

Swift cǎn bản 1



Giới thiệu

- Swift là ngôn ngữ được Apple ra mắt vào năm 2014 và được cải tiến hằng năm
- Phiên bản hiện tại của Swift là 5.x
- Yêu cầu đối với các công cụ để làm việc với Swift 5.x
 - macOS 10.14.x or later
 - Xcode 10.2 or later

Cài đặt môi trường

App Store -> Search -> Xcode -> Downloads -> Installs



Viết Swift ở đâu?

- Để biên dịch ngôn ngữ Swift, chúng ta có thể viết trên các nền tảng: Xcode, Terminal, Swift Online
- Đối với terminal (hoặc iTerm2), chúng ta phải khởi động môi trường Swift



A screenshot of a macOS terminal window titled "taof — lldb --repl=-enable-objc-interop -sdk /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Developer/SDK...". The window shows the following text:

```
Last login: Wed Jul 10 10:40:17 on console
[TaoQuynh:~ taof$ swift ← Gõ lệnh Swift khởi động môi trường
Welcome to Apple Swift version 4.2.1 (swiftlang-1000.11.42 clang-1000.11.45.1).
Type :help for assistance.
[ 1> print("Hello Swift")
Hello Swift
[ 2> |
```

The text "Gõ lệnh Swift khởi động môi trường" is overlaid in red with a white outline, and a red arrow points to the word "swift".

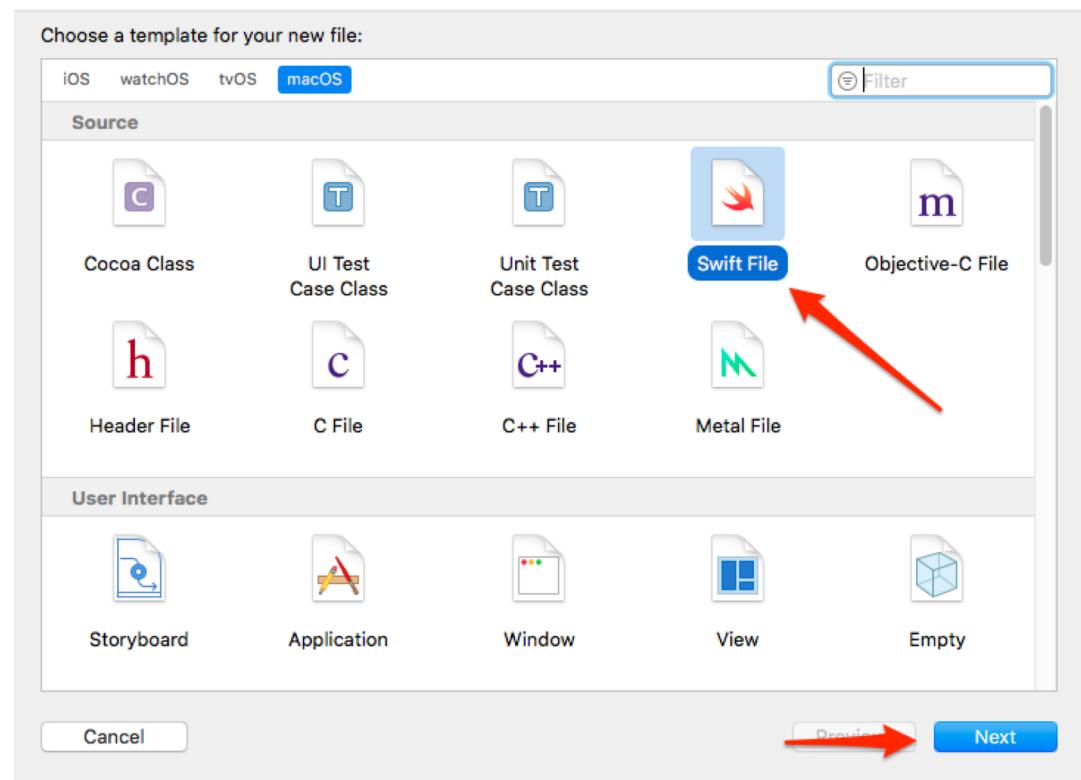
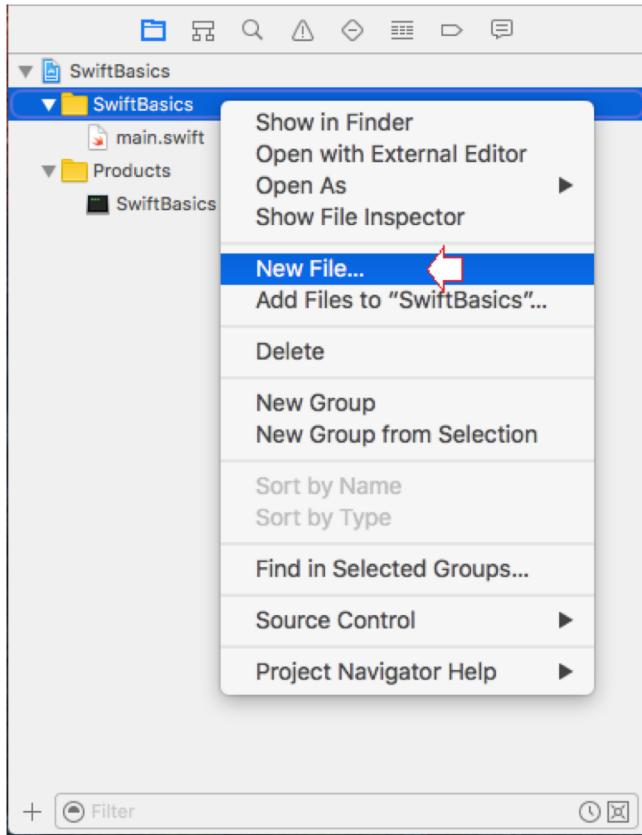
Viết Swift ở đâu?

