# FARIDPUR ENGINEERING COLLEGE

## Department of Computer Science & Engineering

**Course Name:** Artificial Intelligence (Sessional).

**Course Code:** CSE-702.

## Lab Report

**SUBMITTED TO,**

Md Tuhin Reza

Lecturer

Department of CSE

Faridpur Engineering College

# Index

**Experiment No:** 01

**Experiment Name**: Study on Arithmetic Operations (Add, Subtraction, Division, Multiplication) in Python.

**Objective**:

The objective of this lab is to gain a comprehensive understanding of performing basic arithmetic operations (addition, subtraction, division, and multiplication) using the Python programming language.

**Description:**

Arithmetic operations are fundamental in programming and mathematics. Python provides built-in operators and functions to perform these operations on numerical data. In this lab, we will explore how to use Python to perform arithmetic operations and observe the results.

**Code:**

```
#using list
numbers = []

for i in range(5):
    a = int(input())
    numbers.append(a)

sum_result = sum(numbers)

subtraction_result = numbers[0] - sum(numbers[1:])

multiplication_result = 1
for num in numbers:
    multiplication_result *= num

division_result = numbers[0]
for num in numbers[1:]:
    division_result /= num

print("Sum Result:", sum_result)
print("Subtraction Result:", subtraction_result)
print("Multiplication Result:", multiplication_result)
print("Division Result:", division_result)

# using dictionary
marks = {'subject1': 10, 'subject2': 20, 'subject3': 30}
```

```
sum = 0
mul = 1
div = marks['subject1']
sub = marks['subject1']

for value in marks.values():
    sum += value
print("Sum =", sum)

for value in marks.values():
    mul *= value
print("Mul =", mul)

for key, value in marks.items():
    if key != 'subject1':
        div /= value
print("Div =", div)

for key, value in marks.items():
    if key != 'subject1':
        sub -= value
print("Sub =", sub)
```

**Result:**

```
1
2
3
4
5
Sum Result: 15
Subtraction Result: -13
Multiplication Result: 120
Division Result: 0.008333333333333333

#dictionary
Sum = 60
Mul = 6000
Div = 0.016666666666666666
Sub = -40
```

**Conclusion:**

We learned about basic arithmetic operations in Python: addition, subtraction, division, and multiplication. We used variables to store numerical values and performed these operations using appropriate operators. The results of the operations were displayed using the print function. Python's arithmetic operators and built-in functions make it easy to perform mathematical calculations within the program. This lab helped us to understand how to work with numerical data and perform fundamental arithmetic operations using Python.

**Experiment No:** 02

**Experiment Name**: Study on arithmetic operation (Subtraction and Multiplication) in prolog.

**Objective:**

- To learn about prolog basics
- To learn how to implement arithmetic operations.

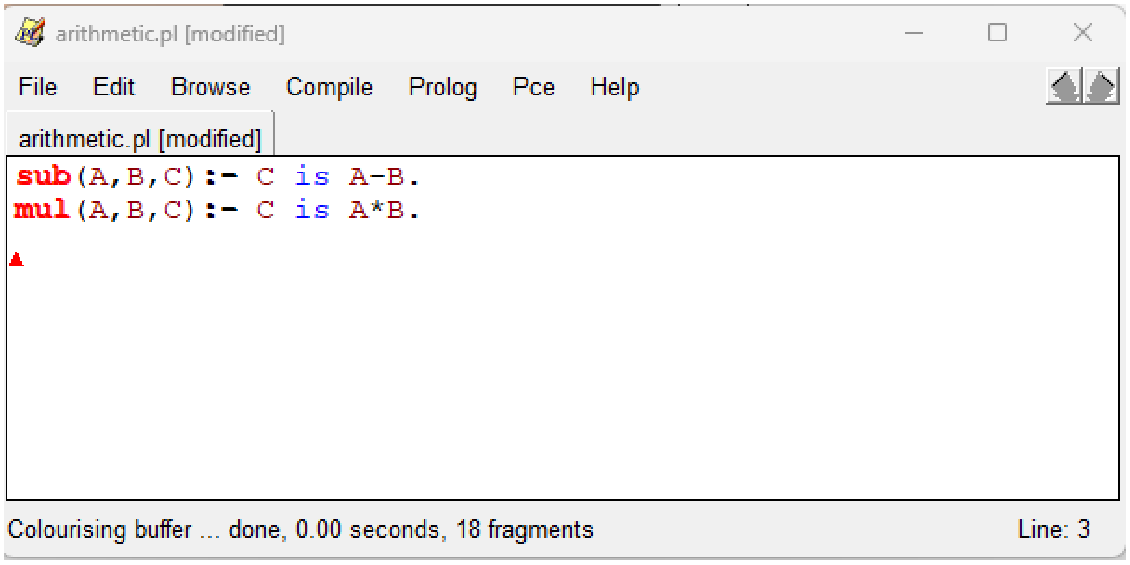
**Description**:

Prolog is a logic programming language that does not have built-in arithmetic operations. However, there are a number of libraries that provide arithmetic operations for Prolog. One popular library is the SWI-Prolog library.

Mathematical operations on numbers are referred to as arithmetic operations. Addition, subtraction, multiplication, and division are the four fundamental arithmetic operations. In this problem, we are working on only subtraction and multiplication.

- ➢ Subtraction: Subtraction is the process of taking away one number from another number. In this case, 5-3=2.
- ➢ Multiplication: The process of multiplying involves repeatedly adding two or more numbers together. For instance, 2 + 3 = 6.

**Code**:



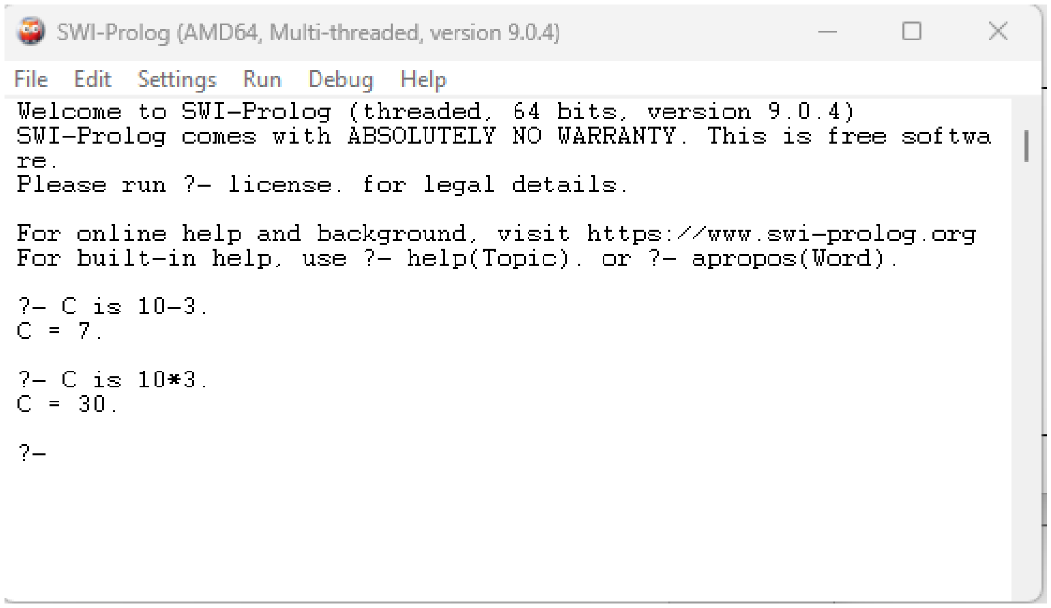
```
sub(A,B,C) :- C is A-B.
mul(A,B,C) :- C is A*B.
```

```
Colourising buffer ... done, 0.00 seconds, 18 fragments                Line: 3
```

*Figure 1:Creating Knowledge Base*

**Result:**

In this problem, we will be subtracting two numbers. 52 and 33 respectively. The outcome is 19, after subtraction. On the other hand, we will use the integers 12 and 2 for the multiplication process. Hence 24 is the result of the multiplication.


SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free softwa
re.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- C is 10-3.
C = 7.

?- C is 10*3.
C = 30.

?-
```

*Figure 2:Accessing using Query Section*

**Discussion**:

For this problem, we must be careful about there is no syntactical error in the coding. The code must be error-free.

**Experiment No:** 03

**Experiment Name:** Write a prolog program for the addition of two numbers in artificial intelligence

**Objective**:

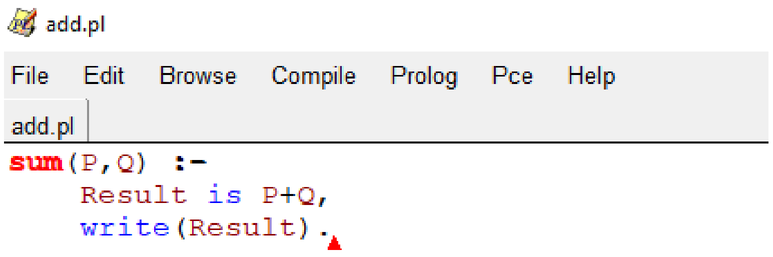
Using artificial intelligence to execute the concept of adding two numbers.

**Description**:

Prolog is a logic programming language associated with artificial intelligence and computational linguistics. As a result, the logic programming language Prolog lacks built-in support for arithmetic operations. Prolog may use a number of libraries that offer arithmetic operations. The SWI-Prolog library is one well-known collection.

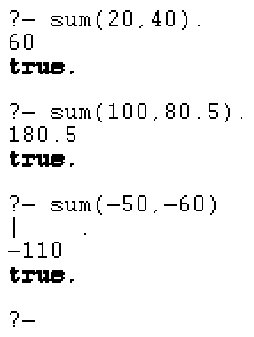
The term "arithmetic operations" refers to mathematical operations on numbers. The four basic operations used in mathematics are addition, subtraction, multiplication, and division. Among of them, addition operation means combining two or more numbers to produce a new number is the process of addition. For instance, 2+3 = 5.

**Code**:



```
sum(P,Q) :-
    Result is P+Q,
    write(Result).
```

**Result**:



```
?- sum(20,40).
60
true.

?- sum(100,80.5).
180.5
true.

?- sum(-50,-60)
|      .
-110
true.

?-
```

**Discussion**:

For this problem, we must take care of this issue to ensure that the coding has no syntactical errors. Exceptionless code is required.

**Experiment No:** 04

**Experiment Name**: Study on a prolog program for finding the sum of all numbers in a given list in artificial intelligence.

**Objective:**

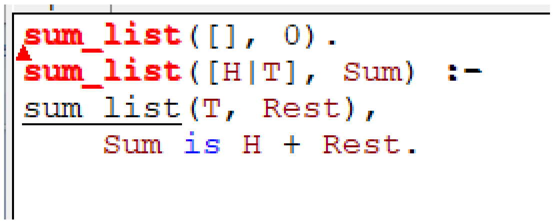
The objective of this lab was to study and implement a Prolog program to calculate the sum of all numbers in a given list.

**Description:**

The base case defines that the sum of an empty list is 0. The recursive rule defines that the sum of a non-empty list is the head of the list added to the sum of the rest of the list.

**Code:**

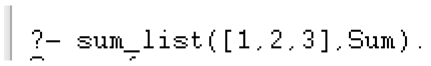
```
sum_list([], 0).
sum_list([H|T], Sum) :-
sum_list(T, Rest),
   Sum is H + Rest.
```



```
sum_list([], 0).
sum_list([H|T], Sum) :-
sum_list(T, Rest),
     Sum is H + Rest.
```

**Prolog interpreter query :**

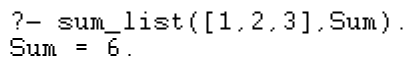
sum_list([1,2,3],Sum).



