**LAB PROJECT**

**Sameer Arif Khan 2020430**

**Shahryar Mubashar 2020447**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**CFG**

**Grammar Suggestions for string manipulation:**

1. **Statement:**

- `<Statement> ::= <Assignment> | <Expression>`

2. **Assignment:**

- `<Assignment> ::= <Identifier> "=" <Expression>`

3. **Expression:**

- `<Expression> ::= <Concatenation> | <SubString> | <Length>`

4. **Concatenation:**

- `<Concatenation> ::= <String> "+" <String>`

5. **SubString:**

- `<SubString> ::= "substring" "(" <String> "," <Index> "," <Length> ")"`

6. **Length:**

- `<Length> ::= "length" "(" <String> ")"`

7. **String:**

- `<String> ::= <Identifier> | "\"" <Text> "\""`

8. **Identifier:**

- `<Identifier> ::= [a-zA-Z]+`

9. **Index:**

- `<Index> ::= [0-9]+`

10. **Text:**

- `<Text> ::= [a-zA-Z0-9\s]+`

This grammar allows for basic string manipulation operations like concatenation, substring extraction, and length calculation.

**Grammar Rules:**

1. **Statement:**

- Explanation: Statements can be assignments or expressions.

- Example: `result = "Hello" + "World"`

2. **Assignment:**

- Explanation: Assignments assign a value to a variable.

- Example: `result = "Hello" + "World"`

3. **Expression:**

- Explanation: Expressions represent operations like concatenation, substring extraction, or length calculation.

- Example: `"Hello".length()`

4. **Concatenation:**

- Explanation: Concatenation joins two strings together.

- Example: `"Hello" + "World"`

5. **SubString:**

- Explanation: Substring extracts a portion of a string.

- Example: `"Hello".substring(0, 3)`

6. **Length:**

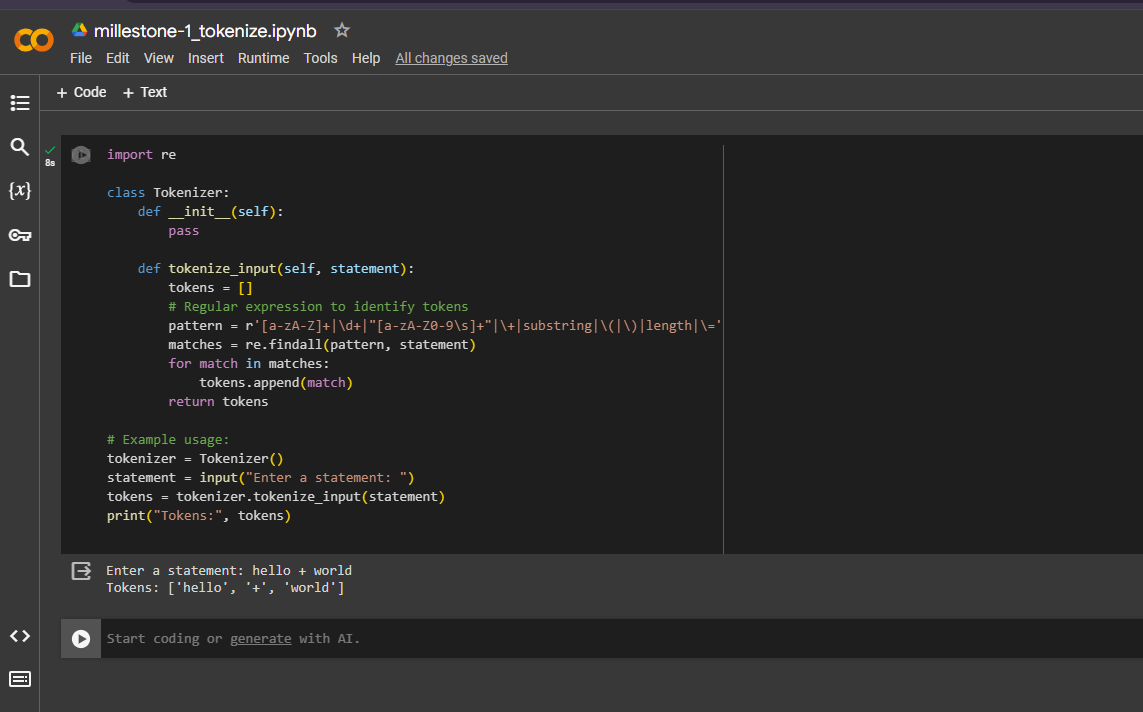
- Explanation: Length calculates the number of characters in a string.

- Example: `"Hello".length()`

**Ambiguities/Questions:**

- None identified for the time being.

**Code:**

****

**Explanation:**

- Implemented a `Tokenizer` class with a `tokenize\_input` method to handle input.

- Used regular expressions to identify tokens efficiently.

- Error handling ensures graceful handling of invalid characters or syntax errors.

- Tokens are stored in a list for further processing.

This implementation aligns with the suggested grammar for a string manipulation language and fulfills the requirements for Milestone 1.