EX.No : 1

Date : 25/08/2021

LEXICAL ANALYZER

AIM:

To create a a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines

ALGORITHM

STEP 1: Import regex library because it will be needed when checking if certain words match a certain regex pattern.

STEP 2: Create an empty list called tokens. This will be used to store all of the tokens we create.

STEP 3: Turn source code into list of words.

STEP 4: Loop through each source code word to check for a match

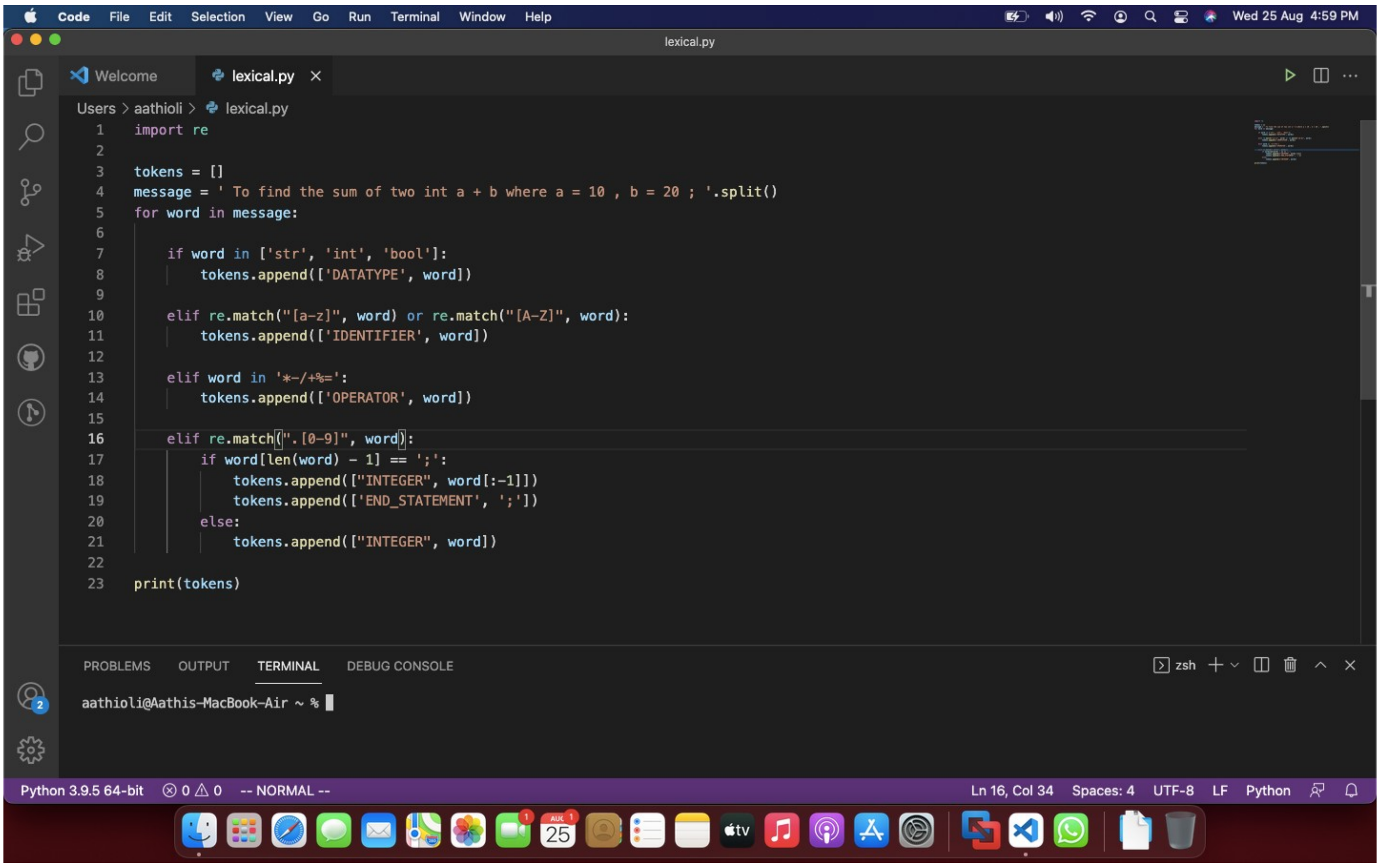
```
• This will check if a token has datatype declaration
    if word in ['str', 'int', 'bool']:

• This will look for an identifier which would be just a word
      elif re.match("[a-z]", word) or re.match("[A-Z]", word):

•  This will look for an operator
      elif word in '*-/+%=':

• This will look for integer items and cast them as a number
        elif re.match(".[0-9]", word):
```

STEP 5: After Performing more checks like the one above identifying each word in our source code and creating a token for it. These tokens will then be passed on to the parser to create an Abstract Syntax Tree (AST).

