# Dasar Bahasa Kotlin

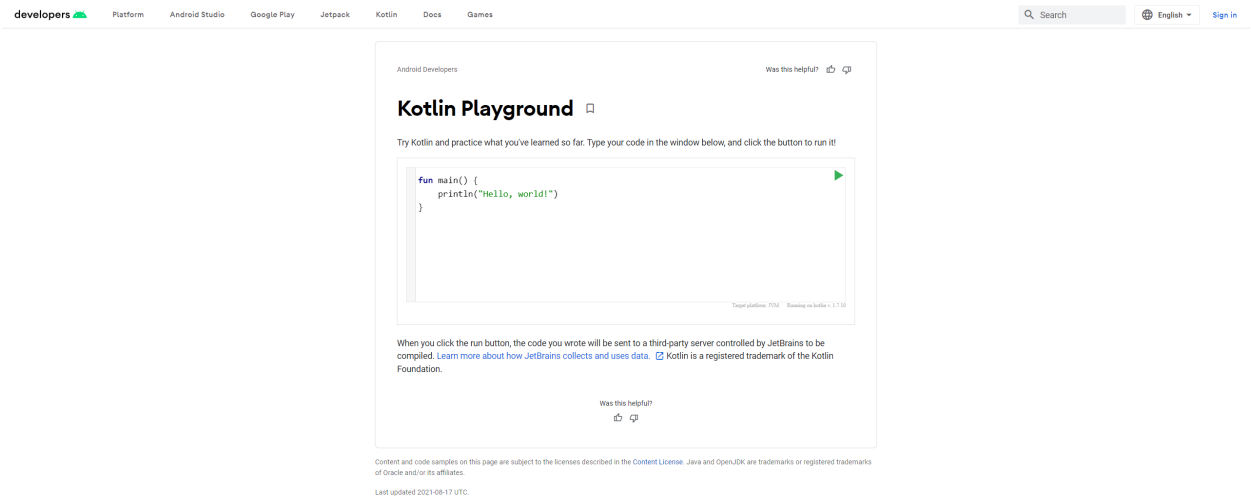
Praktikum Pemrograman Mobile - 01

Wanda Gusdya

2023

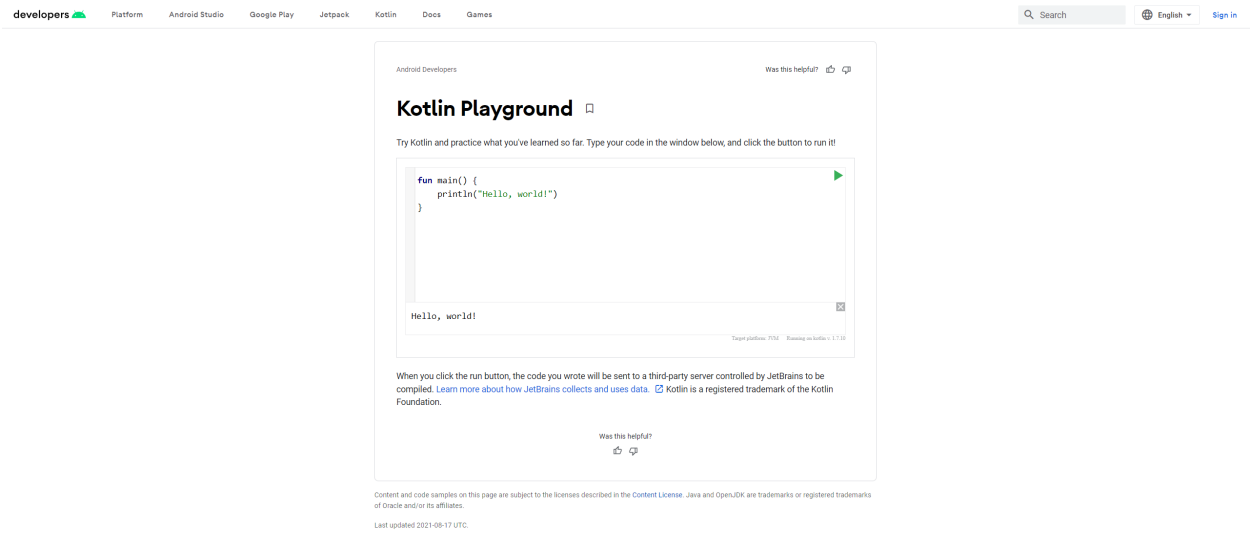
# Program Pertama

Pada pertemuan ini, anda akan belajar dasar pemrograman kotlin. Kotlin adalah bahasa pemrograman yang akan digunakan untuk membuat aplikasi android pada praktikum pemrograman mobile ini. Untuk praktek pemrograman dasar, anda dapat memanfaatkan Kotlin Playground yang telah disediakan oleh situsweb android di alamat https://developer.android.com/training/kotlinplayground.




*Gambar 1 Kotlin Playground*

Gambar 1 menunjukkan pada kotak yang disediakan oleh Kotlin Playground tersebut terdapat sebuah potongan kode untuk menampilkan teks Hello World. Jika anda klik tombol yang ditunjukkan dengan angka 1, output dari kode tersebut akan muncul di bawah kotak seperti pada Gambar 2 di bawah ini.




*Gambar 2 Hasil Running Program*

Program tersebut sangat sederhana, hanya ada sebuah fungsi (ditandai dengan kata kunci fun) bernama main tanpa parameter. Badan fungsi tersebut hanya berisi 1 baris kode memanggil fungsi println dengan parameter teks Hello World untuk menampilkan teks tersebut di console. Kotlin akan mencari fungsi bernama main ketika anda menjalankan sebuah file program.

