codecademy

Collections

Immutable Lists

An immutable list represents a group of elements with read-only operations.

It can be declared with the term `listOf` , followed by a pair of parentheses containing elements that are separated by commas.

```
var programmingLanguages = listOf("C#", "Java", "Kotlin", "Ruby")
```

Mutable Lists

A mutable list represents a collection of ordered elements that possess read and write functionalities.

It can be declared with the term, `mutableListOf` followed by a pair of parentheses containing elements that are separated by commas.

```
var fruits = mutableListOf("Orange", "Apple", "Banana", "Mango")
```

Accessing List Elements

In order to retrieve an element from a list, we can reference its numerical position or index using square bracket notation.

Note: Remember that the first element of a list starts at `0` .

```
var cars = listOf("BMW", "Ferrari", "Volvo", "Tesla")

println(cars[2]) // Prints: Volvo
```

The Size Property

The `size` property is used to determine the number of elements that exist in a collection.

```
var worldContinents = listOf("Asia", "Africa", "North America", "South America", "Antarctica", "Europe", "Australia")

println(worldContinents.size) // Prints: 7
```

List Operations

The list collection supports various operations in the form of built-in functions that can be performed on its elements.
Some functions perform read and write operations, whereas others perform read-only operations.
The functions that perform read and write operations can only be used on mutable lists while read-only operations can be performed on both mutable and immutable lists.

```
var seas = listOf("Black Sea", "Caribbean
Sea", "North Sea")
println(seas.contains("North Sea")) // Pri
nts: true

// The contains() function performs a read
 operation on any list and determines if a
n element exists.

seas.add("Baltic Sea") // Error: Can't per
form write operation on immutable list

// The add() function can only be called o
n a mutable list thus the code above throw
s an error.
```

Immutable Sets

An immutable set represents a collection of unique elements in an unordered fashion whose contents cannot be altered throughout a program.

It is declared with the term, `setOf`, followed by a pair of parentheses holding unique values.

```
var primaryColors = setOf("Red", "Blue", "
Yellow")
```

Mutable Sets

A mutable set represents a collection of ordered elements that possess both read and write functionalities.

It is declared with the term, `mutableSetOf`, followed by a pair of parentheses holding unique values.

```
var womenInTech = mutableSetOf("Ada Lovela
ce", "Grace Hopper", "Radia Perlman", "
Sister Mary Kenneth Keller")
```

Accessing Set Elements

Elements in a set can be accessed using the `elementAt()` or `elementAtOrNull()` functions.

The `elementAt()` function gets appended onto a set name and returns the element at the specified position within the parentheses.

The `elementAtOrNull()` function is a safer variation of the `elementAt()` function and returns `null` if the position is out of bounds as opposed to throwing an error.

```
var companies = setOf("Facebook", "Apple",
 "Netflix", "Google")

println(companies.elementAt(3)) // Prints:
 Google

println(companies.elementAt(4)) // Returns
 and Error

println(companies.elementAtOrNull(4)) // P
rints: null
```

Immutable Maps

An immutable Map represents a collection of entries that cannot be altered throughout a program.

It is declared with the term, `mapOf`, followed by a pair of parentheses. Within the parentheses, each key should be linked to its corresponding value with the `to` keyword, and each entry should be separated by a comma.

```
var averageTemp = mapOf("winter" to 35,  "
spring" to 60,  "summer" to 85, "fall" to
55)
```

Mutable Maps

A mutable map represents a collection of entries that possess read and write functionalities. Entries can be added, removed, or updated in a mutable map.
A mutable map can be declared with the term, `mutableMapOf`, followed by a pair of parentheses holding key-value pairs.

```
var europeanDomains = mutableMapOf("German
y" to "de", "Slovakia" to "sk", "Hungary"
to "hu", "Norway" to "no")
```

Retrieving Map Keys and Values

Keys and values within a map can be retrieved using the `.keys` and `.values` properties.
The `.keys` property returns a list of key elements, whereas the `.values` property returns a list of value elements.
To retrieve a single value associated with a key, the shorthand, `[key]`, syntax can be used.

```
var oscarWinners = mutableMapOf("Parasite"
 to "Bong Joon-
ho", "Green Book" to "Jim Burke", "The Sha
pe Of Water" to "Guillermo del Toro")

println(oscarWinners.keys)
// Prints: [Parasite, Green Book, The Shap
e Of Water]

println(oscarWinners.values)
// Prints: [Bong Joon-
ho, Jim Burke, Guillermo del Toro]

println(oscarWinners["Parasite"])
// Prints: Bong Joon-ho
```

Adding and Removing Map Entries

An entry can be added to a mutable map using the `put()` function. Oppositely, an entry can be removed from a mutable map using the `remove()` function.

The `put()` function accepts a key and a value separated by a comma.

The `remove()` function accepts a key and removes the entry associated with that key.

```
var worldCapitals = mutableMapOf("United S
tates" to "Washington D.C.", "Germany" to
"Berlin", "Mexico" to "Mexico City", "Fran
ce" to "Paris")

worldCapitals.put("Brazil", "Brasilia")
println(worldCapitals)
// Prints: {United States=Washington D.C.,
 Germany=Berlin, Mexico=Mexico City, Franc
e=Paris, Brazil=Brasilia}

worldCapitals.remove("Germany")
println(worldCapitals)
// Prints: {United States=Washington D.C.,
 Mexico=Mexico City, France=Paris, Brazil=
Brasilia}
```