Akhmad Fauzi (PC-11)

Mobile Aplication and Technology

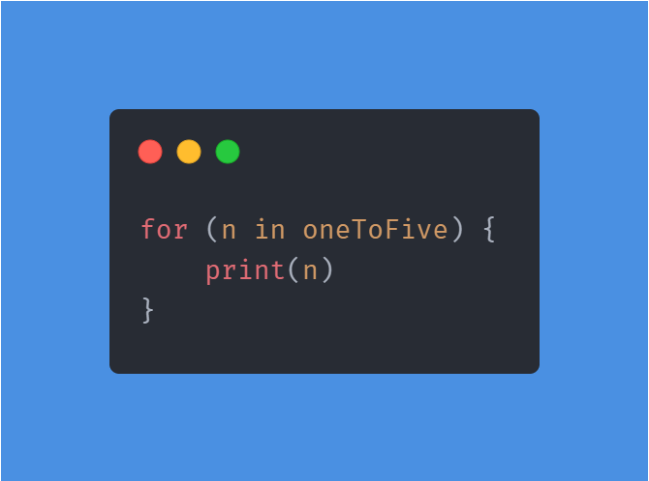
**Ranges**

Sebuah Range di Kotlin adalah tipe unik yang mendefinisikan nilai awal dan nilai akhir. Dengan kata lain, ini adalah interval antara awal dan nilai akhir. Range di Kotlin ditutup, yang berarti bahwa nilai awal dan nilai akhir termasuk dalam range.

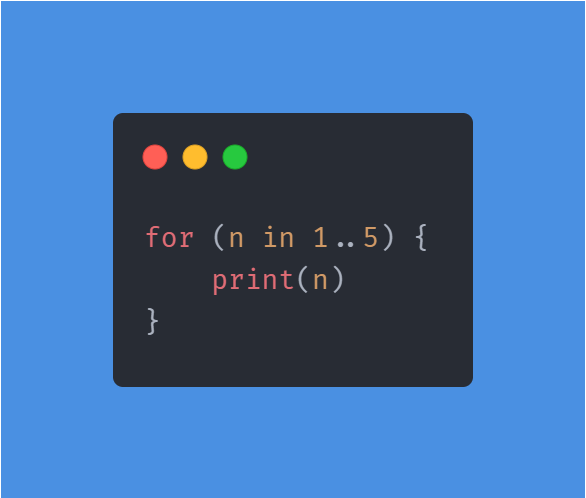
Sekarang kita akan melihat berbagai cara untuk menciptakan range di Kotlin.

**Operator**

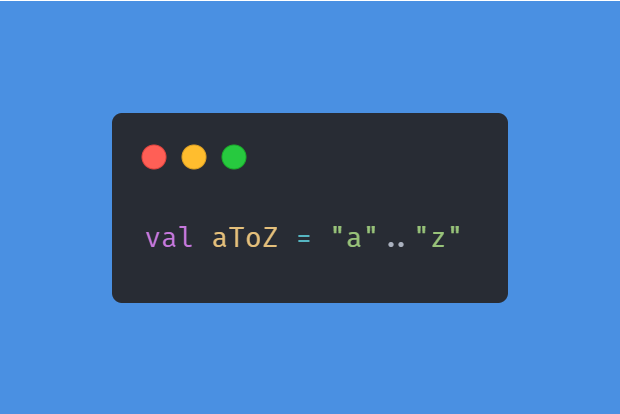


Dalam kode di atas, kami telah membuat rentang tertutup. Variabel oneToFive ini akan menyertakan nilai-nilai berikut: 1, 2, 3, 4, 5. Kita dapat mengulangnya menggunakan for konstruk loop.

Kode di atas dapat dipersingkat menjadi:



Kami juga dapat membuat berbagai karakter:



Variabel aToZ akan memiliki semua huruf dalam abjad Inggris.

### Fungsi rangeTo()

Operator .. dapat diganti dengan fungsi ekstensi rangeTo() untuk membuat range. Sebagai contoh, kita juga dapat melakukan ini 1.rangeTo(5) dan itu akan tetap memiliki hasil yang sama seperti menggunakan operator .. seperti yang dibahas sebelumnya.



### Fungsi downTo()

Ini adalah fungsi ekstensi lain yang akan membuat rentang mulai dari angka yang diberikan hingga yang lain.



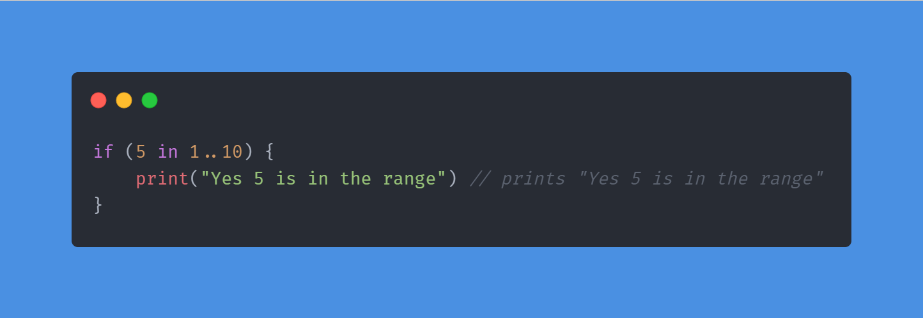
Kita dapat memodifikasi rentang menggunakan fungsi step(). Ini akan memodifikasi delta antara setiap elemen dalam range.



Kode di atas akan berisi angka ganjil antara 1 dan 10.

### Operator in

Operator in digunakan untuk memastikan apakah suatu nilai hadir dalam range yang diberikan.



Dalam kode di atas, kami memeriksa untuk melihat apakah 5 berada dalam rentang 1..10 menggunakan operator in. Kita juga dapat melakukan yang sebaliknya dengan menggunakan !n ​​untuk memeriksa apakah 5 tidak dalam range.

