# The Scala Interpreter

## Install Scala.

- curl -s "https://get.sdkman.io" | bash
- source "$HOME/.sdkman/bin/sdkman-init.sh"
- sdk install scala
- scala -version

## Run "Hello World"

**Create a Scala File**

- nano Hello.scala

**Scala Code**

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

**Save and Exit Nano**

- Ctrl + O to write the file
- Enter to confirm the filename
- Ctrl + X to exit

**Compile the Scala Program**

- scalac Hello.scala

**Run the Program**

- scala Hello

```
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ scala -version
Scala code runner version 2.11.12 -- Copyright 2002-2017, LAMP/EPFL
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ scalac Hello.scala
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ scala Hello
Hello, Scala!
```

**Experiment with Scala Basics**

- scala

**Data types & Variables**

- val x: Int = 5
- var y = "Scala"

**Operators & Conditionals**

- if (x > 3) println("Greater")

**Loops**

- for (i <- 1 to 5) println(i)

```
scala> :quit
hadoop@bmscecse-HP-Elite-Tower-600-G9-Desktop-PC:~$ scala
Welcome to Scala 2.11.12 (OpenJDK 64-Bit Server VM, Java 11.0.26).
Type in expressions for evaluation. Or try :help.

scala> val x: Int = 5
x: Int = 5

scala> var y = "Scala"
y: String = Scala

scala> if (x > 3) println("Greater")
Greater

scala> for (i <- 1 to 5) println(i)
1
2
3
4
5
```

**Work with Functions**

- def add(a: Int, b: Int): Int = a + b
- println(add(3, 4))

**Collections**

- val list = List(1, 2, 3)
- list.foreach(println)
- val map = Map("a" -> 1, "b" -> 2)
- println(map("a"))

```
scala> def add(a: Int, b: Int): Int = a + b
add: (a: Int, b: Int)Int

scala> println(add(3, 4))
7

scala> val list = List(1, 2, 3)
list: List[Int] = List(1, 2, 3)

scala> list.foreach(println)
1
2
3

scala>

scala> val map = Map("a" -> 1, "b" -> 2)
map: scala.collection.immutable.Map[String,Int] = Map(a -> 1, b -> 2)

scala> println(map("a"))
1
```

**Object-Oriented Programming**

- class Person(name: String) {
  def greet() = println(s"Hello, $name")

}

- val p = new Person("Alice")

p.greet()

**Advanced Features**

- **Traits**
  - trait Greeter {
    def greet(): Unit
    }
  - class EnglishGreeter extends Greeter {
    def greet() = println("Hello")
    }
  - val g = new EnglishGreeter()
  - g.greet()
- **Pattern Matching**
  - def describe(x: Any): String = x match {
    case 1 => "One"
    case "two" => "Two"
    case _ => "Something else"
    }

```
scala> class Person(name: String) {
     |   def greet() = println(s"Hello, $name")
     | }
defined class Person

scala> val p = new Person("Alice")
p: Person = Person@79f82fc4

scala> p.greet()
Hello, Alice

scala> trait Greeter {
     |   def greet(): Unit
     | }
defined trait Greeter

scala>

scala> class EnglishGreeter extends Greeter {
     |   def greet() = println("Hello")
     | }
defined class EnglishGreeter

scala>

scala> val g = new EnglishGreeter()
g: EnglishGreeter = EnglishGreeter@64c79b69

scala> g.greet()
Hello

scala> def describe(x: Any): String = x match {
     |   case 1 => "One"
     |   case "two" => "Two"
     |   case _ => "Something else"
     | }
describe: (x: Any)String
```