孙焕然 2001210653

**GitHub使用初探**

目录

[第一部分 Git协议 - 1 -](#_Toc31315)

[1.1 Git是什么？ - 1 -](#_Toc20150)

[1.2 Git的功能特性 - 1 -](#_Toc4677)

[1.3 Git协议 - 2 -](#_Toc19937)

[1.3.1 本地协议 - 2 -](#_Toc28384)

[1.3.2 HTTP协议 - 3 -](#_Toc7262)

[1.3.3 SSH协议 - 3 -](#_Toc32423)

[1.3.4 Git 协议 - 4 -](#_Toc751)

[第二部分 GitHub.com新手入门 - 5 -](#_Toc29276)

[2.1 GitHub简介 - 5 -](#_Toc30262)

[2.2 GitHub与Git的关系 - 5 -](#_Toc24438)

[2.3 GitHub.com的使用 - 6 -](#_Toc17344)

[2.3.1 托管代码、历史版本管理 - 6 -](#_Toc31605)

[2.3.2 搜索开源项目 - 8 -](#_Toc6283)

[2.3.3 协同开发 - 9 -](#_Toc24283)

[2.3.4 学习与能力提升 - 9 -](#_Toc13476)

[2.3.5提升自己的影响力 - 9 -](#_Toc29896)

[第三部分 GitHub的桌面应用 - 11 -](#_Toc31954)

[3.1 安装 - 11 -](#_Toc9351)

[3.2 登录 - 11 -](#_Toc25184)

[第四部分 GitHub那些有趣的项目 - 13 -](#_Toc27801)

[4.1 表情包的博物馆 - 13 -](#_Toc6798)

[第五部分 Fork开源项目并自己修改 - 16 -](#_Toc8959)

[5.1 什么是Fork - 16 -](#_Toc10799)

[5.2 Fork使用实践 - 16 -](#_Toc27848)

[第六部分 心得体会 - 19 -](#_Toc12148)

[参考文献 - 20 -](#_Toc20279)

**第一部分 Git协议**

1.1 Git是什么？

首先，需要明确最基本的定义，也就是Git是什么。

通过bing搜索引擎进行检索，在中英文界面的百科界面分别找到了如下解释：

在Wikipedia中，对Git的解释是这样的：Git(/gɪt/) is a distributed version-control system for tracking changes in source code during software development. It is designed for coordinating work among programmers, but it can be used to track changes in any set of files. Its goals include speed, data integrity, and support for distributed, non-linear workflows.

在百度百科中，可以查到其对Git的简要介绍如下：Git（读音为/gɪt/）是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。

综合中外两种阐述，可以对Git有了基本的了解。Git可以说是一个分布式的、开源的版本控制系统，不同用户可以在不同服务器上进行合作。每台计算机上的每个Git目录都是一个成熟的存储库，具有完整的历史记录和完整的版本跟踪能力，有利于用户高效完成各类任务与项目。

1.2 Git的功能特性

从一般开发者的角度来看，Git有以下功能（来自百度百科）：

1、从服务器上克隆完整的Git仓库（包括代码和版本信息）到单机上。

2、在自己的机器上根据不同的开发目的，创建分支，修改代码。

3、在单机上自己创建的分支上提交代码。

4、在单机上合并分支。

5、把服务器上最新版的代码fetch下来，然后跟自己的主分支合并。

6、生成补丁（patch），把补丁发送给主开发者。

7、看主开发者的反馈，如果主开发者发现两个一般开发者之间有冲突（他们之间可以合作解决的冲突），就会要求他们先解决冲突，然后再由其中一个人提交。如果主开发者可以自己解决，或者没有冲突，就通过。

8、一般开发者之间解决冲突的方法，开发者之间可以使用pull 命令解决冲突，解决完冲突之后再向主开发者提交补丁。

1.3 Git协议

Git 可以使用四种不同的协议来传输资料：本地协议（Local），HTTP 协议，SSH（Secure Shell）协议及 Git 协议。

在Git官方网站中，我找到了其对自身四种协议的详细介绍：



1.3.1 本地协议

本地协议（Local protocol）是最基本的协议，其中的远程版本库就是同一主机上的另一个目录。这常见于团队每一个成员都对一个共享的文件系统（例如一个挂载的NFS）拥有访问权，或者比较少见的多人共用同一台电脑的情况。后者并不理想，因为你的所有代码版本库如果长存于同一台电脑，更可能发生灾难性的损失。

如果你使用共享文件系统，就可以从本地版本库克隆（clone）、推送（push）以及拉取（pull）。 像这样去克隆一个版本库或者增加一个远程到现有的项目中，使用版本库路径作为URL。

本地协议具有很多优点。基于文件系统的版本库的优点是简单，并且直接使用了现有的文件权限和网络访问权限。如果当前团队已经有共享文件系统，建立版本库会十分容易。只需要像设置其他共享目录一样，把一个裸版本库的副本放到大家都可以访问的路径，并设置好读/写的权限就可以了。这也是快速从别人的工作目录中拉取更新的方法。如果你和别人一起合作一个项目，他想让你从版本库中拉取更新时，运行类似git pull /home/john/project的命令比推送到服务器再抓取回来简单多了。

但是，本地协议也存在着缺点。这种方法的缺点是，通常共享文件系统比较难配置，并且比起基本的网络连接访问，这不方便从多个位置访问。另外，如果使用的是类似于共享挂载的文件系统，这个协议不一定是速度最快的。

而且这个协议并不保护仓库避免意外的损坏， 每一个用户都有“远程”目录的完整shell权限，这个协议无法阻止他们修改或删除Git内部文件和损坏仓库。

1.3.2 HTTP协议

在Git中，通过HTTP通信会有两种模式，即“智能HTTP协议”和“哑HTTP协议”。在Git 1.6.6 版本之前只有一个方式可用，十分简单并且通常是只读模式的。Git 1.6.6 版本引入了智能HTTP协议，这是一种新的、更智能的协议，让Git可以像通过SSH那样智能的协商和传输数据。之后几年，这个新的 HTTP 协议因为其简单、智能变的十分流行。

下面，简要介绍一下两种不同的HTTP协议。智能HTTP的运行方式和SSH 及Git协议类似，只是运行在标准的HTTP/S端口上并且可以使用各种HTTP验证机制，这意味着使用起来会比SSH协议简单的多，比如可以使用HTTP协议的用户名/密码授权，免去设置SSH公钥。这种协议或许已经是最流行的使用Git 的方式了。

