Collections

Sun*

Outline

1. Collection types
2. Difference among Array, Set and Dictionary

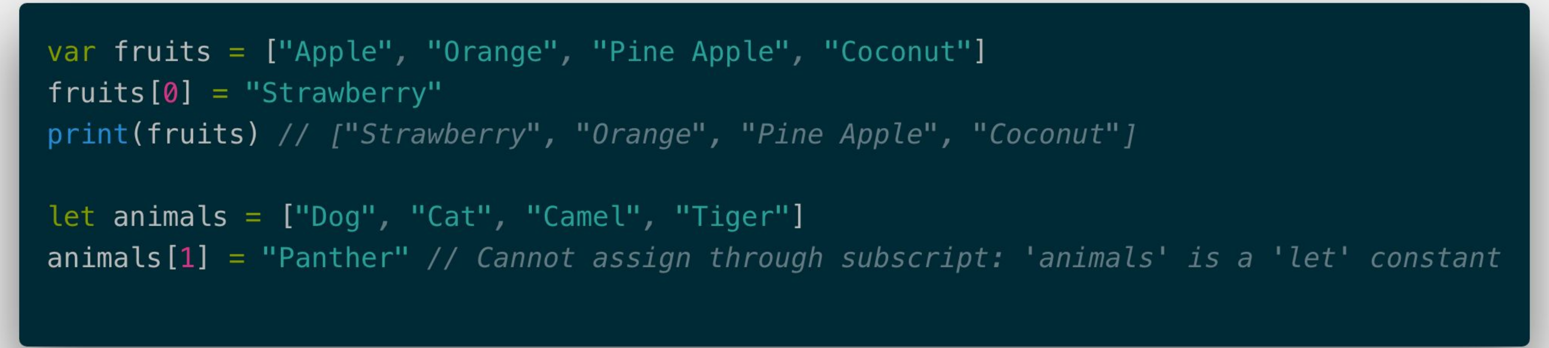
1. Collection types

1. Mutability of Collections
2. Array
3. Set
4. Dictionary

Sun*

1.1 Mutability of Collections

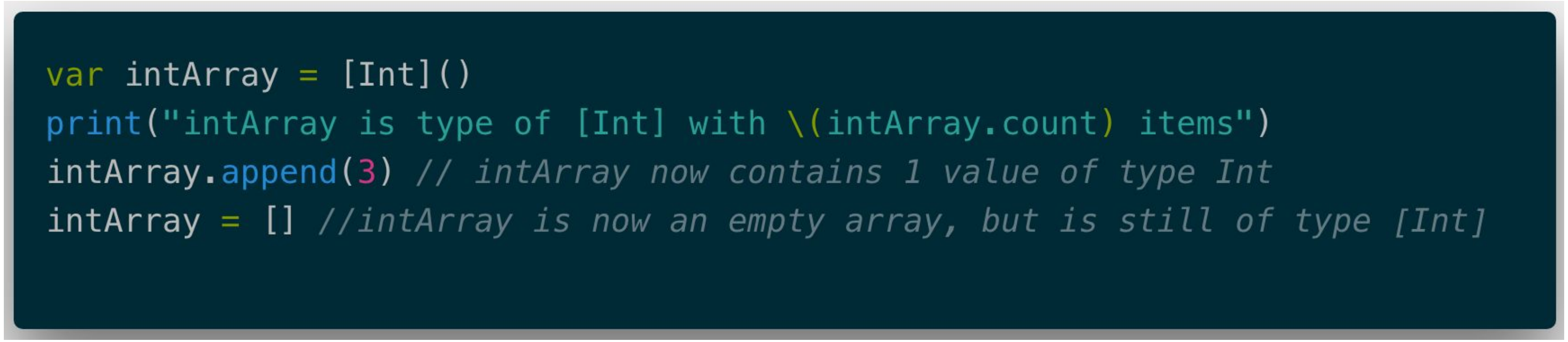
- If you create an array, a set or a dictionary as a variable, it will be *mutable.*
- If you assign an array, a set or a dictionary to a constant, that collection is *immutable* - its size and contents cannot be changed.



```swift
var fruits = ["Apple", "Orange", "Pine Apple", "Coconut"]
fruits[0] = "Strawberry"
print(fruits) // ["Strawberry", "Orange", "Pine Apple", "Coconut"]

let animals = ["Dog", "Cat", "Camel", "Tiger"]
animals[1] = "Panther" // Cannot assign through subscript: 'animals' is a 'let' constant
```

