Subscripts

Classes, structures, and enumerations can define *subscripts*, which are shortcuts for accessing the member elements of a collection, list, or sequence. You use subscripts to set and retrieve values by index without needing separate methods for setting and retrieval. For example, you access elements in an `Array` instance as `someArray[index]` and elements in a `Dictionary` instance as `someDictionary[key]`.

Subscript Syntax

```
subscript(index: Int) -> Int {
    get {
        // Return an appropriate subscript value
here.
    }
    set(newValue) {
        // Perform a suitable setting action
here.
    }
}
```

The type of `newValue` is the same as the return value of the subscript. As with computed properties, you can choose not to specify the setter's `(newValue)` parameter. A default parameter called `newValue` is provided to your setter if you don't provide one yourself.

As with read-only computed properties, you can simplify the declaration of a read-only subscript by removing the `get` keyword and its braces:

```
subscript(index: Int) -> Int {
    // Return an appropriate subscript value
here.
}
```

Here's an example of a read-only subscript

implementation, which defines a `TimesTable` structure to represent an *n*-times-table of integers:

```
struct TimesTable {
    let multiplier: Int
    subscript(index: Int) -> Int {
        return multiplier * index
    }
}
let threeTimesTable = TimesTable(multiplier: 3)
print("six times three is \(threeTimesTable[6])")
// Prints "six times three is 18"
```

For example, Swift's `Dictionary` type implements a subscript to set and retrieve the values stored in a `Dictionary` instance. You can set a value in a dictionary by providing a key of the dictionary's key type within subscript brackets, and assigning a value of the dictionary's value type to the subscript:

```
var numberOfLegs = ["spider": 8, "ant": 6, "cat": 4]
numberOfLegs["bird"] = 2
```

The example above defines a variable called `numberOfLegs` and initializes it with a dictionary literal containing three key-value pairs. The type of the `numberOfLegs` dictionary is inferred to be `[String: Int]`. After creating the dictionary, this example uses subscript assignment to add a `String` key of `"bird"` and an `Int` value of `2` to the dictionary.

Subscripts are typically used as a shortcut for accessing the member elements in a collection, list, or sequence.

Subscripts can take any number of input parameters, and these input parameters can be of any type.

Subscripts can also return a value of any type.