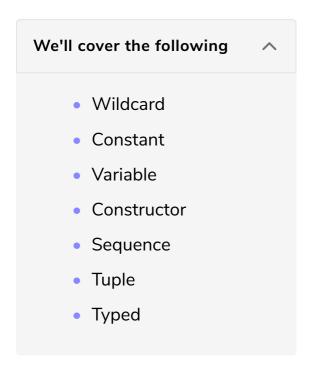
### Types of Patterns

In this lesson, you will be given a brief introduction to the different types of patterns you can work with in pattern matching.



In the previous lesson, we were introduced to *pattern matching* in Scala. In this lesson, we will go over the types of patterns provided in Scala to see the true potential of *pattern matching*.

#### Wildcard #

We saw the *wildcard* pattern in the previous lesson. It is a pattern which matches with any object. It is often used as a default pattern to avoid runtime errors.

```
This code requires the following environment variables to execute:

LANG

C.UTF-8

val wildcardPattern = 75

wildcardPattern match {
   case _ => println(s"You said $wildcardPattern")
}

\[
\begin{align*}
\text{\text{\text{C}}}
\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\text{\tex{
```

Constant

The food example in the previous lesson was matching *constant* patterns. A *constant* pattern matches with itself. Literals can be used as a *constant*.

```
This code requires the following environment variables to execute:

LANG

C.UTF-8

val constantPattern: Any = 75

constantPattern match {
   case 75 => println("case1")
   case "hello" => println("case2")
   case true => println("case3")
   case _ => println(s"You said $constantPattern")
}
```

When declaring our constantPattern variable, we need to specify that it's of type Any. The reason for this is that our patterns are of different types and when the value of constantPattern goes through each case, it will not be able to compare itself with a pattern of a different type which will result in an error and premature termination of the program.

#### Variable #

A *variable* pattern is similar to a *wildcard* pattern in that it matches with any object. It differs by binding the object to a variable. Any variable name can be a *variable* pattern.

```
This code requires the following environment variables to execute:

LANG

C.UTF-8

val variablePattern = 75

variablePattern match {
   case myVariable => println(s"$myVariable has been bound")
}
```

### Constructor #

A *Constructor* pattern matches with a constructor. Constructors are used for creating instances (objects) of a user-built class, a topic which will be discussed in detail in a later chapter. For the sake of completion, let's look at an example.

```
constructorPattern match {
  case binaryOperators("+", e, Number(0)) => println("constructor match")
}
```

This shows the incredible depth of pattern matching in Scala. The pattern above checks if the object is of class <code>binaryOperators</code> and then further checks if the parameters of the object match that of the constructor ("+", e, Number()) and then even further checks if the parameters of the parameters match the constructor (Number(0)).

## Sequence #

A *Sequence* pattern matches with a sequence type collection such as an Array or a List.



case Array(0,\_,\_) will match with any array whose first element is **0** while the next two elements can be anything.

## Tuple #

A tuple pattern matches with a tuple. A tuple is simply an ordered set of elements.

LANG C.UTF-8

```
val tuplePattern: Any = (3,"word",true)

tuplePattern match {
  case (0,"word",_) => println("case1")
  case (1,_,true) => println("case2")
  case (3,_,true) => println("case3")
  case _ => println("default")
}
```

# Typed #

A *Typed* pattern matches with its *type* of object.



With *pattern types*, our discussion on control structures comes to an end. Let's take a quiz for a quick recap before we move on to the next chapter.