1.2 Arrays

- An *array* stores values of the same type in an ordered list. The same value can appear in an array multiple times at different positions.
- Create an empty array:

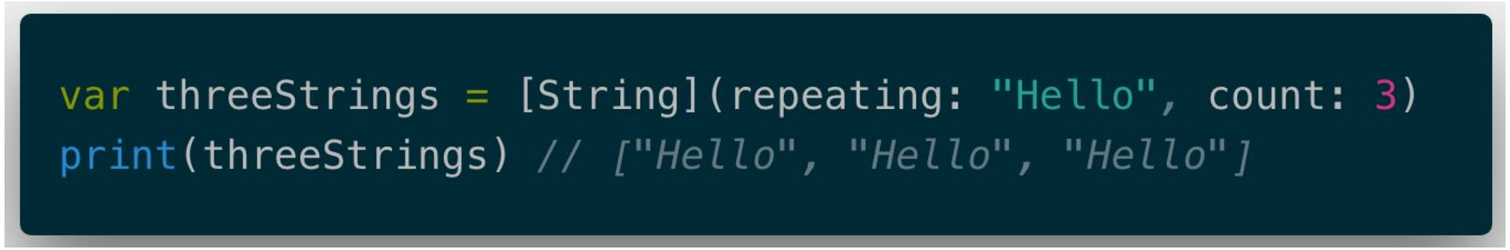


```swift
var intArray = [Int]()
print("intArray is type of [Int] with \(intArray.count) items")
intArray.append(3) // intArray now contains 1 value of type Int
intArray = [] //intArray is now an empty array, but is still of type [Int]
```

Sun*

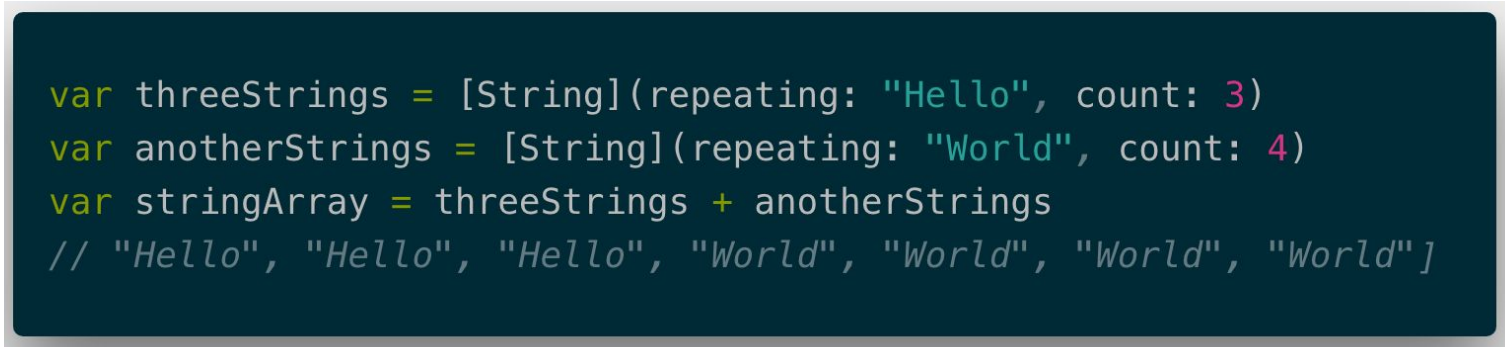
1.2 Arrays (cont)

