

The Swift Programming Language

Part 3 – Array & Tuple

Array

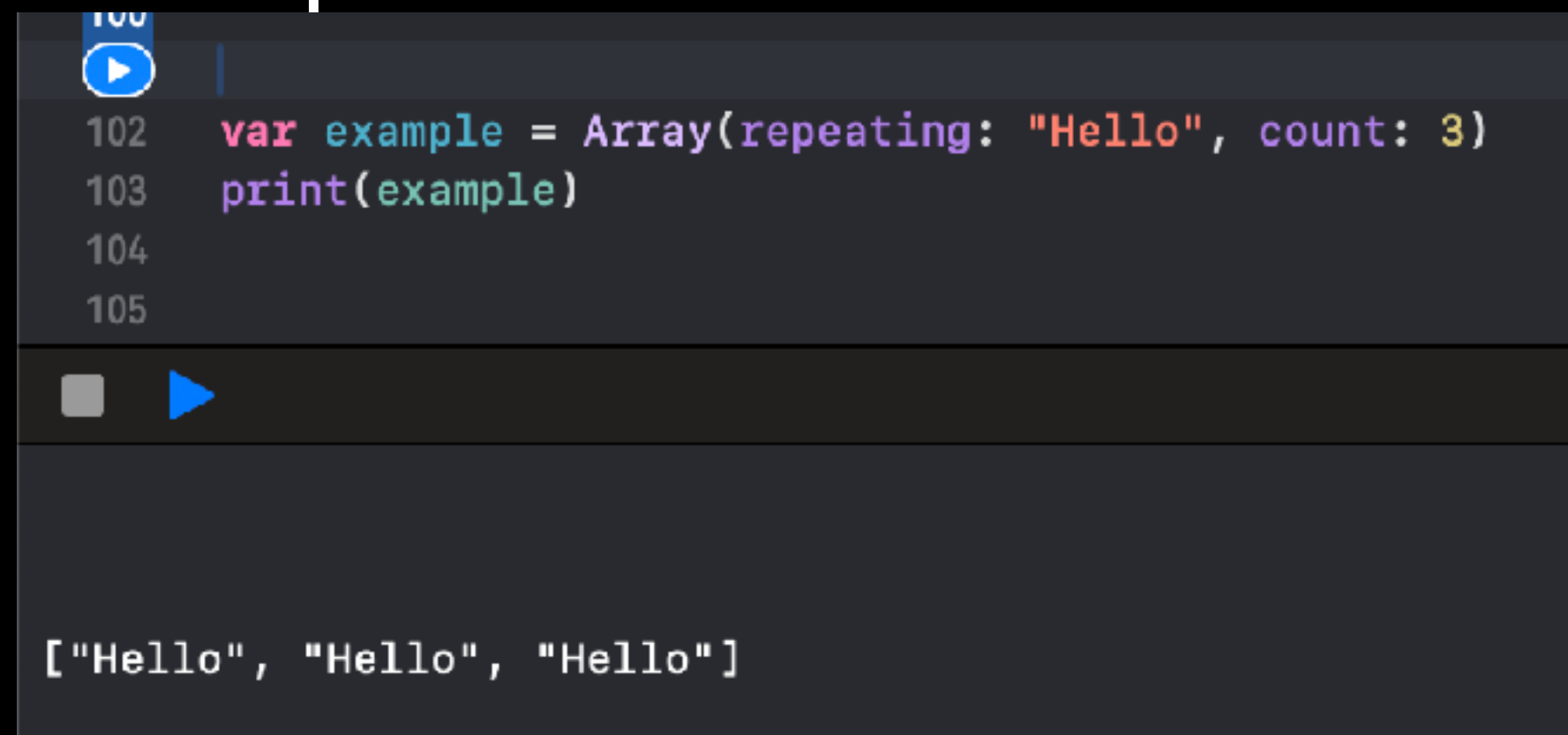
What is Array?

- Arrays function to collect values and store them neatly so that they are easy to access and modify
- Array needs to be same data type
- Declaration `var a = [data type]()`
- Declaration with type annotation `var a: [data type] = []` (safer)
- Array are modifiable

Array

Make repeating array

- Use `Array(...)` to repeat something in between
- Example:



The screenshot shows a Swift Playground interface. The code editor contains the following Swift code:

```
102 var example = Array(repeating: "Hello", count: 3)
103 print(example)
104
105
```

Below the code editor, there is a console area with a play button icon. The console output displays the result of the code execution:

```
["Hello", "Hello", "Hello"]
```

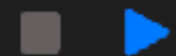
- Works perfectly on repeating `LazyVGrid` or `LazyHGrid` in SwiftUI

Array

Accessing array

- Use **index** to accessing element of array
- Example:

```
16 // Usage example:
17 let umur2 = 20
18 checkAge(umur: umur2)
19 print("\n\n\n\n\n\n\n\n\n")
20
21 var namaHari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat"]
22 let hariPertama = namaHari[0]
23 print(hariPertama)
```



