

# Your first program in Kotlin

## 1. Before you begin



In this course, you will build apps by writing code in the Kotlin programming language, which is the language recommended by Google when creating new Android apps.

Kotlin is a modern programming language that helps developers be more productive. For example, Kotlin allows you to be more concise and write fewer lines of code for the same functionality compared to other programming languages. Apps that are built with Kotlin are also less likely to crash, resulting in a more stable and robust app for users. Essentially, with Kotlin, you can write better Android apps in a shorter amount of time. As a result, Kotlin is gaining momentum in the industry and is the language that the majority of professional Android developers use.

To get started on building Android apps in Kotlin, it's important to first establish a solid foundation of programming concepts in Kotlin. With the codelabs in this pathway, you will learn about Kotlin programming basics before diving into app creation.

### What you'll build

- Short programs in Kotlin that display messages when you run them.

### What you'll learn

- How to write and run a simple Kotlin program.
- How to modify a simple program to change the output.

### What you'll need

- A computer with internet access and a web browser.

## 2. Get started

In this codelab, you explore and modify simple programs in Kotlin. You can think of a *program* as a series of instructions for a computer or mobile device to perform some action, such as display a message to a user or calculate the cost of items in a shopping cart. The step-by-step instructions for what the computer should do is called *code*. When you modify the code in a program, the output can change.

You use a tool called a *code editor* to write and edit your code. It's similar to a text editor in which you can write and edit text, but a code editor also provides functionality to help you write code more accurately. For example, a code editor shows autocomplete suggestions while you type, and displays error messages when code is incorrect.

To practice the basics of the Kotlin language, you will use an interactive code editor called the Kotlin Playground. You can access it from a web browser, so you don't need to install any software on your computer. You can edit and run Kotlin code directly in the Kotlin Playground and see the output.

Note that you can't build Android apps within the Kotlin Playground. In later pathways, you will install and use a tool called Android Studio to write and edit your Android app code.

Now that you have some context on Kotlin, take a look at your first program!

## 3. Open Kotlin Playground

In a web browser on your computer, open the Kotlin Playground (<https://developer.android.com/training/kotlinplayground>).

You should see a web page similar to this image:




There's already some default code populated in the code editor. These three lines of code make up a simple program:

```
fun main() {  
    println("Hello, world!")  
}
```

Even if you've never programmed before, can you guess what the program does?

See if your guess is correct by moving onto the next section!

## 4. Run your first program

Click  to run your program.

When you click the Run button, a lot happens. Code in the Kotlin programming language is meant to be understood by humans, so that people can more easily read, write, and collaborate on Kotlin programs. However, your computer doesn't immediately understand this language.

You need something called the *Kotlin compiler*, which takes the Kotlin code you wrote, looks at it line by line, and translates it into something that the computer can understand. This process is called *compiling* your code.

If your code compiles successfully, your program will run (or execute). When the computer executes your program, it performs each of your instructions. If you've ever followed a cooking recipe, performing each step in the recipe is considered executing each instruction.

Here's a screenshot of what you should see when you run your program.



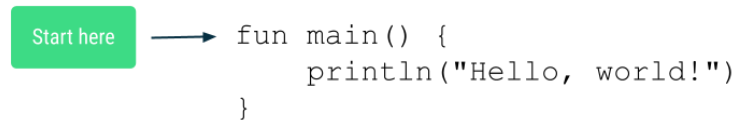
At the bottom of the code editor, you should see a pane that shows the output, or result, of your program:

```
Hello, world!
```

Cool! The purpose of this program is to print or display a message that says `Hello, world!`.

**Warning:** If you don't see this result, copy and paste the three lines of code from the previous section into the Kotlin Playground and try again.

How does this work? A Kotlin program is required to have a main function, which is the specific place in your code where the program starts running. The main function is the entry point, or starting point, of the program.



```
fun main() {  
    println("Hello, world!")  
}
```

Now you may be wondering, what is a function?

