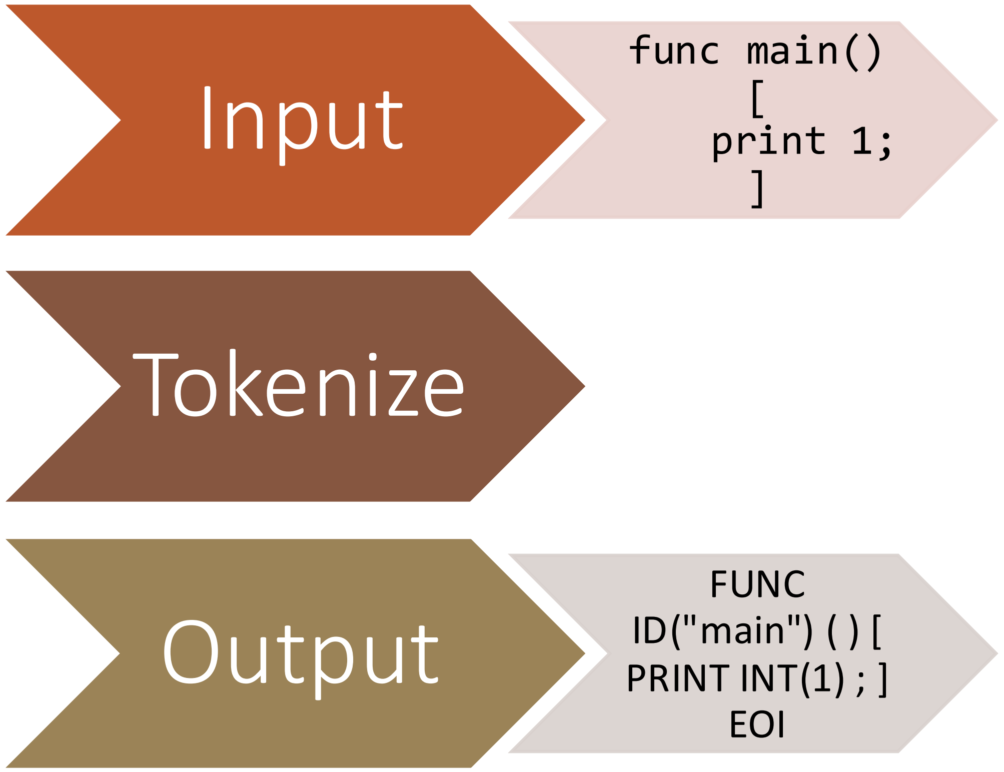
Tiny Programming Language (TPL)

MEGAN EDDER AND KAY PERSCHKA

Lexical Analysis

- Recognizes Tokens:
  - Identifiers
  - Keywords
  - Operators
  - Parenthesis/Brackets
  - End-Of-Input
  - General Error



```
func main()
  [
    print 1;
  ]
```

```
FUNC
ID("main") ( ) [
PRINT INT(1) ; ]
EOI
```

Parsing
Grammar Rules

Program Structure program = { function } ;

Function Definitions function = "func" ID "(" [parameters]")" block ;

parameters = ID { "," ID } ;

Blocks block = "[" { statement } "]" ;

- let_stmt = "let" ID ";" ;
- assign_stmt = ID "=" expression ";" ;
- expr_stmt = expression ";" ;
- if_stmt = "if" expression block "else" block ;
- while_stmt = "while" expression block ;
- return_stmt = "return" expression ";" ;
- print_stmt = "print" expression ";" ;

Pratt Parsing

Expression (Pratt)
expression
= logic_or ;

logic_or =logic_and {
"|" logic_and } ;

logic_and = equality
{"&" equality } ;

equality = relational {
("==" | "!="
)relational } ;

relational = additive {
("<" | ">")additive }
;

additive =
multiplicative { ("+" |
"-")multiplicative } ;

multiplicative = unary
{ ("*" | "/")unary } ;

unary = ("!" | "-")
unary | primary ;

primary = INT | BOOL
| ID | function_call |
"("expression ")" ;

