Basic Syntax I

Rapid tour of the basic syntax of Kotlin

Defining packages

Package specification should be at the top of the source file:

```
package my.demo

import java.util.*

// ...
```

It is not required to match directories and packages: source files can be placed arbitrarily in the file system.

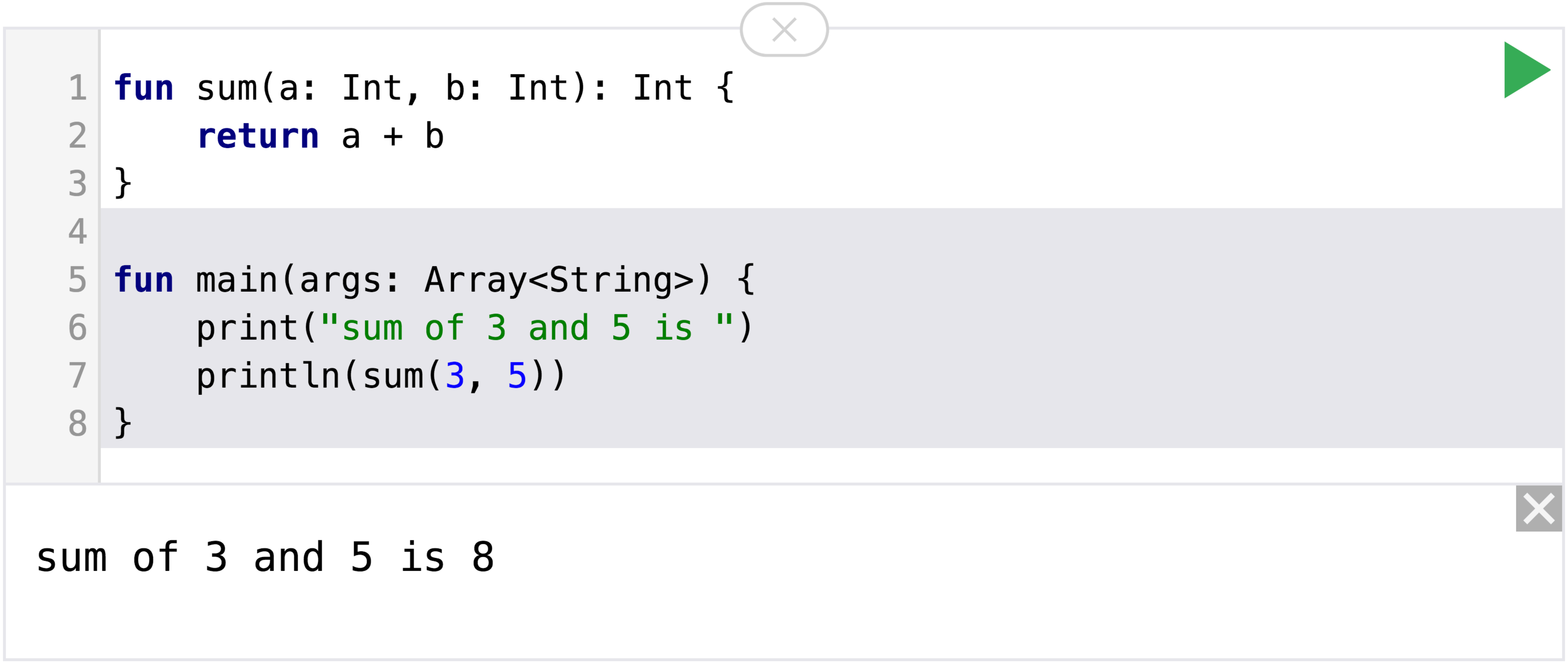
Defining functions

Function having two `Int` parameters with `Int` return type:

```kotlin
fun sum(a: Int, b: Int): Int {
    return a + b
}
```

Defining functions

Function having two `Int` parameters with `Int` return type:



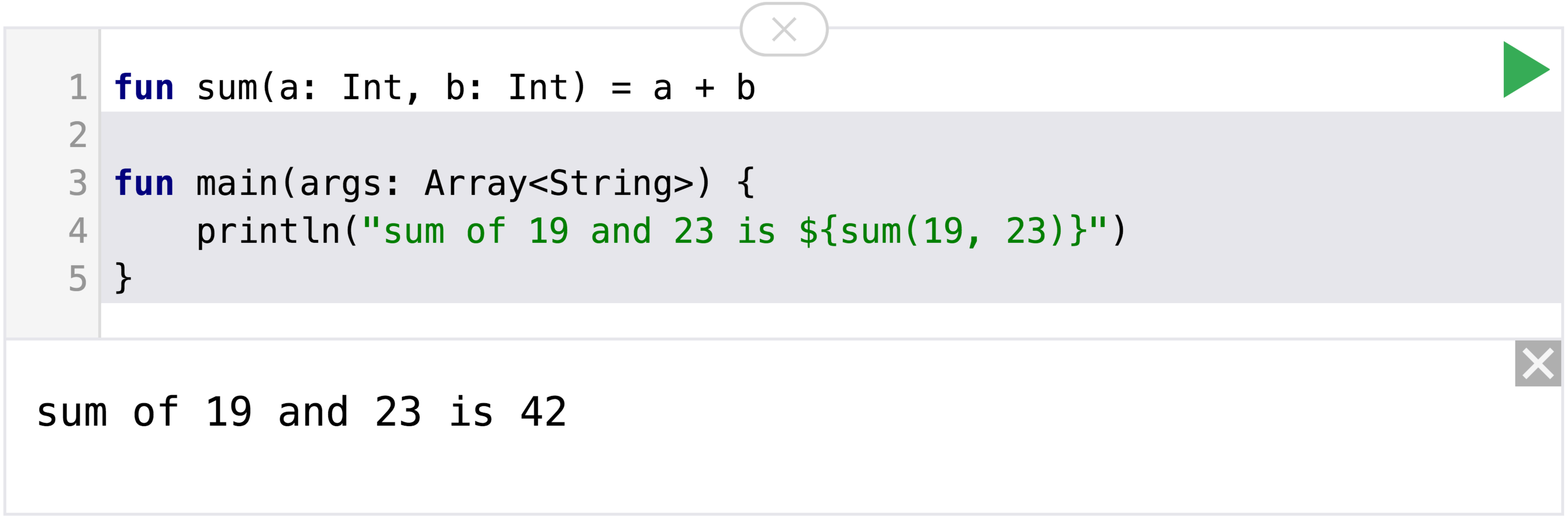
```kotlin
fun sum(a: Int, b: Int): Int {
    return a + b
}

fun main(args: Array<String>) {
    print("sum of 3 and 5 is ")
    println(sum(3, 5))
}
```

```
sum of 3 and 5 is 8
```

Function with an expression body and inferred return type:

```kotlin
fun sum(a: Int, b: Int) = a + b
```

Function with an expression body and inferred return type:



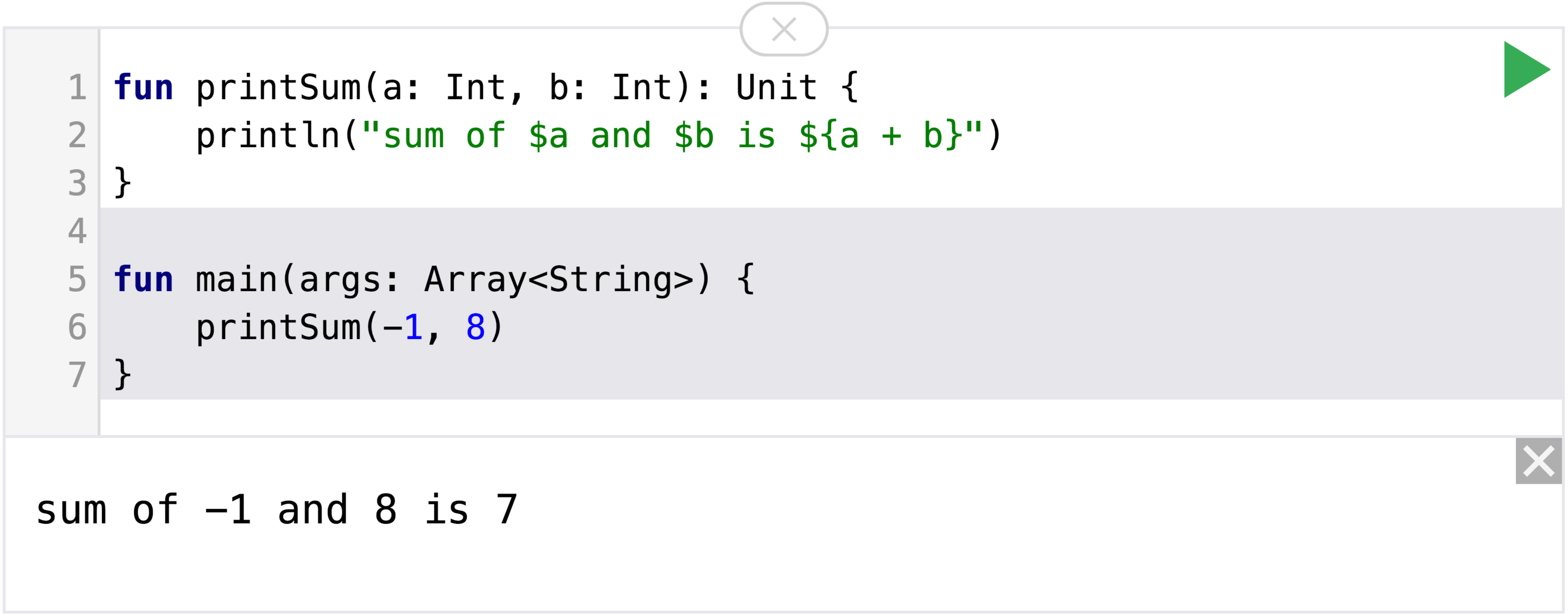
```kotlin
fun sum(a: Int, b: Int) = a + b

fun main(args: Array<String>) {
    println("sum of 19 and 23 is ${sum(19, 23)}")
}
```

```
sum of 19 and 23 is 42
```

Function returning no meaningful value:

```kotlin
fun printSum(a: Int, b: Int): Unit {
    println("sum of $a and $b is ${a + b}")
}
```

Function returning no meaningful value:



```kotlin
fun printSum(a: Int, b: Int): Unit {
    println("sum of $a and $b is ${a + b}")
}

fun main(args: Array<String>) {
    printSum(-1, 8)
}
```

```
sum of -1 and 8 is 7
```

`Unit` return type can be omitted:

```kotlin
fun printSum(a: Int, b: Int) {
    println("sum of $a and $b is ${a + b}")
}
```

Defining variables

Assign-once (read-only) local variable:

```kotlin
val a: Int = 1  // immediate assignment
val b = 2   // `Int` type is inferred
val c: Int  // Type required when no initializer is provide
c = 3       // deferred assignment
```

Target platform: JVM   Running on kotlin v. 1.2.70

Mutable variable:

```kotlin
var x = 5 // `Int` type is inferred
x += 1
```

Top-level variables:

```kotlin
val PI = 3.14
var x = 0

fun incrementX() {
    x += 1
}
```

Comments

Just like Java and JavaScript, Kotlin supports end-of-line and block comments.