- Đối với Xcode, chúng ta có thể sử dụng Playground hoặc Command Line Tools:
 - + Playground: Xcode -> File -> New -> Playground -> Blank
 - + Command Line Tools: Xcode -> New Projects -> chọn tab MacOS -> Command Line Tool

Đặc điểm cơ bản của Swift

- **Top Level:** Một dòng lệnh hoặc một biểu thức không nằm trong một hàm, khối lệnh hoặc một class nghĩa là nó nằm ở Top-Level, là nơi khai báo sử dụng các thư viện, biến, hằng số, hàm, lớp.
- Điểm bắt đầu của chương trình Swift: trong một chương trình Swift file **main.swift** là một file đặc biệt, vì nó là điểm bắt đầu để chạy chương trình. Có thể gọi hàm hoặc viết biểu thức ở Top-Level trên file nguồn **main.swift**, đó là một ngoại lệ dành riêng cho file này.

Thêm mới một file nguồn



Các kiểu dữ liệu trong Swift

Trong Swift có các kiểu dữ liệu cơ bản:

- Kiểu chuỗi: **String**
- Kiểu logic: **Bool**, nhận giá trị **true** hoặc **false**
- Kiểu tập hợp: **Array**, **Set**, **Dictionary**, **Enum**
- Kiểu số:

Kiểu dữ liệu	Độ rộng	Phạm vi giá trị
Int8	1byte	-127 tới 127
UInt8	1byte	0 tới 255
Int32	4bytes	-2147483648 tới 2147483647
UInt32	4bytes	0 tới 4294967295
Int64	8bytes	-9223372036854775808 tới 9223372036854775807
UInt64	8bytes	0 tới 18446744073709551615
Float	4bytes	1.2E-38 tới 3.4E+38 (~6 digits)
Double	8bytes	2.3E-308 tới 1.7E+308 (~15 digits)

Khai báo biến

- Chúng ta sử dụng từ khoá **var** để khai báo biến, cú pháp:

```
// Khai báo một biến.  
var <tên biến>: <kiểu dữ liệu>  
  
// Khai báo một biến đồng thời gán luôn giá trị.  
var <tên biến>: <kiểu dữ liệu> = <giá trị>  
  
// Khai báo một biến không xác định kiểu dữ liệu.  
var <tên biến> = <giá trị>
```

Ví dụ:

```
var strName: String // Khai báo biến kiểu chuỗi  
strName = "Swift"
```

```
var year: Int = 1985 // Khai báo và gán giá trị cho biến
```

```
var a, b, c: Double // Khai báo đồng thời nhiều biến  
var x = 0.0, y = 0.0, z = 0.0
```

Khai báo hằng số

- Chúng ta sử dụng từ khoá **let** để khai báo hằng số, cú pháp:

```
// Khai báo một hằng số xác định kiểu dữ liệu.  
let <tên hằng số>: <kiểu dữ liệu>  
  
// Khai báo một hằng số đồng thời giàn luôn giá trị.  
let <tên hằng số>: <kiểu dữ liệu> = <giá trị>  
  
// Khai báo một hằng số không xác định kiểu dữ liệu.  
let <tên hằng số> = <giá trị>
```

Ví dụ:

```
let pi = 3.14159
```

```
let languageName = "Swift"
```

```
let 🐒 = "Monkey" // Có thể sử dụng unicode để khai báo hằng số
```

Xuất ra màn hình Console

- Sử dụng lệnh print in chuỗi hoặc biến ra màn hình

```
print("My name is Swift") // Kết quả: My name is Swift  
var str = "Hello, Swift"  
print(str) // Kết quả: Hello, Swift
```

- Chúng ta có thể chèn giá trị biến/hằng vào một chuỗi trong câu lệnh print:

```
var name = "Swift"  
var age = 5  
print("I am \(name), I am \(age) years old.") // kết quả: I am Swift, I am 5 years old.
```