## Penggunaan Variabel

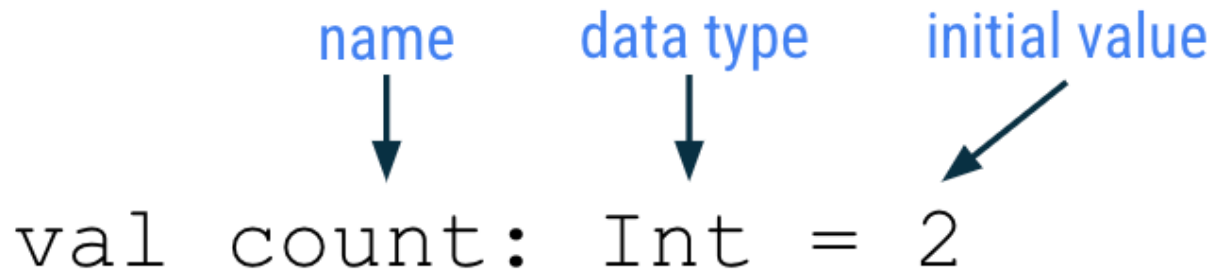
Seperti bahasa pemrograman lain, kita dapat menggunakan variabel untuk menampung nilai dari tipe data tertentu. Perhatikan kode berikut:

```
fun main() {
    val count: Int = 2
    println(count)
}
```

Fungsi main pada kode di atas memiliki sebuah variabel bernama count dengan tipe data Int. Variabel count tersebut diinisialisasi dengan nilai 2. Gambar 3 berikut ini menjelaskan struktur pendeklarasian sebuah variabel untuk bahasa kotlin.


*Gambar 3 Deklarasi Variabel*

Maka, deklarasi variabel count dapat kita petakan seperti pada gambar 4 berikut ini.




*Gambar 4 Deklarasi Variabel Count*

Variabel count menggunakan kata kunci **val** di depannya, artinya nilai variabel tersebut hanya dapat diisi sekali pada saat pendeklarasian, selanjutnya variabel menjadi read-only. Cobalah kode berikut ini di kotlin playground kemudian amati hasilnya.

```
fun main() {
    val cartTotal = 0
    cartTotal = 20
    println("Total: $cartTotal")
}
```

Kode tersebut akan menghasilkan pesan gala: Val cannot be reassigned karena variabel **cartTotal** bersifat *read-only*. Jika kita membutuhkan variabel yang dapat diubah nilainya, kita dapat menggunakan kata kunci **var**.

