## 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>