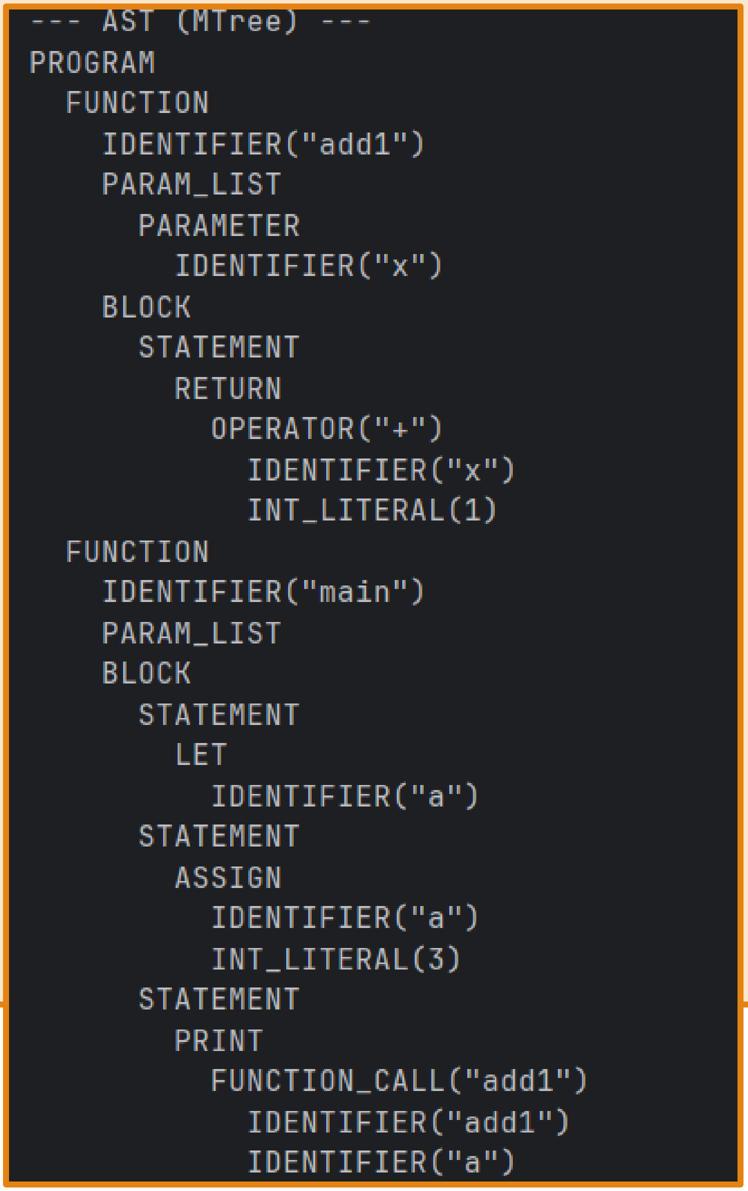
function_call = ID "("
[arguments]")" ;

arguments =
expression { ","
expression } ;

