APKA Programming Language

Introduction

Welcome to the world of APKA, a dynamic programming language designed to empower developers with an imperative paradigm.

Team Behind APKA:

Anjali Bharuka
Prasheel Nandwana
Kushagra Kartik
Asad Shaikh

Overview:

In this presentation, we delve into the intricacies of APKA, covering its design, grammar, features, and future scope.

About APKA:

Name: APKA
File Extension: .apka
Programming Paradigm: Imperative
Programming Languages Used: Python (Lexer), Prolog (Parser and Evaluator)

Design:

Project Pipeline:

.APKA → LEXER.PY → .APKATOKENS → PARSER.PL → EVALUATOR.PL

Components of the Design:

.APKA: The APKA source code file serves as the initial input to the pipeline.

LEXER.PY: The lexer script reads the APKA source code and generates tokenized output.

.APKATOKENS: The tokenized output from the lexer is stored in a file.

PARSER.PL: The parser reads the tokenized output and generates a parse tree.

EVALUATOR.PL: Finally, the evaluator takes the parse tree as input and executes the instructions, producing the desired output.

Features:

Program Structure:

Programs are enclosed within {} and consist of a list of commands.

Commands:

- For Loop
- While Loop
- Enhanced For Loop
- If Statement
- If-Elif-Else Statement
- Print Statement
- Variable Declaration
- Variable Assignment

Variable Types:

- Integer (chotu)
- Boolean (nirnaya)
- String (vakya)

Operations:

- Arithmetic Operations (+, -, *, /)
- Comparison Operations (==, <, >, <=, >=)
- Boolean Operations (&& ,||, !)
- Ternary Operator (? :)
- Unary Operator (++, --)

Reserved Keywords:

Program Structure: {}

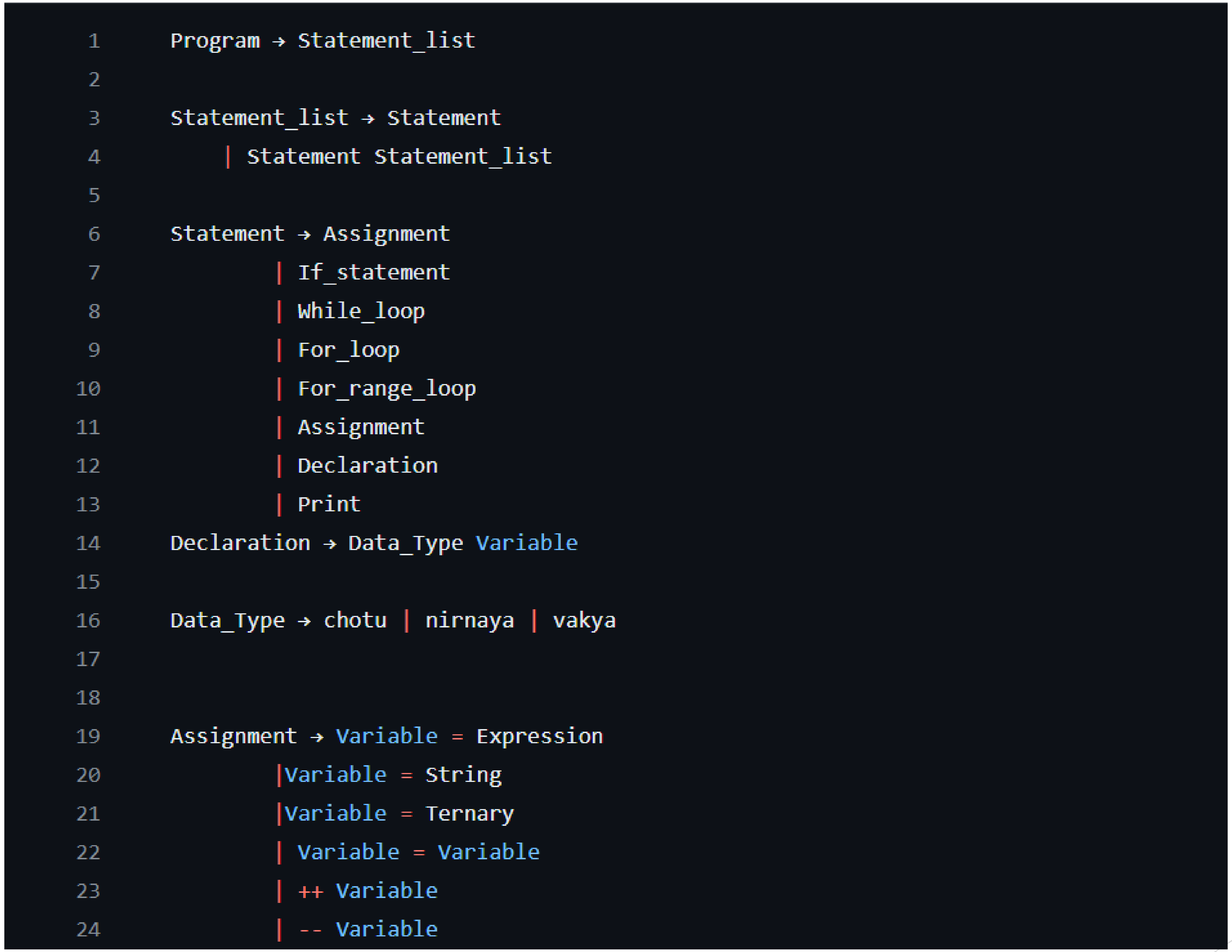
Commands: chaap, jab, nahito, jabtak, tabtak

