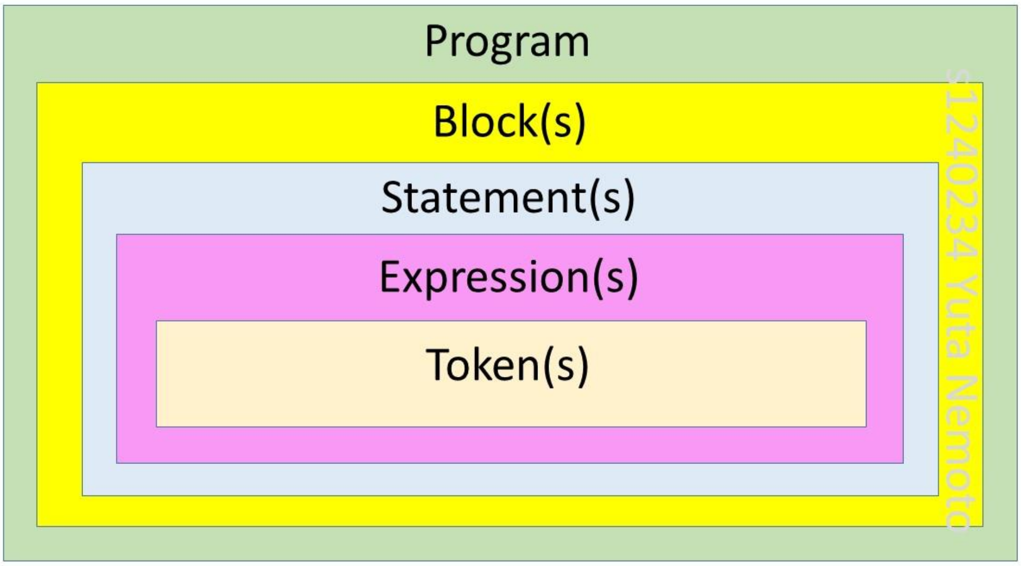
## Quiz 4

### 1. Briefly define the terms: syntax, syntax analysis, and parsing?

- **Syntax**: the way in which words are put together to form phrases, clauses, or sentences. The rules governing the formation of statements in a programming language.
- **Syntax analysis**: the task concerned with fitting a sequence of tokens into a specified syntax.
- **Parsing**: to break a sentence down into its component parts of speech with an explanation of the form, function, and syntactical relationship of each part.

### 2. Informally explain the syntactic structure of programming languages?

The syntactic structure of programming languages can be informally expressed by the following diagram.



### 3. Give 3 examples of parsing (syntax) errors?

1. Misspelling of identifier, keyword, or operator. For example, the missing of operator in the sentence "x = 5    7 + 3".
2. Arithmetic expression with unbalanced parentheses. For example, the error in the structure of the sentence (number of closing brackets) "x = ((a+b*c)".
3. Punctuation errors such as using comma in place of semicolon. For example, the missing of ';' in the sentence "a = 5".

### 4. Define context-free grammars?

A context-free grammar G = (T, N, S, P) consists of:

1. T, a set of terminals (scanner tokens). Such as the factor of "class" or brackets.
2. N, a set of nonterminals (syntactic variables generated by productions). Generally the definitions of the program rules such as the definition of "program", "block", and so on.
3. S, a designated start nonterminal.
4. P, a set of productions. Each production has the form, A::= $\alpha$ , where A is a nonterminal and $\alpha$

is a sentential form, i.e., a string of zero or more grammar symbols (terminals/nonterminals).

## 5. Give the definition of left recursive grammar? And give an example of it?

**Definition:**

A grammar that has at least one production of the form A → A $\alpha$ is a left recursive grammar.

**Example:**

E → E + T | T  
T → T * F | F  
F → ( E ) | **id**

## 6. Briefly explain (with an example) about left factoring in grammars?

For example, the grammar: stmt → if expr then stmt else stmt | if expr then stmt

Cannot be parsed by a predictive parser that looks one element ahead. But the grammar can be re-written like: stmt → if expr then stmt stmt' stmt' → else stmt | $\lambda$

Where $\lambda$ is the empty string.

Rewriting a grammar to eliminate multiple productions starting with the same token is called **left factoring**.

## 7. For the following grammar, find the FOLLOW(T') and FIRST(S)?

**S → E $**  
**E → T E'**  
**E' → + T E' | - T E' | $\lambda$**  
**T → F T'**  
**T' → * F T' | / F T'| $\lambda$**  
**F → id | num | (E)**

**FOLLOW(T') = {+, -, ), $}**  
**FIRST(S) = FIRST(E) = FIRST(T) = FIRST(F) = {id, num, ( }**