而如果服务器没有提供智能HTTP协议的服务，Git客户端会尝试使用更简单的“哑”HTTP协议。在这种协议中，web服务器仅把裸版本库当作普通文件来对待，提供文件服务。他的优点主要在于其相对简单的设置。

1.3.3 SSH协议

架设Git服务器时常用SSH协议作为传输协议。因为大多数环境下服务器已经支持通过SSH访问——即使没有也很容易架设。SSH协议也是一个验证授权的网络协议；并且，因为其普遍性，架设和使用都很容易。

SSH协议的优点有很多。首先，SSH架设相对简单——SSH守护进程很常见，多数管理员都有使用经验，并且多数操作系统都包含了它及相关的管理工具。 其次，通过SSH访问是安全的——所有传输数据都要经过授权和加密。最后，SSH协议很高效，在传输前也会尽量压缩数据。

SSH协议的缺点在于它不支持匿名访问Git仓库。如果使用SSH，那么即便只是读取数据，使用者也必须通过SSH访问主机，这使得 SSH 协议不利于开源的项目。

1.3.4 Git 协议

最后是Git协议。这是包含在Git里的一个特殊的守护进程；它监听在一个特定的端口（9418），类似于SSH服务，但是访问无需任何授权。要让版本库支持 Git 协议，需要先创建一个git-daemon-export-ok文件——它是 Git 协议守护进程为这个版本库提供服务的必要条件——但是除此之外没有任何安全措施。

目前，Git协议是Git使用的网络传输协议里最快的。如果项目有很大的访问量，或者项目很庞大并且不需要为写进行用户授权，架设Git守护进程来提供服务是不错的选择。

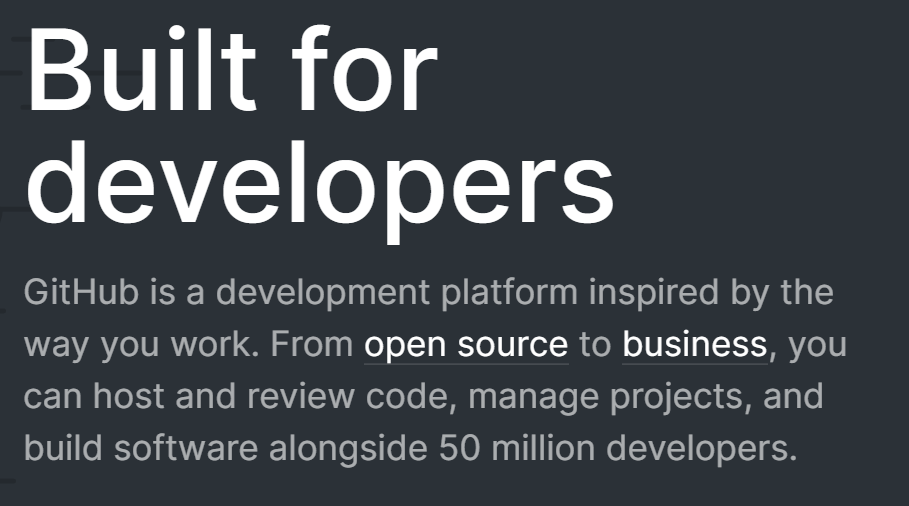
但是，Git 协议缺点是缺乏授权机制。由于没有授权机制，一旦开放推送操作，意味着网络上知道这个项目 URL 的人都可以向项目推送数据。另外，Git 协议也许同时是最难架设的。它要求有自己的守护进程，这就要配置 xinetd、systemd 或者其他的程序，这些工作并不简单。它还要求防火墙开放 9418 端口，但是企业防火墙一般不会开放这个非标准端口。

**第二部分 GitHub.com新手入门**

首先，在开始研究生的学习生活之后，我了解到很多同学，尤其是有过计算机学习的相关经历的同学，都把GitHub作为一个重要的工具。而作为一个GitHub的使用新手，我想通过一个新手的视角，来为大家进行一个GitHub的介绍，同时进行对GitHub一些功能使用的探究。

2.1 GitHub简介

首先，在github的英文网站首页，一打开，我们就可以看到这样一个界面：



我们可以看到，“Develop for developers”十分醒目，显然这是一个适用于一切开发人员的网站。通过搜索引擎进行检索，我们可以总结一下所得：GitHub 是一个面向开源及私有软件项目的托管平台，因为只支持 Git 作为唯一的版本库格式进行托管，故名 GitHub。简单来说，GitHub是一个代码托管云服务网站，帮助开发者存储和管理其项目源代码，且能够追踪、记录并控制用户对其代码的修改。甚至，我们可以简单粗暴的把它当作一个网盘，用来存储任何东西。

GitHub大大提高了开发者的工作效率。

2.2 GitHub与Git的关系

在做这次作业之前，我一直认为GitHub与Git是同一个东西，而开始了解之后才发现，原来两者并不是同一物有两个不同的名字。

Git只是一个命令行工具，一个分布式版本控制系统。正是它在背后管理和跟踪你的代码历史版本，好比一个时光机，让你在代码出错时不至于手忙脚乱，能快速回退之前的历史版本。类似的工具还有SVN。

而GitHub是一个代码托管网站，背后使用Git作为版本管理工具（而非svn）。主要服务是将你的项目代码托管到云服务器上，而非存储在自己本地硬盘上。类似的网站还有gitlab.com，bitbucket.com，coding.com(国内)和gitee.com(国内)等。

