**IOS**

# Section 1: Swift

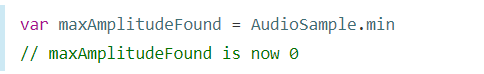
## 1. The Basics

### 1.1 Type Aliases

Type aliases define an alternative name for an existing type. You define type aliases with the typealias keyword



Lúc này ta có thể sử dụng AudioSample giống như UInt16

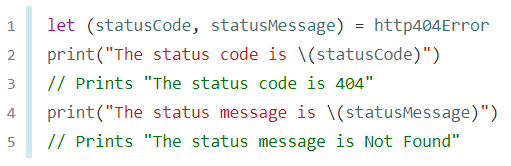


### 1.2 Tuples

Tuples group multiple values into a single compound value. The values within a tuple can be of any type and don’t have to be of the same type as each other.

VD:

Khi khai báo như trên, bên dưới ta có thể phân tách(decompose) nội dung thành các hằng số hoặc các biến để có thể truy cập:



Ta thấy http404Error được tupple thành (404, “Not Found”) sau đó lại được phân ra thành (statuscode, statusMessage). Ta cũng có thể đặt tên cho các thành phần phân ra luôn khi mới couple



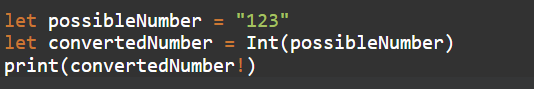
Kết quả in ra: 200

Từ đây ta cũng phát hiện ra để return nhiều giá trị, ta có thể return 1 Tupples

### 1.3 Optionals

Chúng ta sử dụng Optionals khi giá của variable có thể là nil. Một optional đại diện cho 2 khả năng: Nếu có value, ta có thể unwrap optional đó để truy cấp value, TH2 là không có value.

Ta có đoạn code:



Khi ta khai báo như VD trên, thì trình thông dịch sẽ gán ngầm định như sau:



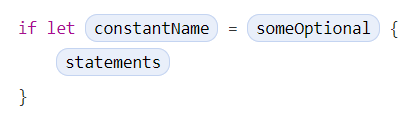
Lúc này biến convertedNumber sẽ là 1 optionals. Lý do là gì?

Lý do lúc này vì biến possibleNumber là 1 chuỗi String, nếu nó là “123” thì việc convert sang int sẽ hoàn tất và gán thành công, nhưng nếu possibleNumber là “aaa” thì việc convert sẽ failed và nó sẽ trả lại giá trị nil. Mà 1 biến thông thường không thể được gán bằng nil, nên lúc này biến convertedNumber sẽ là 1 optional.

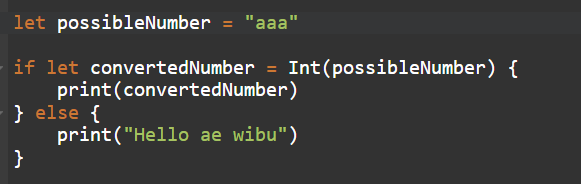
* Optional binding:

Ta có thể sử dụng Optional binding để kiểm tra xem optional có đang chứa value hay không. Optional binding có thể được sử dụng với if và while.

Chúng ta viết 1 optional binding với if như dưới



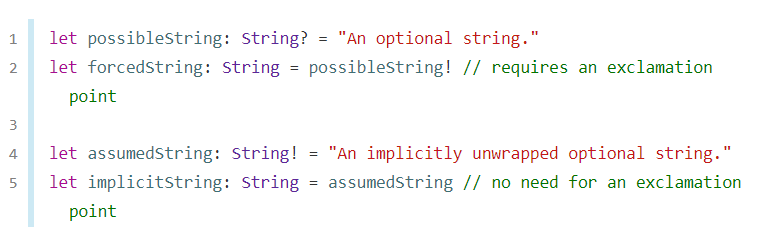
Ta có thể hiểu rằng, 1 biến thông thường không thể được gán trị nil, nên nếu someOptional lúc này có giá trị thì constanName sẽ được gán và chạy các lệnh statements, nhưng nếu someOptional là nil thì sẽ không được gán và sẽ không chạy statements



Kết quả in ra màn hình là “Hello ae wibu” và chương trình không bị crashed.