Parse Tree Example

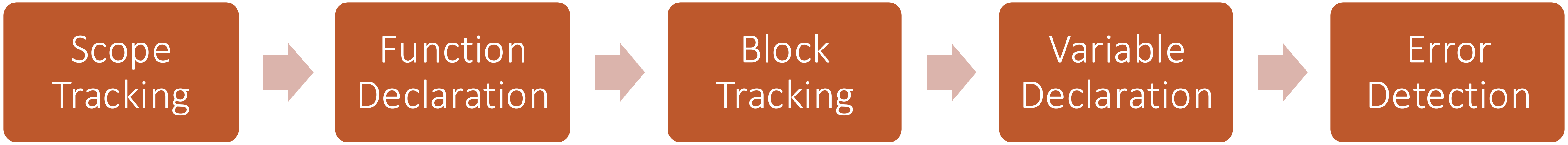
- TreeCode Tokens
- Rc<RefCell<MTree>>

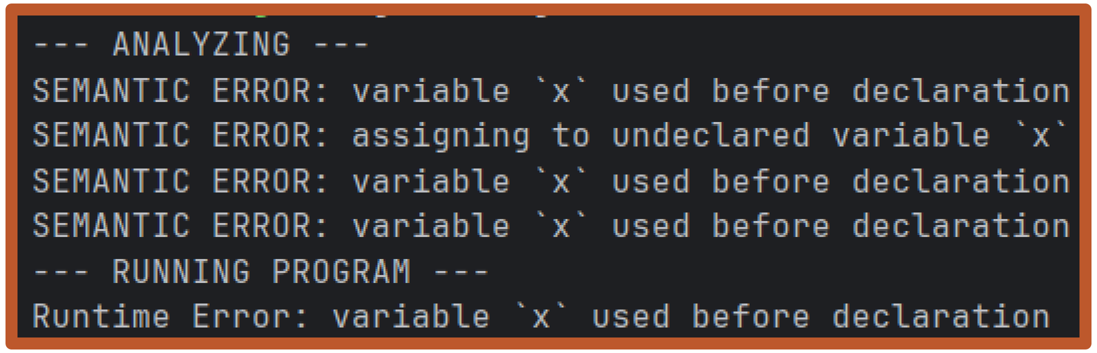
```
func add1(x) [
    return x + 1;
]

func main() [
    let a;
    a = 3;

    print add1(a);
]
```



```
--- AST (MTree) ---
PROGRAM
  FUNCTION
    IDENTIFIER("add1")
    PARAM_LIST
      PARAMETER
        IDENTIFIER("x")
    BLOCK
      STATEMENT
        RETURN
          OPERATOR("+")
            IDENTIFIER("x")
            INT_LITERAL(1)
  FUNCTION
    IDENTIFIER("main")
    PARAM_LIST
    BLOCK
      STATEMENT
        LET
          IDENTIFIER("a")
      STATEMENT
        ASSIGN
          IDENTIFIER("a")
          INT_LITERAL(3)
      STATEMENT
        PRINT
          FUNCTION_CALL("add1")
            IDENTIFIER("add1")
            IDENTIFIER("a")
```

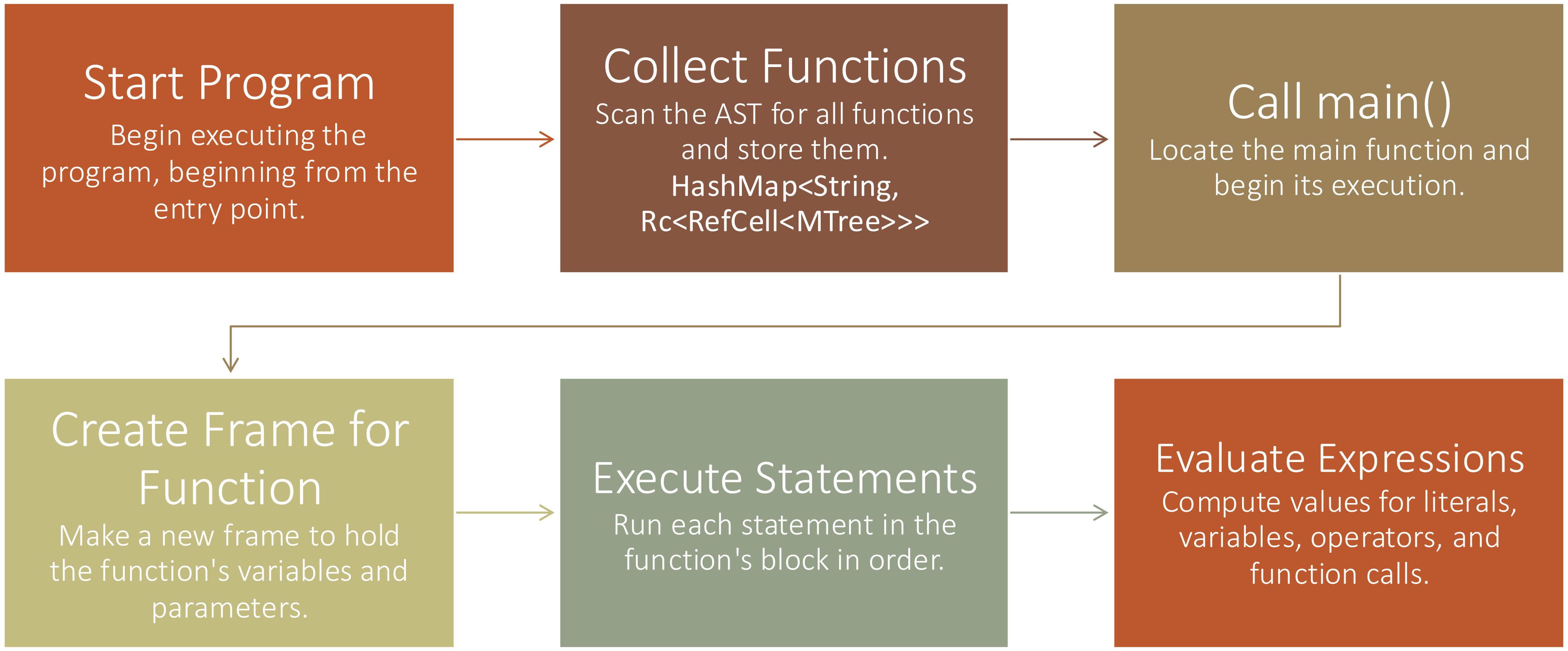
Semantic Analysis





```
--- ANALYZING ---
SEMANTIC ERROR: variable `x` used before declaration
SEMANTIC ERROR: assigning to undeclared variable `x`
SEMANTIC ERROR: variable `x` used before declaration
SEMANTIC ERROR: variable `x` used before declaration
--- RUNNING PROGRAM ---
Runtime Error: variable `x` used before declaration
```