### Latihan 1

Ubahlah kode sebelumnya supaya nilai variabel cartTotal dapat diubah!

## Penggunaan Operator

Untuk dapat mengubah nilai atau menghasilkan nilai baru dari variabel-variabel tersebut, kita dapat menggunakan operator. Seperti bahasa pemrograman pada umumnya, kotlin memiliki operator aritmatika, penggabungan string, operator *assignment*, operator *increment/decrement*, operator perbandingan, dan operator logika. Berikut ini daftar operator-operator tersebut beserta contoh penggunaannya.

*Tabel 1. Operator Aritmatika*

| Operator | Arti |
|---|---|
| + | Operator penambahan |
| - | Operator pengurangan |
| * | Operator perkalian |
| / | Operator pembagian |
| % | Operator modulus untuk mengetahui sisa hasil bagi |

Untuk mempelajari contoh penggunaan operator aritmatika, anda dapat menjalankan kode berikut ini di kotlin playground.

```
fun main(args: Array<String>) {

    val number1 = 12.5
    val number2 = 3.5
    var result: Double

    result = number1 + number2
    println("number1 + number2 = $result")

    result = number1 - number2
    println("number1 - number2 = $result")

    result = number1 * number2
    println("number1 * number2 = $result")

    result = number1 / number2
    println("number1 / number2 = $result")

    result = number1 % number2
    println("number1 % number2 = $result")
}
```

Operator + juga digunakan untuk menggabungkan string, seperti pada contoh kode berikut.

```
fun main(args: Array<String>) {

    val start = "Talk is cheap. "
    val middle = "Show me the code. "
    val end = "- Linus Torvalds"

    val result = start + middle + end
    println(result)
}
```

Berikutnya, terdapat operator assignment yang dapat digunakan untuk menggabungkan nilai bertipe bilangan.

*Tabel 2 Operator Assignment*

| Ekspresi | Setara dengan |
|---|---|
| **a +=b** | a = a + b |
| **a -= b** | a = a - b |
| **a *= b** | a = a * b |
| **a /= b** | a = a / b |
| **a %= b** | a = a % b |

Untuk mempelajari operator assignment, cobalah kode berikut ini di kotlin playground kemudian amati hasilnya.

```
fun main(args: Array<String>) {
    var number = 12

    number *= 5
    println("number = $number")
}
```

Operator assignment di atas dapat dipersingkat menjadi *unary prefix* dan operator increment/decrement seperti berikut.

*Tabel 3 Unary Prefix dan Opeartor Increment/Decrement*

| Operator | Arti | Contoh ekspresi |
|---|---|---|
| **+** | Penambahan unary | +a |
| **-** | Pengurangan unary (inversi tanda positif/negatif) | -a |
| **!** | not (menginversi nilai) | !a |
| **++** | Menambah nilai variabel dengan 1 | ++a |
| **--** | Mengurangi nilai variabel dengan 1 | --a |

Cobalah kode berikut di kotlin playground untuk dapat memahami cara kerja operator yang ditunjukkan di tabel 3 di atas.

```
fun main(args: Array<String>) {
    val a = 1
    val b = true
```

```
    var c = 1

    var result: Int
    var booleanResult: Boolean

    result = -a
    println("-a = $result")

    booleanResult = !b
    println("!b = $booleanResult")

    --c
    println("--c = $c")
}
```

Selanjutnya, untuk nilai boolean, kotlin menyediakan operator perbandingan seperti pada tabel 4 berikut ini.

*Tabel 4 Operator Perbandingan*

| Operator | Arti | Ekspresi |
|---|---|---|
| > | Lebih dari | a > b |
| < | Kurang dari | a < b |
| >= | Lebih dari sama dengan | a >= b |
| <= | Kurang dari sama dengan | a <= b |
| == | Sama dengan | a == b |
| != | Tidak sama dengan | a != b |

Cobalah kode berikut ini di kotlin playground untuk mempelajari cara kerja operator perbandingan.

```
fun main(args: Array<String>) {

    val a = -12
    val b = 12

    // use of greater than operator
    val max = if (a > b) {
        println("a is larger than b.")
        a
    } else {
        println("b is larger than a.")
        b
    }

    println("max = $max")
}
```

Terakhir, kotlin menyediakan operator logika untuk operasi boolean. Berikut ini adalah daftar operator logika.