```
?- sum_list([1,2,3],Sum).
```

**Result:**

Sum = 6.



```
?- sum_list([1,2,3],Sum).
Sum = 6.
```

**Conclusion:**

Successfully studied and implemented a Prolog program to find the sum of all numbers in a given list.

**Experiment No:** 05.

**Experiment Name**: Study on a program to reverse given numbers in Python

**Objective:**

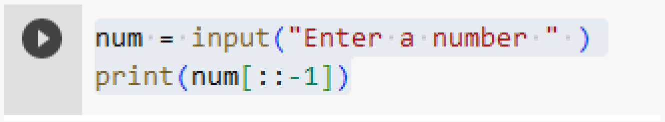
The objective of this lab was to study and implement a Python program that can reverse a given integer number. Reversing a number involves changing the order of its digits while maintaining the sign.

**Description:**

- If the number is non-negative, it converts the number to a string, reverses the string using slicing ([::-1]), and then converts it back to an integer.
- If the number is negative, it works similarly but reverses the absolute value and adds the negative sign back.

**Code:**

```
num = input("Enter a number " )
print(num[::-1])
```

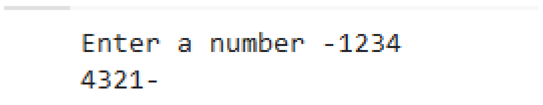


```
num = input("Enter a number " )
print(num[::-1])
```

**Result:**

Enter a number -1234

4321-



```
Enter a number -1234
4321-
```

**Conclusion:**

This lab exercise provided insights into implementing a Python program to reverse a given integer number.

**Experiment No**: 06.

**Experiment Name:** Convert the queries/facts into First Order Logic in Prolog.

**Objective**:

To become proficient in how to translate first-order logic from questions and facts.

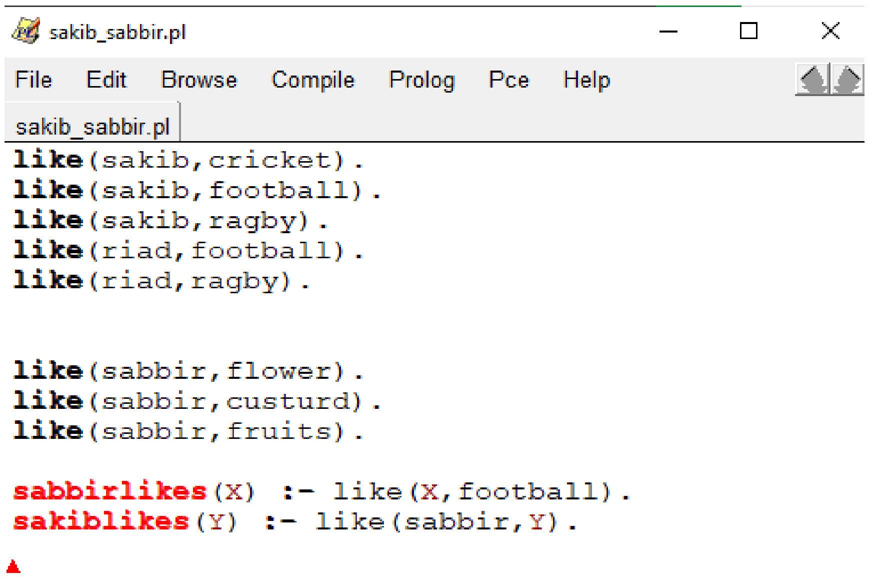
**Description**:

A logic programming language built on predicate calculus is called Prolog. Because it is a declarative language, programs are expressed in terms of what they should accomplish rather than how they should accomplish it. Because of this, Prolog programs are simpler to read and comprehend than those created in procedural languages like C or Java.

FOL is a formal framework for representing and thinking about the world. It is a subset of Predicate Logic, a more comprehensive formal framework for inferring relationships between sets. FOL is founded on the following fundamental principles:

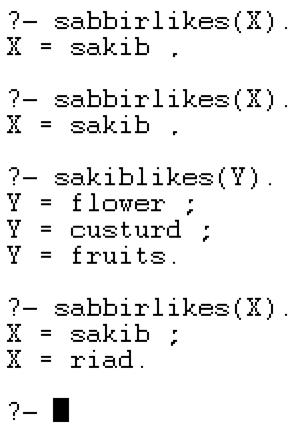
- Variable: Variables are used to symbolize unknowable quantities.
- Predicates: Predicates are used to depict how different variables relate to one another. For instance, the relationship between two variables x and y, where x is greater than y, is represented by the predicate x > y.
- Quantifier: Quantifiers are used to describe how frequently a predicate holds true for a group of variables. For instance, the existential quantifier means "there exists," while the universal quantifier indicates "for all."

**Code**:



```
like(sakib,cricket).
like(sakib,football).
like(sakib,ragby).
like(riad,football).
like(riad,ragby).


like(sabbir,flower).
like(sabbir,custurd).
like(sabbir,fruits).

sabbirlikes(X) :- like(X,football).
sakiblikes(Y) :- like(sabbir,Y).
```

**Result**:



```
?- sabbirlikes(X).
X = sakib .

?- sabbirlikes(X).
X = sakib .

?- sakiblikes(Y).
Y = flower ;
Y = custurd ;
Y = fruits.

?- sabbirlikes(X).
X = sakib ;
X = riad.

?- 
```

**Discussion**:

For this problem, we must ensure that there are no syntactical problems in the coding for this issue, care must be taken. Code that is faultless is necessary. Convert the queries/facts into first-order logic in an appropriate manner.

**Experiment No:** 07

**Experiment Name:** Converting queries/facts into FOL

**Objective:**

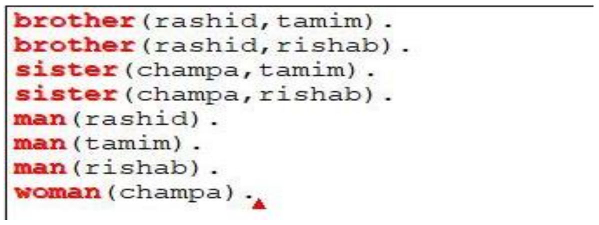
This lab report aims to explore the process of converting natural language queries and factual statements into First-Order Logic (FOL) representations and solve queries based on these relationships.

**Description:**

First-Order Logic (FOL), also known as Predicate Logic, is a mathematical language designed for representing knowledge about the world in a formal, structured way. It provides a means to express relationships between objects, attributes, and actions. FOL consists of predicates, variables, quantifiers, and logical connectives that help capture the complexity of natural language statements and queries.

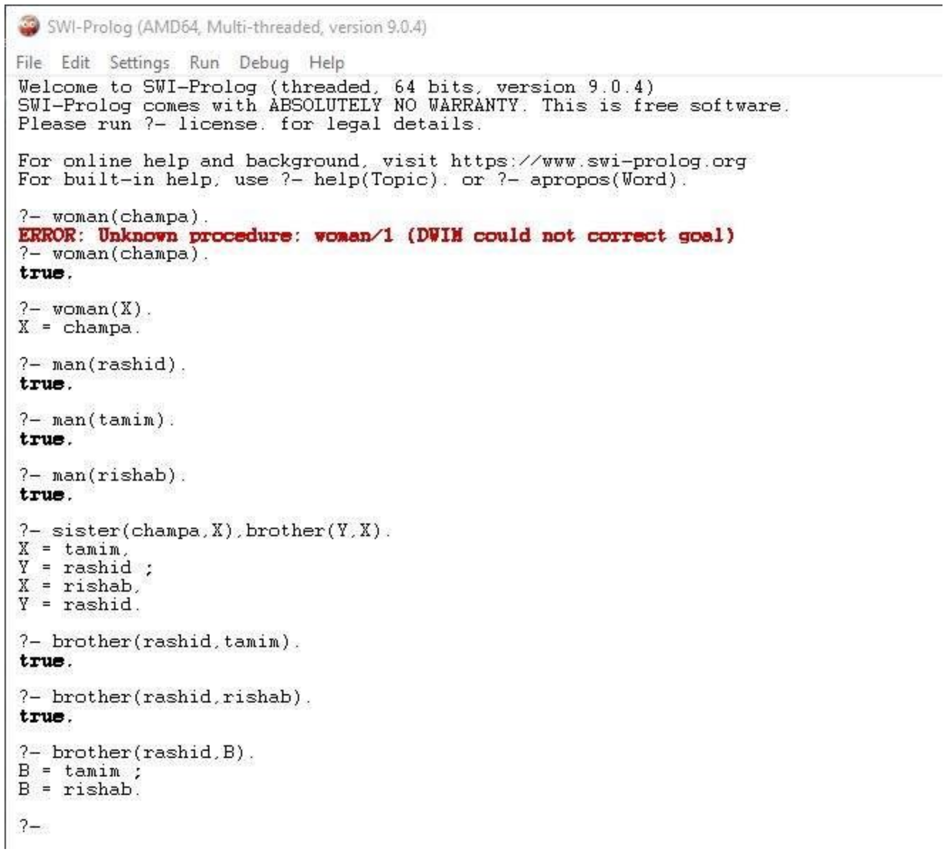
- Investigating logical connections between individuals and genders is the focal point, achieved through Prolog.
- This involves defining facts that categorize individuals as men or women and discerning sibling relationships as brothers or sisters.
- Inference rules are then created, utilizing gender and sibling associations to deduce connections.
- Queries are subsequently posed to Prolog, aiming to extract specific relationship intricacies.
- The precision of the logical framework is assessed by comparing Prolog's responses with the predetermined relationships.

**Code:**