简单来说，Git与GitHub两者一个是工具，一个是平台。两者之间的关系可以用这样一句话来进行表述：Git是一张弓，GitHub是靶子，而你的源代码是箭。

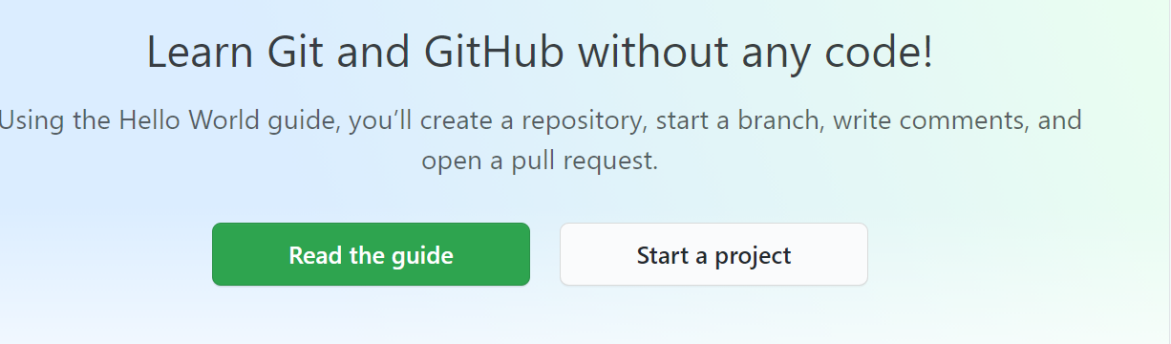
2.3 GitHub.com的使用

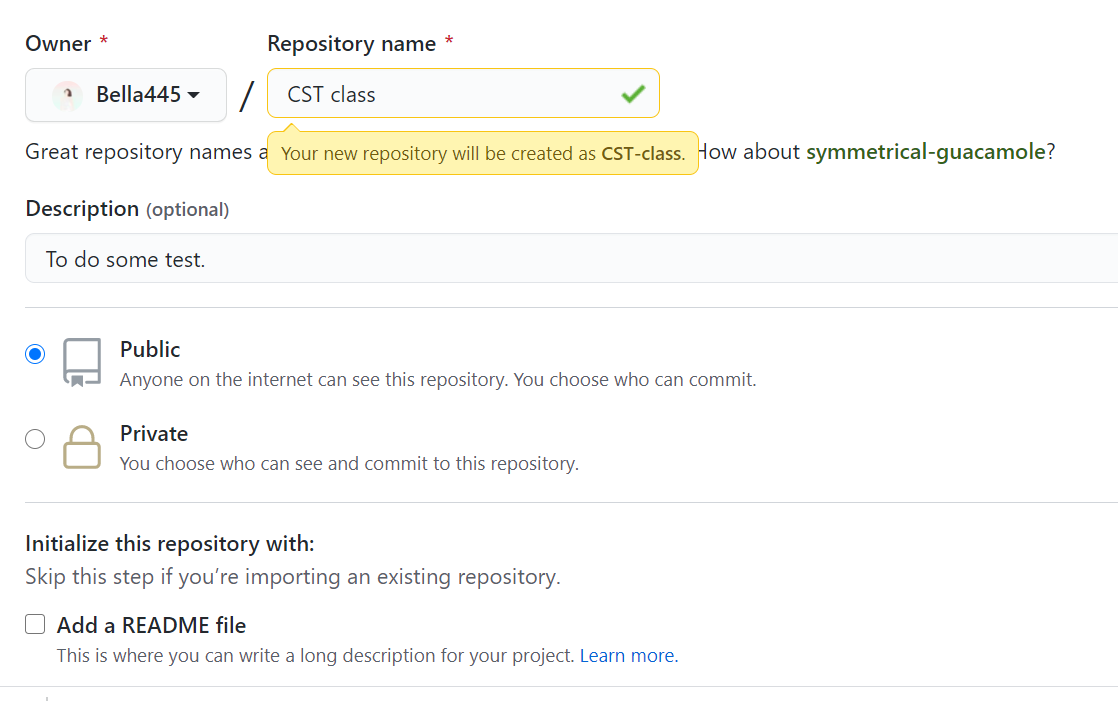
首先，注册并登录GitHub.com。让我来试着探索一下GitHub的强大功能。

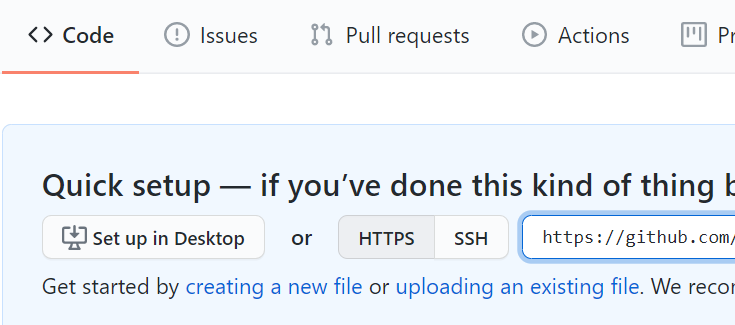
2.3.1 托管代码、历史版本管理

当然不仅仅是代码，任何文件都支持（不少人用GitHub来写博客）。它会自动帮你记录代码的修改，在必要的时候，能够帮助你快速回退到之前的历史版本。

现在，让我来试一下这个功能，看看效果如何。

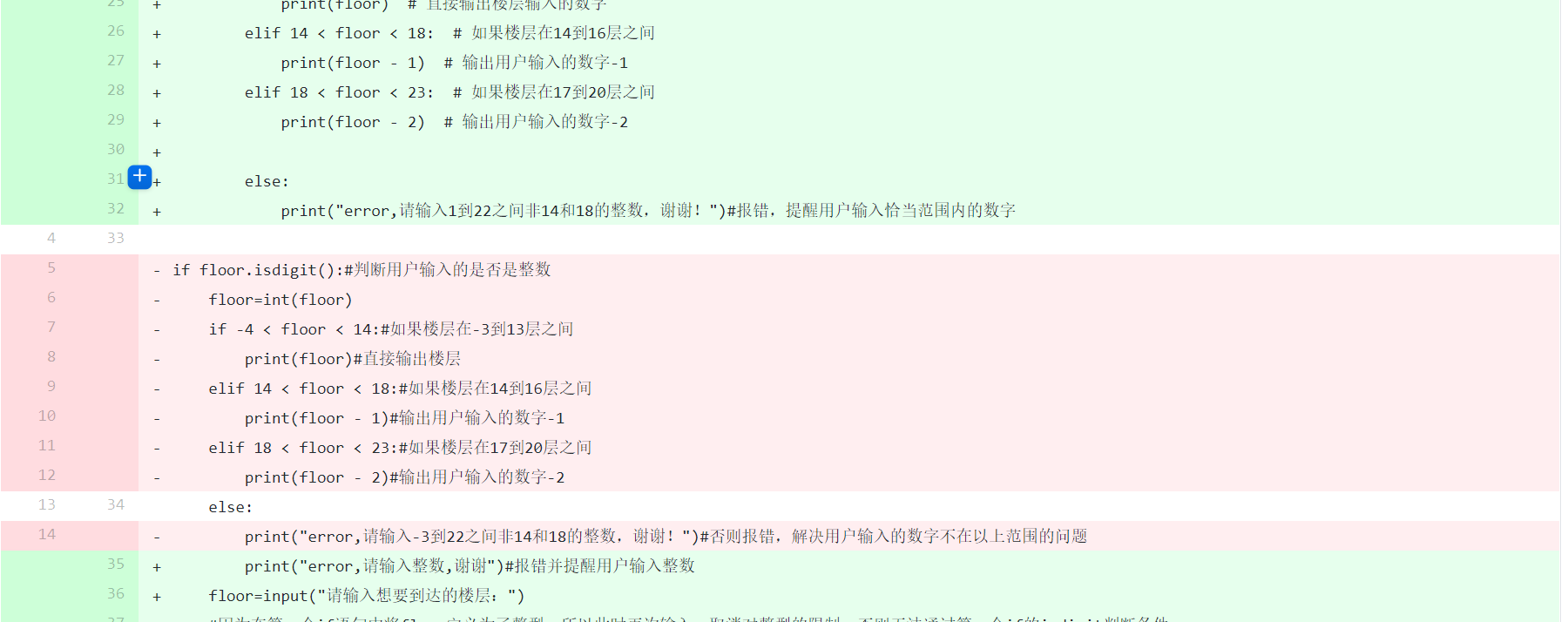
首先，点击“Start a project”，此时会进入这样一个窗口，创建自己的新仓库（repository）。