Variable Types: chotu, nirnaya, vakya

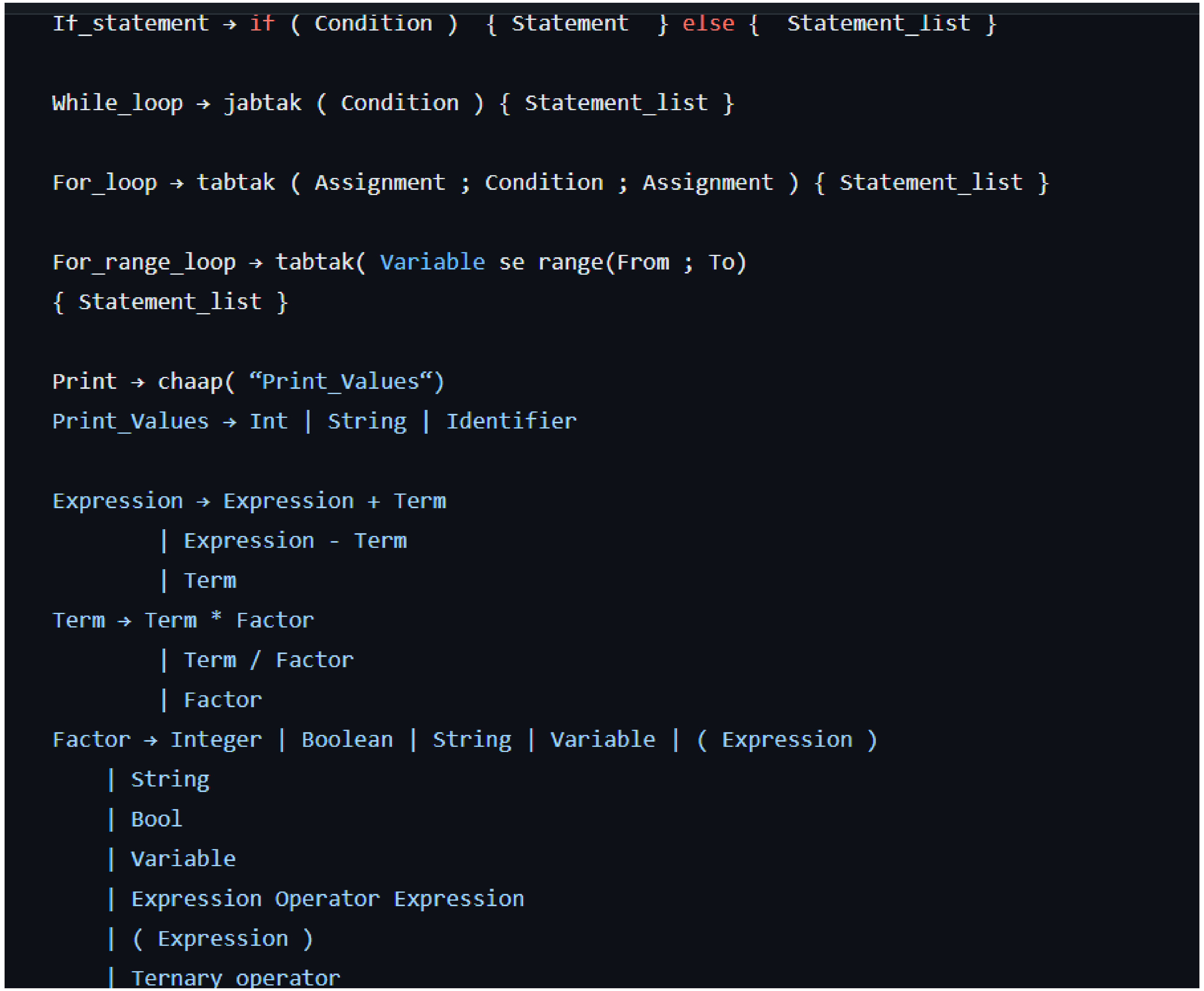
Others: se

Boolean Values: sach, jhoot

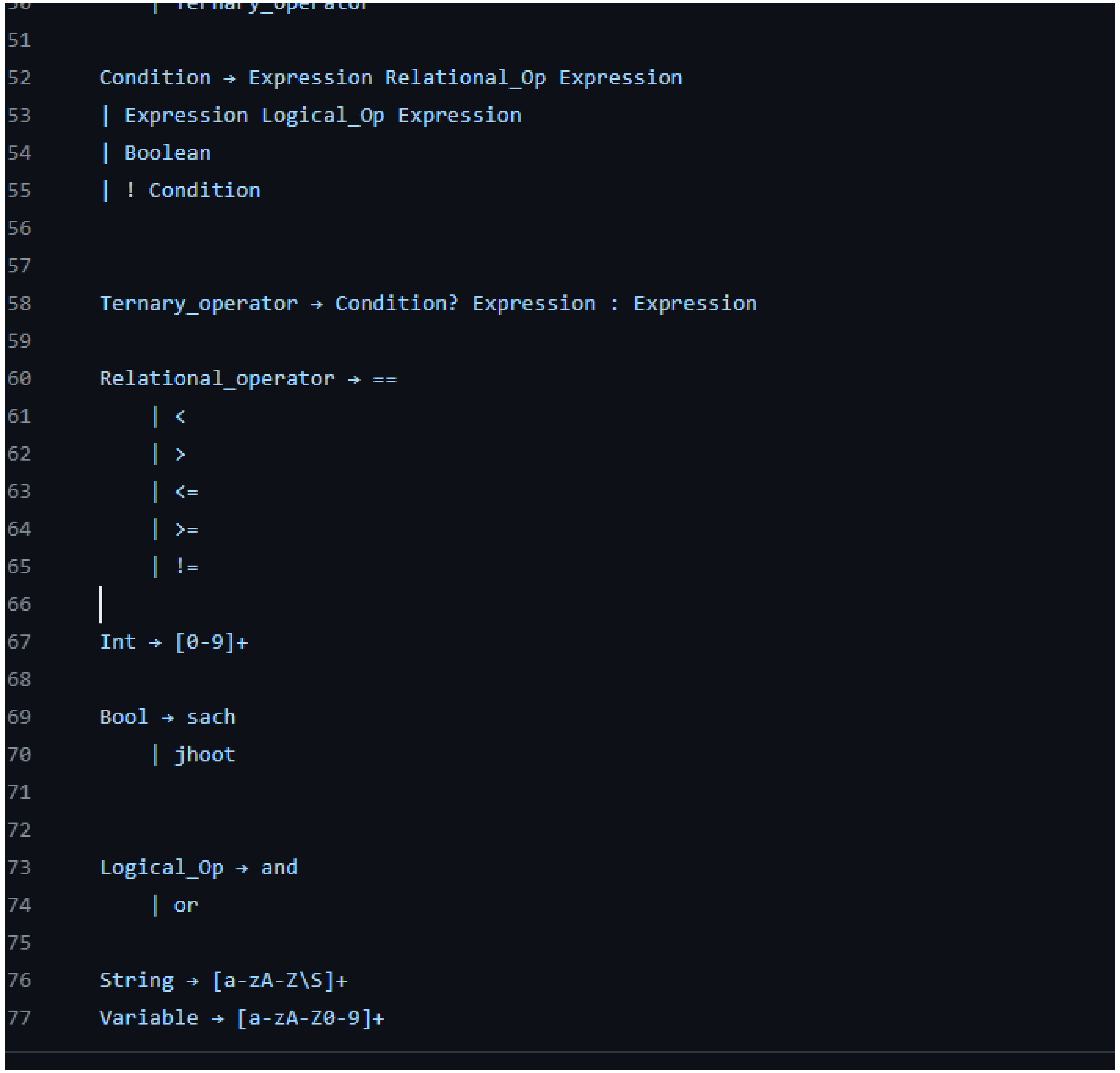
GRAMMAR:



```
Program → Statement_list

Statement_list → Statement
  | Statement Statement_list

Statement → Assignment
    | If_statement
    | While_loop
    | For_loop
    | For_range_loop
    | Assignment
    | Declaration
    | Print
Declaration → Data_Type Variable

Data_Type → chotu | nirnaya | vakya


Assignment → Variable = Expression
    |Variable = String
    |Variable = Ternary
    | Variable = Variable
    | ++ Variable
    | -- Variable
```

