Functions

function 是自包含的代码块

函数的类型包括参数列表和返回类型，可以像使用其他类型一样使用函数类型

定义及调用

定义函数时需要定义参数和返回类型

```
func greet(person: String) -> String {
    let greeting = "Hello, " + person + "!"
    return greeting
}
```

一切需要的信息都包含在定义里了，调用函数时需要显式指明参数名

```
print(greet(person: "Anna"))
// Prints "Hello, Anna!"
```

参数与返回值

函数的参数是可选的，不定义参数也可以，但是定义时仍需写出括号

超过一个参数时，用逗号分隔每个参数

函数的返回值同样是可选的，不需要返回值时不需要写出 `->`

> 严格来讲是有返回值的，返回 `Void`，或者说一个空的 tuple `()`

如果需要返回多个值，可以使用 tuple

```swift
func minMax(array: [Int]) -> (min: Int, max: Int) {
    var currentMin = array[0]
    var currentMax = array[0]
    for value in array[1..<array.count] {
        if value < currentMin {
            currentMin = value
        } else if value > currentMax {
            currentMax = value
        }
    }
    return (currentMin, currentMax)
}
```

返回后可以使用定义时的名来访问

```swift
let bounds = minMax(array: [8, -6, 2, 109, 3, 71])
print("min is \(bounds.min) and max is \(bounds.max)")
```

如果返回的 tuple 可能为空，可以返回 optional tuple，即在括号后加上 `?`

当整个函数体只有一句表达式时，会隐式地返回表达式的值

```swift
func greeting(for person: String) -> String {
    "Hello, " + person + "!"
}
```

参数标签和参数名

每个参数都有一个参数标签（argument label）和参数名（parameter name），前者用于调用函数，而后者用于实现函数，在默认情况下使用 parameter name 作为 argument label

parameter name 必须唯一，但多个参数可以有同样的 argument label

显式使用 argument label 时采用如下语法

```swift
func someFunction(argumentLabel parameterName: Int) {
    // In the function body, parameterName refers to the argument value
    // for that parameter.
}
```

在不需要调用时指明 argument label 时，可以用下划线代替

```swift
func someFunction(_ firstParameterName: Int, secondParameterName: Int) {
    // In the function body, firstParameterName and secondParameterName
    // refer to the argument values for the first and second parameters.
}
someFunction(1, secondParameterName: 2)
```

也可以为参数设置默认值

```swift
func someFunction(parameterWithoutDefault: Int, parameterWithDefault: Int = 12) {
    // If you omit the second argument when calling this function, then
    // the value of parameterWithDefault is 12 inside the function body.
}
```

可变参数

可变参数可以接受 0 个或多个指定类型的值，只需要在参数类型名之后加上 `...`

在函数实现时，可变参数是一个常量数组

```swift
func arithmeticMean(_ numbers: Double...) -> Double {
    var total: Double = 0
    for number in numbers {
        total += number
    }
    return total / Double(numbers.count)
}
```

In-Out Parameter

参数默认是常量，如果想修改其值会产生一个编译时错误，如果需要修改参数的值并且持续到函数调用结束后，需要专门定义 in-out parameter，在类型名前加上 `inout` 关键字

传入时只能传入变量，而不能传入常量或字面量。需要在参数名前加上 `&`

```swift
func swapTwoInts(_ a: inout Int, _ b: inout Int) {
    let temporaryA = a
    a = b
    b = temporaryA
}
```

函数类型

每个函数都有其类型，由参数列表和返回值决定，形如 `(Int, Int) -> Int`

可以像使用其他类型一样使用函数类型来定义常量或变量，或者作为参数类型/返回值类型

```swift
func chooseStepFunction(backward: Bool) -> (Int) -> Int {
    return backward ? stepBackward : stepForward
}
```

嵌套函数

直接定义的函数都位于全局作用域下，在函数定义内部定义的函数可以被其外围的函数使用，但在全局是不可见的

可以将其作为返回值以在函数定义域之外使用