**COLLECTION**

Collections digunakan untuk menyimpan kelompok objek terkait dalam memori. Dalam collection, kita dapat mengambil, menyimpan, atau mengatur objek. Kotlin menyediakan koleksi API-nya sebagai [pustaka standar](https://kotlinlang.org/api/latest/jvm/stdlib/index.html) yang dibangun di atas Java Collections API. (Kami akan membahas antarmuka di Kotlin di posting masa depan.)

Anda harus mencatat bahwa antarmuka ini terkait dengan penerapannya pada waktu kompilasi. Anda tidak dapat melihat kode sumber implementasi di Kotlin, karena koleksi benar-benar diimplementasikan oleh Koleksi Java standar seperti ArrayList, Maps, HashMap, Sets, HashSet, List dan sebagainya. Untuk benar-benar memahami koleksi API di Kotlin, Anda harus terbiasa dengan kelas dasar dan antarmuka di Java ini.

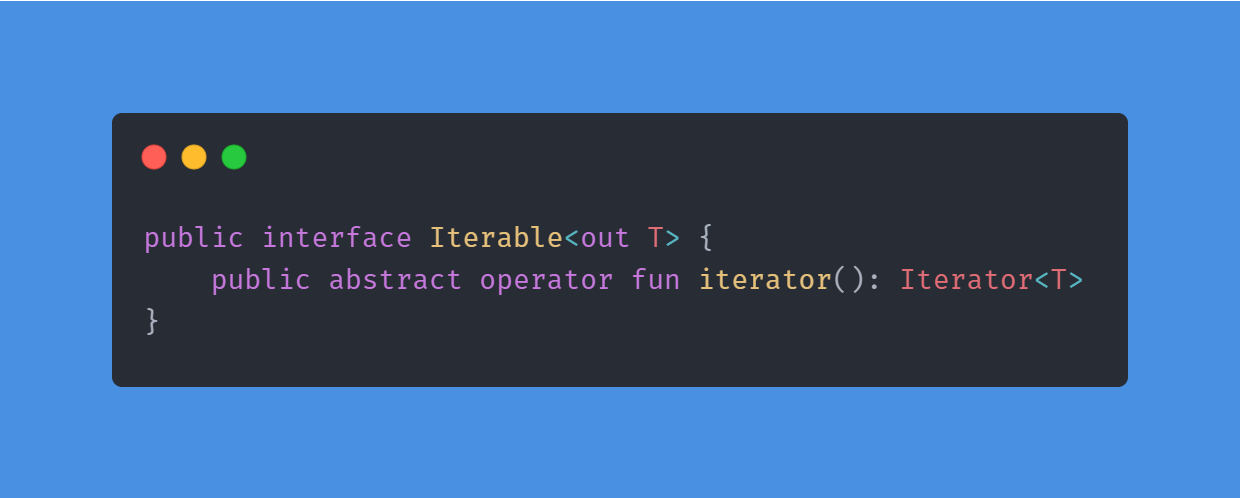
Di bagian ini, kita akan belajar tentang List, Set dan Map collection di Kotlin. (Jika Anda menginginkan penyegaran pada array di Kotlin, silakan kunjungi [tutorial pertama dalam seri ini](https://code.tutsplus.com/tutorials/kotlin-from-scratch-variables-basic-types-arrays-type-inference-and-comments--cms-29328?_ga=2.45579620.1796967716.1502453273-1099501081.1478521852).)

Koleksi Kotlin memberi kita kemampuan untuk mencapai banyak hal dengan hanya sedikit kode—tidak seperti di Java, yang tampaknya membutuhkan banyak kode untuk mencapai sedikit! Kotlin memiliki dua varian collection: bisa berubah dan tidak berubah. Kumpulan yang bisa berubah memberi kita kemampuan untuk memodifikasi koleksi dengan menambahkan, menghapus, atau mengganti elemen. Collection yang tidak dapat diubah tidak dapat dimodifikasi dan tidak memiliki metode penolong ini.

Perhatikan bahwa penambahan, penghapusan atau penggantian elemen dalam collection tidak berubah dimungkinkan melalui fungsi operator (kita akan segera mendapatkannya), tetapi ini akhirnya akan membuat collection baru.

**Antarmuka Iterable**

Antarmuka [Iterable](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-iterable/index.html) Kotlin ada di bagian atas hierarki kelas koleksi. Antarmuka ini memungkinkan koleksi direpresentasikan sebagai urutan elemen (yang dapat diulang secara alami).