- Bản thân lệnh print đã có \n xuống dòng, để ngắt việc xuống dòng, hãy sử dụng terminator:

```
print("I am ", terminator: "")  
print("Swift")
```

Toán tử số học

Toán tử	Ý nghĩa	Ví dụ
- (số âm)	Giá trị âm	<code>var x = -10</code>
*	Phép nhân	<code>var x: Int</code>
/	Phép chia	<code>var y: Int = 10</code> <code>var z: Int = 5</code>
+	Phép cộng	
-	Phép trừ	<code>x = y * 10 + z - 5 / 4</code>
%	Chia lấy dư	<code>var x: Int = 9 % 4 // Kết quả: x = 1</code>

Toán tử so sánh

Toán tử	Ý nghĩa	Ví dụ
>	Lớn hơn	$5 > 4$ là đúng (true)
<	Nhỏ hơn	$4 < 5$ là đúng (true)
\geq	Lớn hơn hoặc bằng	$4 \geq 4$ là đúng (true)
\leq	Nhỏ hơn hoặc bằng	$3 \leq 4$ là đúng (true)
\equiv	Bằng nhau	$1 \equiv 1$ là đúng (true)
\neq	Không bằng nhau	$1 \neq 2$ là đúng (true)
$\&\&$	Và	$a > 4 \&\& a < 10$
$\ $	Hoặc	$a \equiv 1 \ a \equiv 4$

Toán tử gán

Toán tử	Ý nghĩa	Ví dụ
<code>+=</code>	Tính tổng	$a += b$ tương đương $a = a + b$
<code>-=</code>	Tính hiệu	$a -= b$ tương đương $a = a - b$
<code>*=</code>	Tính tích	$a *= b$ tương đương $a = a * b$
<code>/=</code>	Tính thương	$a /= b$ tương đương $a = a / b$
<code>%=</code>	Chia lấy dư	$a %= b$ tương đương $a = a \% b$

Toán tử Ternary (toán tử 3 ngôi)

<biểu thức điều kiện> ? <kết quả 1> : <kết quả 2>

```
var a: Int = 19
```

```
var b: Int = 2
```

```
let min a > b ? b : a // Kết quả: min = 2
```

Chuỗi và kí tự

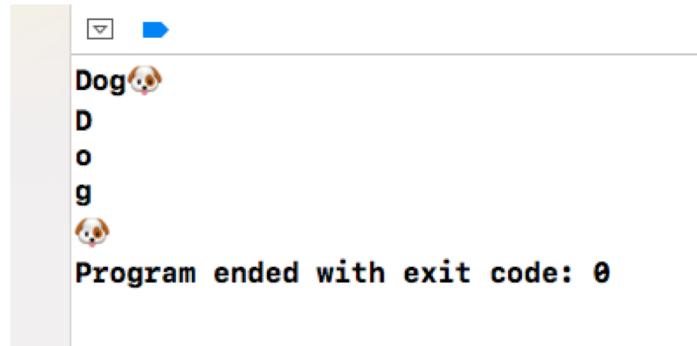
Chuỗi là một tập hợp các kí tự. Một chuỗi có thể tách thành một mảng các kí tự và ngược lại, một mảng các kí tự có thể ghép lại thành một chuỗi.

```
let dogCharacter: [Character] = ["D", "o", "g", "🐶"]
let dogString: String(dogCharacter)

print(dogString)

for character in dogString {
    print(character)
}
```

Kết quả:



```
Dog🐶
D
o
g
🐶
Program ended with exit code: 0
```

Chuỗi và kí tự

```
let currentDay = "Tuesday"  
let prefix = "Today is "
```

```
let today = prefix + currentDay  
print(today)
```

```
// .isEmpty để kiểm tra xem chuỗi có rỗng hay không, isEmpty = true là rỗng  
print(currentDay.isEmpty)
```

```
// nối chuỗi = appending  
let anotherToday = prefix.appending(currentDay)  
print(anotherToday)
```

```
// viết hoa  
print(today.uppercased())
```

```
// viết thường  
print(today.lowercased())
```

Chuỗi và kí tự

```
// Kiểm tra đầu chuỗi, cuối chuỗi  
print(today.prefix(5)); print(today.suffix(6))  
  
// Kiểm tra xem trong chuỗi có chứa chuỗi mình muốn tìm  
print(today.contains("Monday"))  
  
// đảo chuỗi  
today.reversed()  
  
// khai báo một mảng string  
let myArrayString = ["This", "is", "Techmaster", "iOS", "class"]  
  
// nối chuỗi từ một mảng  
print(myArrayString.joined())  
  
// nối chuỗi  
print(myArrayString.joined(separator: " "))  
  
// Cắt chuỗi thành một mảng  
let joinedMyString = myArrayString.joined()  
print(joinedMyString.components(separatedBy: "h"))
```