*Tabel 5 Operator Logika*

| Operator | Deskripsi | Ekspresi |
|---|---|---|
| \|\| | Operator or. Akan menghasilkan true jika ekspresi di sebelah kanan atau kiri-nya bernilai true. | (a>b)\|\|(a<c) |
| && | Operator and. Akan menghasilkan true jika ekspresi di sebelah kanan dan kiri-nya bernilai true. | (a>b)&&(a<c) |

Berikut ini adalah contoh kode penggunaan operator logika. Silakan coba di kotlin playground.

```
fun main(args: Array<String>) {

    val a = 10
    val b = 9
    val c = -1
    val result: Boolean

    result = (a>b) && (a>c) // result = (a>b) and (a>c)
    println(result)
}
```

## Array dan Collection

Bahasa pemrograman kotlin juga mendukung tipe data *array* dan *collection* untuk menampung kumpulan nilai dengan tipe data yang sama. Berikut ini adalah contoh deklarasi variabel dengan tipe *array*.

```
val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
```

Untuk mengakses sebuah elemen dari array, kita dapat menggunakan operator pengakses indeks berupa kurung siku [ ]. Contoh penggunaannya dapat dilihat pada kode berikut.

```
println(cars[1])
```

Untuk mengetahui kapasitas array, kita dapat menggunakan atribut size dari variabel array tersebut.

```
cars.size
```

Selain array, kotlin mendukung collection seperti list yang bersifat lebih fleksibel. Berikut ini adalah contoh deklarasi variabel dengan tipe list.

```
val cars = listOf("Volvo", "BMW", "Ford", "Mazda")
```

Seperti array, list juga memiliki atribut size. Bedanya, size pada list digunakan untuk mengetahui jumlah elemen yang dimilikinya. Kita juga dapat mengakses elemen list dengan operator pengakses indeks.

## Pengkondisian

Terdapat dua jenis pengkondisian yang didukung oleh kotlin, yaitu **if else** dan **when**. Sintaks yang digunakan untuk pengkondisian if else tidak berbeda jauh dengan sintaks if else pada bahasa pemrograman lain, berikut ini contohnya.

```
if (20 > 18) {
  println("20 is greater than 18")
}
```

Ekspresi dalam kurung harus merupakan ekspresi yang bernilai boolean, kita dapat memanfaatkan operator logika dan operator perbandingan seperti lebih dari yang ditunjukkan pada contoh di atas. Sintaks pengkondisian **when** mirip dengan pengkondisian **case** di bahasa pemrograman lain, seperti contoh berikut ini.

```
val day = 4

val result = when (day) {
  1 -> "Monday"
  2 -> "Tuesday"
  3 -> "Wednesday"
  4 -> "Thursday"
  5 -> "Friday"
  6 -> "Saturday"
  7 -> "Sunday"
  else -> "Invalid day."
}
println(result)
```

### Latihan 2

Tambahkan fungsi main untuk kode di atas, kemudian ubah nilai variabel day sehingga kode tersebut menghasilkan output Sunday. Jalankan di kotlin playground.

## Pengulangan

Untuk pengulangan, kotlin mendukung pengulangan **while, do while,** dan **for**. Berikut ini adalah contoh pengulangan **while.**

```
var i = 0
while (i < 5) {
  println(i)
  i++
}
```

Sedangkan untuk pengulangan **do while,** kode dalam badan pengulangan dieksekusi terlebih dahulu dibandingkan dengan pengkondisiannya. Berikut adalah contohnya.

```
var i = 0
```

```
do {
  println(i)
  i++
  }
while (i < 5)
```