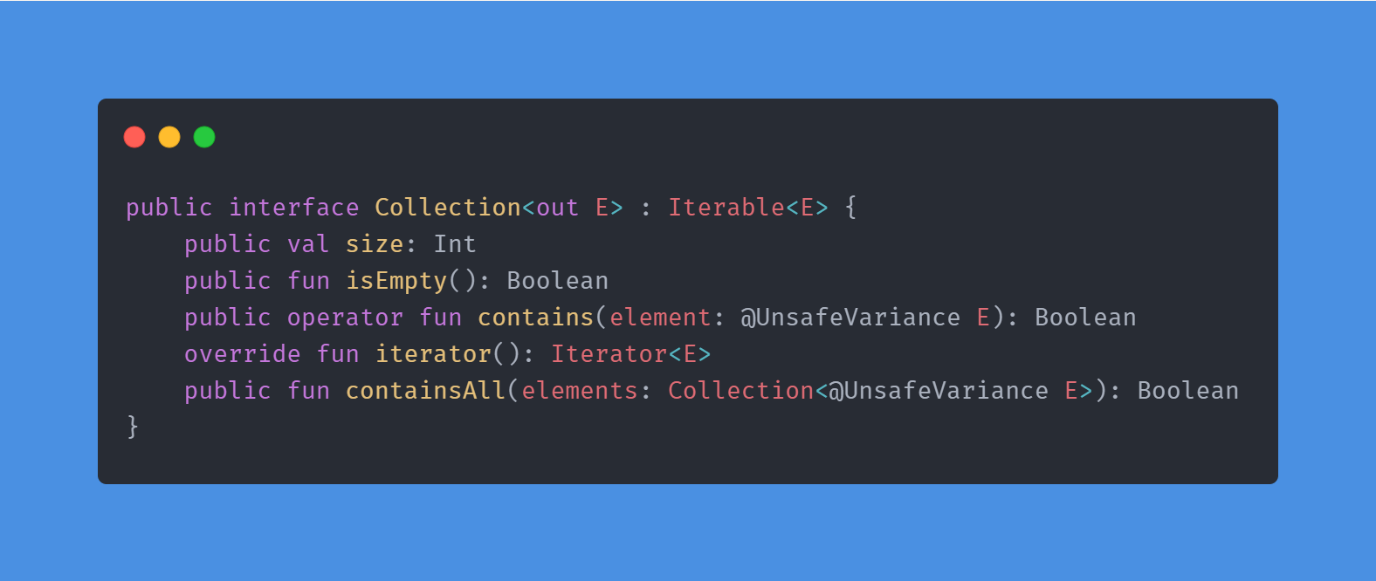


### Antarmuka Collection

Antarmuka [Collection](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-collection/index.html) Kotlin memperluas antarmuka [Iterable](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-iterable/index.html). Antarmuka Collection tidak dapat diubah. Dengan kata lain, Anda memiliki akses hanya-baca ke koleksi. Antarmuka Set dan List (lebih lanjut tentang ini segera) di Kotlin memperpanjang antarmuka ini.

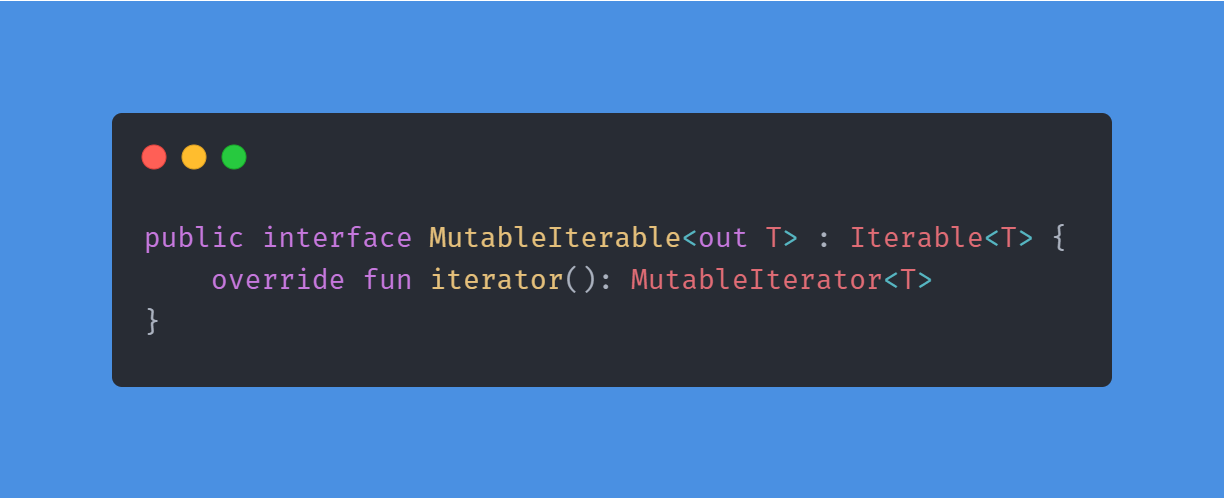
Beberapa fungsi dan properti yang tersedia di antarmuka Collection adalah:

* size: properti ini mengembalikan ukuran collection.
* isEmpty(): mengembalikan nilai true jika collection kosong atau salah jika sebaliknya.
* contains(element: E): mengembalikan nilai true jika elemen yang ditentukan dalam argumen ada dalam collection.
* containsAll(element: Collection<E>): mengembalikan nilai true jika elemen dalam collection dilewatkan ketika argumen ada dalam collection.



### Antarmuka MutableIterable

Antarmuka ini di Kotlin memberi kita iterator yang bisa berubah-ubah dari parent antarmuka Iterable.



**Antarmuka MutableCollection**

Antarmuka [MutableCollection](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-mutable-collection/index.html) di Kotlin adalah antarmuka khusus yang memungkinkan koleksi menjadi bisa berubah. Dengan kata lain, menambah dan menghapus operasi dapat dilakukan pada koleksi yang diberikan. Antarmuka ini memperluas antarmuka Collection dan antarmuka MutableIterable yang sudah dibahas di atas. Antarmuka MutableSet dan MutableList (kita akan segera mendapatkannya) di Kotlin memperluas antarmuka ini. Fungsi-fungsi yang tersedia di antarmuka ini—terlepas dari yang tersedia di parent—adalah:

* add(elemen: E): menambahkan elemen yang dilewatkan sebagai argumen ke collection dan mengembalikan nilai true jika berhasil atau salah jika collection tidak mendukung duplikat dan elemen sudah ada.
* remove(element: E): menghilangkan elemen yang dilewatkan sebagai argumen dari collection. Mengembalikan nilai true jika berhasil atau salah jika tidak ada dalam collection.
* addAll(elements: Collection<E>): menambahkan semua elemen dalam collection yang dilewatkan sebagai argumen ke collection. Mengembalikan nilai true jika berhasil atau salah jika tidak ada yang ditambahkan.
* removeAll(elements: Collection<E>): menghapus semua elemen yang ada dalam collection yang dilewatkan sebagai argumen. Mengembalikan nilai true jika berhasil atau salah jika tidak ada yang dihapus.
* retainAll(elements: Collection<E>): hanya mempertahankan elemen yang ada dalam collection yang dilewatkan sebagai argumen. Mengembalikan nilai true jika berhasil atau salah jika tidak ada yang dipertahankan.
* clear(): menghapus semua elemen dari collection ini.