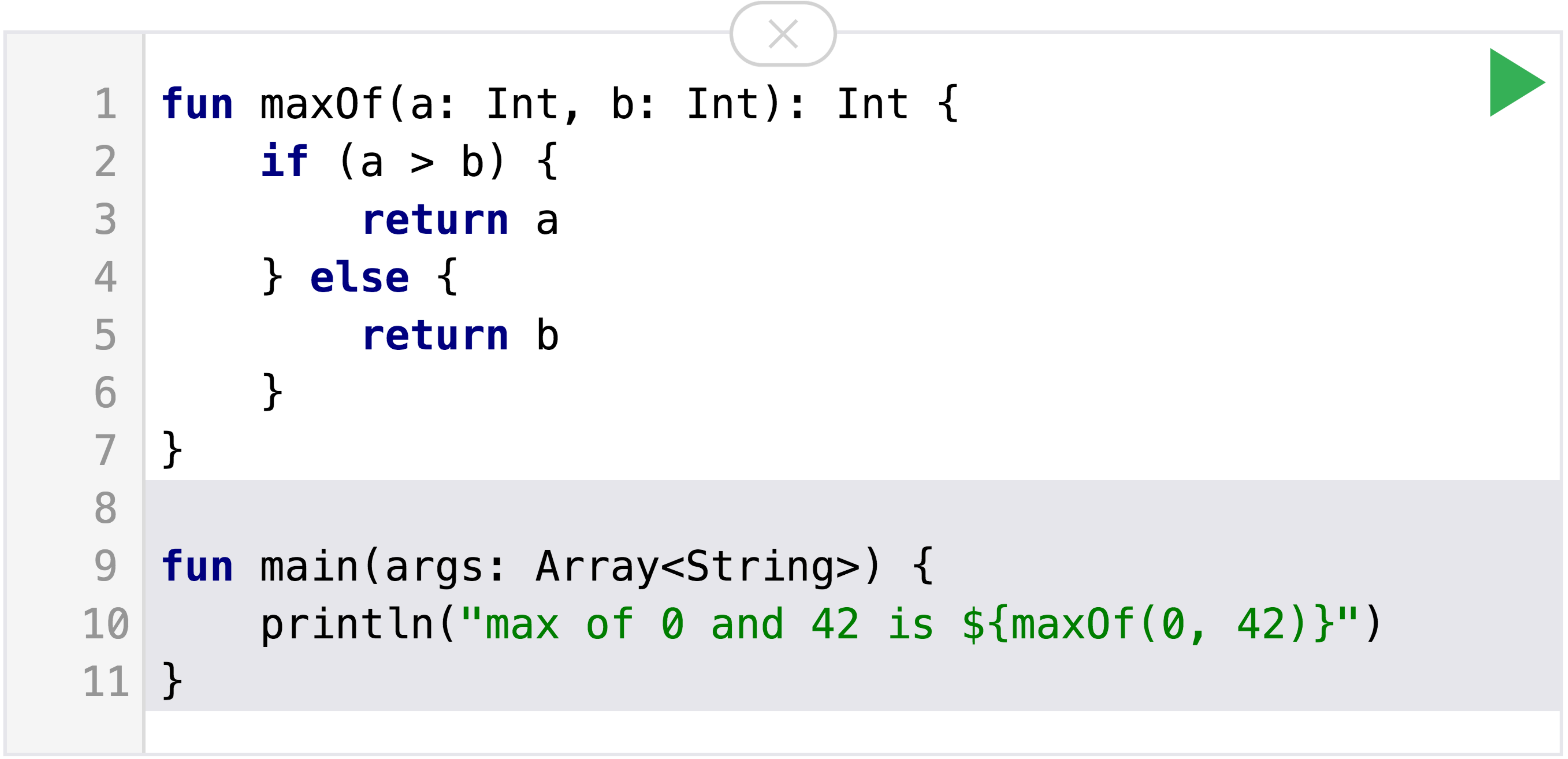
```
// This is an end-of-line comment

/* This is a block comment
   on multiple lines. */
```

Unlike Java, block comments in Kotlin can be nested.

Using string templates

```
var a = 1
// simple name in template:
val s1 = "a is $a"

a = 2
// arbitrary expression in template:
val s2 = "${s1.replace("is", "was")}, but now is $a"
```

Using conditional expressions



```
fun maxOf(a: Int, b: Int): Int {
    if (a > b) {
        return a
    } else {
        return b
    }
}

fun main(args: Array<String>) {
    println("max of 0 and 42 is ${maxOf(0, 42)}")
}
```

Using `if` as an expression:

```kotlin
fun maxOf(a: Int, b: Int) = if (a > b) a else b
```