仓库创建好之后，就可以开始新建或上传代码了。

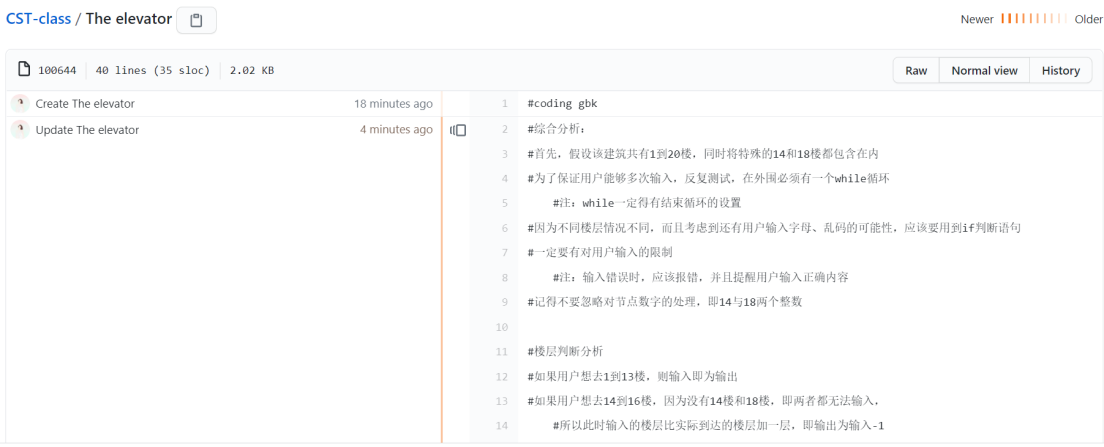
“Creating a new file”，选择创建新文件，如下图：

随后对代码进行改进，如下图：



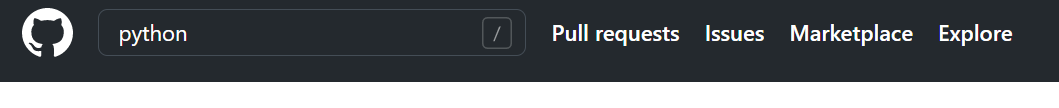
在查看代码时发现，GitHub自动将修改的部分标记为绿色，将未修改部分标记为红色。

同时，在查看界面点击“Blame”按钮，用户就可以查看该项目的全部历史版本，非常便捷直观。

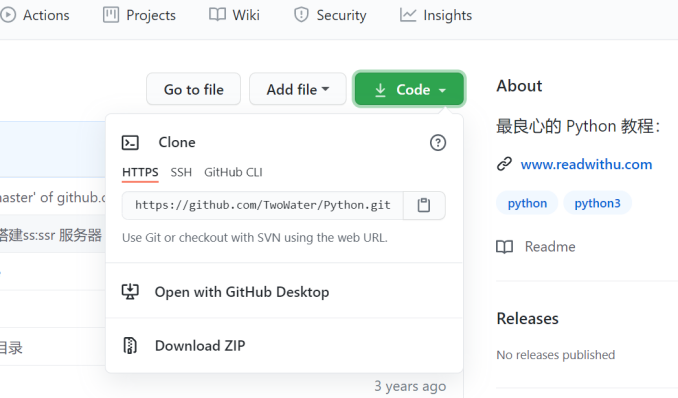


2.3.2 搜索开源项目

GitHub是世界顶级开发者的聚集地，在这里你几乎可以找到任何你想要的东西。上面分享有许多优秀、著名的开源项目，在遵守其License的前提下，你可以免费使用、下载这些开源项目。这也是GitHub最为吸引人的地方。

在GitHub首页检索框中输入“python”进行检索。

结果如下，可以找到大量相关的开源项目。

随意打开一个项目，试着进行下载。

如上图所示，点击右上角的下载符号，选择“Download ZIP”，即可将项目中的你内容下载到本地。

2.3.3 协同开发

GitHub可以说是发动群众力量的最佳实践之一。其一大优势是协同开发，你分享出来的项目，其他开发者也可以参与贡献，帮助你一同完善项目功能，Bug修复，代码质量提升。

关于协同开放，在本报告的后半部分会有详细演示，此处就不再赘述。

2.3.4 学习与能力提升

GitHub上有很多优秀开源项目和教程资源值得你去发现、学习。

比如当前我们在学习python，那么在GitHub中检索“python教程”，会得到大量python教程，由简到难，涉及各个方面，使我们学习编程的一大助力。但是，这些学习资源鱼龙混杂，还需要我们提高自己的鉴别能力，审慎选择学习材料。

2.3.5提升自己的影响力

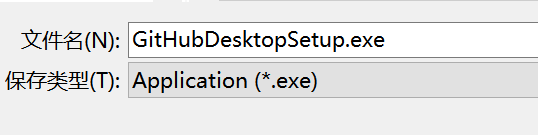
每个人都可以将自己的开源项目发布到GitHub网站上，项目越受欢迎获得的Star越高，说明业界对你的能力越认可。一份优秀的GitHub主页，能为自己的简历润色不少。

