Chapter 11: Dictionaries

Section 11.1: Declaring Dictionaries

Dictionaries are an unordered collection of keys and values. Values relate to unique keys and must be of the same type.

When initializing a Dictionary the full syntax is as follows:

```
var books : Dictionary<Int, String> = Dictionary<Int, String>()
```

Although a more concise way of initializing:

```
var books = [Int: String]()
// or
var books: [Int: String] = [:]
```

Declare a dictionary with keys and values by specifying them in a comma separated list. The types can be inferred from the types of keys and values.

```
var books: [Int: String] = [1: "Book 1", 2: "Book 2"]
//books = [2: "Book 2", 1: "Book 1"]
var otherBooks = [3: "Book 3", 4: "Book 4"]
//otherBooks = [3: "Book 3", 4: "Book 4"]
```

Section 11.2: Accessing Values

A value in a `Dictionary` can be accessed using its key:

```
var books: [Int: String] = [1: "Book 1", 2: "Book 2"]
let bookName = books[1]
//bookName = "Book 1"
```

The values of a dictionary can be iterated through using the `values` property:

```
for book in books.values {
    print("Book Title: \(book)")
}
//output: Book Title: Book 2
//output: Book Title: Book 1
```

Similarly, the keys of a dictionary can be iterated through using its `keys` property:

```
for bookNumbers in books.keys {
    print("Book number: \(bookNumber)")
}
// outputs:
// Book number: 1
// Book number: 2
```

To get all `key` and `value` pair corresponding to each other (you will not get in proper order since it is a Dictionary)

```
for (book,bookNumbers)in books{
print("\(book)  \(bookNumbers)")
}
```

```
// outputs:
// 2  Book 2
// 1  Book 1
```

Note that a `Dictionary`, unlike an `Array`, in inherently unordered-that is, there is no guarantee on the order during iteration.

If you want to access multiple levels of a Dictionary use a repeated subscript syntax.

```
// Create a multilevel dictionary.
var myDictionary: [String:[Int:String]]! =
["Toys":[1:"Car",2:"Truck"],"Interests":[1:"Science",2:"Math"]]

print(myDictionary["Toys"][2]) // Outputs "Truck"
print(myDictionary["Interests"][1]) // Outputs "Science"
```

Section 11.3: Change Value of Dictionary using Key

```
var dict = ["name": "John", "surname": "Doe"]
// Set the element with key: 'name' to 'Jane'
dict["name"] = "Jane"
print(dict)
```

Section 11.4: Get all keys in Dictionary

```
let myAllKeys = ["name" : "Kirit" , "surname" : "Modi"]
let allKeys = Array(myAllKeys.keys)
print(allKeys)
```

Section 11.5: Modifying Dictionaries

Add a key and value to a Dictionary

```
var books = [Int: String]()
//books = [:]
books[5] = "Book 5"
//books = [5: "Book 5"]
books.updateValue("Book 6", forKey: 5)
//[5: "Book 6"]
```

updateValue returns the original value if one exists or nil.

```
let previousValue = books.updateValue("Book 7", forKey: 5)
//books = [5: "Book 7"]
//previousValue = "Book 6"
```

Remove value and their keys with similar syntax

```
books[5] = nil
//books [:]
books[6] = "Deleting from Dictionaries"
//books = [6: "Deleting from Dictionaries"]
let removedBook = books.removeValueForKey(6)
//books = [:]
//removedValue = "Deleting from Dictionaries"
```

Section 11.6: Merge two dictionaries

```
extension Dictionary {
    func merge(dict: Dictionary<Key,Value>) -> Dictionary<Key,Value> {
        var mutableCopy = self
        for (key, value) in dict {
            // If both dictionaries have a value for same key, the value of the other dictionary is
used.
            mutableCopy[key] = value
        }
        return mutableCopy
    }
}
```