SYNC

An Imperative Language

Data Types

SYNC uses val as data type which supports int, bool and string values.

Conditional Statements

SYNC supports both traditional if and else statements and ternary operators.

Loops

while loop

while(BOOLEAN)

{ BLOCK }

for loop

for(INITIALIZATION ; BOOLEAN ; UNARY)

{ BLOCK }

for loops with range
For IDENTIFIER in range (EXPRESSION,EXPRESSION)
{BLOCK}

If loop
if(BOOLEAN)
{ BLOCK }
else if(BOOLEAN)
{ BLOCK }

Print Statements

The print statement in SYNC is “print ”.

ex: print(“Tom Brady is the GOAT”)

- $ indicates end of statement in this language.
- The lexer converts the input program into tokens. These tokens are parsed by parser and a parse tree is generated. This parse tree is then interpreted to give expected output.
- Like in Python, there is no need to declare the variable.

Grammar

P --> PROGRAM

K --> BLOCK

Id --> IDENTIFIER

D --> DECLARATION

I --> INITIALIZATION

E --> EXPRESSION

B --> BOOLEAN EXPRESSION

U --> UNARY

T --> TERNARY

A--> ASSIGN

P ::= K

K ::= Statements K | Statements

Statements ::= D $ | I $ | A $ | IF | while B { K } | FOR | print $ | U $

D::= val Id$

I::= val Id = E$ | val Id = S$

A::= Id = E$ | Id = B$ | Id = S$

U::= Id++$ | Id--$

IF::= if B { K } ELSE_CASE | if E { K } ELSE_IF_CASE

ELSE_IF_CASE::= else if B { K } ELSE_IF_CASE

ELSE_CASE::=else { K }| empty

FOR::= for (I ; B ; U) { K } | for Id in range (E , E) { K }

print::= print(“S”)$ | print(Id)$ | print(“S”,Id)$

B ::= true | false | not B | B or B | B and B | E

C ::= E < E | E > E | E<= E | E >= E | E == E

E ::= E + E | E - E | E * E | E / E | Id | N | T

Id::= [a-z] Id* | [A-Z] Id*

Id* ::= [a-z] | [A-Z] | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | empty

N::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

T::= (B) ? (E : E)

Sample

Sample SYNC Code

val x = 45 $

print(“ value of x = “, x)$

OUTPUT

value of x = 45