Sekarang Anda telah belajar tentang antarmuka teratas dalam hirarki kelas collection di Kotlin, sekarang mari kita lihat bagaimana Kotlin menangani collection seperti Daftar, Kumpulan, dan Peta di bagian yang tersisa dari tutorial.

## Lists

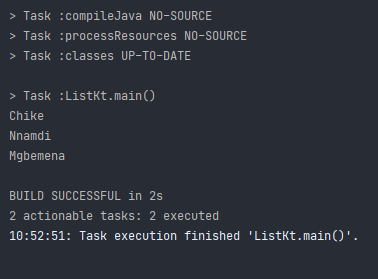
Daftar adalah kumpulan elemen yang dipesan. Ini adalah collection populer yang banyak digunakan. Mari kita lihat cara-cara berbeda untuk membuat daftar di Kotlin.

### Menggunakan fungsi listOf()

Di Kotlin, kita dapat membuat daftar yang tidak dapat diubah (read-only) menggunakan fungsi helip listOf() dari perpustakaan standar Kotlin. Fungsi ini mengembalikan tipe antarmuka Kotlin [List](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-list/index.html).



Kode di atas mencetak:

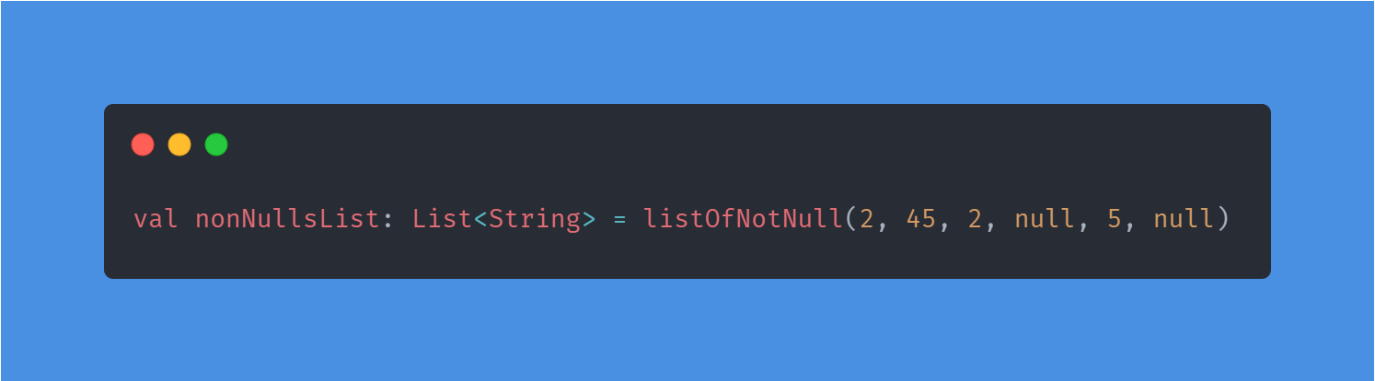


Selain itu, kita dapat memberikan nilai dari berbagai jenis ke dalam listOf() sebagai argumen dan hasilnya akan tetap berfungsi—ini akan menjadi daftar tipe campuran.



### Menggunakan fungsi listOfNotNull()

Fungsi ini membuat daftar baru yang tidak berubah hanya berisi elemen yang tidak null. Perhatikan bahwa fungsi ini mengembalikan jenis antarmuka Kotlin List juga.



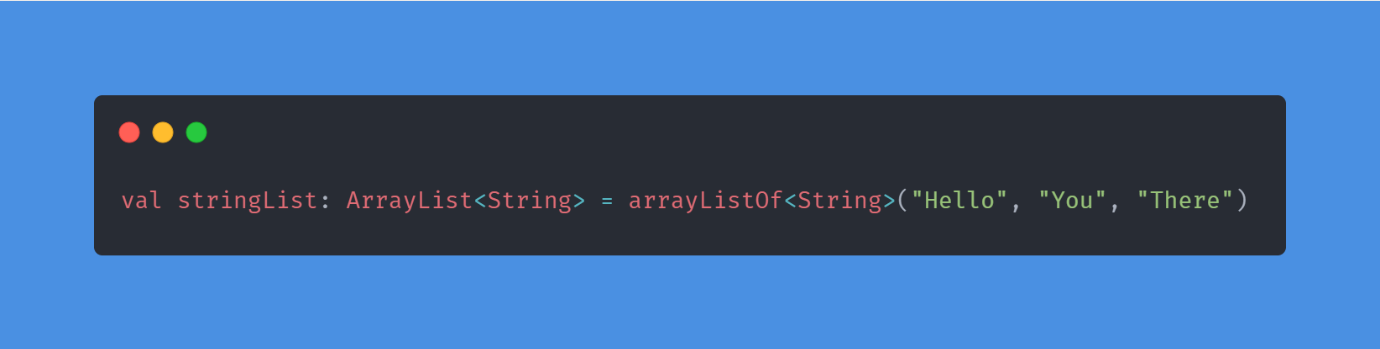
Antarmuka [List](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-list/index.html) dari perpustakaan standar Kotlin hanya memperluas antarmuka Collection. Dengan kata lain, satu-satunya parent adalah antarmuka Collection. Ini mengabaikan semua fungsi dalam antarmuka orangtua untuk memenuhi kebutuhan khusus dan juga mendefinisikan fungsi sendiri, seperti:

* get(index: Int): operator fungsi yang mengembalikan elemen pada indeks yang ditentukan.
* indexOf(element: E): mengembalikan indeks dari kemunculan pertama dari elemen yang dilewatkan sebagai argumen dalam daftar, atau -1 jika tidak ada yang ditemukan.
* lastIndexOf(elemen: E): mengembalikan indeks kemunculan terakhir dari elemen yang dilewatkan sebagai argumen dalam daftar, atau -1 jika tidak ada yang ditemukan.
* listIterator(): mengembalikan daftar iterator atas elemen-elemen dalam daftar.
* subList(fromIndex: Int, toIndex: Int): mengembalikan daftar yang berisi bagian daftar antara awal dan akhir indeks yang ditentukan.



