MSIB
magang dan studi independen bersertifikat

INFINITE
LEARNING

Kotlin Collection

Lists, Sets, Maps,
Collection Functions

INFINITE
LEARNING

Collections

- Collections are objects that can store a collection of other objects.
- Collections can store a lot of data at once.
- In collections there are several child objects, including List, Set, and Map
- Kotlin provides the following types of collection:
  Collection or Immutable Collection, Mutable Collection

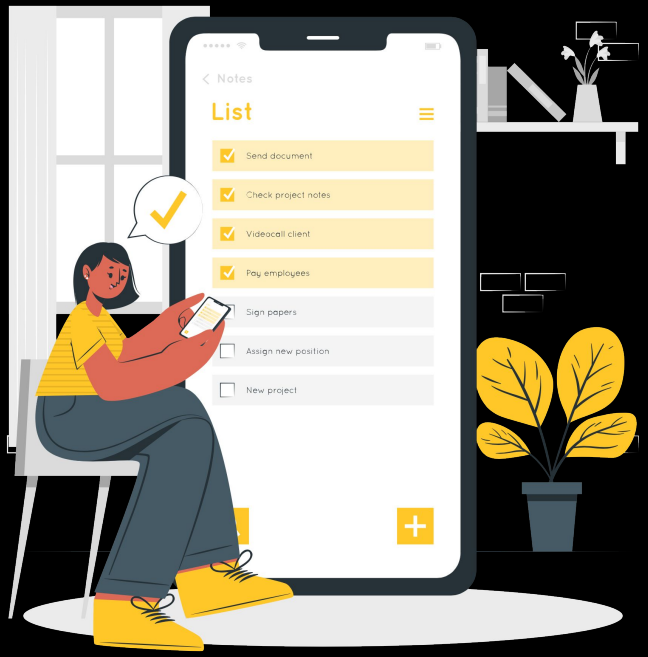
INFINITE
LEARNING

Table of Content

- List: listOf, mutableListOf
- Set: setOf
- Map: mapOf, .toMutableMap
- Collection Functions

INFINITE
LEARNING

Lists



List

INFINITE LEARNING

- Kotlin **List** is an ordered collection with access to elements by indices.
- A Kotlin list can be either mutable (mutableListOf) or read-only (listOf).
- The elements of list can be accessed using indices.
- Kotlin mutable or immutable lists can have duplicate elements.
- List can store data with different data types.

List: listOf()

INFINITE LEARNING

- For list creation, use the standard library functions **listOf()** for read-only lists.
- If all items are Strings, then Kotlin will define a list object of type String.

Let's code!

```
fun main() {

    val numberList = listOf("One", "Two", "Tree")
    println(numberList)

}
```

List: mutableListOf()

INFINITE LEARNING

- For list creation, use the standard library functions **mutableListOf()** for mutable lists.
- If all items are Integers, then Kotlin will define a list object of type Integer.
- So it can not change the value of an element of the object list with a different type.

Let's code!

```
fun main() {

    val numberList = mutableListOf(1, 2, 3)
    println(numberList)

}
```

List: mutableListOf()

INFINITE LEARNING

- For list creation, use the standard library functions **mutableListOf()** for mutable lists.
- So it can not change the value of an element of the object list with a different type.

Let's code...

```
fun main() {

    val numberList = mutableListOf(1, 2, 3)
    println(numberList)

    numberList[1] = "Two" // Type mismatch: inferred type is String but Int was expected

    println(numberList)

}
```

List<Any>

INFINITE LEARNING

Meanwhile, to create a List with different data types, just enter the value of the object list containing items that have different data types.

Let's code...

```
fun main() {

    val numberList = mutableListOf(false, "One", 2, 3.5)
    println(numberList)

    numberList.remove(false)
    numberList.set(0, 1)
    numberList[2] = 3
    numberList.add(3, 4)

    println(numberList)

}
```

INFINITE
LEARNING

Set

Kotlin set is an unordered collection of items.

Set

INFINITE LEARNING

- Kotlin **Set** is an unordered collection of items and can only store unique values.
- A Kotlin set can be either mutable (**mutableSetOf**) or read-only (**setOf**).
- Kotlin mutable or immutable sets do not allow to have duplicate elements.
- Useful when you want no identical or duplicate data in a collection.