Execution Flow



Start Program
Begin executing the program, beginning from the entry point.

Collect Functions
Scan the AST for all functions and store them.
HashMap<String, Rc<RefCell<MTree>>>

Call main()
Locate the main function and begin its execution.

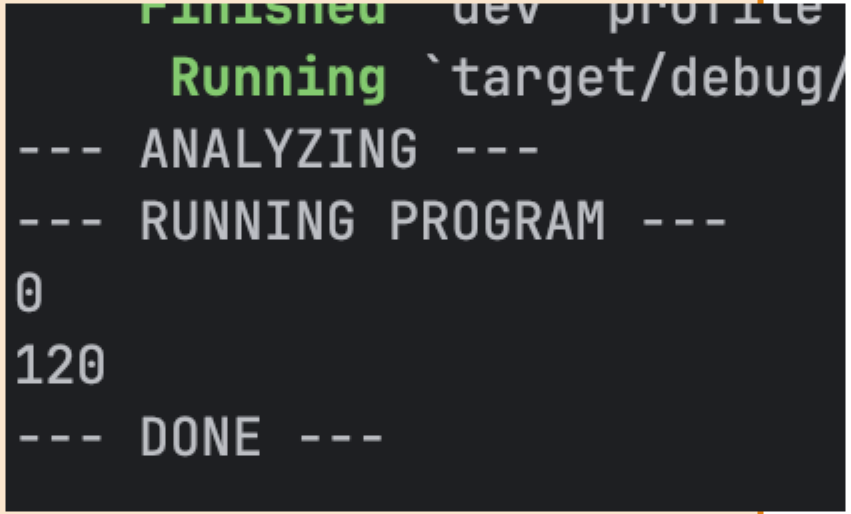
Create Frame for Function
Make a new frame to hold the function's variables and parameters.

Execute Statements
Run each statement in the function's block in order.

Evaluate Expressions
Compute values for literals, variables, operators, and function calls.