### Menggunakan fungsi arrayListOf()

Ini menciptakan daftar yang bisa berubah dan mengembalikan tipe Java ArrayList.



### Menggunakan Fungsi mutableListOf()

Untuk menambah, menghapus atau mengganti nilai dalam daftar, kita perlu membuat daftar yang dapat diubah. Kita dapat mengubah daftar tidak berubah menjadi yang bisa berubah dengan memanggil fungsi toMutableList() pada daftar. Namun, perhatikan bahwa metode ini akan membuat daftar baru.



Untuk membuat daftar yang dapat diubah dari jenis tertentu dari awal, mis. String, kita menggunakan mutableListOf<String>(), sedangkan untuk tipe campuran kita bisa menggunakan fungsi mutableListOf().

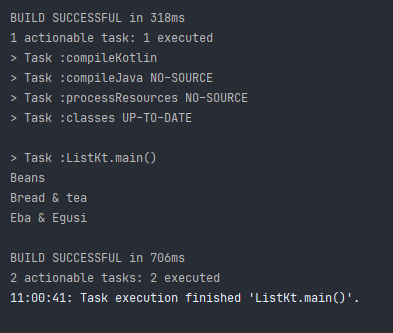


Fungsi-fungsi ini akan mengembalikan tipe antarmuka Kotlin MutableList. Antarmuka ini memperluas kedua antarmuka MutableCollection dan List dibahas sebelumnya di bagian ini. Antarmuka MutableList menambahkan metode untuk pengambilan atau penggantian item berdasarkan posisinya:

* set(index: Int, element: E): mengganti elemen dalam daftar dengan elemen lain. Ini mengembalikan elemen sebelumnya pada posisi yang ditentukan.
* add(index: Int, element: E): menyisipkan elemen pada indeks yang ditentukan.
* removeAt(index: Int): menghilangkan elemen pada indeks tertentu.



Menjalankan kode di atas, kami menghasilkan hasil sebagai berikut:



Perhatikan bahwa semua fungsi ini membuat Java ArrayList di belakang layar.

## Sets

Sebuah set adalah koleksi elemen unik yang tidak disusun. Dengan kata lain, tidak dapat memiliki duplikat! Mari kita lihat beberapa cara berbeda untuk membuat sebuah set di Kotlin. Masing-masing menciptakan struktur data yang berbeda, yang masing-masing dioptimalkan untuk jenis tugas tertentu.

### Menggunakan fungsi setOf()

Untuk membuat set immutable (read-only) di Kotlin, kita bisa menggunakan fungsi setOf(), yang mengembalikan tipe antarmuka Kotlin [Set](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-set/index.html).



Perhatikan bahwa antarmuka Kotlin Set hanya memperluas antarmuka Collection Kotlin dan menimpa semua properti yang tersedia di induknya.

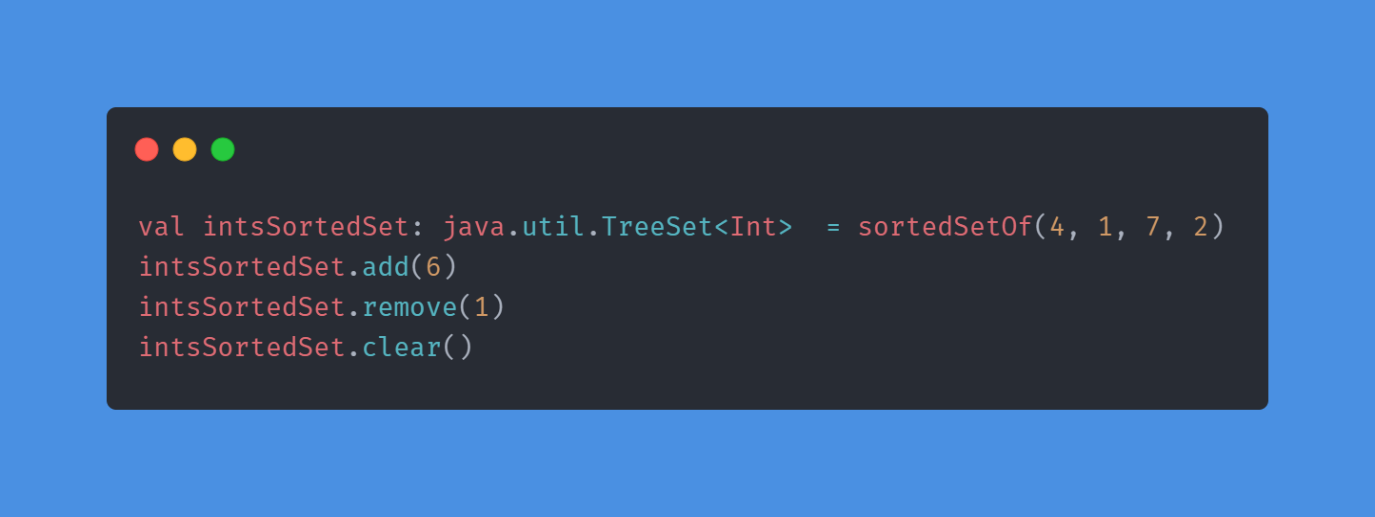
### Menggunakan Fungsi hashSetOf()

Menggunakan fungsi hashSetOf() menciptakan koleksi Java [HashSet](https://docs.oracle.com/javase/7/docs/api/java/util/HashSet.html) yang menyimpan elemen dalam tabel hash. Karena fungsi ini mengembalikan tipe Java HashSet, kita bisa menambahkan, menghapus, atau menghapus elemen di set. Dengan kata lain, itu bisa berubah.