**第三部分 GitHub的桌面应用**

3.1 安装

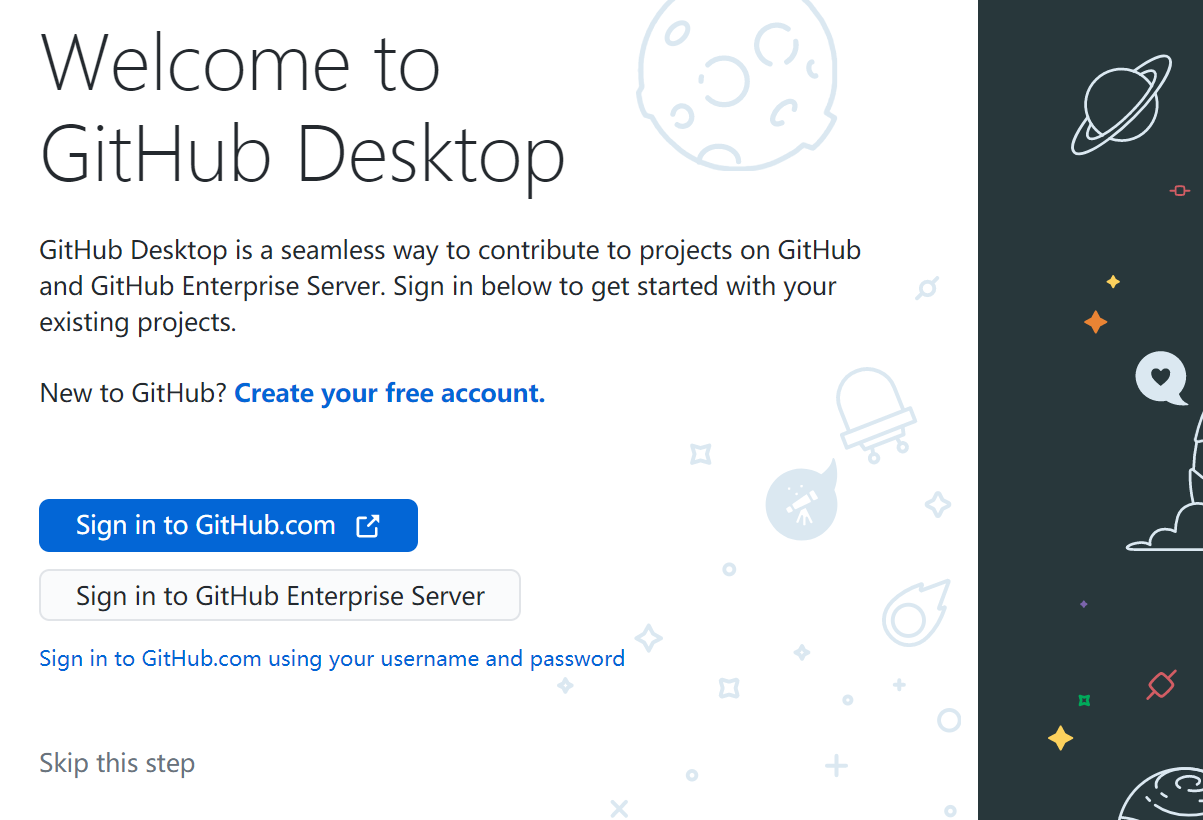
首先去官网https://desktop.github.com/，下载64位的GitHub Desktop，如下图所示：

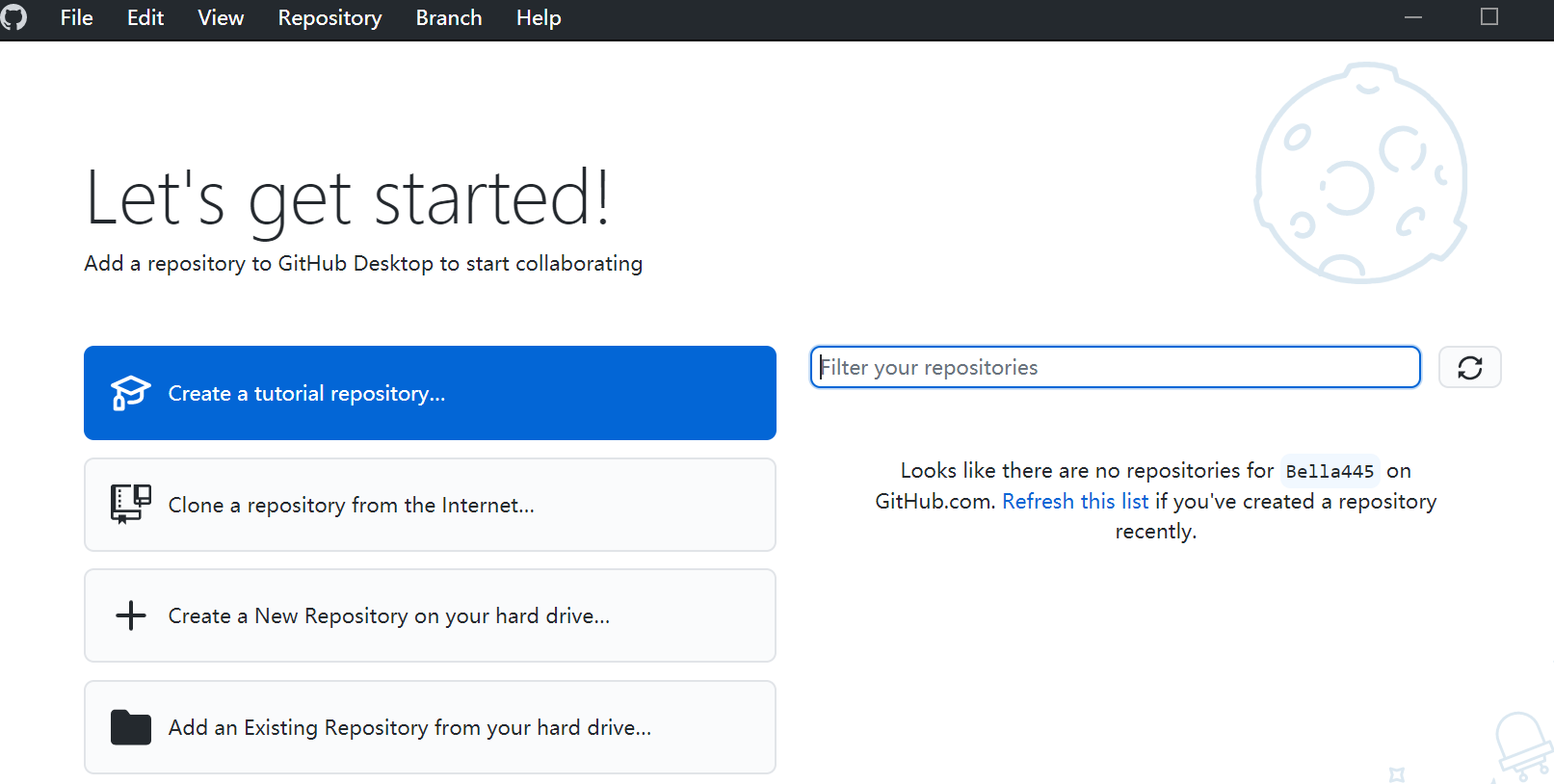




安装包下载好之后，进行安装。

3.2 登录

安装完成后，软件会提醒你注册或登录，点击“Sign in to GitHub.com”进行注册：

进入网页端进行登录，随后在网页端对软件进行授权，就可以成功在软件端进行登录，如下图：

**第四部分 GitHub那些有趣的项目**

GitHub是世界上最大的代码托管平台，超5千万开发者正在使用。鉴于GitHub超高的用户量，在这里能找到各种各样的项目也就不足为奇了，这里我试着去搜索了一些好玩的项目来跟大家进行分享。

