Program & Blocks

```
program              =     ( expression | block ) { expression |
block }

block                =     while_block | for_loop_block | if_block

while_block          =     "while" expression":"block_body
{block_body}

for_loop_block       =     "for" id "in" ("range" "(" digit_list
")" | list | tuple | id ) ":" block_body {block_body}

if_block             =     if_header  block_body { else_if_header
block_body} [ else_header block_body]

if_header            =     "if" expression ":"

else_if_header       =     "elif" expression ":"

else_header          =     "else" ":"

block_body           =     NEWLINE TAB expression
```

Functions

```
function_dec         =     "def" id "(" [ parameter { "," parameter
} ] ")" "->" type ":" function_body

function_name        =     id | id"."id

function_body        =     {block_body} NEWLINE [TAB "return"
expression]

function_call        =     function_name"(" [ expression { ","
expression  } ] ")"

parameter            =     id ":" type
```

Expressions

```
expression           =     numeric_term | bool | character | list |
tuple | str | arth_expression | comp_expression | function_call |
"(" expression ")"

arth_expression      =     (expression arth_operator expression) |
(unary_arth_operator arth_expression)
```

```
comp_expression       =    (expression binary_comperator
expression) | (unary_comp_operator expression)
```

Operators

```
unary_arth_operator   =    "-"

binary_arth_operator  =    "+" | "-" | "*" | "/" | "%"

unary_comp_operator   =    "not" | "!"

binary_comp_operator  =    "+" | "-" | "*" | "/" | "%"|">"["="] |
"<"["="] | "==" | "!=" | "or" | "and" | "&" | "|" | "^"
```

Types

```
type                  =    primative_type | non_primative_type

primative_type        =    "int" | "str" | "float" | "bool"

non_primative_type    =    "[" type "]" | "(" type ")"

numeric_term          =    integer | float | id

integer               =    [ "-" | "+" ] digit_list

float                 =    integer ["." {digit} ]

list                  =    "[" [ expression { "," expression } ]
"]"

tuple                 =    ("(" ")") | ("("  expression { ","
expression }  ")")

bool                  =    "True" | "False"

character             =    alphabet | digit | symbol_char

str                   =    "\"" {character} "\"" | "\’" {character}
"\’"
```

Variables & Variable Assignments

```
var_assignment        =    id [":" type] "=" expression
```

```
id                    =     ("_" | character) { character | digit |
"_"}
```

Low-level Definitions

```
NEWLINE               =     "\n"

TAB                   =     "\t"

digit_list            =     "0" | ( non_zero_digit {digit} )

digit                 =     "0" | non_zero_digit

non_zero_digit        =     "1" | "2" | "3" | "4" | "5" | "6" | "7"
| "8" | "9"

alphabet              =     "a" | "b" | "c" | "d" | "e" | "f" | "g"
| "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r"
| "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z" |"A" | "B" | "C" |
"D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" |
"O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" |
"Z"

symbol_char           =     " " | "!" | "@" | "#" | "$" | "%" | "^"
| "&" | "*" | "(" | ")" | "-" | "_" | "+" | "=" | "{" | "}" | "["
| "]" | "<" | ">" |"."  | "," | "?" | "/" | ";" | ":" | "\" | "|"
| "\'" | "\"" | "`" | "~"
```