1. if

**var** optionalName: String? = "John Appleseed"

**if** let name = optionalName**{**

**}**如果可选的optionalName不为空，那么将其赋值给name，然后在接下来的语句中可以使用name

1. switch case

switch 支持任意类型的数据以及各种比较操作——不仅仅是整数以及测试相等。每个case执行完了之后不会执行下一个case，如果需要继续执行下一个case可以使用fallthrough

**case "celery":**

**case "cucumber", "watercress":**

**case let x where x.hasSuffix("pepper"):**

**default://必须要有default语句，并且每一个case都要有执行语句**

另外可以使用switch查找键值对：

**let interestingNumbers = [**

**"Prime": [2, 3, 5, 7, 11, 13],**

**"Fibonacci": [1, 1, 2, 3, 5, 8],**

**"Square": [1, 4, 9, 16, 25],**

**]**

**var largest = 0**

**for (kind, numbers) in interestingNumbers {**

**for number in numbers {**

**if number > largest {**

**largest = number**

**}**

**}**

**}**

1. for

**for i in 0..3 {}//不包含上界**

**for i in 0...3 {}//包含上界**

1. 函数

使用 func 来声明一个函数，使用名字和参数来调用函数。使用->来指定函数返回值。

**func greet(name: String, day: String) -> String {**

**return "Hello \(name), today is \(day)."**

**}**

使用一个元组来返回多个值。

**func getGasPrices() -> (Double, Double, Double) {**

**return (3.59, 3.69, 3.79)  
}**

函数的参数数量是可变的，用一个数组来获取它们：

**func sumOf(numbers: Int...) -> Int {**

**var sum = 0**

**for number in numbers {  
 sum += number  
}  
 return sum**

**}**

**sumOf()  
sumOf(42, 597, 12)**

函数可以嵌套。被嵌套的函数可以访问外侧函数的变量，你可以使用嵌套函数来重构一个太长或者太复杂的函数。

func returnFifteen() -> Int {

**var** y = 10

func add() {

y += 5

}

add()

**return** y

}  
returnFifteen()

函数可以作为另一个函数的返回值。

//此函数返回的是一个函数，这个返回函数的参数是int，返回值是int

func makeIncrementer() -> (Int -> Int) {

func addOne(number: Int) -> Int {

**return** 1 + number  
}

**return** addOne

}  
**var** increment = makeIncrementer()//返回一个函数int参数  
increment(7)//8

函数也可以当做参数传入另一个函数。

**func hasAnyMatches(list: Int[], condition: Int -> Bool) -> Bool {**

**for item in list {**

**if condition(item) {//此处实际是在调用lessTanTen函数**

**return true**

**}**

**}**

**return false**

**}**

**func lessThanTen(number: Int) -> Bool {**

**return number < 10**

**}**

**var numbers = [20, 19, 7, 12]**

**hasAnyMatches(numbers, lessThanTen)//传入函数**

1. 对象和类

使用 class 和类名来创建一个类。类中属性的声明和常量、变量声明一样，唯一的区别就是它们的上下文是类。同样，方法和函数声明也一样。

**class Shape {**

**var numberOfSides = 0**

**func simpleDescription() -> String {**

**return "A shape with \(numberOfSides) sides."**

**}**

**}**

**var shape = Shape()**

**shape.numberOfSides = 7**

**var shapeDescription = shape.simpleDescription()**

这个版本的 Shape 类缺少了一些重要的东西：一个构造函数来初始化类实例。使用 init 来创建一个构造器。

**class NamedShape {**

**var numberOfSides: Int = 0**

**var name: String**

**init(name: String) {**

**self.name = name**

**}**

**func simpleDescription() -> String {**

**return "A shape with \(numberOfSides) sides."**

**}**

**}**

如果你需要在删除对象之前进行一些清理工作，使用 deinit 创建一个析构函数。