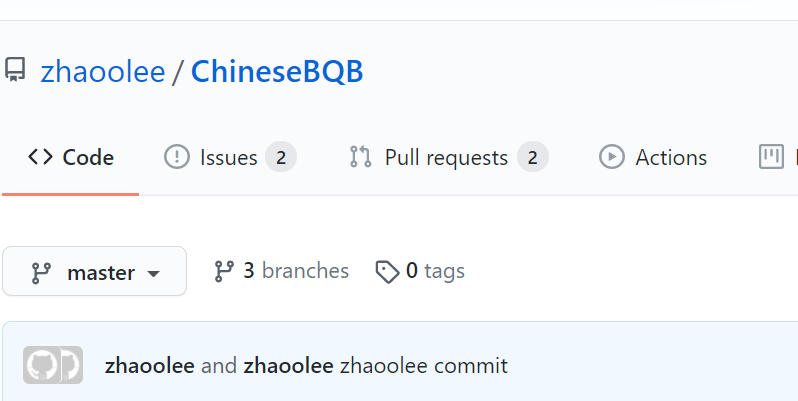
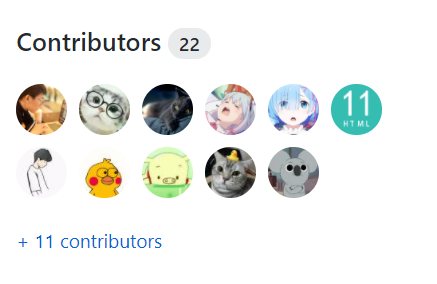
4.1 表情包的博物馆

在检索的时候，我找到了这样一个神奇的项目，叫做“ChineseBQB”，项目的简介是：

Chinese sticker pack, More joy / 表情包的博物馆，Github最有毒的仓库，中国表情包大集合，聚欢乐~

对于当代年轻人来说，表情包不仅是我们线上聊天中不可或缺的一部分，也是我们日常生活中的一大快乐源泉，能在GitHub上挖到这个宝藏项目并且分享给大家，我也感到十分高兴。

首先，我们可以看到该项目的创始人是一个名为“zhaolee”的GitHub用户，同时，有22名用户贡献了他们的contribution。



下面，我将展示一些项目的内容。

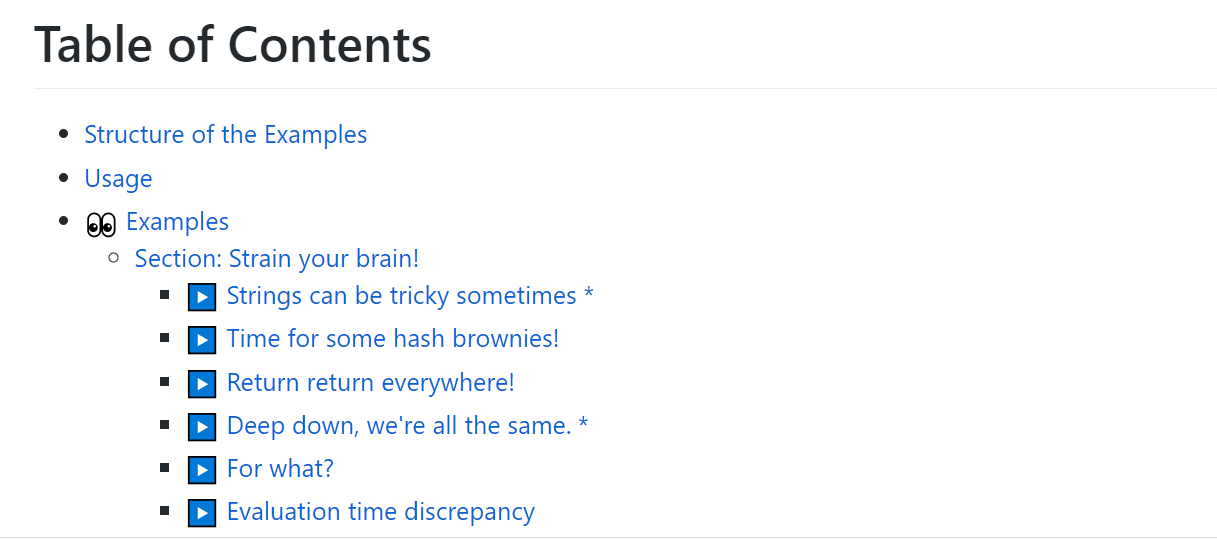


4.2 What the f\*ck Python!

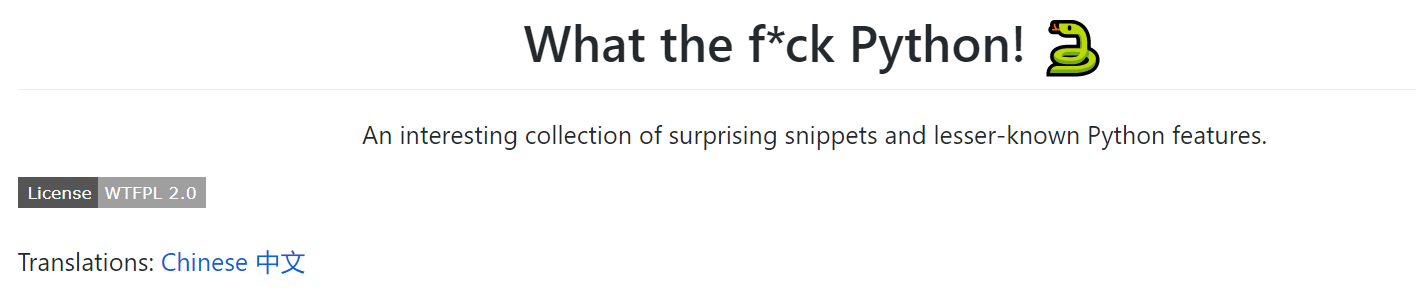
由于近期大家都在学python，于是想分享一个有趣的python项目，在知乎上逛了逛，找到这么一个有趣的项目。

