Program Structure

<program> ::= <function_definitions> <main_block>

<function_definitions> ::= <function_definition> <function_definitions> | ϵ

<function_definition> ::= def <func_name>(): NL_T(\n) TAB_T(indent) <statements> NL_T(\n) TAB_T(dedent)

<main_block> ::= <statements>

<func_name> ::= MNID_T

<comments> ::= CMT_T

Statements and Data Declarations

<statements> ::= <statement> <statements> | <statement> | ϵ

<statement> ::= <variable_declaration> | <assignment_statement> | <selection_statement> | <iteration_statement> | <input_statement> | <output_statement> | <function_call_statement>

<variable_declaration> ::= <variable_initialization> COL_T <type_annotation>

<type_annotation> ::= int | float | str | bool

<variable_initialization> ::= VAR_T = <expression>

Assignment, Input, and Output Statements

<assignment_statement> ::= VAR_T = <expression>

<input_statement> ::= input(<variable_list>);

<output_statement> ::= print(<expression>);

<variable_list> ::= VAR_T | <variable_list>, VAR_T

Expressions

<expression> ::= <arithmetic_expression> | <string_expression> | <variable> | <function_call> | <conditional_expression> | <logical_expression> | <relational_expression>

<arithmetic_expression> ::= <additive_arithmetic_expression>

<additive_arithmetic_expression> ::=
<additive_arithmetic_expression> + <multiplicative_arithmetic_expression>
| <additive_arithmetic_expression> - <multiplicative_arithmetic_expression>
| <multiplicative_arithmetic_expression>

<multiplicative_arithmetic_expression> ::=
<multiplicative_arithmetic_expression> * <exponential_expression>
| <multiplicative_arithmetic_expression> / <exponential_expression>
| <multiplicative_arithmetic_expression> % <exponential_expression>
| <exponential_expression>

<exponential_expression> ::=
<exponential_expression> ^ <primary_expression>
| <primary_expression>

<primary_expression> ::= INL_T | FLT_T | <variable> | (<expression>)

<string_expression> ::= STR_T | <string_expression> + <expression>

<conditional_expression> ::= <logical_expression> | <relational_expression>

<logical_expression> ::= <logical_OR_expression>

<logical_OR_expression> ::= <logical_AND_expression> | <logical_OR_expression> || <logical_AND_expression>

<logical_AND_expression> ::= <logical_NOT_expression> | <logical_AND_expression> && <logical_NOT_expression>

<logical_NOT_expression> ::= ! <relational_expression> | <relational_expression>

<relational_expressions> ::= <expression> <relational_operator> <expression>

<relational_operator> ::= OP_EQ | OP_NE | OP_GT | OP_LT

Control Structures

```
<selection_statement> ::=     If (<conditional_expression>) COL_T
                                      NL_T(\n) TAB_T(indent) <opt_statements>
                              NL_T(\n) TAB_T(dedent)
                              <optional_elif_statements>
                              <optional_else_statement>

<optional_elif_statements> ::= <elif_statement> <optional_elif_statements> | ϵ

<elif_statement> ::=          elif (<conditional_expression>) COL_T
                                      NL_T(\n) TAB_T(indent) <opt_statements>
                              NL_T(\n) TAB_T(dedent)

<optional_else_statement> ::= else COL_T
                                      NL_T(\n) TAB_T(indent) <opt_statements>
                              NL_T(\n) TAB_T(dedent) | ϵ

<iteration_statement> ::=     while (<conditional_expression>) COL_T
                                      NL_T(\n) TAB_T(indent) <statements>
```

Function Call

```
<function_call_statement> ::= <func_name>() | <func_name>(<expression_list>)

<expression_list> ::= <expression> | <expression_list>, <expression>
```