Program & Blocks

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

block                =    while_block | for_loop_block | if_block

while_block          =    "while" (comp_expression | cond_term)
":"  block_body {block_body}

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

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

if_header            =    "if" (comp_expression | cond_term) ":"

else_if_header       =    "elif" (comp_expression | cond_term) ":"

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           =    id | number | bool | character | list |
tuple | str | arth_expression | comp_expression | function_call


arth_expression      =    arth_no_paren_expression | "("
arth_no_paren_expression ")"
```

```
comp_expression      =    comp_no_paren_expression | "("
comp_no_paren_expression ")"

arth_no_paren_expression   =    (numeric_term | arth_expression)
arth_operator (numeric_term | arth_expression) |
(unary_arth_operator (numeric_term | arth_expression))


comp_no_paren_expression   =    (numeric_term | comp_expression)
comp_operator (numeric_term | comp_expression) |
(unary_comp_operator (numeric_term | comp_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         =    number | id

number               =    integer | float

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

float                =    [ "-" | "+" ] digit_list ["." {digit} ]

cond_term            =    cond_no_paren_term | "("
cond_no_paren_term ")"

cond_no_paren_term   =    bool | number | character

iterable_term        =    list | tuple
```

```
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            =    lowercase | uppercase

lowercase           =    "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"

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