刻意思考

随着时间的推移，你会体会到它的威力：

> 刻意思考哪儿需要刻意练习

只不过是一句话而已，却竟然知道与不知道之间终究会形成天壤之别的差异，也是神奇。

刻意思考，就是所谓的琢磨，琢磨这事儿，一旦开始就简单得要死，可无从下手的时候就神秘无比。让我们再看一个“刻意思考” —— 即，琢磨 —— 的应用领域：

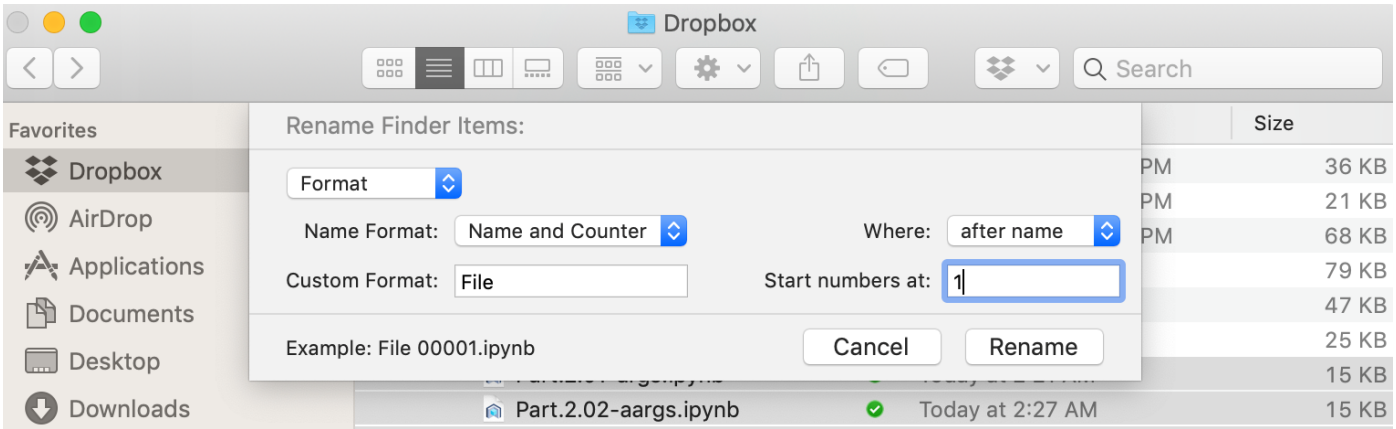
> **这东西能用在哪儿呢？**

很多人学了却没怎么练，有一个很现实的原因 —— 没什么地方用得上。

这也怪我们的应试教育，大学前上 12 年学，“学”（更多是被逼的）的绝大多数东西，只有一个能够切实体会到的用处，考试 —— 中考、高考，以及以它们为目标的无数“模考”..... 于是，反过来，不管学什么东西，除了考试之外，几乎无法想象其他的用处。

一旦我们启动了对某项技能的自学之后，在那过程中，最具价值的刻意思考就是，时时刻刻琢磨“这东西能用在哪儿呢？”

比如，当你看到字符串的 Methods 中有一个 `str.zfill()` 的时候，马上就能想到，“嗯！这可以用来批量更名文件.....”



虽然现在的 Mac OS 操作系统里已经有相当不错的批量更名工具内建在 Finder 之中（选中多个文件之后，在右键菜单中能看到 rename 命令），但，这是近期才加进去的功能，几年前却没有 —— 也就是说，几年前的时候，有人可以用 `str.zfill()` 写个简单的程序完成自己的工作，而另外一些人仅因为操作系统没有提供类似的功能就要么手工做，要么干脆忍着忘了算了.....

