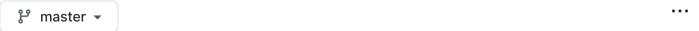
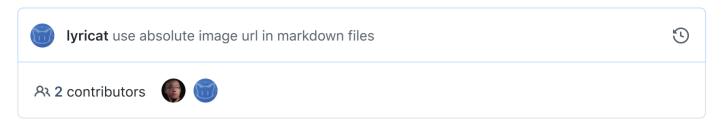
## \$ SchOng / the-craft-of-selfteaching

forked from selfteaching/the-craft-of-selfteaching

Code Pull requests Actions Projects Wiki Security Insights Settings



### the-craft-of-selfteaching / markdown / README.md





159 lines (125 sloc) 7.78 KB

# 

One has no future if one couldn't teach themself<sup>[1]</sup>.

# 自学是门手艺

没有自学能力的人没有未来

作者: 李笑来

特别感谢**霍炬**(@virushuo)、**洪强宁**(@hongqn)两位良师诤友在此书写作过程中给予 我的巨大帮助!

```
# pseudo-code of selfteaching in Python

def teach_yourself(anything):
    while not create():
        learn()
        practice()
    return teach_yourself(another)

teach_yourself(coding)
```

## 目录

- 01.preface (前言)
- 02.proof-of-work (如何证明你真的读过这本书?)
- Part.1.A.better.teachyourself (为什么一定要掌握自学能力?)
- Part.1.B.why.start.from.learning.coding(为什么把编程当作自学的入口?)
- Part.1.C.must.learn.sth.only.by.reading(只靠阅读习得新技能)
- Part.1.D.preparation.for.reading (开始阅读前的一些准备)
- Part.1.E.1.entrance (入口)
- Part.1.E.2.values-and-their-operators (值及其相应的运算)
- Part.1.E.3.controlflow (流程控制)
- Part.1.E.4.functions (函数)
- Part.1.E.5.strings (字符串)
- Part.1.E.6.containers(数据容器)
- Part.1.E.7.files (文件)
- Part.1.F.deal-with-forward-references (如何从容应对含有过多"过早引用"的知识?)
- Part.1.G.The-Python-Tutorial-local (官方教程: The Python Tutorial)
- Part.2.A.clumsy-and-patience (笨拙与耐心)
- Part.2.B.deliberate-practicing (刻意练习)
- Part.2.C.why-start-from-writing-functions (为什么从函数开始?)
- Part.2.D.1-args(关于参数(上))
- Part.2.D.2-aargs (关于参数(下))
- Part.2.D.3-lambda(化名与匿名)
- Part.2.D.4-recursion (递归函数)
- Part.2.D.5-docstrings (函数的文档)
- Part.2.D.6-modules (保存到文件的函数)
- Part.2.D.7-tdd (测试驱动的开发)
- Part.2.D.8-main (可执行的 Python 文件)
- Part.2.E.deliberate-thinking (刻意思考)
- Part.3.A.conquering-difficulties (战胜难点)
- Part.3.B.1.classes-1 (类 —— 面向对象编程)
- Part.3.B.2.classes-2 (类 —— Python 的实现)
- Part.3.B.3.decorator-iterator-generator (函数工具)
- Part.3.B.4.regex (正则表达式)
- Part.3.B.5.bnf-ebnf-pebnf (BNF 以及 EBNF)

- Part.3.C.breaking-good-and-bad (拆解)
- Part.3.D.indispensable-illusion (刚需幻觉)
- Part.3.E.to-be-thorough (全面 —— 自学的境界)
- Part.3.F.social-selfteaching (自学者的社交)
- Part.3.G.the-golden-age-and-google (这是自学者的黄金时代)
- Part.3.H.prevent-focus-drifting (避免注意力漂移)
- Q.good-communiation (如何成为优秀沟通者)
- R.finale (自学者的终点)
- S.whats-next (下一步干什么?)
- T-appendix.editor.vscode (Visual Studio Code 的安装与配置)
- T-appendix.git-introduction (Git 简介)
- T-appendix.jupyter-installation-and-setup (Jupyterlab 的安装与配置)
- T-appendix.symbols(这些符号都代表什么?)

