为什么把编程当作自学的入口？

很多人误以为“编程”是很难的事情。

…… 实则不然 —— 这恰恰是我们选择“编程”作为自学的第一个“执行项目”的原因。

一本关于自学能力的书，若是真的能够起到作用，那么它就必须让读者在读之前和读之后不一样 —— 比如，之前可能没有自学能力，或者自学能力很差，之后就有了一定的自学能力……

然而，这很难。不但对读者来说很难，对作者来说更难 —— 我当过那么多年被学生高度评价的老师，出版过若干本畅销且长销的书籍，所以更是清楚地知道例子的重要性。

道理当然很重要；可是，在传递道理的时候，例子相对来看好像更重要。

同样的道理，例子不准，人就可能会理解错；例子不精彩，人就可能听不进去；例子居然可以令人震惊，那就可以做到让听众、让读者“永生不忘”。

许多年前，有位后来已经在美国读书博士毕业了的学生来信，大意是说：

> 好多年前，我在新东方上课，听您讲，人学习就好像是动物进化一样…… 很多人很早就开始停止了进化，本质上跟猴子没啥区别。
>
> 那段类比好长，我记不太清楚细节了…… 可是，当时我是出了一身汗的，因为我忽然觉得自己是一只猴子。可是，突然之间，我不想继续做猴子，更不想一直做猴子！
>
> 从那之后，我好像变了一个人似的…… 现在我已经博士毕业了，觉得应该写封信告诉您，我不再是猴子了，最起码是大猩猩，而且我保证，我会一直进化。
>
> ……

所以啊，在我看来，写书讲课之前，最重要的工作，也是做得最多的事情，其实就是“找到好例子” —— 那即意味着说，先要找到很多很多恰当合适的例子，而后再通过反复比较试验，挑出那个效果最好的例子。了解了这一点，将来你准备任何演讲，都会不由自主地多花一点时间在这方面，效果肯定比“把幻灯片做得更花哨一些”要好太多了罢？

后来，我选中了一个例子：“**自学编程**” —— “*尽量只通过阅读学会编程*”。

（一）

选择它的理由，首先就在于：

> 事实证明，**它就是无论是谁都能学会的** —— 千万别不信。

它老少皆宜 —— 也就是说，“只要你愿意”，根本没有年龄差异。十二岁的孩子可以学；十八岁的大学生可以学；在职工作人员可以学…… 就算你已经退休了，想学就能学，谁也拦不住你。

它也不分性别，男性可以学，女性同样可以学，性别差异在这里完全不存在。

它也不分国界，更没有区域差异 —— 互联网的恩惠在于，你在北京、纽约也好，老头沟、门头沟也罢，在这个领域里同样完全没有任何具体差异。

尤其是在中国。现状是，中国的人口密度极高，优质教育资源的确就是稀缺…… 但，在计算机科学领域，所有的所谓“优质教育资源”事实上完全没有任何独特的竞争力 —— 编程领域，实际上是当今世上极为罕见的“**教育机会公平之地**”。又，不仅在中国如此，事实上，在全球范围内也都是如此。

（二）

编程作为“讲解如何习得自学能力的例子”，实在是太好了。

首先，编程这个东西反正要自学 —— 不信你问问计算机专业的人，他们会如实告诉你的，学校里确实也教，但说实话都教得不太好……

其次，编程这个东西最适合“仅靠阅读自学” —— 这个领域发展很快，到最后，新东西出来的时候，没有老师存在，任由你是谁，都只能去阅读“官方文档”，只此一条路。

然后，也是最重要的一条，别管是不是很多人觉得编程是很难的东西，事实上它就是每个人都应该具备的技能。

许多年前，不识字，被称为文盲……

后来，人们反应过来了，不识英文，也是文盲，因为科学文献的主导语言是英文，读不懂英文，什么都吃不上热乎的；等菜好不容易端上来了吧，早就凉了不说，味道都常常会变……

再后来，不懂基本计算机操作技能的，也算是文盲，因为他们无论做什么事情，效率都太低下了，明明可以用快捷键一下子完成的事情，却非要手动大量重复……

到了最近，不懂数据分析的，也开始算作文盲了。许多年前人们惊呼信息时代来了的时候，其实暂时体会不到什么太多的不同。然而，许多年过去，互联网上的格式化数据越来越多，不仅如此，实时产出的格式化数据也越来越多，于是，数据分析不仅成了必备的能力，而且早就开始直接影响一