❖ Create an array with a default value:



```
var threeStrings = [String](repeating: "Hello", count: 3)
print(threeStrings) // ["Hello", "Hello", "Hello"]
```

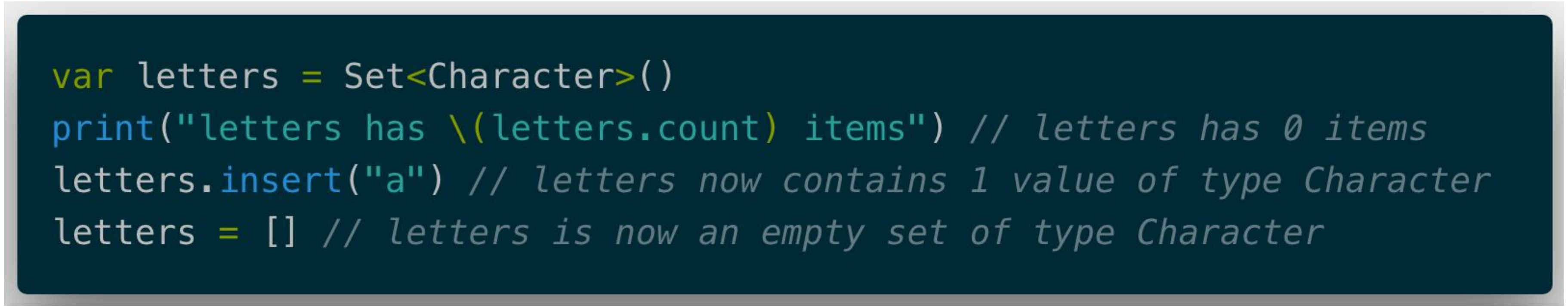
❖ Create an array by adding two arrays together:



```
var threeStrings = [String](repeating: "Hello", count: 3)
var anotherStrings = [String](repeating: "World", count: 4)
var stringArray = threeStrings + anotherStrings
// "Hello", "Hello", "Hello", "World", "World", "World", "World"]
```

1.3 Sets

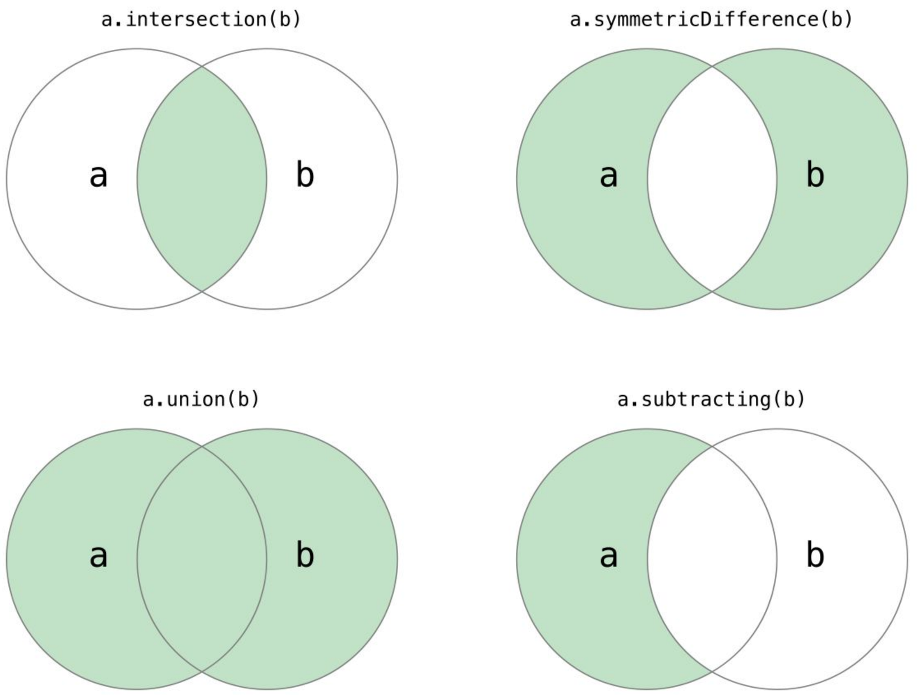
- A set stores distinct values of the same type in a collection with no defined ordering.
- A type must be *hashable* in order to be stored in a set.



```
var letters = Set<Character>()
print("letters has \(letters.count) items") // letters has 0 items
letters.insert("a") // letters now contains 1 value of type Character
letters = [] // letters is now an empty set of type Character
```

