Function Composition in Scala

✅ 1. Basic Example

Let's define two simple functions:

```
val f: Int => Int = x => x * 2
val g: Int => Int = x => x + 10
```

✔ Using `compose`

```
(f compose g)(x)  = f(g(x))

val result1 = (f compose g)(5)
println(result1)   // (5 + 10) * 2 = 30
```

✔ Using `andThen`

```
(f andThen g)(x)  = g(f(x))

val result2 = (f andThen g)(5)
println(result2)   // (5 * 2) + 10 = 20
```

✅ 2. Real Example with Strings

```
val toUpper: String => String = _.toUpperCase
val trim: String => String = _.trim
```

First trim, then upper:

```
val c1 = toUpper compose trim
println(c1("  hello "))   // HELLO
```

First upper, then trim:

```
val c2 = toUpper andThen trim
println(c2(" hello  "))   // HELLO
```

✅ 3. Chain multiple functions

```
val f1: Int => Int = _ + 2
val f2: Int => Int = _ * 3
val f3: Int => Int = _ - 5

val chain = f1 andThen f2 andThen f3
println(chain(5))   // ((5+2)*3) - 5 = 16
```

✅ 4. Function Composition with Methods

```
def double(x: Int) = x * 2
def square(x: Int) = x * x

val h = double _ andThen square _
println(h(3))   // (3*2)^2 = 36
```

`double _` converts method → function.

✅ 5. Higher-Order Function Example

```
def compose[A,B,C](f: B => C, g: A => B): A => C = {
  x => f(g(x))
}

val add2 = (x: Int) => x + 2
val mul3 = (x: Int) => x * 3
```

```
val newFunc = compose(mul3, add2)
println(newFunc(10))  // (10+2)*3 = 36
```

✅ 6. Real ETL Example (practical)

```
val trimStr: String => String = _.trim
val removeComma: String => String = _.replace(",", "")
val toInt: String => Int = _.toInt

val parseAmount = trimStr andThen removeComma andThen toInt

println(parseAmount("  1,200 "))  // 1200
```

This is very useful in Spark transformations.

SUMMARY TABLE

Syntax	Order	Meaning
`f compose g`	g → f	Apply g first, then f
`f andThen g`	f → g	Apply f first, then g