Senin

Array

Add value to array, Part 1

- Use **append** to add element end of array
- Example:

```
22  
23  var namaHari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"]  
24  namaHari.append("Minggu")  
25  print(namaHari)
```



```
["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
```

Array

Add value to array, Part 2

- Use **insert** to add between element in array
- Example:

```
23  var namaHari = ["Senin", "Selasa", "Kamis", "Jumat", "Sabtu", "Minggu"]
24  namaHari.insert("Rabu", at: 2)
25  print(namaHari)
26
```



```
["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu", "Minggu"]
```

Array

Count elements in array



- Use **count** to count elements in array
- Will return as Integer

Array

Loop through elements in array, Part 1

- Use **for** loop to iterate
- Example:

```
23  var namaHari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu",  
    "Minggu"]  
24  for hari in namaHari {  
25      print("Hari", hari)  
26  }  
27
```

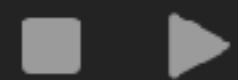

Hari Senin
Hari Selasa
Hari Rabu
Hari Kamis
Hari Jumat
Hari Sabtu
Hari Minggu

Array

Loop through elements in array, Part 2

- Use `.enumerated()` and `tuple` to loop as index
- Example:

```
23  var namaHari = ["Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu",  
    "Minggu"]  
24  for (indeks, hari) in namaHari.enumerated() {  
25      print("Hari ke-\(indeks + 1): \(hari)")  
26  }  
27
```



```
Hari ke-1: Senin  
Hari ke-2: Selasa  
Hari ke-3: Rabu  
Hari ke-4: Kamis  
Hari ke-5: Jumat  
Hari ke-6: Sabtu  
Hari ke-7: Minggu
```

Summary

By using arrays in Swift, we can easily store, access, and manipulate sets of values in a single data structure.

Tuple

What is tuple?

- Data structure used to store a set of data
- Tuple serves to group multiple values into one compound value
- Values in a tuple can consist of different data types and do not have to be the same
- Tuple are NOT modifiable

Tuple

Example of Code #1 – Simple Tuple Assignment, Part 1

```
104 let http404Error = (404, "Not Found")
105 let (statusCode, statusMessage) = http404Error
106 print("http404Error memiliki kode status \"(statusCode)\")
107 print("http404Error memiliki pesan error \"\n(statusMessage)\n")
```



```
http404Error memiliki kode status 404
http404Error memiliki pesan error "Not Found"
```

You can use different data type to a single tuple.

Example given is Int and String in a single tuple

Tuple

Example of Code #2 – Simple Tuple Assignment, Part 2

```
103
104  let http404Error = (404, "Not Found")
105  let (justTheStatusCode, _) = http404Error
106  print("http404Error memiliki kode status \ (justTheStatusCode)")
```

Use `_` in tuple to deactivate tuple variable

```
http404Error memiliki kode status 404
```

Tuple

Example of Code #3 – Use Indexing in Tuple

```
104 let http404Error = (404, "Not Found")
105 print("http404Error memiliki kode status \%(http404Error.0)s")
106 print("http404Error memiliki pesan error \"\%(http404Error.1)s\"")
```



```
http404Error memiliki kode status 404
http404Error memiliki pesan error "Not Found"
```

You can use by using `.` followed by index start from 0 (just same as array)

Tuple

Example of Code #4 – Declare Variable in Tuple

```
21 // Deklarasi terlebih dahulu
22 let http200Status = (statusCode: 200, description: "OK")
23
24 // Cetak apa yang udah dideklarasikan
25 print("http200Status memiliki kode status \(http200Status.statusCode)")
26 print("http200Status memiliki pesan error "\(http200Status.description)\")")
27
```



```
http200Status memiliki kode status 200
http200Status memiliki pesan error "OK"
```

Directly declare variable
inside a tuple

Tuple

Example of Code #5 – A Returning Tuple in Function

```
23 func hitungStatistik(_ data: [Int]) -> (rata: Double, total: Int) {  
24     let total = data.reduce(0, +)  
25     let rata = Double(total) / Double(data.count)  
26     return (rata, total)  
27 }  
28  
29 let data = [10, 20, 30, 40, 50]  
30 let statistik = hitungStatistik(data)  
31 print("Rata-rata: \(statistik.rata), Total: \(statistik.total)")  
32
```



Rata-rata: 30.0, Total: 150

Tuple vs Array

What's the difference?

- **Tuples** have a fixed size and cannot be changed after being declared, while **arrays** have a changeable size.
- **Tuples** are useful for storing small amounts of related values, while **arrays** are suitable for storing larger collections of values.

Summary

Tuple is a powerful feature in Swift that allows you to group multiple values into one. They can be used in a variety of situations, such as returning the value of a function, temporary use of limited data, and more.

Thanks For Your Attendance Today!

iCodeWave Community