这个项目叫“What the f\*ck Python!”，专门介绍Python里那些奇奇怪怪的语言坑，每个都有示例，目前，该项目已经有了约22万的标星。





同时，一位名为“leisurelicht”的用户对这个项目进行了全文翻译，中英文的两位作者都将另一种语言的链接放在了自己项目的开头位置，方便不同语言的用户进行阅读与学习。

如图，为中英文两个版本的互相指引。

下图是中文版的项目截图，这位译者我很佩服，显然他不知对python这门编程语言很是了解，中英文转换的功底也颇为深厚，语言十分生动。



**第五部分 Fork开源项目并自己修改**

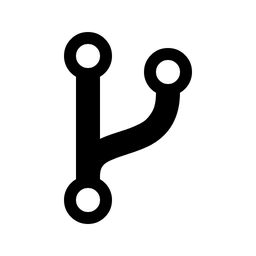
5.1 什么是Fork

首先，我们需要明确什么是fork。

Fork在英文中的本意是“叉子”，比较自然的引申成“分叉（动词）”，就像下面叉子，从左到从右，一条线变成多条了。



在计算机领域中，fork一词又有了新的含义。

此处我们只探究GitHub中fork的含义。GitHub中，会用下面的图来表达 fork，意为“分叉、克隆出一个（仓库的）新拷贝”。

在知乎用户李鼎的回答中，他做出了如下说明：

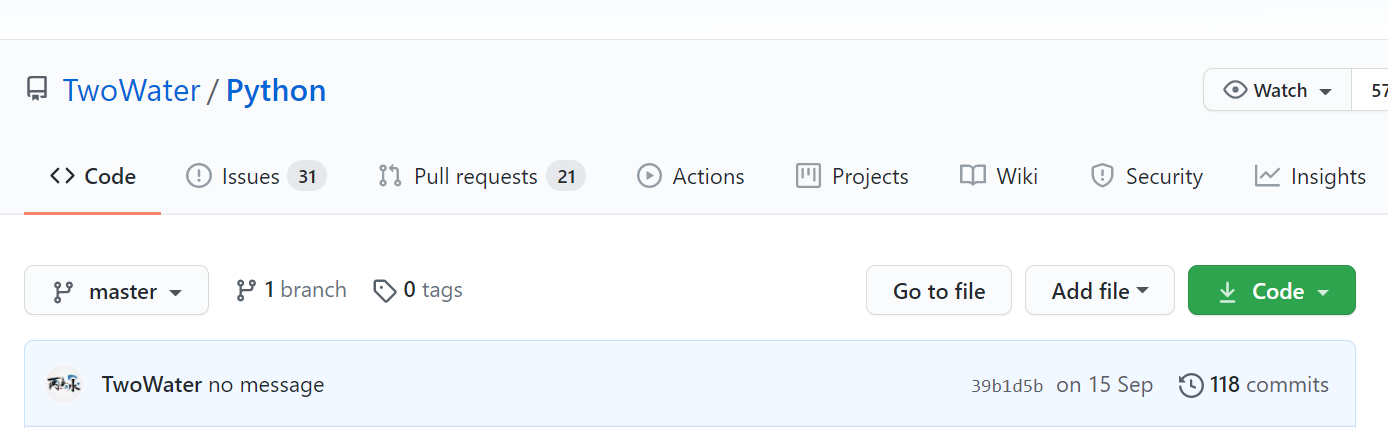
GitHub中Fork是服务端的代码仓库克隆（即新克隆出来的代码仓库在远程服务端），包含了原来的仓库（即upstream repository，上游仓库）所有内容，如分支、Tag、提交。代码托管服务（如Github、BitBucket）提供了方便的完成Fork操作的功能（在仓库页面点一下Fork按钮）。这样有了一个你自己的可以自由提交的远程仓库，然后可以通过的 Pull Request把你的提交贡献回原仓库。而对于原仓库Owner来说，鼓励别人Fork自己的仓库，通过Pull Request给自己的仓库做贡献，也能提高了自己仓库的知名度。

Fork是GitHub的一个核心功能，下面我们来试着使用一下这个功能。

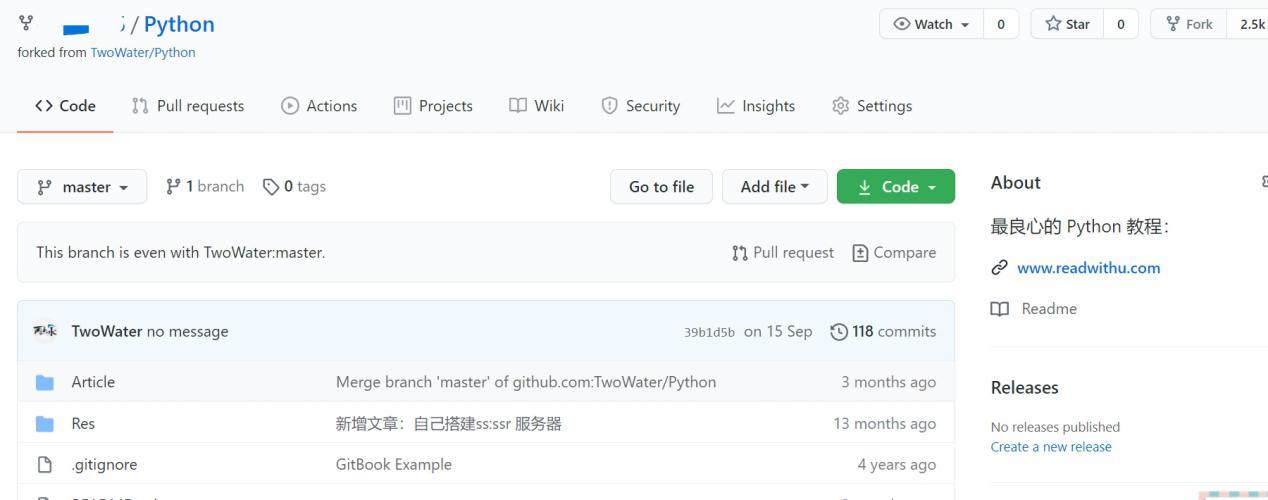
5.2 Fork使用实践

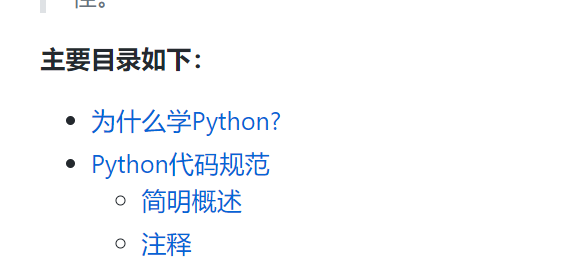
fork功能既然是需要克隆一个原有仓库，那我们肯定需要先找到一个现有的仓库，才能开始fork。由于我们现在正在学习python，所以我在GitHub中检索“python”，希望找到相关教程，并且进行fork。