Pengulangan for dalam bahasa kotlin hanya dapat digunakan untuk menelusuri **array** atau **collection**, dengan menggunakan operator **in**. Seperti pada contoh kode berikut ini.

```
val nums = arrayOf(1, 5, 10, 15, 20)
for (x in nums) {
  println(x)
}
```

Namun demikian, kita dapat menelusuri rentang nilai tertentu seperti contoh kode berikut.

```
val x = 10
val y = 9
if (x in 1..y+1) {
    println(x)
}
```

### Latihan 3

Buatlah sebuah program untuk menelusuri array berikut ini.

```
val cars = arrayOf("Volvo", "BMW", "Ford", "Mazda")
```

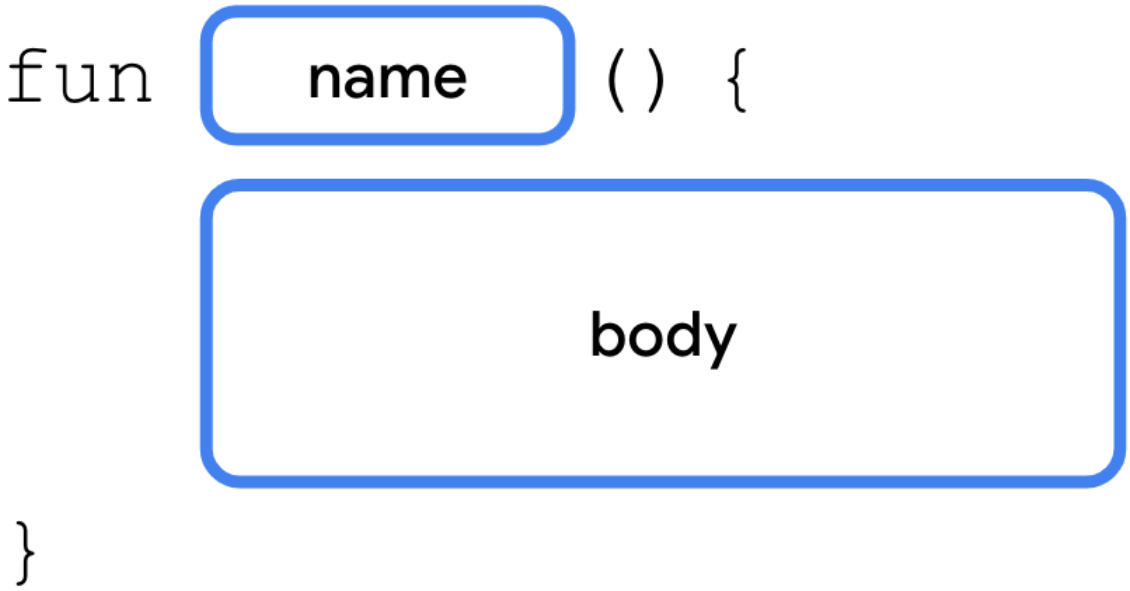
Tampilkan setiap nilai dari elemen array tersebut kecuali Ford.

## Fungsi

Seperti yang telah dijelaskan sebelumnya, kotlin memiliki kemampuan untuk membuat fungsi (seperti fungsi main) dengan atau tanpa *return value*. Perhatikan contoh berikut ini.

```
fun main() {
    println("Happy Birthday, Rover!")
    println("You are now 5 years old!")
}
```

Potongan kode di atas memperlihatkan sebuah fungsi bernama **main**, tanpa parameter, dengan badan berisi 2 baris kode memanggil fungsi lain yaitu **println**.