## 关于 .ipynb 文件转换为 .md 文件的备注:

- # 需提前安装 nbconvert 插件, Terminal 下执行:
- \$ jupyter nbconvert --to markdown \*.ipynb

而后将所有 `.md` 文件移到 `markdown/` 目录之下 — 除 `README.md` 文件之外

`README.md` 文件复制一份到 `markdown/` 目录之下,而后编辑为当前文件

# 需使用 VSCode 批量 Find and Replace:

将所有(https://raw.githubusercontent.com/selfteaching/the-craft-of-selfteak所有(Part.1.A.better.teachyourself\_files/替换为(https://raw.githubuserc将所有(Part.1.E.6.containers\_files/替换为(https://raw.githubusercontent.c将所有 ```\n\n 替换为 ```\n
将所有 \n\n`` 替换为 \n```
将所有 .ipynb)替换为 .md)

`Part.1.E.3.controlflow.md` 文件中有过长的 output 需要编辑

`Part.1.E.7.files.md` 文件中有过长的 output 需要编辑

推荐读者在自己的浏览器上安装 Stylus 这类终端 CSS 定制插件,

Chrome/Firefox/Opera 都支持 Stylus 插件。以便拥有更好的阅读体验。以下 gif 图片展示的是使用自定义 css 前后的效果:

### 布尔运算

从定义上来看,所谓程序 (Programs) 其实一点都不神秘。

因为程序这个东西,不过是按照一定*顺序*完成任务的**流程**(Procedures)。根据定义,日常生活中你做盘蛋炒饭给自己吃,也是完成了一个"做蛋炒饭"的程序——你按部就班完成了一系列的步骤,最终做好了一碗蛋炒饭给自己吃——从这个角度望过去,所有的菜谱都是程序……

只不过,菜谱这种程序,编写者是人,执行者还是人;而我们即将要学会写的程序,编写者是人,执行者是计算机 —— 当然,菜谱用自然语言编写,计算机程序由程序员用编程语言编写。

然而,这些都不是最重要的差异——最重要的差异在于计算机能做布尔运算(Boolean Operations)。

于是,一旦代码编写好之后,计算机在执行的过程中,除了可以"按照顺序执行任务"之外,还可以"根据不同情况执行不同的任务",比如,"如果条件尚未满足则重复执行某一任务"。

计算器和计算机都是电子设备,但计算机更为强大的原因,用通俗的说法就是它"**可编程**"(Programable)—— 而所谓可编程的核心就是*布尔运算*及其相应的**流程控制**(Control Flow);没有布尔运算能力就没有办法做*流程控制*;没有流程控制就只能"按顺序执行",那就显得"很不智能"……

#### 布尔值

True

在 Python 语言中,布尔值 (Boolean Value) 用 True 和 False 来表示。

注意:请小心区分大小写 —— 因为 Python 解释器是对大小写敏感的,对它来说, True 和 true 不是一回事。

任何一个逻辑表达式都会返回一个布尔值。

```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
# 请暂时忽略以上两行......
1 == 2
1 != 2
False
```

我用的 Stylus 定制 CSS (针对 github.com) 是这样的:

```
.markdown-body {font-family: "PingFang SC";}
strong {color:#6392BF;}
em {color: #A9312A; font-style: normal !important;}
table {font-size: 95% !important;}
.CodeMirror, pre {font-size: 90%;}
pre {
    padding: 10px 25px;
    background-color: #fafafa;
    border-left: 4px solid #dadada;
    border-radius: 10px;
}
pre code {
    background-color: #fafafa;
}
h1 code,
h2 code,
h3 code,
h4 code,
```

```
p code,
li code,
blockquote p code,
blockquote li code,
td code {
    background-color: #f6f6f6;
    font-size: 90%;
    color:#2e2e2e;
    padding: 4px 4px;
    margin: 0 8px;
    box-shadow: 0px 1px 2px 0px rgba(0,0,0,0.2);
    border-radius: 4px;
}
```

我写的内容里,为了重点突出,特别定制了 strong 和 em 两个元素的显示,让它们以不同的颜色展示;又因为中文并不适合斜体展示,所以,把 em 的 font-style 设定为 normal ......

本书的版权协议为 CC-BY-NC-ND license。



署名-非商业性使用-禁止演绎 3.0 未本地化版本 (CC BY-NC-ND 3.0)

### 脚注

[1]: 'Themselves' or 'themself'?-- Oxford Dictionary

↑ Back to Content ↑