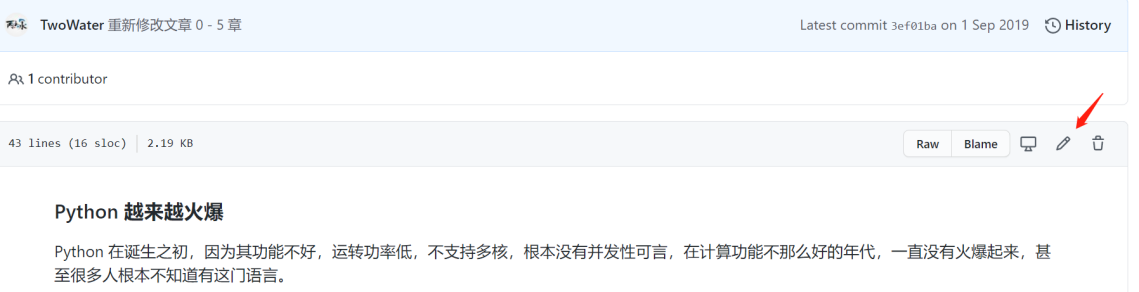
最终，我选择了对用户“TwoWater”的Python教程进行fork，这样一来也可以对后续的python学习带来一定的帮助。



点击仓库右上角的“fork”标志，就可以很轻松的将原有仓库fork到自己的仓库中。

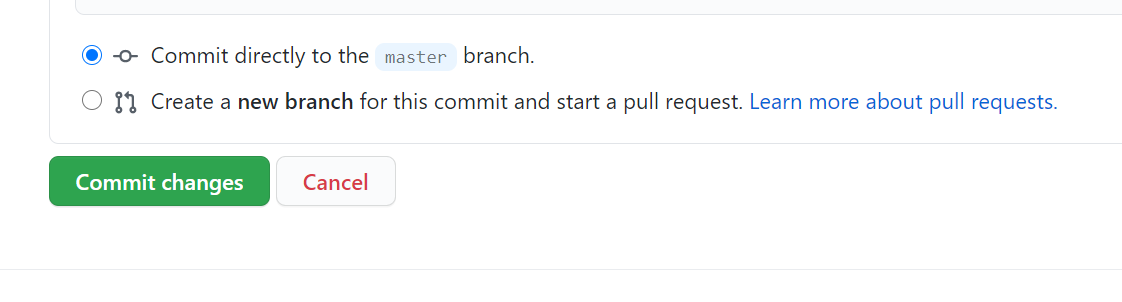


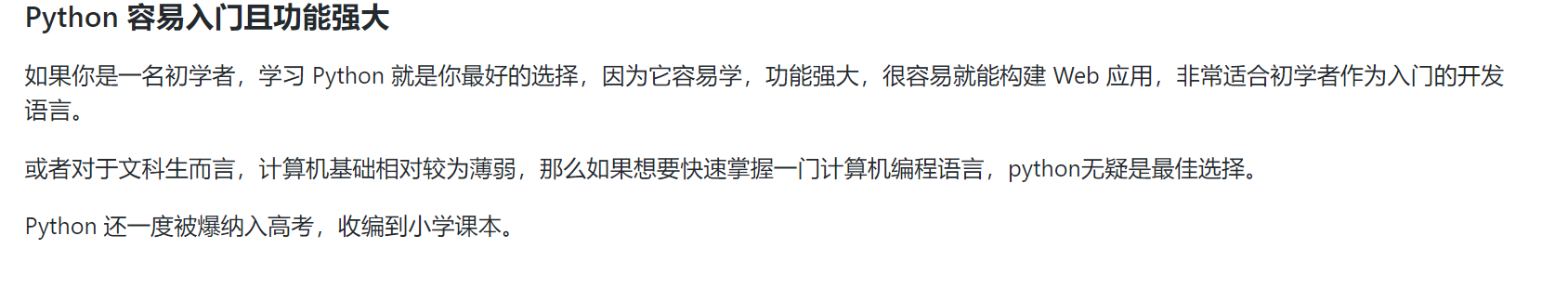
点击进入“为什么学python？”，然后自己尝试修改。

点击右上角的铅笔状图标，便可以开始对此进行编辑修改。

我在27行处增加了一句话，并且保存到自己的仓库。



最后，生成的界面如图：



**第六部分 心得体会**

说实话，在进入研究生的学习阶段之前，作为一名文科生，我很少听说周围同学有使用GitHub，包括我自己。后来，在开学之后，我从各种途径与渠道，了解到很多同学，尤其是技术类的同学很喜欢用GitHub，甚至GitHub已经成为他们生活中非常重要的一个部分。很惭愧，我一直没有对GitHub进行尝试，也没有去查一下相关资料，了解一下这个平台的作用。

通过这次作业，在真正注册并开始使用GitHub之后，我才发现GitHub并不像我想象中一样，只是程序员才会使用，也不是只是一个储存代码的平台。这是一个知识的仓库，或者说我更愿意称之为一个乐园。不同人在这里记录自己的成长，储存与分享自己所掌握的知识、技能与信息等等。由于其开源的特点，这些内容在这个平台上得到了最大化的利用。

我想，这次作业只是一个开端，感谢老师push有拖延症和懒惰病的我开始了自己的GitHub之旅，下一步我会不断探索其他功能，通过这个平台不断提升自我。

本报告中如有不当之处，欢迎大家指正，谢谢！

**参考文献**

1. Wikipedia-Git.

https://en.wikipedia.org/wiki/Git, 访问时间2020年11月3日.

1. 百度百科-Git.

https://baike.baidu.com/item/GIT/12647237, 访问时间2020年11月3日.

1. Git官方网站.

https://git-scm.com/book/zh/v2/. 访问时间2020年11月3日.

1. GitHub中文社区.

https://www.githubs.cn/. 访问时间2020年11月3日.

1. 知乎-GitHub上有哪些优秀的项目？

https://www.zhihu.com/question/20584141. 访问时间2020年11月3日.