SOURCE CODE

```
import re
tokens = []                                       source_code =
'int result = 100;'.split()
for word in source_code:

   if word in ['str', 'int', 'bool']:
        tokens.append(['DATATYPE', word])

   elif re.match("[a-z]", word) or re.match("[A-Z]",
word):
        tokens.append(['IDENTIFIER', word])

   elif word in '*-/+%=':
        tokens.append(['OPERATOR', word])

   elif re.match(".[0-9]", word):
        if word[len(word) - 1] == ';':
            tokens.append(["INTEGER", word[:-1]])
            tokens.append(['END_STATEMENT', ';'])
        else:
            tokens.append(["INTEGER", word])

print(tokens)
```

OUTPUT

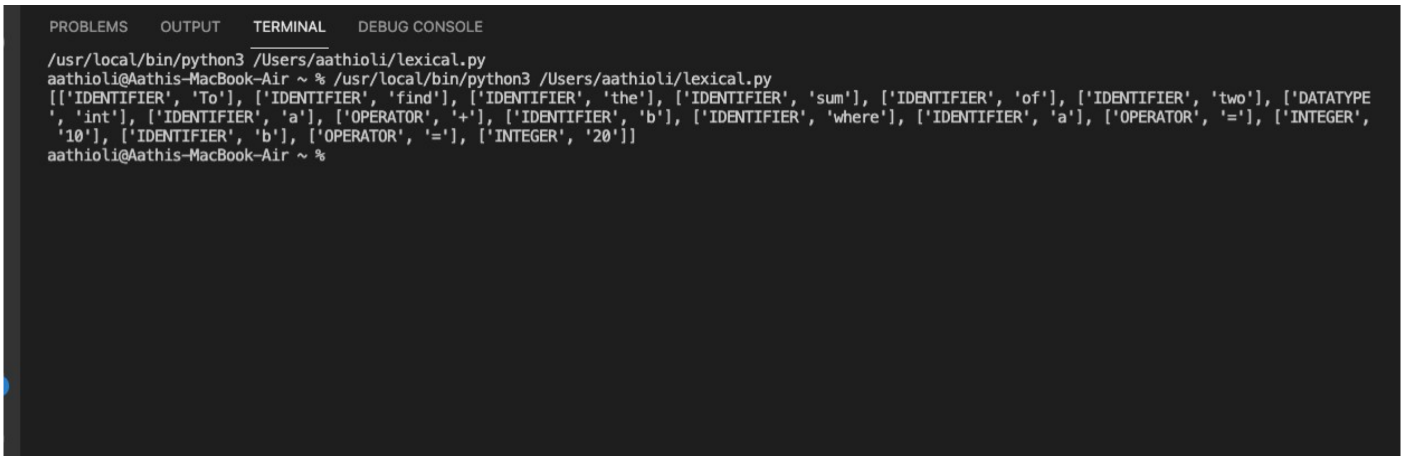


Code File Edit Selection View Go Run Terminal Window Help
Wed 25 Aug 4:59 PM
lexical.py
Welcome
lexical.py
Users > aathioli > lexical.py

```python
import re

tokens = []
message = ' To find the sum of two int a + b where a = 10 , b = 20 ; '.split()
for word in message:

    if word in ['str', 'int', 'bool']:
        tokens.append(['DATATYPE', word])

    elif re.match("[a-z]", word) or re.match("[A-Z]", word):
        tokens.append(['IDENTIFIER', word])

    elif word in '*-/+%=':
        tokens.append(['OPERATOR', word])

    elif re.match(".[0-9]", word):
        if word[len(word) - 1] == ';':
            tokens.append(["INTEGER", word[:-1]])
            tokens.append(['END_STATEMENT', ';'])
        else:
            tokens.append(["INTEGER", word])

print(tokens)
```

```
aathioli@Aathis-MacBook-Air ~ %
```

Python 3.9.5 64-bit ⊗ 0 ⚠ 0 -- NORMAL --
Ln 16, Col 34 Spaces: 4 UTF-8 LF Python



PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
/usr/local/bin/python3 /Users/aathioli/lexical.py
aathioli@Aathis-MacBook-Air ~ % /usr/local/bin/python3 /Users/aathioli/lexical.py
[['IDENTIFIER', 'To'], ['IDENTIFIER', 'find'], ['IDENTIFIER', 'the'], ['IDENTIFIER', 'sum'], ['IDENTIFIER', 'of'], ['IDENTIFIER', 'two'], ['DATATYPE
', 'int'], ['IDENTIFIER', 'a'], ['OPERATOR', '+'], ['IDENTIFIER', 'b'], ['IDENTIFIER', 'where'], ['IDENTIFIER', 'a'], ['OPERATOR', '='], ['INTEGER',
 '10'], ['IDENTIFIER', 'b'], ['OPERATOR', '='], ['INTEGER', '20']]
aathioli@Aathis-MacBook-Air ~ %
```

RESULT

A lexical analyzer has been created which ignore redundant spaces, tabs and new lines and desired output was obtained .