1.3 Sets (cont)

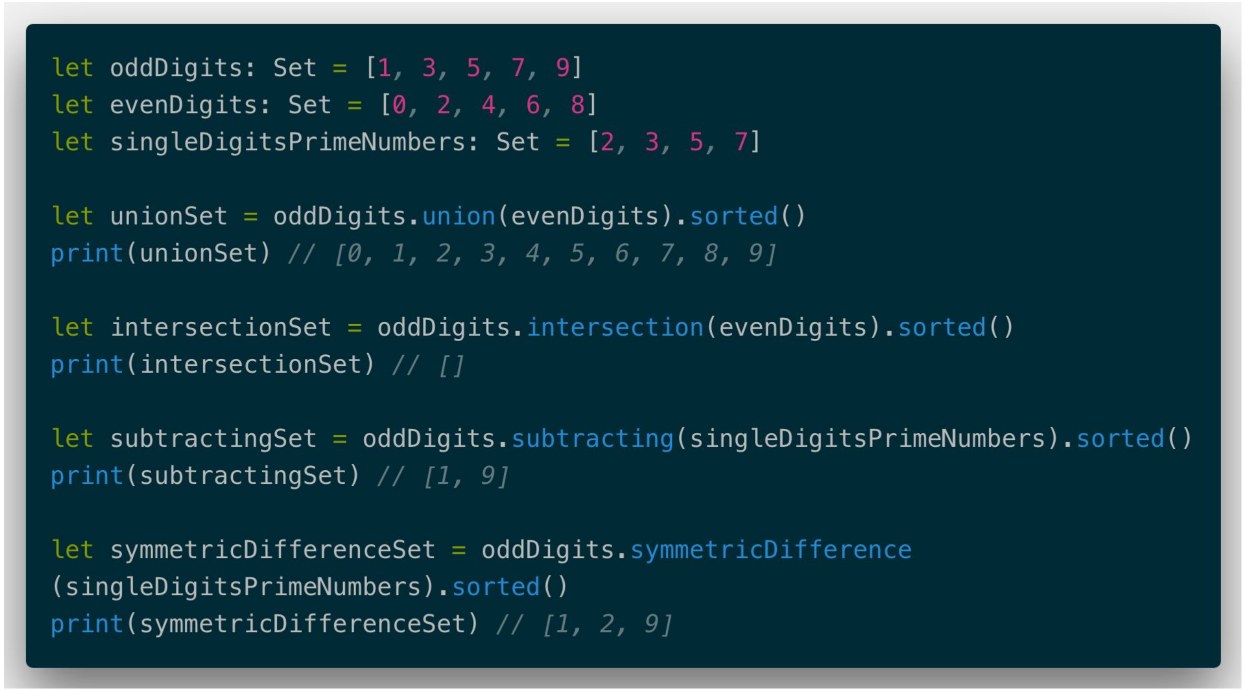
Set operations:



1.3 Sets (cont)

Set operations:

- Union
- intersection
- substracting
- symmetricDifference

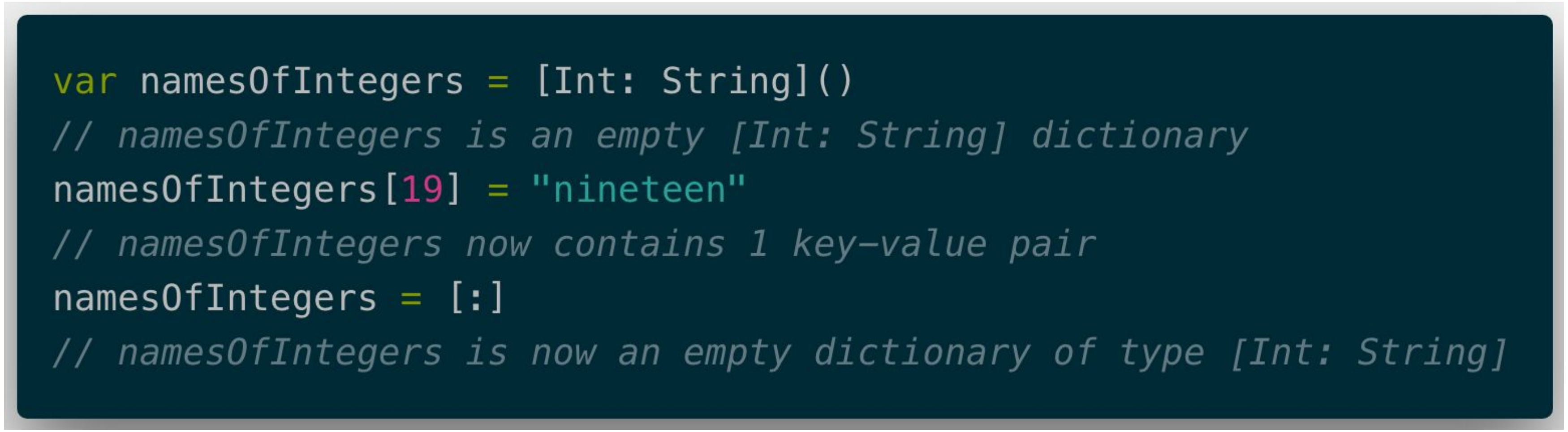


```swift
let oddDigits: Set = [1, 3, 5, 7, 9]
let evenDigits: Set = [0, 2, 4, 6, 8]
let singleDigitsPrimeNumbers: Set = [2, 3, 5, 7]

let unionSet = oddDigits.union(evenDigits).sorted()
print(unionSet) // [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

let intersectionSet = oddDigits.intersection(evenDigits).sorted()
print(intersectionSet) // []

let subtractingSet = oddDigits.subtracting(singleDigitsPrimeNumbers).sorted()
print(subtractingSet) // [1, 9]

let symmetricDifferenceSet = oddDigits.symmetricDifference
(singleDigitsPrimeNumbers).sorted()
print(symmetricDifferenceSet) // [1, 2, 9]
```

1.4 Dictionaries

❖ A *dictionary* stores associations between keys of the same type and vales of the same type in a collection with no defined ordering.

❖ Each value is associated with a unique key.



```swift
var namesOfIntegers = [Int: String]()
// namesOfIntegers is an empty [Int: String] dictionary
namesOfIntegers[19] = "nineteen"
// namesOfIntegers now contains 1 key-value pair
namesOfIntegers = [:]
// namesOfIntegers is now an empty dictionary of type [Int: String]
```

Sun*

2. Difference among Array, Set and Dictionary

	Array	Set	Dictionary
Type of collection	Ordered collection	Unordered collection	Unordered collection with key-value pair
Can contain duplicates?	Yes	No	No
Types of elements	Any kind of elements	Hashable types or custom types that conforms to Hashable protocol	- Key must be Hashable types, Value can be any kind of types
Elements in collection	Same type, ordered as a sequence	Same type, unordered	Dictionaries store associations between keys of the same type with values of the same type.
When to use	when ordering is needed or to store non hashable types	when order is not important, and we need to ensure that each element appears only one time in collection	when we need quick access to elements in a collection using its identifier

Question & Answer?

Sun*