* Implicitly Unwrapped Optionals

Ta là 1 lập trình viên, và ta biết rõ hơn máy, trong trường hợp này, nó chắc chắn sẽ có value và không thể là nil nên đôi khi ta không cần check và unwrap optional. Lúc này ta sử dụng dấu ! sử dụng Implicit Unwrapp

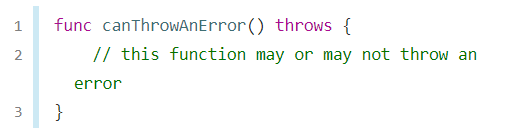


Ta thấy có 2 cách khác nhau, nhưng bên trên ta bắt buộc phải unwrap, bên dưới thì không. Ta hiểu đơn giản là “an implicitly unwrapped optional as giving permission for the optional to be force-unwrapped if needed”.

### 1.4 Error Handling

Chúng ta sử dụng Error handling để phản hồi(respond) mỗi khi chương trình gặp lỗi trong lúc thực thi.  Error handling cho phép ta xác định nguyên nhân cơ bản của lỗi và, nếu cần, ta sẽ thông báo lỗi sang một phần khác trong chương trình của mình.

“When a function encounters an error condition, it throws an error. That function’s caller can then catch the error and respond appropriately: ”



“ When you call a function that can throw an error, you prepend the try keyword to the expression. Swift automatically propagates errors out of their current scope until they’re handled by a catch clause.”



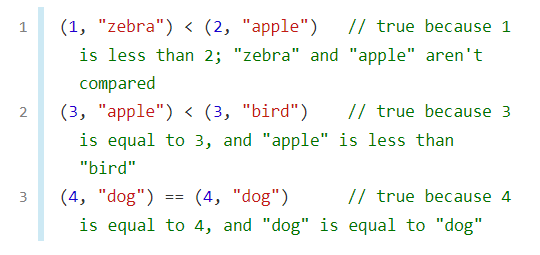
Ta hiểu như sau: “In this example, the makeASandwich() function will throw an error if **no clean dishes are available or if any ingredients(thành phần) are missing**. Because makeASandwich() can throw an error, the function call is wrapped in a try expression. By wrapping the function call in a do statement, any errors that are thrown will be propagated(tuyên truyền, thông báo) to the provided catch clauses.

If no error is thrown, the eatASandwich() function is called. If an error is thrown and it matches the SandwichError.outOfCleanDishes case, then the washDishes() function will be called. If an error is thrown and it matches the SandwichError.missingIngredients case, then the buyGroceries(\_:) function is called with the associated [String] value captured by the catch pattern”

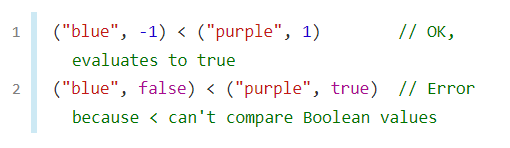
## 2. Basic Opearators

* Comparison Operators

“You can compare two tuples if they have the same type and the same number of values. Tuples are compared from left to right, one value at a time, until the comparison finds two values that aren’t equal. Those two values are compared, and the result of that comparison determines the overall result of the tuple comparison. If all the elements are equal, then the tuples themselves are equal. ”



“In the example above, you can see the left-to-right comparison behavior on the first line. Because 1 is less than 2, (1, "zebra") is considered less than (2, "apple"), regardless of any other values in the tuples. It doesn’t matter that "zebra" isn’t less than "apple", because the comparison is already determined by the tuples’ first elements. However, when the tuples’ first elements are the same, their second elements are compared—this is what happens on the second and third line.”



“ As demonstrated in the code above, you can compare two tuples of type (String, Int) because both String and Int values can be compared using the < operator. In contrast, two tuples of type (String, Bool) can’t be compared with the < operator because the < operator can’t be applied to Bool values.”

* Range Operators

“The closed range operator (a...b) defines a range that runs from a to b, and includes the values a and b. The value of a must not be greater than b.”