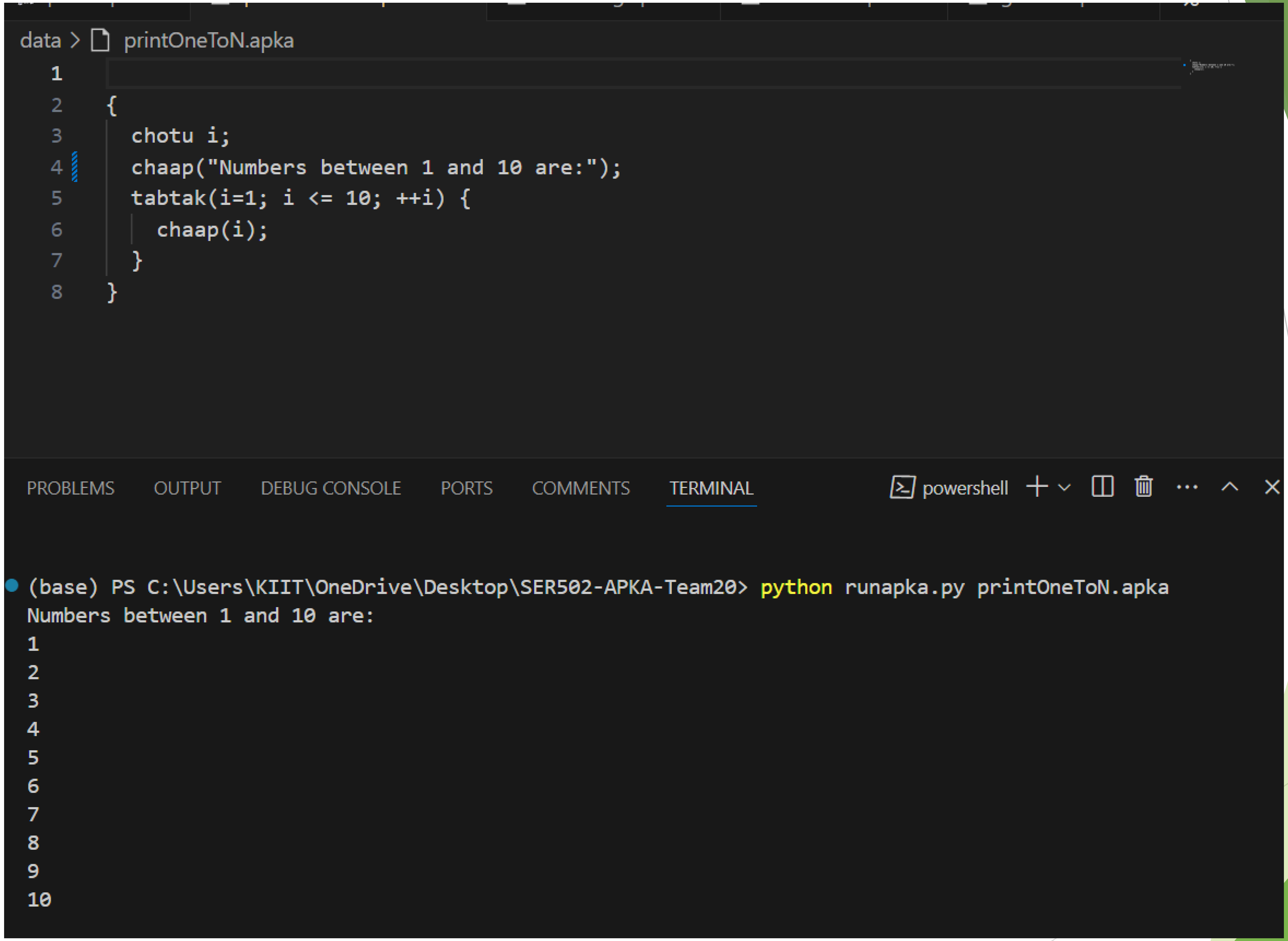


```
If_statement → if ( Condition )  { Statement  } else {  Statement_list }

While_loop → jabtak ( Condition ) { Statement_list }

For_loop → tabtak ( Assignment ; Condition ; Assignment ) { Statement_list }

For_range_loop → tabtak( Variable se range(From ; To)
{ Statement_list }

Print → chaap( “Print_Values“)
Print_Values → Int | String | Identifier

Expression → Expression + Term
        | Expression - Term
        | Term
Term → Term * Factor
        | Term / Factor
        | Factor
Factor → Integer | Boolean | String | Variable | ( Expression )
    | String
    | Bool
    | Variable
    | Expression Operator Expression
    | ( Expression )
    | Ternary operator
```