个人的薪资水平。

你作为一个个体，每天都在产生各种各样的数据，然后时时刻刻都在被别人使用着、分析着…… 然而你自己却全然没有数据分析能力，甚至不知道这事儿很重要，是不是感觉很可怕？你看看周边那么多人，有多大的比例想过这事儿？反正那些天天看机器算法生成的信息流的人好像就是全然不在意自己正在被支配……

怎么办？学呗，学点编程罢 —— 巧了，*这还真是个正常人都能学会的技能*。

（三）

编程作为"讲解如何习得自学能力的例子"最好的地方在于，这个领域的知识结构，最接近每个人所面对的人生中的知识结构。

这是什么意思呢？

编程入门的门槛之所以高，有个比较特殊的原因：

> **它的知识点结构不是线性的。**

我们在中小学里所遇到的教科书，其中每个章节所涉及到的知识点之间，全都是线性关联。第一章学好了，就有基础学第二章；在第二章的概念不会出现在第一章之中……

很遗憾，编程所涉及到的知识点没办法这样组织 —— 就是不行。编程教材之所以难以读懂，就是因为它的各章中的知识点结构不是线性排列的。你经常在某一章读到不知道后面第几章才可能讲解清楚的概念。

比如，几乎所有的 Python 编程书籍上来就举个例子：

```
print('Hello, world!')
```

姑且不管这个例子是否意义非凡或者意义莫名，关键在于， print() 是个函数，而函数这个概念，不可能一上来就讲清楚，只能在后面若干章之后才开始讲解……

> 想要理解当前的知识点，需要依赖对以后才能开始学习的某个甚至多个知识点的深入了解……

这种现象，可以借用一个专门的英文概念，叫做"**Forward References**" —— 原本是计算机领域里的一个术语 (https://en.wikipedia.org/wiki/Forward_declaration)。为了配合当前的语境，姑且把它翻译为"**过早引用**"罢，或者"**前置引用**"也行。

学校里的课本，都很严谨 —— 任何概念，未经声明就禁止使用。所以，学完一章，就能学下一章；跳到某一章遇到不熟悉的概念，往前翻肯定能找到……

在学校里习惯了这种知识体系的人，离开学校之后马上抓瞎 —— **社会的知识结构不仅不是这样的，而且*几乎全都不是这样的*。**工作中、生活里，充满了各式各样的"*过早引用*"。为什么总是要到多年以后你才明白父母曾经说过的话那么有道理？为什么总要到孩子已经长大之后才反应过来当初自己对孩子做错过很多事情？为什么在自己成为领导之前总是以为他们只不过是在忽悠你？为什么那么多人创业失败了之后才反应过来当初投资人提醒的一些观念其实是千真万确的？—— 因为很多概念很多观念是"过早引用"，在当时就是非常难以理解……

自学编程在这方面的好处在于，在自学的过程中，其实你相当于过了一遍"模拟人生" —— 于是，面对同样的"*过早引用*"，你不会觉得那么莫名其妙，你有一套你早已在"模拟人生"中练就的方法论去应对。

（四）

另外一个把编程作为"讲解如何习得自学能力的例子"最好的地方在于，你在这个过程中将不得不习得英语 —— 起码是英文阅读能力，它能让你在不知不觉中"脱盲"。

学编程中最重要的活动就是"阅读官方文档"。学 Python 更是如此。Python 有很多非常优秀的地方，其中一个令人无法忽视的优点就是它的文档完善程度极好。它甚至有专门的文档生成工具，Sphinx (http://www.sphinx-doc.org/en/master/)：

> Sphinx is a tool that makes it easy to create intelligent and beautiful documentation, written by Georg Brandl and licensed under the BSD license.
>
> It was originally created for the Python documentation (https://docs.python.org/), and it has excellent facilities for the documentation of software projects in a range of languages. Of course, this site is also created from reStructuredText sources using Sphinx!