Set: setOf()

INFINITE LEARNING

- For set creation, use the standard library functions **setOf()** for read-only sets.
- The setOf function will automatically discard the same number.
- The order of the sets is not important, if you compare two sets with the same value, they will be considered equal.

Let's code...

```
fun main() {
    val setA = setOf(1, 2, 4, 2, 1, 5)
    val setB = setOf(1, 2, 4, 5)
    println(setA == setB) // true
}
```

Check value exist in Set

INFINITE LEARNING

Check if a value is in the Set by using the **in** keyword.

Let's code...

```
fun main() {
    val setData = setOf(1, 2, 4, 2, 1, 5)
    println(4 in setData) // true
}
```

Union and Intersect

INFINITE LEARNING

Collection Set supports union and intersect functions to find out the union and intersection of 2 (two) sets. And this also applies to List.

Let's code...

```
fun main() {
    val list1 = listOf(1, 1, 2 ,3, 5, 8, -1)
    val list2 = setOf(1, 1, 2, 2 ,3, 5)
    val list3 = mutableSetOf(6, 7)

    val intersect = list1 intersect list2
    val union = list1 union list2 union list3

    println(intersect) // Will display the same data
    println(union) // Merge data and eliminate duplicates
}
```

Set: mutableSetOf()

INFINITE LEARNING

mutableSetOf can only add and remove items, and can't change values as in List.

Let's code...

```
fun main() {
    val setItems = mutableSetOf(1, 2, 3, 5, 1, 4)
    // setItems[2] = 7 // can't change immutable set
    setItems.add(6) // add items at the end of the set
    setItems.remove(2) //remove items that have a value of 2

    println(setItems)
}
```

INFINITE
LEARNING

Map

Map

INFINITE LEARNING

- **Map** is a collection that can store data in **key-value** format.
- Where each key is **unique**, and it can only be associated with **one value**.
- The **same** value can be associated with **multiple** keys though.
- Can declare the keys and values to be **any type**; there are no **restrictions**.

Map

INFINITE LEARNING

- A Kotlin map can be either mutable (**mutableMapOf**) or read-only (**mapOf**).

Let's code...

```
fun main() {
    val groupMap = mapOf(
        1 to "Group 1",
        2 to "Group 2",
        3 to "Group 3",
        4 to "Group 4",
        5 to "Group 5",
        6 to "Group 6",
        7 to "Group 3",
    )
    println(groupMap)
    println(groupMap[3])
    println(groupMap.getValue(3))
    println(groupMap.values)
    println(groupMap.keys)
}
```

.toMutableMap()

INFINITE LEARNING

To add a **key-value** to a map, make sure that the map used is **mutable**.

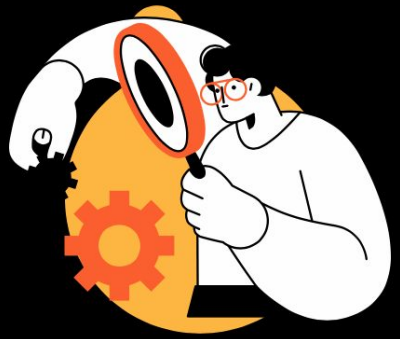
Let's code...

```
fun main() {
    val groupMap = mapOf(
        1 to "Group 1",
        2 to "Group 2",
        3 to "Group 3",
        4 to "Group 4",
        5 to "Group 5",
        6 to "Group 6",
        7 to "Group 3",
    )
    println(groupMap)

    val mutableGroupMap = groupMap.toMutableMap()
    println(mutableGroupMap)

    mutableGroupMap[1] = "Group Satu"
    mutableGroupMap.put(8, "Group 8")

    println(mutableGroupMap)
}
```

INFINITE
LEARNING

Collection Functions

Collection has several operating functions that can be used to access the data in it.



filter() dan filterNot()

INFINITE LEARNING

- The **filter()** and **filterNot()** functions will generate a new list of selections based on the conditions we provide.
- Used to filter or **filter data** in a collection.

