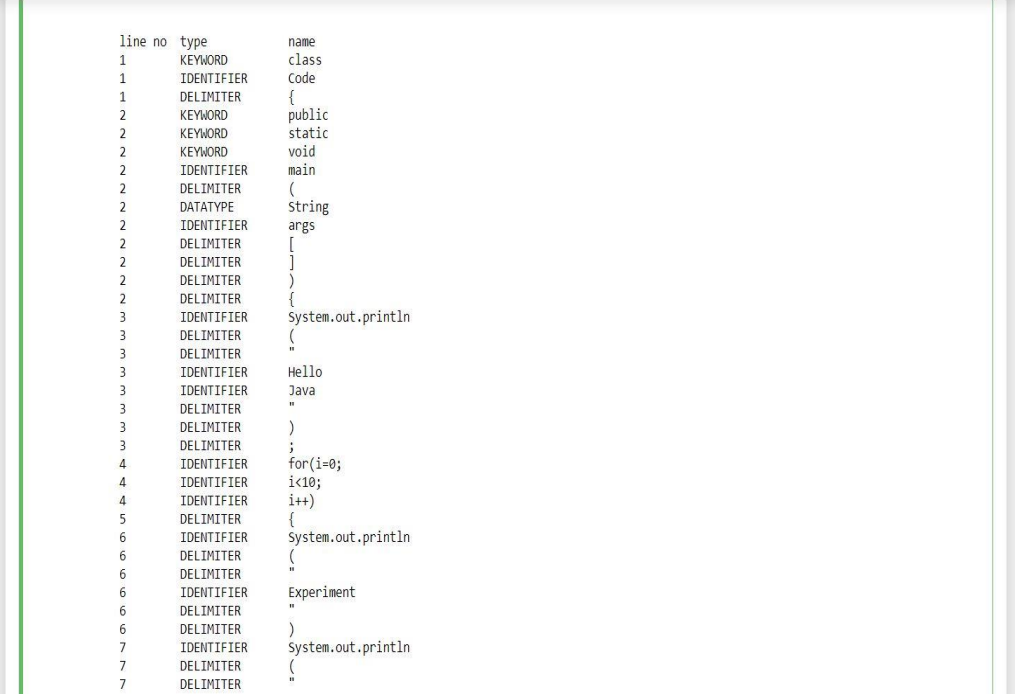
```
            System.out.println ( " Lexical analyzer code " )
```

```
        }
```

```
    }
```

```
}
```

OUTPUT



Jupyter Notebook interface showing a Java code snippet and its tokenization.

The code defines a `Code` class with a `main` method. The `main` method prints "Hello Java" and then enters a loop that prints "Experiment" 10 times.

The tokenization table below the code lists the tokens and their types:

line no	type	name
1	KEYWORD	class
1	IDENTIFIER	Code
1	DELIMITER	{
2	KEYWORD	public
2	KEYWORD	static
2	KEYWORD	void
2	IDENTIFIER	main
2	DELIMITER	(
2	DATATYPE	String
2	IDENTIFIER	args
2	DELIMITER	[
2	DELIMITER]
2	DELIMITER)
2	DELIMITER	{
3	IDENTIFIER	System.out.println
3	DELIMITER	(
3	DELIMITER	"
3	IDENTIFIER	Hello
3	IDENTIFIER	Java
3	DELIMITER	"
3	DELIMITER)
3	DELIMITER	;
4	IDENTIFIER	for(i=0;
4	IDENTIFIER	i<10;
4	IDENTIFIER	i++)
5	DELIMITER	{
6	IDENTIFIER	System.out.println
6	DELIMITER	(
6	DELIMITER	"
6	IDENTIFIER	Experiment
6	DELIMITER	"
6	DELIMITER)
7	IDENTIFIER	System.out.println
7	DELIMITER	(
7	DELIMITER	"

The tokenization table continues with the following tokens:

7	IDENTIFIER	Lexical
7	IDENTIFIER	analyzer
7	IDENTIFIER	code
7	DELIMITER	"
7	DELIMITER)
8	DELIMITER	}
9	DELIMITER	}
10	DELIMITER	}