但，更多的时候，需要你花时间去琢磨，才能找到用处。

找到用处，有时候还真挺难的 —— 因为人都一样，容易被自己的眼界所限，放眼望过去，没有用处，自然也就不用了，甚至不用学了，更不用提那就肯定是感觉不用练了……

所以，仔细想想罢 —— 那些在学校里帮老师干活的小朋友们，更多情况下还真不是很多人以为的"拍马屁"（不排除肯定有哈），只不过是在"主动找活干"……

找活干，是应用所学的最有效方式，有活干，所以就有问题需要解决，所以就有机会反复攻关，在这个过程中，**以用带练**……

所以，很多人在很多事儿上都想反了。

人们常常取笑那些呼哧呼哧干活的人，笑着说，"能者多劳"，觉得他们有点傻。

这话真的没错。但，这么说更准：**劳者多能** —— 你看，都想反了吧？

到最后，一切自学能力差的人，外部的表现都差不多，都起码包括这么一条：眼里没活。他们也不喜欢干活，甚至也没想过，玩乐也是干活（每次逢年过节玩得累死那种） —— 从消耗或者成本的角度来看根本没啥区别 —— 只不过那些通常都是没有产出的活而已。

在最初想不出有什么用处的时候，还可以退而求其次，看看"别人想出什么用处没有？" —— 比如，我去 Google best applications of python skill ，在第一个页面我就发现了这么篇文章："What exactly can you do with Python? (https://medium.freecodecamp.org/what-can-you-do-with-python-the-3-main-applications-518db9a68a78)"，翻了一会儿觉得颇有意思……

再高阶一点的刻意思考（琢磨），无非是在"这东西能用在哪儿呢？"这句话里加上一个字而已：

> 这东西**还**能用在哪儿呢？

我觉得这个问题对思维训练的帮助非常深刻 —— 别看只是多了一个字而已。

当我读到在编程的过程中有很多的"约定"的时候，就琢磨着：

> - 哦，原来约定如此重要……
> - 哦，原来竟然有那么多人不重视约定……
> - 哦，原来就应该直接过滤掉那些不遵守约定的人…… —— 那这个原理（东西）还能用在哪儿呢？ —— 哦，在生活中也一样，遇到不遵守约定的人或事，直接过滤，不要浪费自己的生命……

学编程真的很有意思，因为这个领域是世界上最聪明的人群之一开辟出来并不断共同努力着发展的，所以，在这个世界里有很多思考方式，琢磨方式，甚至可以干脆称为"做事哲学"的东西，可以普遍应用在其它领域，甚至其它任何领域。

比如，在开发方法论中，有一个叫做 MoSCoW Method (https://en.wikipedia.org/wiki/MoSCoW_method) 的东西，1994 年由 Clegg Dai 在《 Case Method Fast-Track: A RAD Approach》一书中提出的 —— 两个 o 字母放在那里，是为了能够把这个缩写读出来，发音跟莫斯科一样。

简单说，就是，凡事儿都可以分为：

> - Must have
> - Should have
> - Could have
> - Won't have

于是，在开发的时候，把所谓的需求打上这 4 个标签中的某一个，以此分类，就很容易剔除掉那些实际上做了还不如不做的功能……

琢磨一下罢，这个东西还可以用在什么地方？

显然，除了编程之外，其他应用领域挺多的，这个原则相当地有启发性……

我写书就是这样的。在准备的过程中 —— 这个过程比绝大多数人想象得长很多 —— 我会罗列所有我能想到的相关话题…… 等我觉得已经再也没有什么可补充的时候，再为这些话题写上几句话构成大纲…… 这时候就会发现很多话题其实应该是同一个话题。如此这般，一次扩张，一次搜索之后，就会进行下一步，应用 MoSCow 原则，给这些话题打上标签 —— 在这过程中，总是发现很多之前感觉必要的话题，其实可以打上 Won't have 的标签，于是，把它们剔除，然后从 Must have 开始写起，直到 Should have ，至于 Could have 看时间是否允许，看情况，比如，看有没有最后期限限制……

在写书这事儿上，我总是给人感觉很快，事实上也是，因为有方法论 —— 但显然，那方法论不是从某一本"如何写书"的书里获得的，而是从另外一个看起来完全不相关的领域里习得后琢磨到的……

所谓的"活学活用"，所谓的"触类旁通"，也不过如此。