代码注释版

```
import UIKit

class ViewController: UIViewController {


    //@IBOutlet：需要与StoryBoard或Xib中的view关联
    //weak：属于内存管理的部分，只要是从StoryBoard或Xib拖拽过来的就是weak
    @IBOutlet weak  var  lbName:UILabel?

    override func viewDidLoad() {
        super.viewDidLoad()

        // Command + T

        //打印
        print("Hello, Swift!")


        //MARK:- var let String

        //var声明后可以修改，let不可以
        var name :String = "Yung"

        print(name)

        //字符串拼接
        let fullName :String = name + "Fan"

        print(fullName)

        lbName?.text = fullName

        // 代码折叠功能
```

```
// command + shift + option + 方向左右 —— 收起/展开所有方法的代码块
// command + option + 方向左右 —— 收起/展开当前方法


//MARK:- Array(有序)

//数组初始化方法1
var strArray = ["A", "C", "B", "D"]
//数组排序
strArray.sort()

print(strArray)

//数组初始化方法2
let intArray:[Int] = [1, 2, 3, 4]

print(intArray)

//数组遍历方法1
//A..<B：表示大于等于A小于B的一个区间
//count：数组元素的个数
for index in 0..<intArray.count{

    //索引访问数组中的值
     print("\(intArray[index])")
}

//数组初始化方法3
var floatArray:Array<Float> = Array()
//追加数据
floatArray.append(3.0)

floatArray.append(4.0)

floatArray.append(5.0)

floatArray.append(6.0)
```

```
print(floatArray)

//数组遍历方法2
for floatValue in floatArray {

    print(floatValue)
}


//MARK:- Dictionary(无序)

//字典初始化方法，字典里面的每个元素为 key:value 的形式
let pDic  = [1:"张三", 2:"李四", 3:"王五", 4:"赵六"]

print(pDic)

//字典遍历
//():  元组类型，这种方式是通过元组遍历字典
for (key, value) in pDic {

    print("\(key):\(value)" )
}

//MARK:- Closures与排序
// 闭包语法
//{
//  (parameters) -> return type in
//     Statements
//     return parameters
//}


//完整写法
//obj1, obj2：待排序的两个字典的值
//-> Bool：返回值类型
//in：用于隔开前后两部分
```

```
        let sortedDic1 = pDic.sorted { (obj1: (key: Int, value: String), obj2:
(key: Int, value: String)) -> Bool in

            return obj1.key > obj2.key
        }

        print(sortedDic1)

        //第一次简化 类型自动推导 可以根据参数推断
        let sortedDic2 = pDic.sorted { (obj1, obj2) in

            return obj1.key > obj2.key
        }

        print(sortedDic2)

        //第二次简化 如果函数体只包含一句 return 代码，可省略 return
        let sortedDic3 = pDic.sorted { (obj1, obj2) in

            obj1.key > obj2.key
        }

        print(sortedDic3)

        //第三次简化 由于Swift具有强大的推测能力，参数只有两个，可以从
下标从0开始，可通过"$"获取。编译器也可以自动推测出参数。所以继续
省略参数列表 （obj1, obj2）和 关键字 in
        let sortedDic4 = pDic.sorted(by: {$0.key > $1.key})

        print(sortedDic4)

        //甚至可以直接进化到如下版本
        let sortedDic5 = pDic.sorted(by: {$0.0 > $1.0})

        print(sortedDic5)


        //MARK:- Timer(定时器，按照固定周期循环执行某段代码)
```

```
        //withTimeInterval：每隔多长时间执行一次
        //repeats：是否重复执行
        //block：执行的代码段
        Timer.scheduledTimer(withTimeInterval: 1.0, repeats: true, block: {

            //smalltimer表示传进来的参数，其实就是当前的这个Timer，也可以省略，因为Swift具有强大的推测能力
            smalltimer in

            print("我在不断打印 \(smalltimer.isValid)")

        })

        //调用函数
        //self：调用的是当前的对象中的方法
        //_：不关心函数的返回值
        _ =  self.add(num1: 3, num2: 4)

    }



    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    //MARK:- Function(函数)
    //函数语法
    //func funcname(形参) -> returntype
    //{
    //    Statements
    //    return parameters
    //}

    //num1与num2：给调用者看的，通常取一个有意义的名字，让调用者知道参数的含义
    func add(num1:Int, num2:Int) ->Int {
```

```
        return num1 + num2
    }
}
```