## 5. Parts of a function

A *function* is a segment of a program that performs a specific task. Your program may have one or more functions.

### Define versus call a function

In your code, you *define* a function first. That means you specify all the instructions needed to perform that task.

Once the function is defined, then you can *call* that function, so the instructions within that function can be performed or executed.

Here's an analogy. You write out step-by-step instructions on how to bake a chocolate cake. This set of instructions can have a name: `bakeChocolateCake`. Every time you want to bake a cake, you can execute the `bakeChocolateCake` instructions. If you want 3 cakes, you need to execute the `bakeChocolateCake` instructions 3 times. The first step is defining the steps and giving it a name, which is considered defining the function. Then you can refer to the steps anytime you want them to be executed, which is considered calling the function.

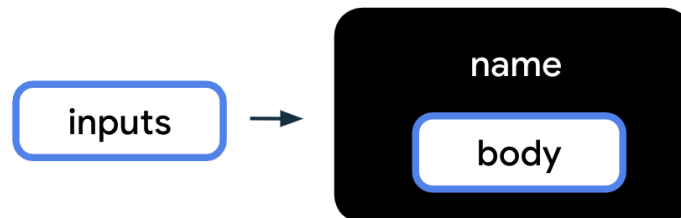
**Note:** You may hear the alternate phrase "declare a function." The words declare and define can be used interchangeably and have the same meaning. You may also hear the term "function definition" or "function declaration", which refer to the exact code that defines a function. In some other programming languages, declare and define have different meanings.

### Define a function

These are the key parts needed to define a function:

- The function needs a **name**, so you can call it later.

- The function can also require some **inputs**, or information that needs to be provided when the function is called. The function uses these inputs to accomplish its purpose. Requiring inputs is optional, and some functions do not require inputs.
- The function also has a **body** which contains the instructions to perform the task.



To translate the above diagram into Kotlin code, use the following syntax, or format, for defining a function. The order of these elements matters. The `fun` word must come first, followed by the function name, followed by the inputs in parentheses, followed by curly braces around the function body.

```

fun [name] ( [inputs] ) {
    [body]
}
  
```

Notice the key parts of a function within the `main` function example you saw in the Kotlin Playground:

- The function definition starts with the word `fun`.
- Then the name of the function is `main`.
- There are no inputs to the function, so the parentheses are empty.
- There is one line of code in the function body, `println("Hello, world!")`, which is located between the opening and closing curly braces of the function.

```

      name inputs
      ↓   ↓
fun main() {
    println("Hello, world!") ← body
}
  
```

Each part of the function is explained in more detail below.

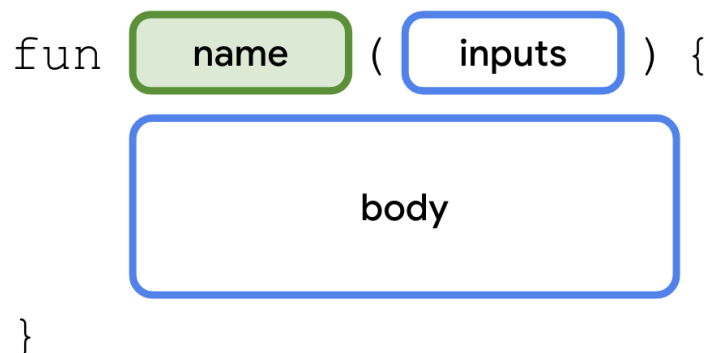
## Function keyword

To indicate that you're about to define a function in Kotlin, use the special word `fun` (short for function) on a new line. You must type `fun` exactly as shown in all lowercase letters. You can't use `func`, `function`, or some alternative spelling because the Kotlin compiler won't recognize what you mean.

These special words are called keywords in Kotlin and are reserved for a specific purpose, such as creating a new function in Kotlin.