### Menggunakan Fungsi sortedSetOf()

Menggunakan fungsi sortetSetOf() membuat koleksi Java [TreeSet](https://docs.oracle.com/javase/7/docs/api/java/util/TreeSet.html) di belakang layar, yang memerintahkan elemen berdasarkan pada pemesanan alami mereka atau oleh komparator. Set ini juga bisa berubah.



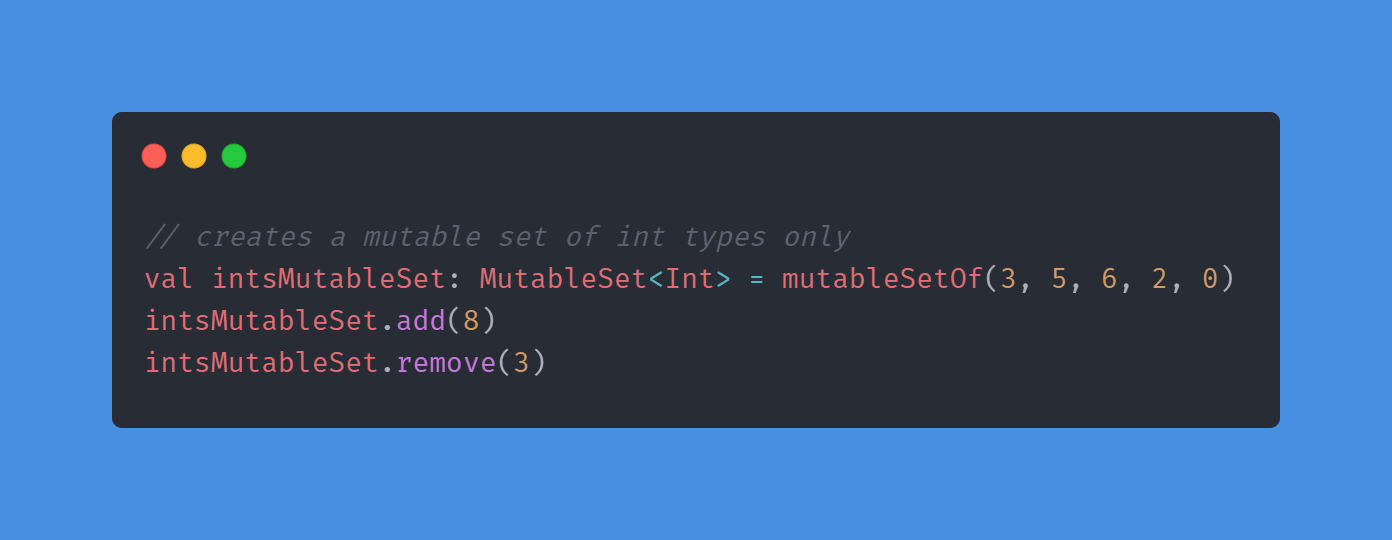
### Menggunakan Fungsi linkedSetOf()

Fungsi ini mengembalikan tipe Java [LinkedHashSet](https://docs.oracle.com/javase/7/docs/api/java/util/LinkedHashSet.html). Set yang dapat berubah ini mempertahankan daftar yang ditautkan dari entri di set, dalam urutan di mana mereka disisipkan.



### Menggunakan Fungsi mutableSetOf()

Kita bisa menggunakan mutableSetOf() untuk membuat set yang bisa berubah. Fungsi ini mengembalikan tipe antarmuka Kotlin [MutableSet](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-mutable-set/index.html). Di belakang layar, fungsi ini hanya menciptakan Java LinkedHashSet.



Antarmuka MutableSet memperluas antarmuka MutableCollection dan Set.

## Maps

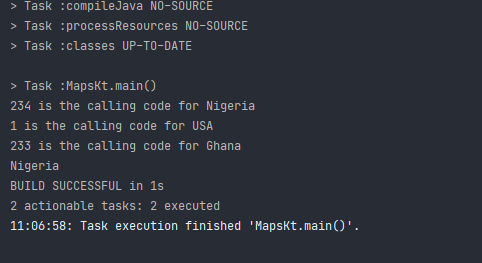
Maps mengaitkan kunci dengan nilai. Kunci harus unik, tetapi nilai yang terkait tidak perlu. Dengan begitu, setiap kunci dapat digunakan untuk mengidentifikasi nilai terkait secara unik, karena peta memastikan Anda tidak memiliki kunci duplikat dalam koleksi. Di belakang layar, Kotlin menggunakan koleksi Map Java untuk mengimplementasikan jenis koleksi peta.

### Menggunakan Fungsi mapOf()

Untuk membuat koleksi Map yang tidak berubah atau hanya-baca di Kotlin, kami menggunakan fungsi mapOf(). Kami membuat peta dengan fungsi ini dengan memberikan daftar pasangan—nilai pertama adalah kunci, dan yang kedua adalah nilainya. Memanggil fungsi ini mengembalikan tipe antarmuka Kotlin [Map](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/-map/index.html).



Menjalankan kode di atas akan menghasilkan hasil:



Tidak seperti List dan Set interface di Kotlin yang memperluas antarmuka Collection, antarmuka Map tidak memperpanjang sama sekali. Beberapa properti dan fungsi yang tersedia di antarmuka ini adalah:

* size: properti ini mengembalikan ukuran koleksi peta.
* isEmpty(): mengembalikan nilai true jika peta kosong atau salah jika sebaliknya.
* containsKey(key: K): mengembalikan nilai true jika peta berisi kunci dalam argumen.
* containsValue(value: V): mengembalikan nilai true jika peta memetakan satu atau lebih kunci ke nilai yang dilewatkan sebagai argumen.
* get(key: K): mengembalikan nilai yang cocok dengan kunci yang diberikan atau "null" jika tidak ada yang ditemukan.
* keys: properti ini mengembalikan Set yang tidak berubah dari semua kunci di peta.
* values: mengembalikan Collection immutable dari semua nilai di peta.

