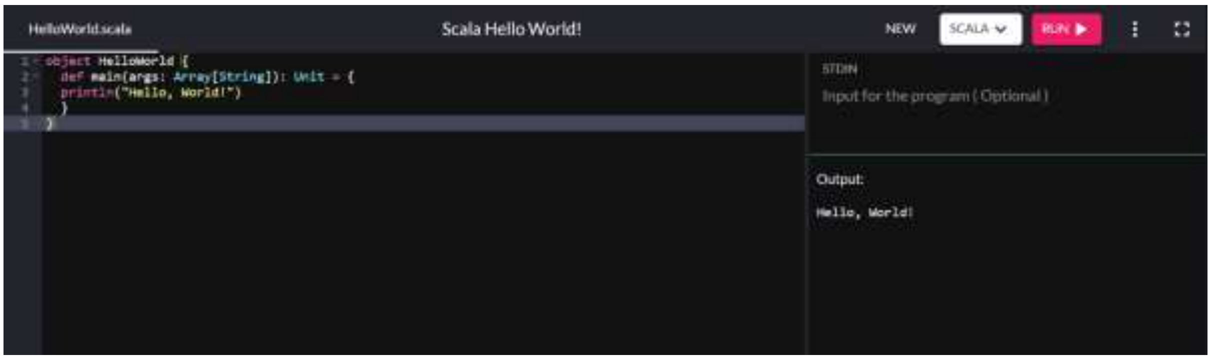
# Experiment-1

Q1) Write a Hello World program in scala.



```
object HelloWorld {
  def main(args: Array[String]): Unit = {
  println("Hello, World!")
  }
}
```

```
Hello, World!
```

Q2) Different Identifiers in scala.

In Scala, we have 4 different types of identifiers defined. They are:

1. Alphanumeric Identifiers
2. Operator Identifiers
3. Mixed Identifiers
4. Literal Identifiers

**1) Alphanumeric Identifiers**

Alphanumeric identifiers, as the name implies, are composed of letters, numbers, and underscores. To comply with the rule that identifiers cannot begin with a number, an alphanumeric identifier starts with an underscore (_) or a character followed by a number.

Example:

Scala123, is12thDigit, _value1

**2) Operator Identifiers**

Operator identifiers consist exclusively of operators, meaning they are composed of one or more operators in Scala. The included operators are '+', '#', '*', '<='.

**3) Mixed Identifiers**

True to their name, mixed identifiers combine elements from the two aforementioned types. They consist of an alphanumeric identifier followed by an operator identifier, with an underscore in between. Example: avg_+, val_=

**4) Literal Identifiers**

A string literal used as an identifier in Scala is a literal identifier. These are strings enclosed inside ticks (‘...’).

Example:

‘scala’, ‘value’

Q3) Implementation of ‘var’ and ‘val’, and multiple variable assignment.

In Scala, the keywords 'var' and 'val' are employed for declaring variables, each possessing distinct characteristics concerning mutability. Moreover, Scala allows the concise and expressive practice of multiple variable assignments in a single line.

**var - Mutable Variables:**

- Variables declared with 'var' are mutable, signifying that their values can be altered or reassigned after initialization.
- Utilize 'var' when a variable's value needs to be modifiable throughout the program.

Example:

var counter: Int = 10

counter = counter + 1

**val - Immutable Variables:**

- Variables declared with 'val' are immutable, indicating that their values cannot be changed once assigned.
- Use 'val' when creating a constant or ensuring that the variable's value remains unchanged

Example:

```
val pi: Double = 3.14159

// Attempting to reassign will result in a compilation error

// pi = 3.14  // This line will produce an error
```

**Multiple Variable Assignment:**

Scala permits multiple variable assignments in a single line using the pattern-matching syntax. The number of variables on the left side should match the number of values on the right side.

Syntax:

```
val (var1, var2, ..., varN) = (value1, value2, ..., valueN)
```

Example:

```
val (name, age, country) = ("Alice", 25, "Wonderland")

// Now, name is "Alice", age is 25, and country is "Wonderland"
```