## Function name

Functions have names so they can be distinguished from each other, similar to how people have names to identify themselves. The name of the function is found after the `fun` keyword.



```
fun [name] ([inputs]) {  
    [body]  
}
```

The diagram illustrates the syntax of a Kotlin function. It shows the keyword `fun` followed by a green box labeled `name`, an opening parenthesis `(`, a blue box labeled `inputs`, a closing parenthesis `)`, an opening curly brace `{`, a large blue box labeled `body`, and a closing curly brace `}`.

Choose an appropriate name for your function based on the purpose of the function. The name is usually a verb or verb phrase. It's recommended to avoid using a Kotlin keyword (<https://kotlinlang.org/docs/keyword-reference.html>) as a function name.

Function names should follow the *camel* case convention, where the first word of the function name is all lower case. If there are multiple words in the name, there are no spaces between words, and all other words should begin with a capital letter.

Example function names:

- `calculateTip`
- `displayErrorMessage`
- `takePhoto`

## Function inputs

Notice that the function name is always followed by parentheses. These parentheses are where you list inputs for the function.

```
fun name ( inputs ) {  
    body  
}
```

An input is a piece of data that a function needs to perform its purpose. When you define a function, you can require that certain inputs be passed in when the function is called. If there are no inputs required for a function, the parentheses are empty `()`.

Here are some examples of functions with a different number of inputs:

The following diagram shows a function that is called `addOne`. The purpose of the function is to add 1 to a given number. There is one input, which is the given number. Inside the function body, there is code that adds 1 to the number passed into the function.



In this next example, there is a function called `printFullName`. There are two inputs required for the function, one for the first name and one for the last name. The function body prints out the first name and last name in the output, to display the person's full name.

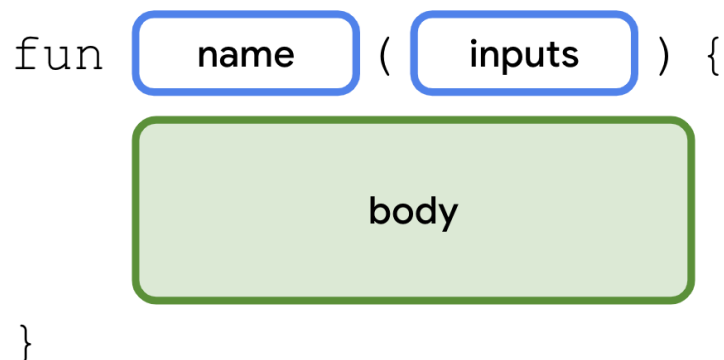


This last example shows a function that doesn't require any inputs to be passed in when the function is called. When you call the `displayHello()` function, a Hello message gets printed to the output.



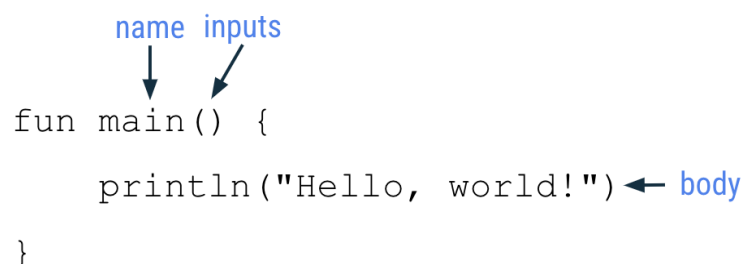
## Function body

The *function body* contains the instructions needed to achieve the purpose of the function. You can locate the function body by looking for the lines of code enclosed within the opening and closing curly braces.



## Simple program explained

Look back at the simple program that you saw earlier in the codelab.



The program contains one function: the `main` function. `main` is a special function name in Kotlin. When you're writing your code in the Kotlin Playground, your code should be written inside the `main()` function or called from the `main()` function.

