Chapter 7: Code Blocks

Exercise:

1. Practice using code blocks by writing a series of statements in them and assigning a value to a variable to see what is returned and stored.
2. Study the caveats of what value gets returned when you use pattern matching, i.e., if you return different values in different case blocks, make sure you understand what happens.

Answer :

Block is multiple lines of code that are enclosed between braces i.e. everything between { and } is in one block code.

```
scala> val x3:String= {
     |     val d = new java.util.Date()
     |     d.toString()
     | }
x3: String = Wed Nov 02 19:50:34 WAT 2022

scala> var age =25
age: Int = 25

scala> if(age>=21){
     | println("Come in")
     | } else {
     | println("Get")}
Come in
```

```
scala> {
     |   val a = "Hello"
     |   val b = "world"
     |   s"$a $b"
     | }
res0: String = Hello world

scala> val n = 42
n: Int = 42

scala> val result = if (n == 42) {
     | "That's the answer to everything"
     | }
result: Any = That's the answer to everything

scala> println(result)
That's the answer to everything
```

Pattern matching is one of Scala's most interesting features. It allows you to check if a value matches one of the defined patterns and runs the associated block of code.

Pattern matching is a way of checking the given sequence of tokens for the presence of the specific pattern. It is the most widely used feature in Scala. It is a technique for checking a value against a pattern. It is similar to the ***switch statement of Java*** and ***C***.

Here, "match" keyword is used instead of switch statement. "match" is always defined in Scala's root class to make its availability to the all objects. This can contain a sequence of alternatives. Each alternative will start from case keyword. Each case statement includes a pattern and one or more expression which get evaluated if the specified pattern gets matched. To separate the pattern from the expressions, arrow symbol(=>) is used.

```
scala> :paste
// Entering paste mode (ctrl-D to finish)

import scala.util.Random

val x: Int = Random.nextInt(10)

x match {
  case 0 => "zero"
  case 1 => "one"
  case 2 => "two"
  case _ => "other"
}
```

The val x above is a random integer between 0 and 10. x becomes the left operand of the matchoperator and on the right is an expression with four cases. The last case _is a "catch all" case for any other possible Int values. Cases are also called *alternatives*.