```
brother(rashid,tamim).
brother(rashid,rishab).
sister(champa,tamim).
sister(champa,rishab).
man(rashid).
man(tamim).
man(rishab).
woman(champa).
```

**Input & output:**


SWI-Prolog (AMD64, Multi-threaded, version 9.0.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.0.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- woman(champa).
ERROR: Unknown procedure: woman/1 (DWIM could not correct goal)
?- woman(champa).
true.

?- woman(X).
X = champa.

?- man(rashid).
true.

?- man(tamim).
true.

?- man(rishab).
true.

?- sister(champa,X),brother(Y,X).
X = tamim,
Y = rashid ;
X = rishab,
Y = rashid.

?- brother(rashid,tamim).
true.

?- brother(rashid,rishab).
true.

?- brother(rashid,B).
B = tamim ;
B = rishab.

?-
```

**Discussion:**

Converting queries and facts into First-Order Logic enables us expressing complex relationships in a structured language. Exploring logical relationships and queries in Prolog involved defining genders and sibling connections. Queries efficiently retrieved information about genders and siblings, showcasing Prolog's logical programming prowess in real-world problem-solving

**Experiment No:** 08

**Experiment Name:** Implement a program to count all words of a given file using NLP.

**Objective:** To count the number of words from a text file.

**Theory:**

Natural Language Processing (NLP) is a field of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. It involves the development of algorithms and models to process and analyze text and speech data.