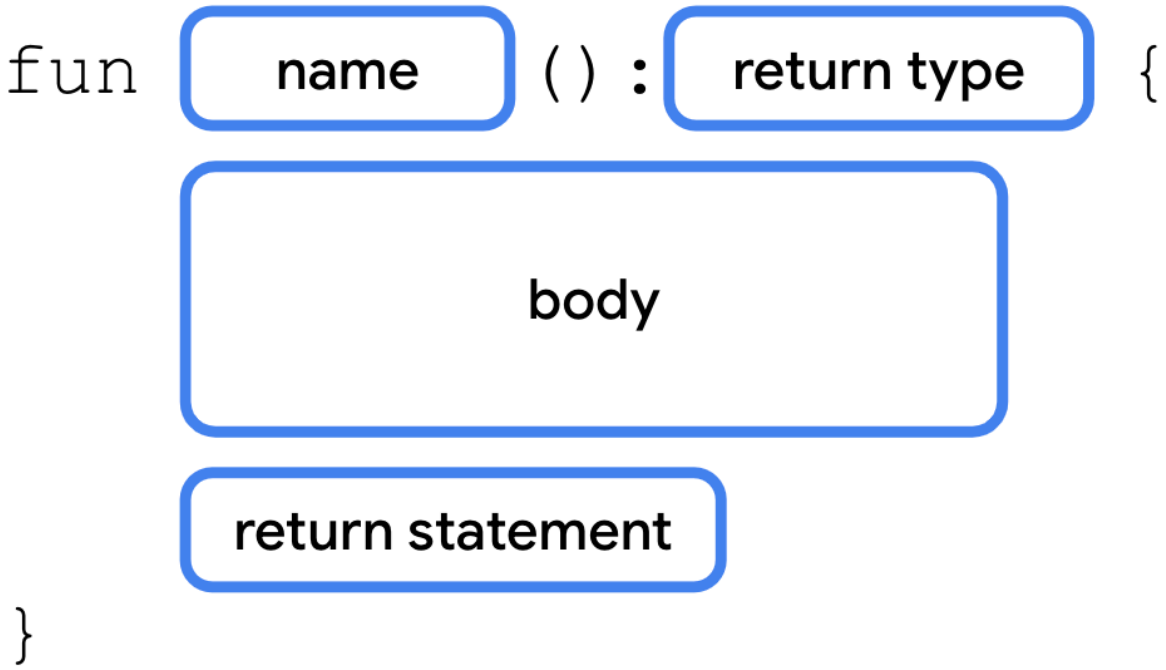

*Gambar 5 Struktur Fungsi*

Untuk membuat fungsi baru di sebuah file program, anda dapat menulis fungsi tersebut di atas atau di bawah fungsi **main**, seperti pada contoh berikut ini.

```
fun main() {
    birthdayGreeting()
}

fun birthdayGreeting() {
    println("Happy Birthday, Rover!")
    println("You are now 5 years old!")
}
```

Fungsi **birthdayGreeting** dapat dipanggil dengan kode **birthdayGreeting()** dari mana saja, seperti yang ditunjukkan pada kode diatas (**birthdayGreeting** dipanggil di fungsi **main**). Untuk fungsi yang memiliki *return value*, kita harus menambahkan tipe data pada *header* fungsi dan kata kunci **return** diikuti dengan nilai kembalian pada badan fungsi seperti pada gambar 6 berikut.



