为什么学习Haskell / Python使你成为更差劲的程序员  
  
与你经常读到相反的是，学习Python和Haskell并没有改善其他语言的编程水平。特别是Haskell与命令式语言有太多的不同，即便对于不使用Haskell的人来说也应该会给编程带来一些新的有益见解。我目前的经验与此不完全相符，以下便是原因：  
  
    失去动力。  
  
虽然我很少使用Haskell，但是我在一定程度上以Python或是Haskell的思维方式进行思考：我一直不断寻找这些语言中我想要使用的特性，如果用这些语言可以少写多少代码（虽然这些语言之间彼此非常不同，他们都远比我工作中使用的语言C#强大。我意识到通常使用这两种语言，我可以减少2-5倍的代码，偶尔可以减少10-20倍。

 此外，我的 Haskell使用经验使我看到命令式代码中潜藏着各种bug。在用过Haskell之前，我就已经意识到程序的状态以及副作用会导致无尽的调试，但没有任何替代品，我想，我刚刚与居住。现在我知道有其他的方式做这些事情，我觉得它很难得到满意我写的代码。我不断地知道我写的陷阱，其他人都可能往里钻  
  
    我也觉得很丑陋相比，Python和Haskell的C＃代码。视觉上的水平，括号到处确定，他们是不是强制性的无处不在，但执行编码标准，具有良好的理由强制使用，使代码包含了很多线路噪音和空的空间，并结合库的详细和类型声明等，你会发现，C＃中的页面几乎做任何事情。我还以为美丽的数学水平，C＃是一个肮脏的泥相比，令人叹为观止，优雅的塔，是Haskell的小屋。  
  
    这些东西的最终结果是让我非常沮丧，士气低落。我觉得像一个人的编译器，转化为Haskell或在我脑海中的Python语言，这是一个整体水平较低。  
  
    使用其他语言时，使用功能的风格混淆您的代码。  
  
    C＃中已经开始得到一些功能，更友好的功能风格的编程。因此，面临着一个非常普遍的情况时，有一天，我尝试了功能的解决方案。我有一个Foo对象的名单，每一个描述（）方法返回一个字符串。我需要连接的所有非空的描述，它们之间插入换行符。  
  
    我想写的代码是Python代码：  
  
    “\ N”。加入（foo.description（）在我清单美孚  
                             ，如果foo.description（）=“”）  
  
    或本哈斯克尔：  
  
    Concat的$ List.intersperse“\ N”过滤器（/ =“”）$图描述馆藏资料  
  
    从C仿制药＃2.0和它们所包含的方法，我得到的最好的：  
  
    的string.join（“\ N”，mylist.ConvertAll <string>的（  
                委托（富富）  
                {  
                        返回foo.Description（）;  
                }）。的FindAll（  
                委托（字符串x）  
                {  
                        返回X =“”;  
                。}）的toArray（））;  
  
    如果我已经开始用不同的数据结构，C＃版本将更糟 - 和C＃似乎有数百个不同的集合类，用于在NET库不一致。我也应该指出，如果你写的任何接受“代表”（最近的事情，以一流的功能）的方法，你必须声明函数签名为他们分开，如果不存在（或者如果你不知道在哪里能找到一个已经存在的），进一步增加任何功能代码风格的膨胀。  
  
    有C＃版本的一些大问题。首先是，很少有减少大小，与势在必行样式代码，如有。比较繁琐的循环，否则我会写：  
  
    字符串RETVAL =“”;  
    的foreach（美孚美孚在我清单）  
    {  
        字符串DESC = foo.description（）;  
        （DESC =“”）  
        {  
                （RETVAL！=“”）  
                   RETVAL + =“\ N”  
                RETVAL + =递减;  
        }  
    }  
  
    没有太多中。  
  
    第二，我花了更长的时间来写。我不得不做一些实验看到多少类型的信息，我不得不添加到它来编译（如添加横空出世不是必要明确委托演员，但我没有指定，而不是ConvertAll ConvertAll <string>的）。  
  
    最后，还有一个问题，此代码将进入我的麻烦与我的同事。为什么我写这些复杂的代码 - 使用匿名委托等先进功能 - 当一个简单的循环就足够了吗？实际上，我离开了我的功能版本，但如此狼狈，我不得不添加一个简短的解释性说明。  
  
    事实是，严重的语言功能成语的工作，不必为他们的句法支持。 Java中，从我所知道的，会变得更糟。 Ç＃患有虽然一些功能能够实现更多的功能规划已经在C＃2.0（以及各种其他语言的改进）抵达。NET库的巨大块没有被更新，以利用他们，和我们自己的代码肯定有不。  
  
    有人可能会说，你仍然可以使用功能规划的原则（无副作用，功能，只依赖自己的投入等），并得到好处，这样，即使你不能使用成语。在现实中，库和命令式语言的设计框架，只是没有这样。 ASP.NET是一个特别坏的榜样。你从Control继承和覆盖各种方法开发的控制。大多数这些方法没有返回值，并没有投入，完全变异的对象（或其他对象）的状态和工作。然后，他们被称为有点微妙和复杂的顺序（是的，这是一场噩梦调试）的框架。  
  
    事实上，应用功能规划的原则，会导致我尽可能使用静态方法（无实例方法），避免任何变异状态（甚至有变异状态的可能性）。我会用几无处不在，“哑”数据类型，并保持单独从他们的算法。苍蝇在面对的教诲，或至少在主编程范式普及的今天，面向对象的做法，。我不能申请，我认为是好的原则，不拒绝非常范式的语言和我周围的库编写代码。这是相当绝望。  
  
因此，学习Python和Haskell意志消沉我，鼓励我写的是奇怪和难以理解的代码，在OOP代码基地的背景下，提供了命令式编程没有什么好处。我毫不怀疑，在一般情况下，我更好的学习这些语言的程序员，但我目前的状况，我不是一个更好的软件开发商 - 我的生产力实际上江河下泻。当我沮丧的C＃代码我写的，然后我去，并再次把它写在Haskell和Python，他们是如何更好地证明自己，这是一个毫无意义的运动，只有我进一步失去动力。  
  
这个故事的寓意：不要打扰提高自己，除非你有相应的自由，以改善您的环境。这是相当令人沮丧。任何人都可以放在一个更好的自旋和我高兴起来吗？  
  
更新：我也许应该更加明显，对某些人来说，文章标题是不完全严重，主要是我只是发牢骚。