FP定义:

简单的说函数式编程是一种编程范式，也就是如何编写程序的方法论，主要思想是把运算过程尽量写成一系列嵌套的函数调用。

特性:

1.闭包和高阶函数 Map,Filter,Reduce

函数编程支持函数作为第一类对象，有时称为闭包或者仿函数（functor）对象。实质上，闭包是起函数的作用并可以像对象一样操作的对象。与此类似，FP 语言支持高阶函数。高阶函数可以用另一个函数（间接地，用一个表达式） 作为其输入参数，在某些情况下，它甚至返回一个函数作为其输出参数。这两种结构结合在一起使得可以用优雅的方式进行模块化编程，这是使用 FP 的最大好处。

2.惰性计算 Lazy

在使用惰性计算时，表达式不在它被绑定到变量之后就立即求值，而是在该值被取用的时候求值。

3.模块化

相较于把程序认为是一系列赋值和方法调用，函数式开发者更倾向于强调每个程序都能够被反复分解为越来越小的模块单元，而所有这些块可以通过函数装配起来，以定义一个完整的程序。

4.递归

特点:

1.函数是"第一等公民"

举例来说，下面代码中的print变量就是一个函数，可以作为另一个函数的参数。

let printInt = { print($0) }

(1...5).forEach(printInt)

2.只用"表达式"，不用"语句"

"表达式"（expression）是一个单纯的运算过程，总是有返回值；"语句"（statement）是执行某种操作，

没有返回值。函数式编程要求，只使用表达式，不使用语句。也就是说，每一步都是单纯的运算，而且都有返回值。

3.没有"副作用"

所谓"副作用"（side effect），指的是函数内部与外部互动（最典型的情况，就是修改全局变量的值），产生运算以外的其他结果。 函数式编程强调没有"副作用"，意味着函数要保持独立，所有功能就是返回一个新的值，没有其他行为，尤其是不得修改外部变量的值。

4.不修改状态

其他类型的语言中，变量往往用来保存"状态"（state）。不修改变量，意味着状态不能保存在变量中。函数式编程使用参数保存状态，最好的例子就是递归

5.引用透明性

函数程序通常还加强引用透明性，即如果提供同样的输入，那么函数总是返回同样的结果

优点:

1. 代码简洁，开发快速

2. 接近自然语言，易于理解

表达式(1 + 2) * 3 - 4，

subtract(multiply(add(1,2), 3), 4)

变形

add(1,2).multiply(3).subtract(4)

下面的代码，大家应该一眼就能明白它的意思吧

merge([1,2],[3,4]).sort().search("2")

因此，函数式编程的代码更容易理解。

3. 更方便的代码管理

缺点

1、数据不可变，占用资源 2、晦涩难懂 3、大工程不适用