**Menggunakan Fungsi mutableMapOf()**

Fungsi MutableMapOf() membuat peta yang bisa diubah agar kami dapat menambah dan menghapus elemen di peta. Ini mengembalikan jenis antarmuka Kotlin MutableMap.



Antarmuka MutableMap tidak memperluas antarmuka MutableCollection; hanya orang tua adalah antarmuka Map. Ini menimpa keys, entries dan values properti dari antarmuka induk untuk mendefinisikan ulang mereka. Berikut adalah beberapa fungsi tambahan yang tersedia di antarmuka MutableMap:

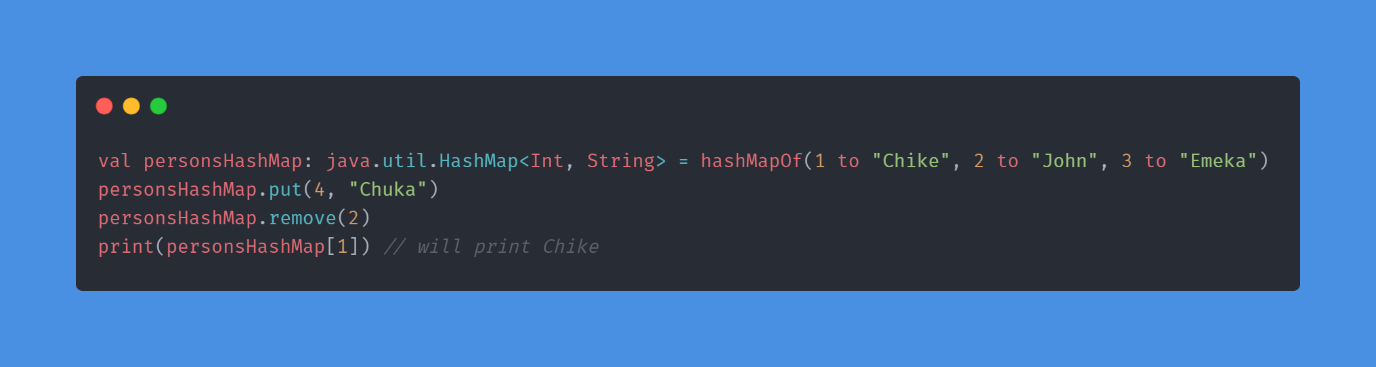
* put(key: K, value: V): menyisipkan kunci, pasangan nilai ke dalam peta. Ini akan mengembalikan nilai sebelumnya yang terhubung dengan kunci atau null jika kunci tersebut tidak digunakan sebelumnya.
* remove(key: K): menghapus kunci dan nilai terkait dari peta.
* putAll(from: Map<out K, V>): memperbarui peta dengan semua data dari peta yang diberikan. Kunci baru akan ditambahkan, dan kunci yang ada akan diperbarui dengan nilai-nilai baru.
* clear(): menghapus semua elemen dari peta.

Kita bisa mendapatkan nilai untuk kunci menggunakan fungsi get(). Kita juga bisa menggunakan notasi braket persegi sebagai cara pintas untuk get().



### Menggunakan Fungsi hashMapOf()

Menggunakan fungsi ini mengembalikan tipe Java [HashMap](https://docs.oracle.com/javase/7/docs/api/java/util/HashMap.html) yang bisa berubah. Kelas HashMap menggunakan tabel hash untuk mengimplementasikan antarmuka Map Java.



### Menggunakan Fungsi linkedHashMap()

Fungsi ini mengembalikan tipe Java [LinkedHashMap](https://docs.oracle.com/javase/7/docs/api/java/util/LinkedHashMap.html) yang bisa berubah. Kelas LinkedHashMap memperluas Java HashMap dan mempertahankan daftar terkait dari entri di peta dalam urutan di mana mereka disisipkan.



### Menggunakan Fungsi sortedMapOf()

Fungsi ini mengembalikan sejenis [SortedMap](https://docs.oracle.com/javase/7/docs/api/java/util/SortedMap.html) Jawa yang bisa berubah. Kelas Java SortedMap melihat bahwa entri dalam peta diselenggarakan dalam urutan menaik kunci.



Ingat, penerapan antarmuka koleksi ini di Kotlin terjadi pada waktu kompilasi.

## Fungsi Pengoperasian Koleksi

Kotlin memberi kita banyak fungsi operator yang berguna yang disebut fungsi ekstensi yang dapat dipanggil pada koleksi. Mari kita lihat beberapa yang paling berguna.

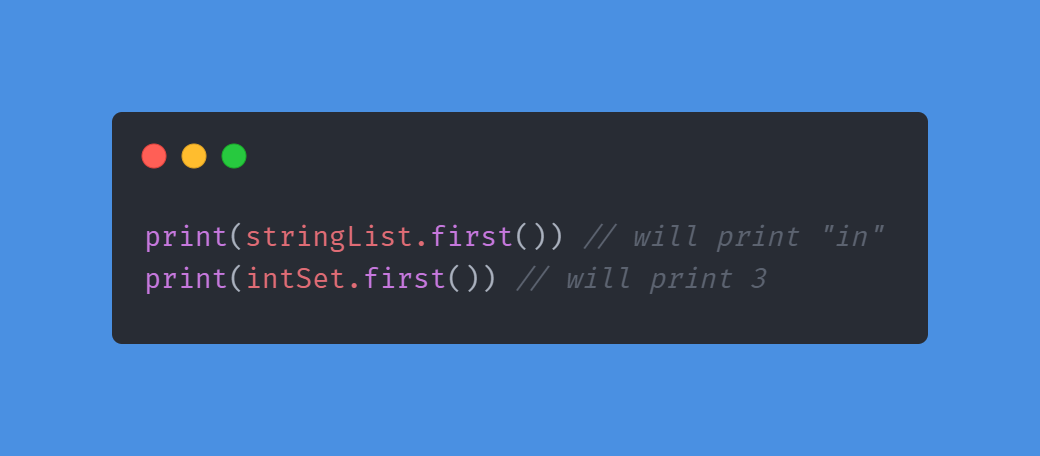
### Fungsi last()

Fungsi operator ini mengembalikan elemen terakhir dalam koleksi seperti daftar atau set. Kami juga dapat menyediakan predikat untuk mencari di dalam subset elemen.