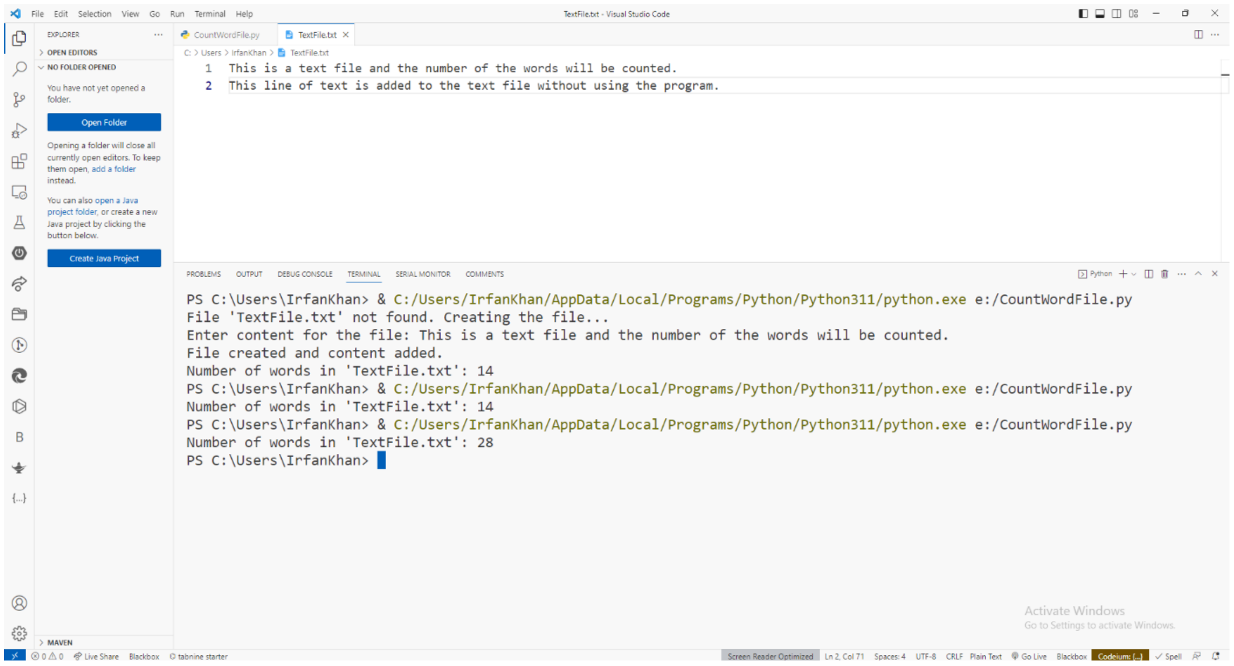
A Python program can create a text file and read the texts residing in it. By reading the text lines, the words of the text line can be counted and the number of words can be printed.

**Program:**

```
def count_words(filename):

    try:

        with open(filename, 'r') as file:

            content = file.read()

            words = content.split()

            word_count = len(words)

            return word_count

    except FileNotFoundError:

        print(f"File '{filename}' not found. Creating the file...")

        with open(filename, 'w') as file:

            content = input("Enter content for the file: ")

            file.write(content)

            print("File created and content added.")

            words = content.split()

            word_count = len(words)

            return word_count

        return 0
```

```
filename = 'TextFile.txt'

word_count = count_words(filename)

print(f"Number of words in '{filename}': {word_count}")
```

## Output:



```
This is a text file and the number of the words will be counted.
This line of text is added to the text file without using the program.
```

```
PS C:\Users\IrfanKhan> & C:/Users/IrfanKhan/AppData/Local/Programs/Python/Python311/python.exe e:/CountWordFile.py
File 'TextFile.txt' not found. Creating the file...
Enter content for the file: This is a text file and the number of the words will be counted.
File created and content added.
Number of words in 'TextFile.txt': 14
PS C:\Users\IrfanKhan> & C:/Users/IrfanKhan/AppData/Local/Programs/Python/Python311/python.exe e:/CountWordFile.py
Number of words in 'TextFile.txt': 14
PS C:\Users\IrfanKhan> & C:/Users/IrfanKhan/AppData/Local/Programs/Python/Python311/python.exe e:/CountWordFile.py
Number of words in 'TextFile.txt': 28
PS C:\Users\IrfanKhan>
```

*Figure 3:Created Text file and Counting The number of words.*

## Discussion:

This program successfully creates a file called TextFile.txt and writes some text in the file. Then the number of words in that file is counted and the counted number is printed. If the text file is already present, then the number of words in that text file is counted also.