There's only one line of code in the body of this `main()` function:

```
println("Hello, world!")
```



This line of code is a *statement* because it performs a specific action, which is to print the `Hello, world!` text in the output pane. More specifically, the `println()` function is being called on this line of code. `println()` is a function that's already defined in the Kotlin language. That means the team of engineers who created the Kotlin language already wrote out the function declaration for the `println()` function. The function requires one input, which is the message that should be printed.

When you call the `println()` function, place the message text within parentheses after the function name. Be sure to use quotation marks around the text to display, such as `"Hello, world!"`.

When the program is executed, the message that was passed into the `println()` function is printed to the output:

```
Hello, world!
```

Try it

Now look back at the original code in the program. Can you modify the code in the Kotlin Playground so that the output shows this message instead?

```
Hello, Android!
```

## 6. Modify your program

1. In order to change the message that's displayed in the output, modify the `println()` function call on the second line of the program. Replace `world` with `Android` in the `println()` function. Make sure that `"Hello, Android!"` is still within quotation marks and within the parentheses.

```
fun main() {  
    println("Hello, Android!")  
}
```

2. Run the program.
3. The output should show this message:

```
Hello, Android!
```

Nice job, you modified your first program!

Now can you change your code so that the message gets printed twice? See the desired output:

```
Hello, Android!  
Hello, Android!
```

## Print more than one message

You can put as many lines of instructions inside a function as you need to accomplish a task. However, note that there should only be one statement per line in Kotlin. If you want to write another statement, put it on a new line of the function.

To change your program to print multiple lines of text:

1. Copy the initial `println()` statement and paste the second statement below it within the function body. Both `println()` statements must be contained within the curly braces of the main function. Now you have two statements within the function body.

```
fun main() {  
    println("Hello, Android!")  
    println("Hello, Android!")  
}
```

2. Run the program.
3. When you run your program, the output should be:

```
Hello, Android!  
Hello, Android!
```

You can see how changes to the code affects the output.

4. Change the code so that it says `Hello, YOUR_NAME!`.

## 7. Kotlin style guide

Throughout this course, you'll learn about good coding practices to follow as an Android developer. One such practice is to follow Google's Android coding standards for code written in Kotlin. The complete guide is called a style guide and explains how code should be formatted in terms of visual appearance and the conventions to follow when writing your code. For example, the style guide includes recommendations on use of whitespace, indentation, naming, and more.

The purpose of following the style guide is to make your code easier to read and more consistent with how other Android developers write their code. This consistency is important when collaborating on large projects together, so that the style of code is the same throughout all the files in the project.

Here are some of the relevant style guide recommendations for what you've learned in Kotlin so far:

- Function names should be in camel case and should be verbs or verb phrases.
- Each statement should be on its own line.

- The opening curly brace should appear at the end of the line where the function begins.
- There should be a space before the opening curly brace.

space  
↓

```
fun main() {
    println("Hello, world!")
}
```

- The function body should be indented in by 4 spaces. Do not use a tab character to indent your code, type in 4 spaces.

indent by 4  
spaces →

```
fun main() {
    println("Hello, world!")
}
```

- The closing curly brace is on its own line after the last line of code in the function body. The closing brace should line up with the `fun` keyword at the beginning of the function.

aligned  
vertically

```
fun main() {
    println("Hello, world!")
}
```

You will learn more Android coding conventions as you build your knowledge in Kotlin. The full style guide is here (<https://developer.android.com/kotlin/style-guide>), but don't worry that it covers other topics in Kotlin you haven't learned yet.

## 8. Fix errors in your code

When you learn a human language, there are rules of syntax and grammar for the correct way to use words and form sentences. Similarly, for programming languages, there are specific rules for what makes valid code, which means code that compiles successfully.

A normal part of the coding process involves making mistakes and accidentally writing invalid code. As a beginner, it may be confusing or overwhelming to encounter these mistakes. But don't worry, this is expected. Code rarely works perfectly the first time

that it's written. Just like how writing a document takes many drafts, writing code can take many iterations until it works as expected.