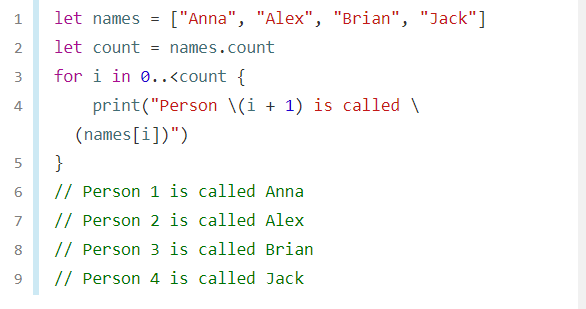
“The closed range operator is useful when iterating(lặp đi lặp lại) over a range in which you want all of the values to be used, such as with a for-in loop:”

# 

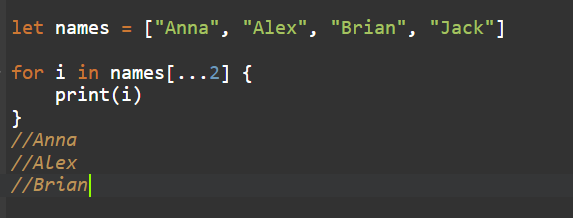
Ta cũng có các khái niệm như Half-Open Range Operator, One-Sided Ranges

* Half-Open range

Giá trị của i sẽ từ 0 đến 3 và sẽ quét hết mảng 4 phần tử

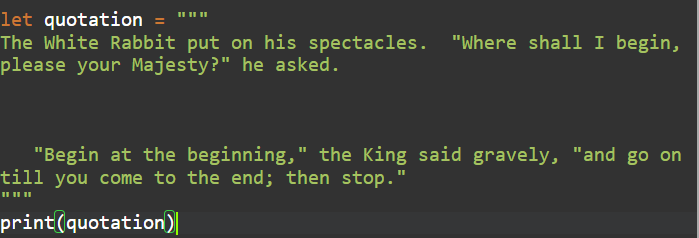


* One-Sided Ranges

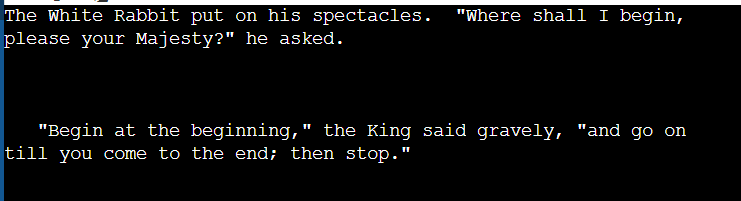


### 3. String and Character

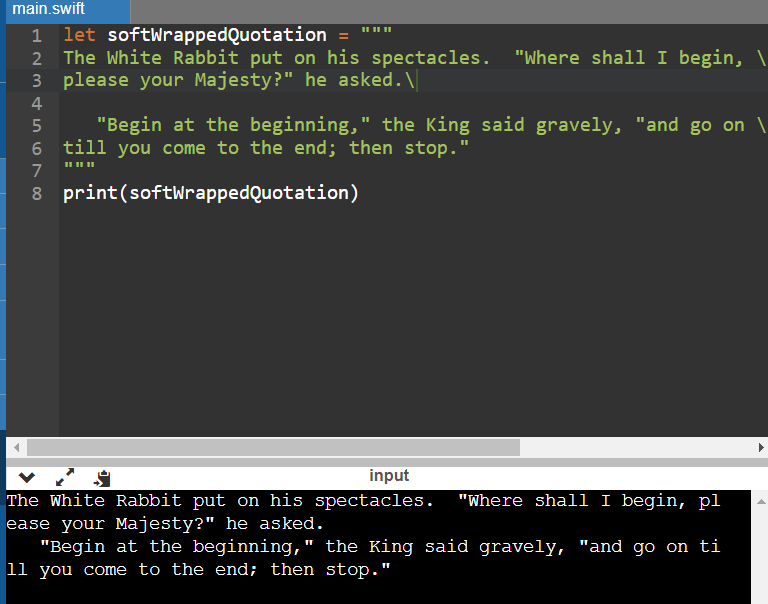
* Multiline String Literals

****

Kết quả:

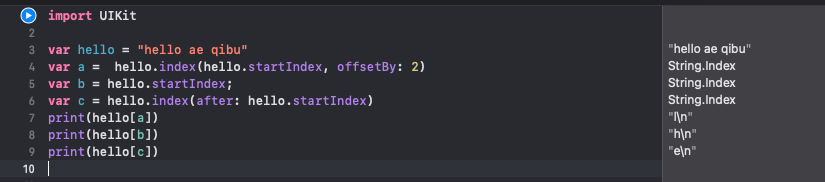


Ta sử dụng \ để nối liền dòng trên và dòng dưới



* Access and Modify a String
* String Indices(Chỉ số):

