# Funciones en Kotlin

Una función es un bloque de código que sólo se ejecuta cuando se llama.

Puede pasar datos, conocidos como parámetros, a una función.

Las funciones se utilizan para realizar determinadas acciones y también se conocen como métodos.

## Funciones predefinidas

Entonces resulta que ya sabes qué es una función. ¡Lo has estado usando todo el tiempo!

Por ejemplo, println() es una función. Se utiliza para enviar/imprimir texto en la pantalla:

Ejemplo
```
fun main() {
  println("Hello World")
}
```

## Crea tus propias funciones

Para crear su propia función, use la palabra clave fun y escriba el nombre de la función, seguido de paréntesis () :

Ejemplo
Cree una función llamada "myFunction" que debería generar algo de texto:

```
fun myFunction() {
  println("I just got executed!")
}
```

## Llamar a una función

Ahora que ha creado una función, puede ejecutarla llamándola.

Para llamar a una función en Kotlin, escriba el nombre de la función seguido de dos paréntesis ().

En el siguiente ejemplo, myFunction() se imprimirá algo de texto (la acción) cuando se llame:

Ejemplo
```
fun main() {
  myFunction() // Call myFunction
}

// Outputs "I just got executed!"
```

Se puede llamar a una función varias veces, si lo desea:

Ejemplo
```
fun main() {
  myFunction()
  myFunction()
  myFunction()
}
```

## Parámetros de función

La información se puede pasar a funciones como parámetro.

Los parámetros se especifican después del nombre de la función, dentro del paréntesis. Puedes agregar tantos parámetros como quieras, simplemente sepáralos con una coma. Solo tenga en cuenta que debe especificar el tipo de cada parámetro (Int, String, etc.).

El siguiente ejemplo tiene una función que toma un String fname llamado como parámetro. Cuando se llama a la función, pasamos un nombre, que se usa dentro de la función para imprimir el nombre completo:

Ejemplo
```
fun myFunction(fname: String) {
  println(fname + " Doe")
}

fun main() {
  myFunction("John")
  myFunction("Jane")
  myFunction("George")
}

// John Doe
// Jane Doe
// George Doe
```

Cuando se pasa un parámetro a la función, se le llama argumento. Entonces, según el ejemplo anterior: fname es un parámetro , mientras Johny, Jane y George son argumentos.

## Múltiples parámetros

Puedes tener tantos parámetros como quieras:

Ejemplo
```
fun myFunction(fname: String, age: Int) {
  println(fname + " is " + age)
}

fun main() {
```

```
  myFunction("John", 35)
  myFunction("Jane", 32)
  myFunction("George", 15)
}

// John is 35
// Jane is 32
// George is 15
```

Nota: Cuando se trabaja con múltiples parámetros, la llamada a la función debe tener la misma cantidad de argumentos que parámetros, y los argumentos deben pasarse en el mismo orden.

## Valores de retorno

En los ejemplos anteriores, usamos funciones para generar un valor. En el siguiente ejemplo, usaremos una función para devolver un valor y asignarlo a una variable.

Para devolver un valor, use la palabra clave return y especifique el tipo de retorno después de los paréntesis de la función (Int en este ejemplo):

Ejemplo
Una función con un parámetro Int y tipo Int de retorno:

```
fun myFunction(x: Int): Int {
  return (x + 5)
}

fun main() {
  var result = myFunction(3)
  println(result)
}

// 8 (3 + 5)
```

Usando dos parámetros:

Ejemplo
Una función con un parámetro Int y tipo Int de retorno:

```
fun myFunction(x: Int, y: Int): Int {
  return (x + y)
}

fun main() {
  var result = myFunction(3, 5)
  println(result)
}

// 8 (3 + 5)
```

# Funciones en Kotlin

## Sintaxis más corta para valores devueltos

También hay una sintaxis más corta para devolver valores. Puede utilizar el operador = en lugar de return sin especificar el tipo de devolución. Kotlin es lo suficientemente inteligente como para descubrir automáticamente qué es:

Ejemplo

```
fun myFunction(x: Int, y: Int) = x + y

fun main() {
  var result = myFunction(3, 5)
  println(result)
}

// 8 (3 + 5)
```