If the code can't compile successfully, there's an error. For example, if you have a typo, such as a missing quotation mark or parenthesis, the compiler won't understand your code and can't translate it into steps for the computer to perform. If your code doesn't work as expected, or you see an error message in your code editor, you have to go back to your code and fix it. The process of solving these errors is called *troubleshooting*.

1. Copy and paste the following code snippet into the Kotlin Playground and run the program. What do you see?

```
fun main() {  
    println("Today is sunny!")  
}
```

Ideally, you want to see the message `Today is sunny!` displayed. Instead, in the output pane, you see exclamation point icons with error messages.



❗ Expecting '''  
❗ Expecting ')'

#### *Error message from Kotlin Playground*

The error messages start with the word "Expecting" because the Kotlin compiler is "expecting" something but not finding it in the code. In this case, the compiler is expecting a closing quotation mark and a closing parenthesis for the code on the second line of your program.

In the `println()` statement, notice that the message to display has an opening quotation mark, but no closing quotation mark. Even though there is a closing parenthesis in the code, the compiler thinks the parenthesis is part of the text to print because there is no closing quotation mark before it.

2. Add the closing quotation mark after the exclamation point, before the closing parenthesis.

```
fun main() {  
    println("Today is sunny!")  
}
```

The main function contains one line of code, which is a `println()` statement, where the text is enclosed in quotation marks and placed within the parentheses: `"Today is sunny!"`.

3. Run the program again.

There shouldn't be any errors and the output pane should display this text:

```
Today is sunny!
```

Good work on fixing the error! Once you gain more experience in writing code and troubleshooting errors, you'll realize how important it is to pay attention to capitalization, spelling, spacing, symbols, and names when you're typing your code.

In the next section, you work on a series of exercises to practice what you've learned. Solutions are provided at the end of the codelab, but try your best to find the answer on your own first.

## 9. Exercises

1. Can you read the code in this program and guess what the output is (without running it in Kotlin Playground)?

```
fun main() {  
    println("1")  
    println("2")  
    println("3")  
}
```

Once you have a guess, copy and paste this code into the Kotlin Playground to check your answer.

2. Use the Kotlin Playground to create a program that outputs the following messages:

```
I'm  
learning  
Kotlin!
```

3. Copy and paste this program into the Kotlin Playground.

```
fun main() {  
    println("Tuesday")  
    println("Thursday")  
    println("Wednesday")  
    println("Friday")  
    println("Monday")  
}
```

Fix the program so that it prints this output:

Monday  
Tuesday  
Wednesday  
Thursday  
Friday

For some early practice on troubleshooting, fix the errors in the following exercises. For each exercise, copy the code into the Kotlin Playground in your browser. Try to run the program and you'll see an error message appear.

4. Fix the error in this program, so that it produces the desired output.

```
fun main() {  
    println("Tomorrow is rainy")  
}
```

Desired output:

Tomorrow is rainy

5. Fix the error in this program, so that it produces the desired output.

```
fun main() {  
    println("There is a chance of snow")  
}
```

Desired output:

There is a chance of snow

6. Fix the error in this program, so that it produces the desired output.

```
fun main() {  
    println("Cloudy") println("Partly Cloudy")  
    println("Windy")  
}
```

Desired output:

Cloudy  
Partly Cloudy  
Windy

7. Fix the error in this program, so that it produces the desired output.

```
fun main() {  
    println("How's the weather today?")  
}
```

Desired output:

```
How's the weather today?
```

After you complete these exercises, check your answers against the solutions in the next section.

## 10. Solutions

1. The output of the program is:

```
1  
2  
3
```

2. The code in your program should look like:

```
fun main() {  
    println("I'm")  
    println("learning")  
    println("Kotlin!")  
}
```

3. This is the correct code for the program:

```
fun main() {  
    println("Monday")  
    println("Tuesday")  
    println("Wednesday")  
    println("Thursday")  
    println("Friday")  
}
```

4. The closing curly brace that indicates the end of the function body for the `main` function is missing on the third line of the program.