Let's code...

```
fun main() {
    val numberList = listOf(1, 2, 3, 4, 5)

    val eventList = numberList.filter { it % 2 == 0 } // [2, 4]

    val notEventList = numberList.filterNot { it % 2 == 0 } // [1, 3, 5]
}
```

map()

INFINITE LEARNING

The **map()** function will create a **new collection** according to the **changes** that will be made from the **previous** collection.

Let's code...

```
fun main() {
    val numberList = listOf(1, 2, 3, 4, 5)

    val multipliedBy5 = numberList.map { it * 5 } // [5, 10, 15, 20, 25]
    print(multipliedBy5)
}
```

count()

INFINITE LEARNING

The **count()** function can be used to count the **number of items** in a collection.

Let's code...

```
fun main() {
    val days = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

    val totalDays = days.count()
    print(totalDays) // 7
}
```

find(), firstOrNull(), and lastOrNull()

INFINITE LEARNING

- The **find()** function is used to **find** items in a collection
- The **firstOrNull()** function is used to find the **first** item that matches the specified condition.
- The **lastOrNull()** function is used to find the **last** item that matches the specified condition.

Let's code...

```
val numberList = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
val oddNumber = numberList.find { it % 2 == 1 }
```

find(), firstOrNull(), and lastOrNull()

INFINITE LEARNING

The **find()**, **firstOrNull()**, and **lastOrNull()** functions work the same way as if **no** matching data is found in the collection, the function will return a **null** value.

Let's code...

```
fun main() {
    val numberList = listOf(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
    val oddNumber = numberList.find { it % 2 == 1 }
    val firstOrNullNumber = numberList.firstOrNull { it % 2 == 3 }
    val lastOrNullNumber = numberList.lastOrNull { it % 2 == 3 }

    println(oddNumber)
    println(firstOrNullNumber)
    println(lastOrNullNumber)

}
```

first() and last()

INFINITE LEARNING

The **first()** and **last()** functions can be used to **filter** the **first** or **last** item from a collection.

Let's code...

```
fun main() {
    val days = arrayOf("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday")

    val firstData = days.first()
    val latestData = days.last()

    println(firstData) // Monday
    println(latestData) // Sunday
}
```

sum()

INFINITE LEARNING

- The **sum()** function is used to **add up** every data in the collection.
- The **special** sum() function can **only** be used for collections of type **number**.

Let's code...

```
fun main() {
    val numberList = listOf(1, 3, 2, 6, 5, 4)

    val total = numberList.sum()
    print(total) // 21
}
```

sorted()

INFINITE LEARNING

The **sorted()** is used to **sort** items in a collection in **ascending** order

Let's code...

```
fun main() {
    val numberList = listOf(1, 3, 2, 6, 5, 4)
    val hackerChar = listOf('h', 'a', 'c', 'k', 'e', 'r')

    val ascendingSort1 = numberList.sorted()
    val ascendingSort2 = hackerChar.sorted()

    println(ascendingSort1) // [1, 2, 3, 4, 5, 6]
    println(ascendingSort2) // [a, c, e, h, k, r]
}
```

sortedDescending()

INFINITE LEARNING

The **sortedDescending()** is used to **sort** the items in a collection in **descending** order

Let's code...

```
fun main() {
    val numberList = listOf(1, 3, 2, 6, 5, 4)
    val hackerChar = listOf('h', 'a', 'c', 'k', 'e', 'r')

    val ascendingSort1 = numberList.sortedDescending()
    val ascendingSort2 = hackerChar.sortedDescending()

    println(ascendingSort1) // [6, 5, 4, 3, 2, 1]
    println(ascendingSort2) // [r, k, h, e, c, a]
}
```

INFINITE
LEARNING

Any question?

GUIDING RESOURCE

INFINITE LEARNING

Guiding Resources:

1. https://kotlinlang.org/docs/collections-overview.html
2. https://www.tutorialspoint.com/kotlin/kotlin_collections.htm
3. https://www.tutorialspoint.com/kotlin/kotlin_lists.htm
4. https://www.tutorialspoint.com/kotlin/kotlin_sets.htm
5. https://www.tutorialspoint.com/kotlin/kotlin_maps.htm

Design Asset:

https://storyset.com

INFINITE
LEARNING

Cheers

@infinitelearning_id
Infinite Learning Indonesia
infinitelearning_id