return keyword in scala

- ✓ return is a keyword in scala programming language.
- ✓ This return keyword we can apply only on functions and methods concept.
- ✓ Based on return statement we can divide functions are two types.
    - Function without return statement
    - Function with return statement.

1. Function without return statement

- ✓ If a function cannot contain return statement, then that function is called as a function without return statement.

Program Name: A function without return statement
Demo1.scala

```
object Demo1
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                balance()
        }

        def balance()
        {
                println("My balance is: ")
        }
}
```

Compile: scalac Demo1.scala
Run: scala Demo1

Output

```
Welcome to main
My balance is:
```

2. Function with return statement

- ✓ Based on requirement a function can contain return statement.
- ✓ The purpose of writing return statement with function is,
  - a function with return statement can return the result.
- ✓ Let's understand by doing practically.
  - Syntactically we can write return statement to function, while creating function with return then we need to use
    - **:** symbol,
    - Type of the value
    - **=** equals symbol

Syntax

```
def functionName(): Type =
{
        // function body

        return value
}
```

If function having return statement then,

- ✓ That function can,
  - Take input,
  - Process it,
  - returns output.

Program Name: A function with return statement
Demo2.scala

```
object Demo2
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                balance()
        }

        def balance(): Int=
        {
                println("My balance is: ")
                return 100
        }
}
```

Compile: scalac Demo2.scala
Run: scala Demo2

Output

```
Welcome to main
My balance is:
```

Important point on return statement

- ✓ If a function contains return statement then while calling that function, that function calling we need to assign to a variable.

Program Name: A function with return statement
Demo3.scala

```
object Demo3
{
        def main(args: Array[String])
        {
                println("Welcome to main")

                var b=balance()

                print(b)
        }

        def balance(): Int=
        {
                println("My balance is: ")
                return 100
        }
}
```

Compile: scalac Demo3.scala
Run: scala Demo3

Output

```
Welcome to main
My balance is:
100
```

Program Name: A function with return statement
Demo4.scala

```
object Demo4
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                println(balance())
        }

        def balance(): Int=
        {
                println("My balance is: ")
                return 100
        }
}
```

Compile: scalac Demo4.scala
Run: scala Demo4

Output

```
Welcome to main
My balance is:
100
```

Why we need to assign function calling to a variable?

- ✓ So, this assigned variable will be holding the result of function returned value.
- ✓ This variable we can use further in coding.

Program Name: A function with return statement
Demo5.scala

```
object Demo5
{
	def main(args: Array[String])
	{
		println("Welcome to main")

		var b=balance()

		if(b>=0)
		{
			println("Balance is: "+b)
		}

		else
		{
			println("Balance is negative please deposit")
		}
	}

	def balance(): Int=
	{
		return 100
	}
}
```

Compile: scalac Demo5.scala
Run: scala Demo5

Output

```
Welcome to main
Balance is: 100
```

Program Name: A function with return statement
Demo6.scala

```
object Demo6
{
	def main(args: Array[String])
	{
		println("Welcome to main")

		var b=balance()

		if(b>=0)
		{
			println("Balance is: "+b)
		}

		else
		{
			println("Balance is negative please deposit")
		}
	}

	def balance(): Int=
	{
		return -100
	}
}
```

Compile: scalac Demo6.scala
Run: scala Demo6

Output

```
Balance is negative please deposit
```

return vs Unit type

- ✓ If any function is not return any value, then by default that function returns Unit type.
- ✓ We can also say as, a function which is not having return statement still that function is returning Unit type value.

Program Name: function which having Unit return type
Demo7.scala

```
object Demo7
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                var b=balance()
                print(b)
        }

        def balance()
        {
                println("My balance is: ")
        }
}
```

Compile: scalac Demo7.scala
Run: scala Demo7

Output

```
Welcome to main
My balance is:
()
```

Unit type

- ✓ If any function is not return any value, then by default that function returns Unit type value.
- ✓ So, in this scenario we can assign function return type value as a Unit type, anyway writing Unit type after function name is an optional.

Program Name: function which having Unit return type
Demo8.scala

```
object Demo8
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                var b=balance()
                println(b)

        }

        def balance(): Unit =
        {
                println("My balance is: ")
        }
}
```

Compile: scalac Demo8.scala
Run: scala Demo8

Output

```
Welcome to main
My balance is:
()
```

Syntax surprise

- ✓ Syntactically writing return keyword is an optional in scala programming language.

Program Name: function which having return type
Demo9.scala

```
object Demo9
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                var b=balance()
                println(b)

        }

        def balance(): Int=
        {
                println("My balance is: ")
                return 100
        }
}
```

Compile: scalac Demo9.scala
Run: scala Demo9

Output

```
Welcome to main
My balance is:
100
```

Program Name: function which having return type but return keyword is optional
Demo10.scala

```
object Demo10
{
	def main(args: Array[String])
	{
		println("Welcome to main")
		var b=balance()
		println(b)
	}


	def balance(): Int=
	{
		println("My balance is: ")
		100
	}
}
```

Compile: scalac Demo10.scala
Run: scala Demo10

Output

```
Welcome to main
My balance is:
100
```

Make a note

- ✓ So, we can directly write a value in end of the function without return statement.

Important point about return statement

- ✓ Make sure, function should return corresponding value means,
  - If a function returns type is Int then it should return integer value otherwise we will get error.
  - If a function return type is String, then it should return String value otherwise we will get error

Program Name: A function which is returning String value.
Demo11.scala

```
object Demo11
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                var b= one()
                println(b)
        }

        def one(): String=
        {
                println("My balance is: ")
                return "Hello"
        }
}
```

Compile: scalac Demo11.scala
Run: scala Demo11

Output

```
Welcome to main
This is Nireekshan
Hello
```

Program Name: Error: function return type is Int but returning String value
Demo12.scala

```
object Demo12
{
        def main(args: Array[String])
        {
                println("Welcome to main")
                var b=one()
                println(b)
        }

        def one(): Int=
        {
                println("This is Nireekshan: ")
                return "Hello"
        }
}
```

Compile: scalac Demo12.scala
Run: scala Demo12

Error

```
Demo1.scala:13: error: type mismatch;
found   : String("Hello")
required: Int
                return "Hello"
                       ^
```

Program Name: Error: function return type is Int but returning String value
Demo13.scala

```
object Demo13
{
	def main(args: Array[String])
	{
		println("Welcome to main")
		var b=one()
		println(b)
	}


	def one(): String=
	{
		println("This is Nireekshan: ")
		return 100
	}
}
```

Compile: scalac Demo13.scala
Run: scala Demo13

Error

```
Demo1.scala:14: error: type mismatch;
found   : Int(100)
required: String
            return 100
                   ^
```