Để truy cập vào vị trí của phần tử trong String ta sử dụng startIndex để truy cập vị trí đầu tiên của String và endIndex để truy cập vị trí cuối cùng đằng sau String(ý là ta phải lùi lại 1 đơn vị để lấy giá trị cuối cùng). Ta cũng có thể sử dụng them index(before:)và index(after:)và index(\_:offsetBy:)để lấy giá trị ở 1 vị trí cụ thể:



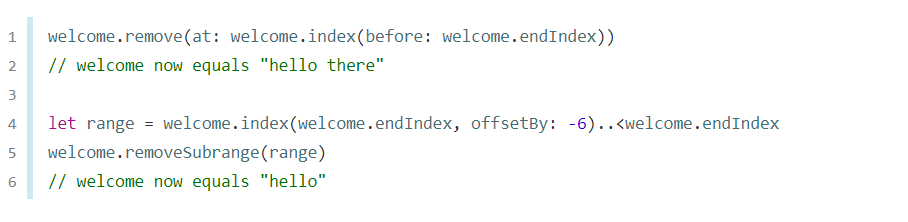
**Từ nay việc truy cập, thay đổi, thêm, sửa xóa String ta đều thông qua các phương thức như endIndex, startIndex, index(after: ), index(before: ), index(\_ offset: )**

* Inserting and Removing:

- Insert



-Remove

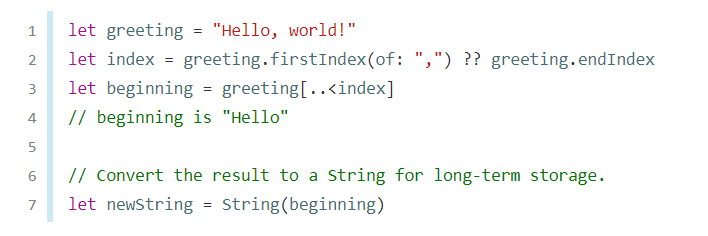


Chú ý: Ta thấy method là removeSubrange là xóa đi 1 khoảng kí tự trong String

Và let range kia là 1 biến có kiểu range 😊)) nghĩa là nó kiểu let range = 1…5

* Substrings

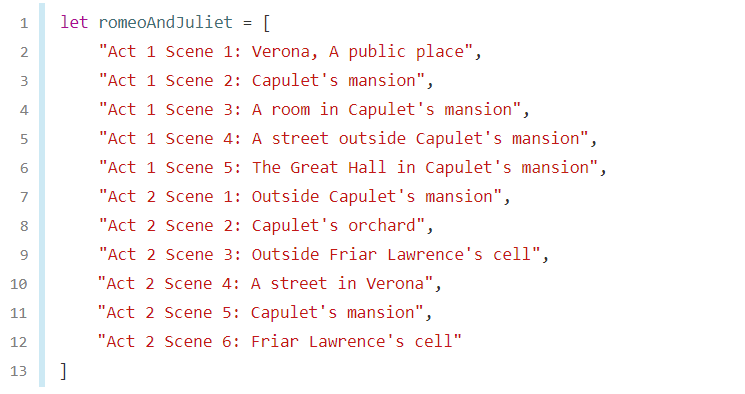
Là ta thu được 1 chuỗi nhỏ từ 1 chuỗi ta muốn