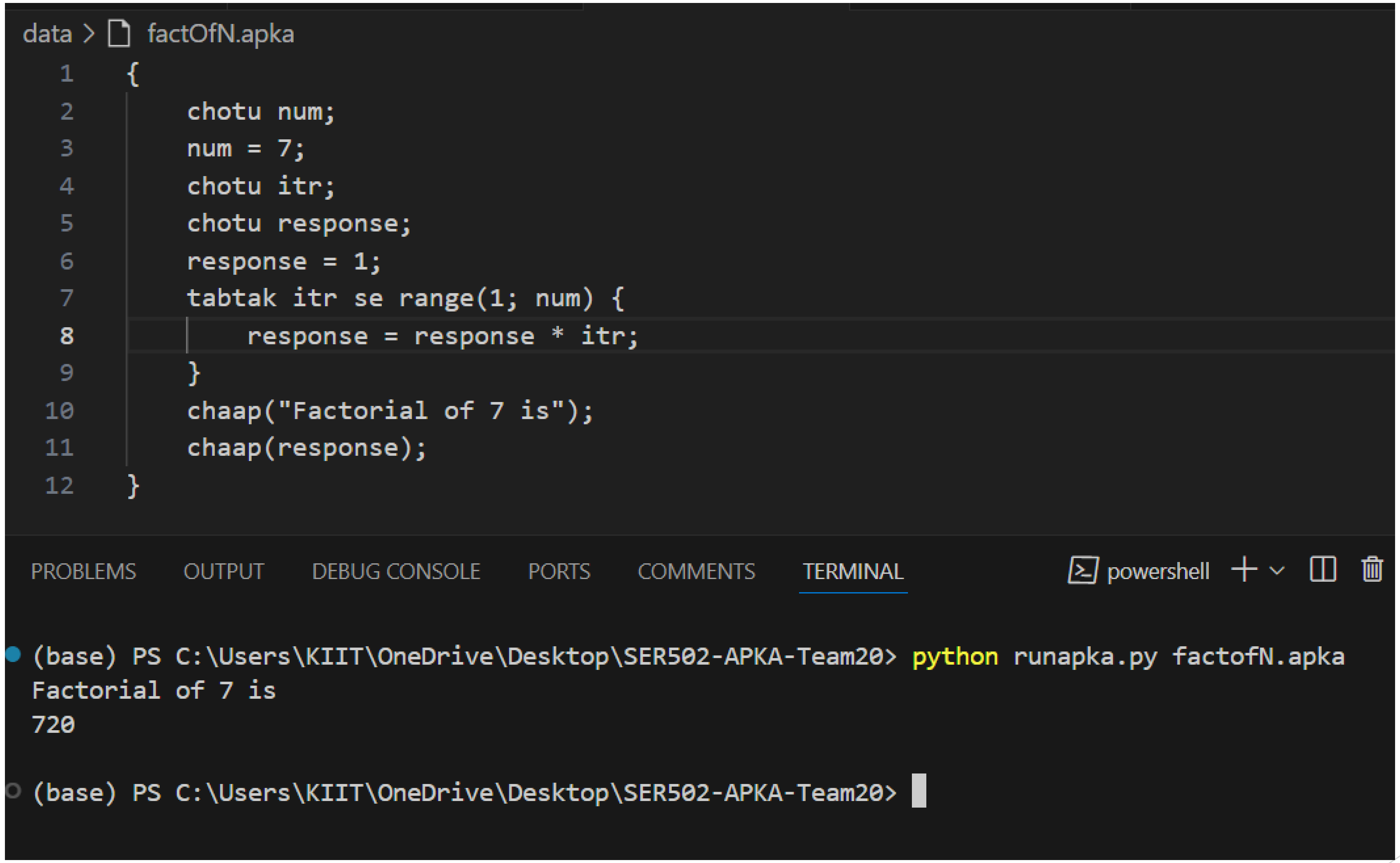
```
    | Ternary_operator

Condition → Expression Relational_Op Expression
| Expression Logical_Op Expression
| Boolean
| ! Condition


Ternary_operator → Condition? Expression : Expression

Relational_operator → ==
    | <
    | >
    | <=
    | >=
    | !=

Int → [0-9]+

Bool → sach
    | jhoot


Logical_Op → and
    | or

String → [a-zA-Z\S]+
Variable → [a-zA-Z0-9]+
```

Output Snippets



```

{
  chotu i;
  chaap("Numbers between 1 and 10 are:");
  tabtak(i=1; i <= 10; ++i) {
    chaap(i);
  }
}
```

```
(base) PS C:\Users\KIIT\OneDrive\Desktop\SER502-APKA-Team20> python runapka.py printOneToN.apka
Numbers between 1 and 10 are:
1
2
3
4
5
6
7
8
9
10
```



```
{
    chotu num;
    num = 7;
    chotu itr;
    chotu response;
    response = 1;
    tabtak itr se range(1; num) {
        response = response * itr;
    }
    chaap("Factorial of 7 is");
    chaap(response);
}
```

```
(base) PS C:\Users\KIIT\OneDrive\Desktop\SER502-APKA-Team20> python runapka.py factofN.apka
Factorial of 7 is
720

(base) PS C:\Users\KIIT\OneDrive\Desktop\SER502-APKA-Team20>
```

THANK YOU!