最好的 Python 教程，是官方网站上的 The Python Tutorial (https://docs.python.org/3/tutorial/index.html)，读它就够了。我个人完全没兴趣从头到尾写一本 Python 编程教材，不仅因为人家写得真好，而且它就放在那里。

虽然你在官方网站上就是很难找到它的中文版，虽然就不告诉你到底在哪里也显得很不厚道，但是，我建议你就只看英文版 —— 因为离开了这个教程之后，还是要面对绝大多数都是英文的现实。

为了照顾那些也想读完本书，但因为种种原因想着读中文可以快一些的人，链接还是放在这里：

> - https://docs.python.org/zh-cn/3/tutorial/index.html (https://docs.python.org/zh-cn/3/tutorial/index.html) (for v.3.7.2)
> - http://www.pythondoc.com/pythontutorial3/ (http://www.pythondoc.com/pythontutorial3/) (for v.3.6.3)

我曾经专门写过一本书发布在网上，叫《人人都能用英语 (https://github.com/xiaolai/everyone-can-use-english)》。其中的观点就是，大多数人之所以在英语这事儿上很挫，是因为他们花无数的时间去*"学"*，但，就是*"不用"*。学以致用，用以促学。可就是不用，无论如何就是不用，那英语学了那么多年能学好吗？

自学编程的一个"副作用"就是，**你不得不用英语**。而且还是天天用，不停地用。

当年我上大学的时候，最初英语当然也不好。不过，因为想读当时还是禁书的《动物庄园》（Animal Farm (https://www.marxists.org/subject/art/literature/children/texts/orwell/animal-farm/index.htm)），就只好看原版（当时好不容易搞到的是本英法对照版）..... 然后英语阅读就基本过关了。

这原理大抵上是这样，刚开始，英语就好像一层毛玻璃，隔在你和你很想要了解的内容之间。然而，由于你对那内容的兴趣和需求是如此强烈，乃至于即便隔着毛玻璃你也会挣扎着去看清楚..... 挣扎久了（其实没两天就不一样），你的"视力"就进化了，毛玻璃还在那里，但你好像可以穿透它看清一切.....

自学编程，也算是一举两得了！

（五）

当然，把编程作为"讲解如何习得自学能力的例子"，实在是太好了的最重要原因在于，自学编程对任何人来说都绝对是：

> - 现实的（Practical）
> - 可行动的（Actionable）
> - 并且还是真正是可达成的（Achievable）

最重要的就是最后这个"可达成的"。虽然对读者和作者来说，一个做到没那么容易，另一个讲清楚也非常难，但是，既然是所有人都"可达成的"，总得试试吧？但是，请相信我，这事儿比减肥容易多了 —— 毕竟，你不是在跟基因作斗争。

这只是个起点。

尽量只靠阅读学会编程，哪怕仅仅是入门，这个经历和经验都是极为宝贵的。

自学是门手艺。只不过它并不像卖油翁的手艺那样很容易被别人看到，也不是很容易拿它出来炫耀——因为别人看不到么！然而，经年累月，就不一样了，那好处管他别人知不知道，自己却清楚得很！

你身边总有些人能把别人做不好的事儿做得极好，你一定很羡慕。可他们为什么能做到那样呢？很简单啊，他们的自学能力强，所以他们能学会大多数自学能力差的人终生学不到的东西。而且他们的自学能力会越来越强，每学会一样新东西，他们就积累了更多自学经验，难以对外言表的经验，再遇到什么新东西，相对没那么吃力。

另外，自学者最大的感受就是万物相通。他们经常说的话有这么一句："…… **到最后，都是一样的呢**。"

（六）

最后一个好处，一句话就能说清楚，并且，随着时间的推移，你对此的感触会越来越深：

> 在这个领域里，自学的人最多……

没有什么比这句话更令人舒心的了：**相信我，你并不孤独**。