Ejercicios

**E901**: Confeccionar una aplicación que muestre una presentación en pantalla del programa. Solicite la carga de dos valores y nos muestre la suma. Mostrar finalmente un mensaje de despedida del programa. Implementar estas actividades en tres funciones.

```
fun presentacion() {
    println("Programa que permite cargar dos valores por teclado.")
    println("Efectua la suma de los valores")
    println("Muestra el resultado de la suma")
    println("******************************")
}

fun cargarSumar() {
    print("Ingrese el primer valor:")
    val valor1 = readln().toInt()
    print("Ingrese el segundo valor:")
    val valor2 = readln().toInt()
    val suma = valor1 + valor2
    println("La suma de los dos valores es: $suma")
}

fun finalizacion() {
    println("******************************")
    println("Gracias por utilizar este programa")
}

fun main() {
    presentacion()
    cargarSumar()
    finalizacion()
}
```

**E902**: Desarrollar una función que solicite la carga de tres valores y muestre el menor. Desde la función main del programa llamar 2 veces a dicha función (sin utilizar una estructura repetitiva)

```
fun menorValor() {
    print("Ingrese primer valor:")
    val valor1 = readln().toInt()
    print("Ingrese segundo valor:")
    val valor2 = readln().toInt()
    print("Ingrese tercer valor:")
    val valor3 = readln().toInt()
    print("Menor de los tres:")
    when {
        valor1 < valor2 && valor1 < valor3 -> println(valor1)
        valor2 < valor3 -> println(valor2)
        else -> println(valor3)
    }
}

fun main() {
```

```
    menorValor()
    menorValor()
}
```

**E903**: Confeccionar una función que le enviemos como parámetro el valor del lado de un cuadrado y nos retorne su superficie.

```
fun retornarSuperficie(lado: Int): Int {
    val sup = lado * lado
    return sup
}

fun main() {
    print("Ingrese el valor del lado del cuafrado:")
    val la = readln().toInt()
    val superficie = retornarSuperficie(la)
    println("La superficie del cuadrado es $superficie")
}
```

**E904**: Confeccionar una función que le enviemos como parámetro el valor del lado de un cuadrado y nos retorne su superficie.

```
fun retornarSuperficie(lado: Int) = lado * lado

fun main() {
    print("Ingrese el valor del lado del cuafrado:")
    val la = readln().toInt()
    println("La superficie del cuadrado es ${retornarSuperficie(la)}")
}
```

**E905**: Confeccionar una función que reciba un String como parámetro y en forma opcional un segundo String con un caracter. La función debe mostrar el String subrayado con el caracter que indica el segundo parámetro

```
fun tituloSubrayado(titulo: String, caracter: String = "*") {
    println(titulo)
    for(i in 1..titulo.length)
        print(caracter)
    println()
}

fun main() {
    tituloSubrayado("Sistema de Administracion")
    tituloSubrayado("Ventas", "-")
}
```

**E906**: Confeccionar una función que reciba el nombre de un operario, el pago por hora y la cantidad de horas trabajadas. Debe mostrar su sueldo y el nombre. Hacer la llamada de la función mediante argumentos nombrados.

```
fun calcularSueldo(nombre: String, costoHora: Double, cantidadHoras:
Int) {
```

```
    val sueldo = costoHora * cantidadHoras
    println("$nombre trabajó $cantidadHoras horas, se le paga por hora
$costoHora por lo tanto le corresponde un sueldo de $sueldo")
}

fun main(parametro: Array<String>) {
    calcularSueldo("juan", 10.5, 120)
    calcularSueldo(costoHora = 12.0, cantidadHoras = 40, nombre="ana")
    calcularSueldo(cantidadHoras = 90, nombre = "luis", costoHora =
7.25)
}
```

**E907**: Confeccionar una función que permita ingresar 10 valores por teclado y contar cuantos son múltiplos de 2 y cuantos son múltiplos de 5.

**E908**: Se desea almacenar los sueldos de operarios. Cuando se ejecuta el programa se debe pedir la cantidad de sueldos a ingresar. Luego crear un vector con dicho tamaño. Definir una función de carga y otra de impresión.

**E909**: Crea la función muestraPares(n: Int) que muestra por consola los primeros n números pares

**E910**: Escribe una función a la que se pase como parámetros una cantidad de días, horas y minutos. La función calculará y devolverá el número de segundos que existen en los datos de entrada.

**E911**: Implementa la función divisoresPrimos() que muestra, por consola, todos los divisores primos del números que se pasa como parámetro.