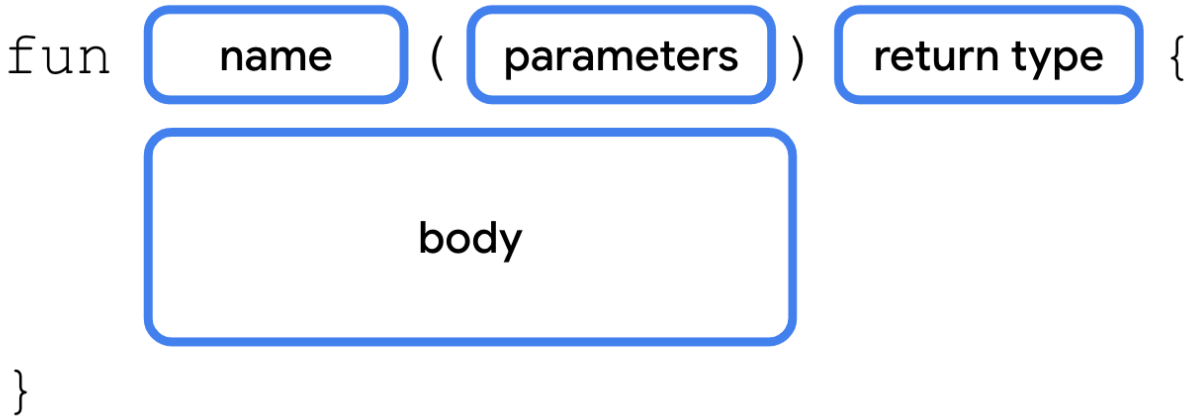

*Gambar 6 Struktur Fungsi dengan Return Value*

Dengan demikian, jika kita ingin mengubah fungsi birthdayGreeting agar mengembalikan ucapan selamat ulang tahun dan tidak mencetaknya langsung di console, kita dapat mengubah kode fungsi tersebut menjadi seperti berikut ini.

```
fun birthdayGreeting(): String {
    val nameGreeting = "Happy Birthday, Rover!"
    val ageGreeting = "You are now 5 years old!"
    return "$nameGreeting\n$ageGreeting"
}
```

**Catatan: kotlin memiliki kemampuan untuk menggabungkan string di dalam literal (kutip dua) dengan tanda $ di depan nama variabel**

Kita juga dapat menambahkan parameter pada fungsi tersebut dengan struktur sebagai berikut.




*Gambar 7 Struktur Fungsi dengan Parameter*

Contohnya, kita dapat menambahkan parameter nama untuk fungsi birthdayGreeting seperti pada kode berikut.

```
fun birthdayGreeting(name: String): String {
    val nameGreeting = "Happy Birthday, $name!"
    val ageGreeting = "You are now 5 years old!"
    return "$nameGreeting\n$ageGreeting"
}
```

Untuk fungsi dengan lebih dari satu parameter, kita dapat menggunakan pemisah koma seperti berikut ini.

```
fun birthdayGreeting(name: String, age: Int): String {
    val nameGreeting = "Happy Birthday, $name!"
    val ageGreeting = "You are now $age years old!"
    return "$nameGreeting\n$ageGreeting"
}
```

Terdapat dua cara untuk memanggil fungsi berparameter, pertama dengan memperhatikan urutan seperti:

```
birthdayGreeting("Rover", 5)
```

Kedua, kita dapat menggunakan named parameter tanpa harus memperhatikan urutan parameter, contohnya:

```
birthdayGreeting(name = "Rover", age = 5)
birthdayGreeting(age = 5, name = "Rover")
```

Kotlin juga memungkinkan parameter sebuah fungsi memiliki nilai default, contohnya seperti pada kode berikut:

```
fun birthdayGreeting(name: String, age: Int = 0): String {
    val nameGreeting = "Happy Birthday, $name!"
    val ageGreeting = "You are now $age years old!"
    return "$nameGreeting\n$ageGreeting"
}
```

Fungsi birthdayGreeting di atas dapat dipanggil hanya dengan menggunakan 1 parameter saja, yaitu name. Pemanggilan birthdayGreeting("Rover") misalnya, akan menghasilkan:

```
Happy Birthday, Rover!
You are now 0 years old!
```

Karena kotlin mengenal named parameter, maka nilai default parameter dapat dipasang hanya untuk parameter pertama seperti pada kode berikut. Hal ini membedakan kotlin dari *PHP* misalnya, yang mengharuskan parameter kedua memiliki nilai default jika parameter pertama memiliki nilai *default*.

```
fun birthdayGreeting(name: String = "John", age: Int): String {
    val nameGreeting = "Happy Birthday, $name!"
    val ageGreeting = "You are now $age years old!"
    return "$nameGreeting\n$ageGreeting"
}
```

Untuk menggunakan nilai *default* dari parameter **name**, kita tidak bisa mengandalkan urutan parameter sehingga harus menggunakan named parameter untuk **age**.

```
birthdayGreeting(age = 5)
```

### Latihan 4

Tambahkan parameter ke-3 dengan nama date bertipe string untuk fungsi birthdayGreeting. Ubah kode fungsi tersebut sehingga jika kita inputkan name = "Rover", age = 5, date = 17/08/2022 maka fungsi akan mengembalikan:

```
Today is 17/08/2022.
Happy Birthday, Rover!
You are now 5 years old!
```