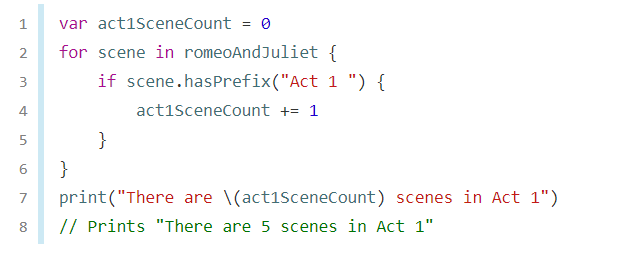


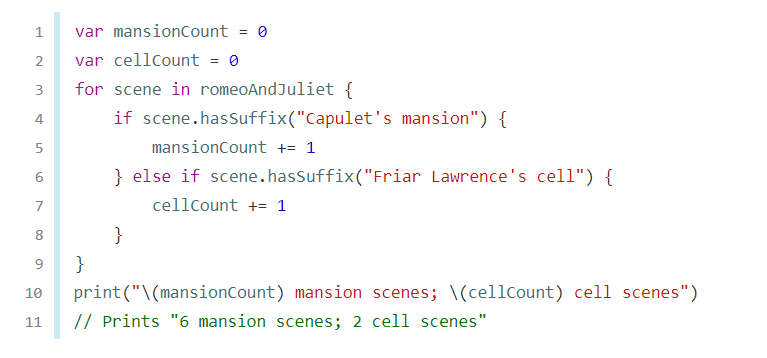
Ta có greeting.firstIndex(of: ",") là trả về vị trí đầu tiên gặp dấu , nếu không có dấu phẩy nào thì lấy giá trị nil.

* Prefix and Suffix Equality

Ta có 2 phương thức hasPrefix(\_:) và  hasSuffix(\_:), phương thức này sẽ kiểm tra xem có tồn tại chuỗi ta mong muốn trong String không, nếu có return true.

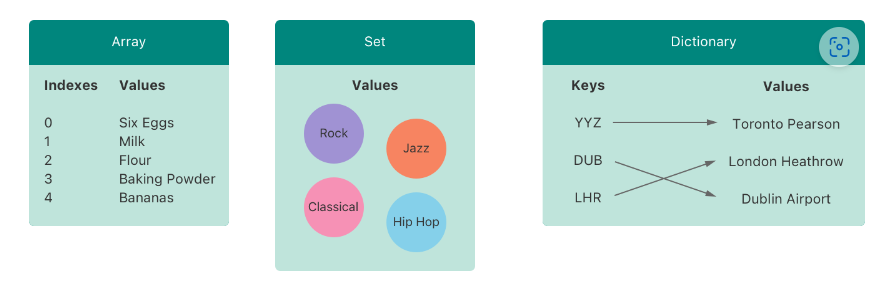






### 4. Collection types

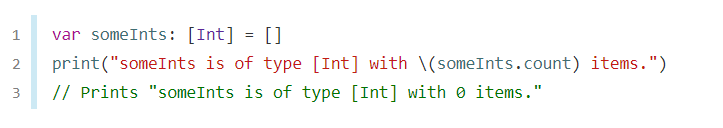
Swift cung cấp cho ta 3 collection types chính là: Arrays, Set và Dicionaries. Arrays tập hợp các giá trị được sắp xếp theo thứ tự. Set là tập giá trị không được sắp xếp theo thứ tự và Dictionaries cũng không theo thứ tự sử dụng cặp key-value



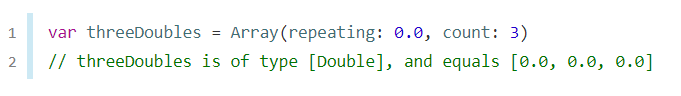
#### 4.1 Array

Khởi tạo array 4 cách:

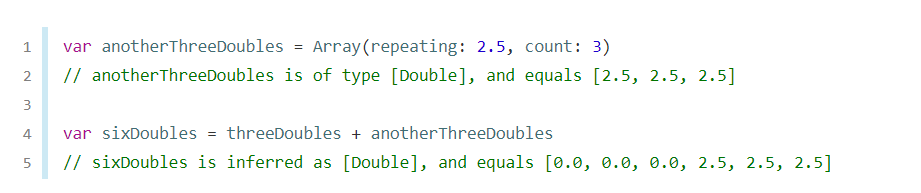
* Khởi tạo 1 empty array



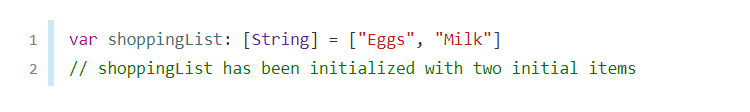
* Khởi tạo array với default value



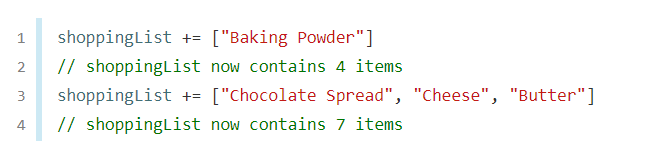
* Khởi tạo array với adding 2 arrays together