Execution Example

```
func factorial_recursion(n)
[
    if n < 2 [
        return 1;
    ] else [
        return n * factorial_recursion(n-1);
    ]
]

func factorial_loop(n)
[
    let p;
    p = n;
    while n > 0 [
        n = n - 1;
        p = p * n;
    ]
    return p;
]

func main()
[
    let n;
    n = 5;
    print factorial_loop(n);
    print factorial_recursion(n);
]
```



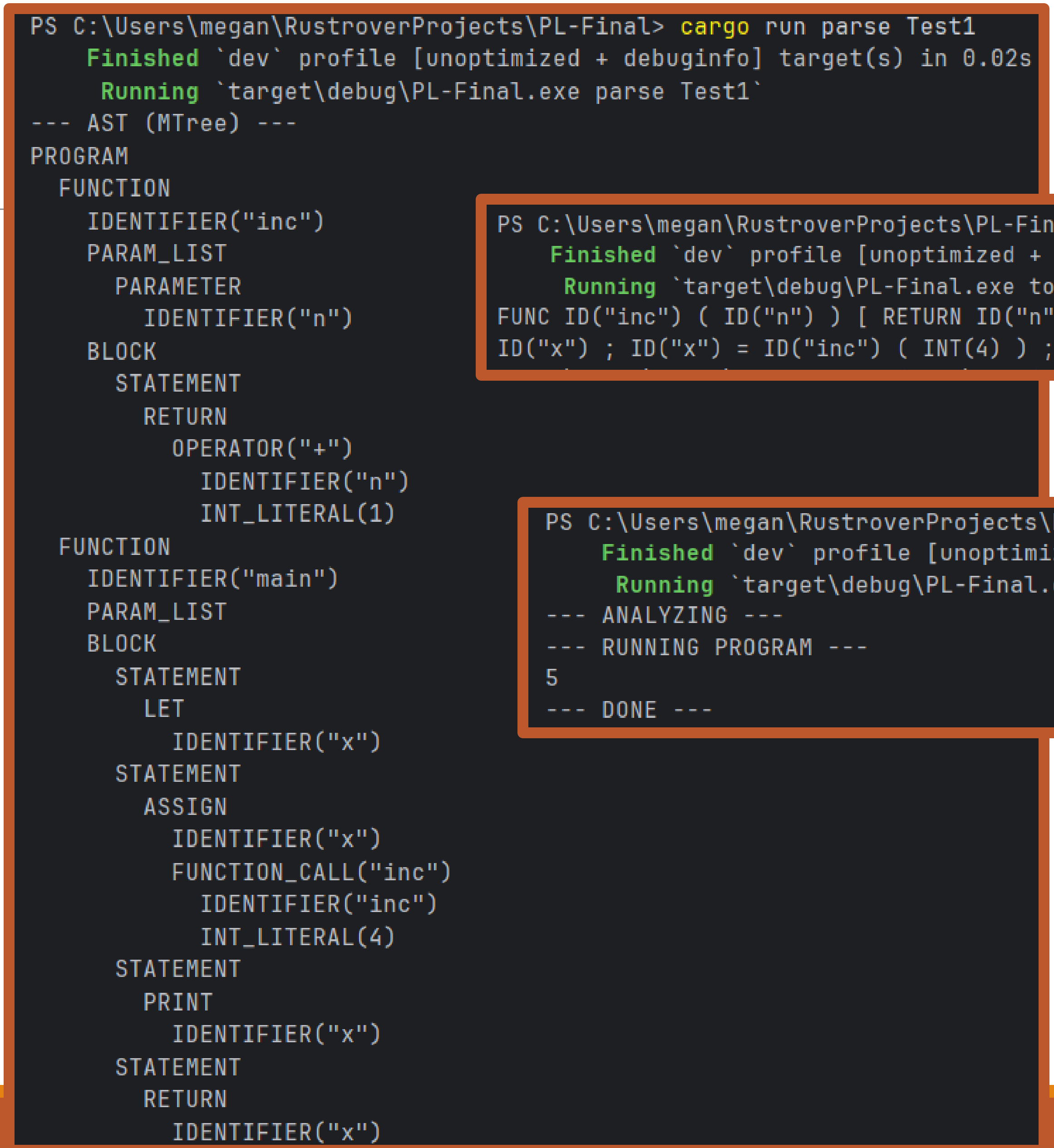
```
    Finished   dev   profile
     Running `target/debug/
--- ANALYZING ---
--- RUNNING PROGRAM ---
0
120
--- DONE ---
```