### Fungsi first()

Fungsi operator ini mengembalikan elemen pertama saat dipanggil pada koleksi seperti daftar atau set. Jika predikat diberikan, ia kemudian menggunakan predikat untuk membatasi operasi ke subset elemen.



### Fungsi max()

Meminjam fungsi operator ini pada koleksi seperti daftar atau set mengembalikan elemen terbesar, atau null jika tidak ada elemen terbesar ditemukan.



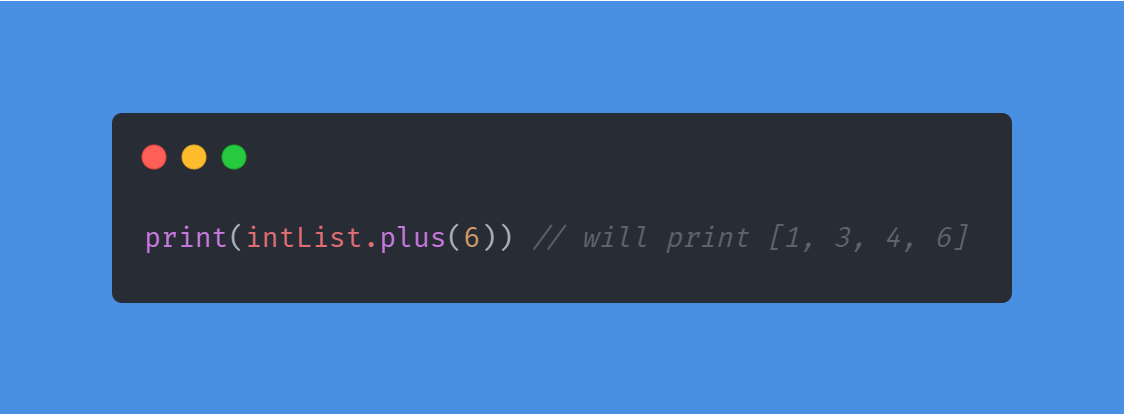
### Fungsi drop()

Memanggil fungsi operator ini mengembalikan daftar atau set baru yang berisi semua elemen kecuali elemen pertama.



### Fungsi plus()

Fungsi operator ini mengembalikan koleksi yang berisi semua elemen asli dan kemudian elemen yang diberikan jika belum ada dalam koleksi. Ini akan berakhir dengan membuat daftar baru, bukan memodifikasi daftar.



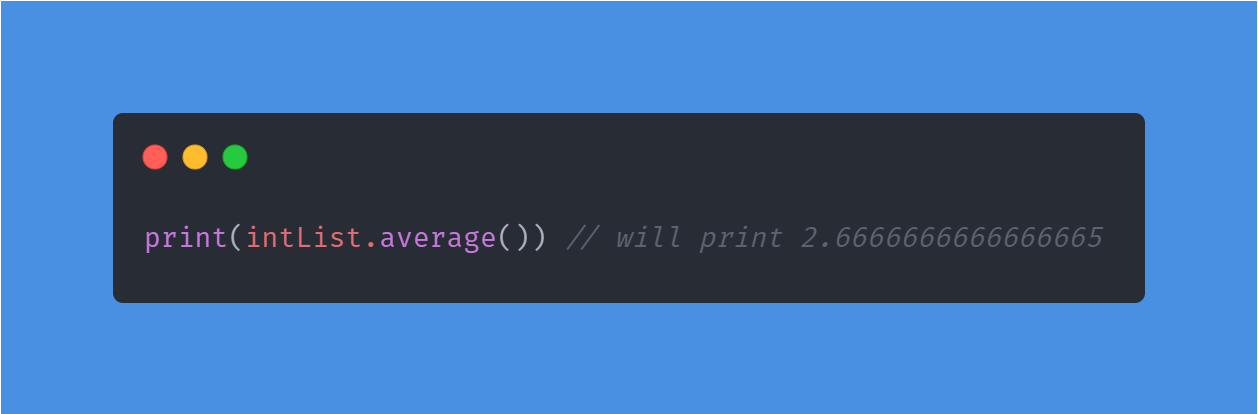
### Fungsi minus()

Kebalikan dari fungsi plus() adalah fungsi minus(). Ini mengembalikan koleksi yang berisi semua elemen dari set asli kecuali elemen yang diberikan. Ini juga akhirnya membuat daftar baru, bukannya mengubah daftar.



### Fungsi average()

Memanggil fungsi operator ini akan menghasilkan jumlah elemen rata-rata dalam koleksi.



Sebagian besar fungsi ekstensi ini tersedia di [perpustakaan standar koleksi Kotlin](https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/index.html). Anda disarankan untuk memeriksa dokumentasi untuk mempelajari tentang yang lain.

## Kesimpulan

Dalam tutorial ini, Anda belajar tentang jangkauan dan koleksi API dalam bahasa pemrograman Kotlin. Dalam tutorial berikutnya dalam seri Kotlin From Scratch, Anda akan diperkenalkan pada fungsi-fungsi di Kotlin. Sampai jumpa lagi!