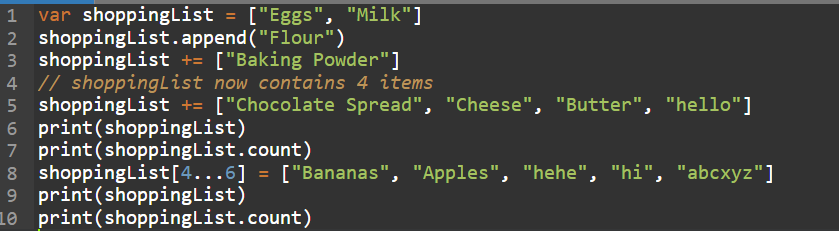
* Khởi tạo với việc gắn giá trị trước



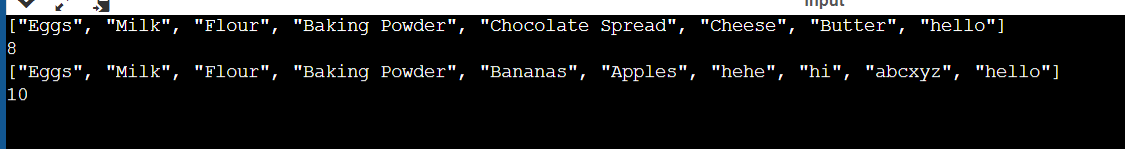
* Các phương thức chính của Array hay dung
  + Cộng Array



* + Sử dụng subscript để change range của array, nó kiểu là phải đọc ví dụ dưới



Kết quả:



Nó kiểu như sẽ thay thế từ vị trí 4 đến 6, sau đó nếu thừa thì sẽ ghi chèn vào đằng sau, chứ ko ghi đè lên phần tử tiếp theo.



Thì sẽ gặp lỗi, vì quá size

#### 4.2 Set(Tạm bỏ)

“A set stores distinct values of the same type in a collection with no defined ordering. You can use a set instead of an array when the order of items isn’t important, or when you need to ensure that an item only appears once.”

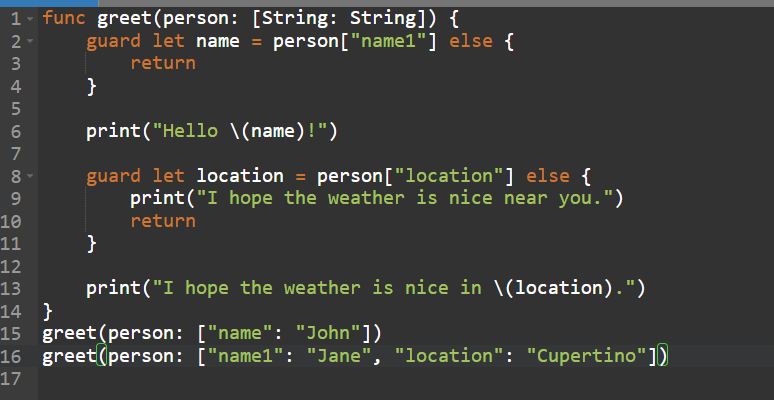
Tạm bỏ

#### 4.3 Dictionaries(tạm bỏ)

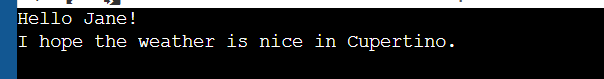
### 5. Control Flow(làm chưa kĩ)

#### 5.1. Early Exit

A guard statement, like an if statement, executes statements depending on the Boolean value of an expression. You use a guard statement to require that a condition must be true in order for the code after the guard statement to be executed. Unlike an if statement, a guard statement always has an else clause—the code inside the else clause is executed if the condition isn’t true.



Kết quả:



Giải thích: Nếu không tồn tại key là “name1” thì name = nil và sẽ không được gán, dẫn đến hàm else sẽ được thực hiện nên với lời gọi hàm greet() đầu tiên sẽ không in ra gì. Nhưng với lời gọi hàm sau, ta có key là “location” nên việc gán thành công và else không được thực hiện.

#### 5.2 Checking API(chưa làm)

### 6. Functions