Command Line Integration

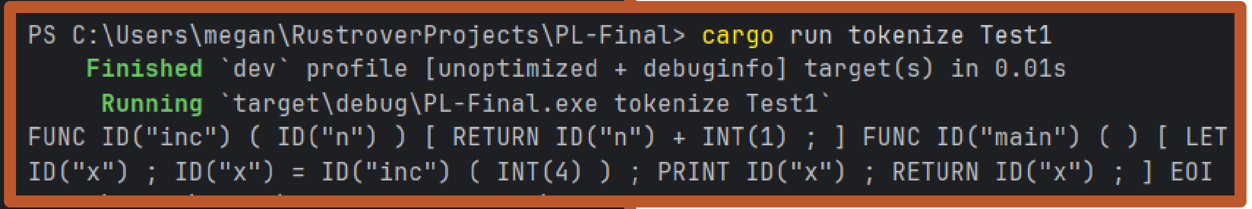
Test1 Contents:

```
func inc(n) [
    return n + 1;
]

func main() [
    let x;
    x = inc(4);
    print x;
    return x;
]
```

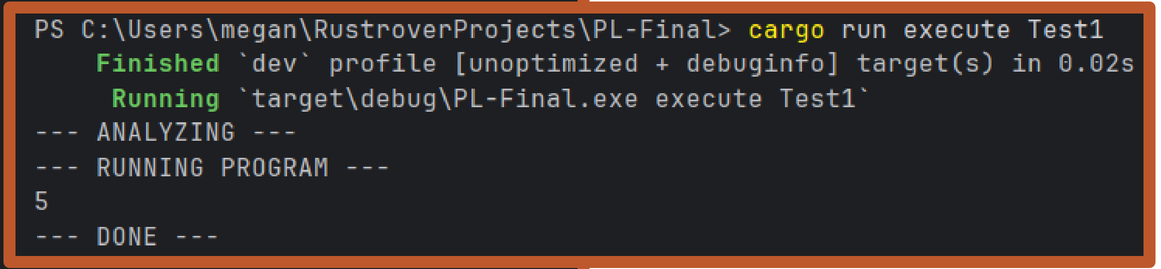
$ cargo run <tokenize|parse|execute> <file>



```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run parse Test1
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.02s
     Running `target\debug\PL-Final.exe parse Test1`
--- AST (MTree) ---
PROGRAM
  FUNCTION
    IDENTIFIER("inc")
    PARAM_LIST
      PARAMETER
        IDENTIFIER("n")
    BLOCK
      STATEMENT
        RETURN
          OPERATOR("+")
            IDENTIFIER("n")
            INT_LITERAL(1)
  FUNCTION
    IDENTIFIER("main")
    PARAM_LIST
    BLOCK
      STATEMENT
        LET
          IDENTIFIER("x")
      STATEMENT
        ASSIGN
          IDENTIFIER("x")
          FUNCTION_CALL("inc")
            IDENTIFIER("inc")
            INT_LITERAL(4)
      STATEMENT
        PRINT
          IDENTIFIER("x")
      STATEMENT
        RETURN
          IDENTIFIER("x")
```



```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run tokenize Test1
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.01s
     Running `target\debug\PL-Final.exe tokenize Test1`
FUNC ID("inc") ( ID("n") ) [ RETURN ID("n") + INT(1) ; ] FUNC ID("main") ( ) [ LET
ID("x") ; ID("x") = ID("inc") ( INT(4) ) ; PRINT ID("x") ; RETURN ID("x") ; ] EOI
```



```
PS C:\Users\megan\RustroverProjects\PL-Final> cargo run execute Test1
    Finished `dev` profile [unoptimized + debuginfo] target(s) in 0.02s
     Running `target\debug\PL-Final.exe execute Test1`
--- ANALYZING ---
--- RUNNING PROGRAM ---
5
--- DONE ---
```