Correct code:

```
fun main() {  
    println("Tomorrow is rainy")  
}
```

Output:

```
Tomorrow is rainy
```

5. When you run the program, you see an `Unresolved reference: println` error. This is because `println()` isn't a recognized function in Kotlin. You can also see the part of the code causing the error highlighted in red in the Kotlin Playground. Change the function name to `println` to print a line of text to the output, which fixes the error.

Correct code:

```
fun main() {  
    println("There is a chance of snow")  
}
```

Output:

```
There is a chance of snow
```

6. When you run the program, you see an `Unresolved reference: println` error. This message doesn't directly say how to fix the problem. This can sometimes happen when you troubleshoot an error, and it requires you to take a deeper look at the code to resolve the unexpected behavior.

Upon closer look, the second `println()` function call in the code is red, which signals that's where the problem is. Kotlin expects just one statement on each line. In this case, you can move the second and third `println()` function calls onto separate new lines to solve the issue.

Correct code:

```
fun main() {  
    println("Cloudy")  
    println("Partly Cloudy")  
    println("Windy")  
}
```

Output:

```
Cloudy  
Partly Cloudy  
Windy
```



7. If you run the program, you see the error: Function 'main' must have a body. A function body should be enclosed within an opening curly brace and a closing curly brace { }, not opening and closing parentheses ( ).

Correct code:

```
fun main() {  
    println("How's the weather today?")  
}
```

Output:

```
How's the weather today?
```

## 11. Conclusion

Great job on completing this introduction to Kotlin!

You worked with simple programs in Kotlin and ran them to see text printed to the output. You modified the programs in different ways and observed how those changes affected the output. It's normal to make mistakes when programming, so you also began to learn about how to troubleshoot and correct errors in your code, which is an important skill that will help you in the future.

Move on to the next codelab to learn how to use variables in Kotlin so that you can create more interesting programs!

## Summary

- A Kotlin program requires a `main` function as the entry point of the program.
- To define a function in Kotlin, use the `fun` keyword, followed by the name of the function, any inputs enclosed in parentheses, followed by the function body enclosed in curly braces.
- The name of a function should follow camel case convention and start with a lowercase letter.
- Use the `println()` function call to print some text to the output.
- Refer to the Kotlin style guide for formatting and code conventions to follow when coding in Kotlin.
- Troubleshooting is the process of resolving errors in your code.

Learn more

Hello World	<a href="https://play.kotlinlang.org/byExample/01_introduction/01_Hello%20world">https://play.kotlinlang.org/byExample/01_introduction/01_Hello%20world</a>
-------------	---

Program entry point	<a href="https://kotlinlang.org/docs/basic-syntax.html#program-entry-point">https://kotlinlang.org/docs/basic-syntax.html#program-entry-point</a>
Print to the standard output	<a href="https://kotlinlang.org/docs/basic-syntax.html#print-to-the-standard-output">https://kotlinlang.org/docs/basic-syntax.html#print-to-the-standard-output</a>
Basic syntax of functions	<a href="https://kotlinlang.org/docs/basic-syntax.html#functions">https://kotlinlang.org/docs/basic-syntax.html#functions</a>
Keywords and operators	<a href="https://kotlinlang.org/docs/keyword-reference.html">https://kotlinlang.org/docs/keyword-reference.html</a>
Functions Concept	<a href="https://kotlinlang.org/docs/functions.html">https://kotlinlang.org/docs/functions.html</a>
<code>println()</code>	<a href="https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.io/println.html">https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.io/println.html</a>
Kotlin style guide	<a href="https://developer.android.com/kotlin/style-guide">https://developer.android.com/kotlin/style-guide</a>