实验一 Git和Markdown基础

班级： 21计科2

学号： B20210302219

姓名： 罗天爱

Github地址：https://github.com/linaliaa/python-lianxi.git

实验目的

1. Git基础，使用Git进行版本控制
2. Markdown基础，使用Markdown进行文档编辑

实验环境

1. Git
2. VSCode
3. VSCode插件

实验内容和步骤

第一部分 实验环境的安装

1. 安装git，从git官网下载后直接点击可以安装：git官网地址
2. 从Github克隆课程的仓库：课程的仓库地址，运行git bash应用（该应用包含在git安装包内），在命令行输入下面的命令（命令运行成功后，课程仓库会默认存放在Windows的用户文件夹下）

```
git clone https://github.com/zhoujing204/python_course.git
```

如果你在使用 `git clone` 命令时遇到SSL错误，请运行下面的git命令(这里假设你的Git使用了默认安装目录)：

```
git config --global http.sslCAInfo "C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt"
```

或者运行下面的命令:

```
git config --global http.sslVerify false
```

该仓库的课程材料后续会有更新，如果需要更新课程材料，可以在本地课程仓库的目录下运行下面的命令：

```
git pull
```

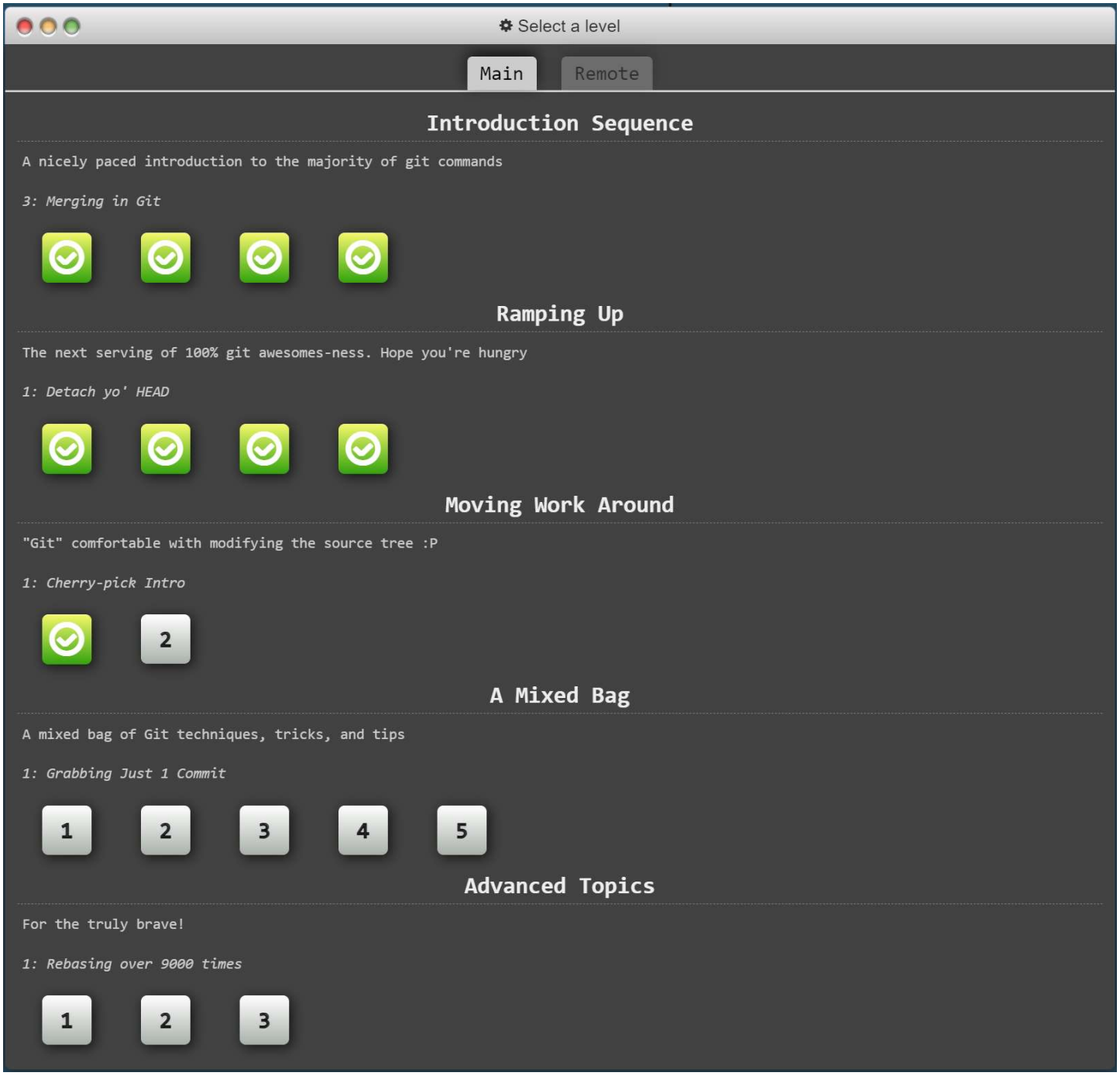
3. 注册Github账号，创建一个新的仓库，用于存放实验报告和实验代码。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件
    - GitLens
    - Git Graph
    - Git History
    - Markdown All in One
    - Markdown Preview Enhanced
    - Markdown PDF
    - Auto-Open Markdown Preview
    - Paste Image
    - markdownlint

第二部分 Git基础

教材《Python编程从入门到实践》P440附录D：使用Git进行版本控制，按照教材的步骤，完成Git基础的学习。

第三部分 learngitbranching.js.org

访问learngitbranching.js.org，如下图所示完成Main部分的Introduction Sequence和Ramping Up两个小节的学习。



上面你学习到的git命令基本上可以应付百分之九十以上的日常使用，如果你想继续深入学习git，可以：

- 继续学习learngitbranching.js.org后面的几个小节（包括Main和Remote）
- 在日常的开发中使用git来管理你的代码和文档，用得越多，记得越牢
- 在git使用过程中，如果遇到任何问题，例如：错误删除了某个分支、从错误的分支拉取了内容等等，请查询git-flight-rules

第四部分 Markdown基础

查看Markdown cheat-sheet，学习Markdown的基础语法

使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括实验过程与结果、实验考查和实验总结，并将其导出为 **PDF格式** 来提交。

实验过程与结果

git基础操作

一：安装Git,配置Git

```
$ git config --global user.name "linaliaa"
$ git config --global user.email "3214178200@qq.com"
```

让git知道我的用户名和电子邮箱地址，必须提供用户名，但可使用虚构的电子邮箱地址

```
$ git config --global init.defaultBranch main
```

上述意味着，使用git管理的每个新项目中，一开始都只有一个分支，该分支名为main

二：创建项目

```
print("Hello Git world!")
```

在系统中创建一个文件夹，命名为git_practice。在这个文件夹中创建一个简单的python程序。

三：忽略文件

```
_pycache_/
```

创建一个名为.gitignore的特殊文件，可以避免混乱，让项目开发起来更容易

四：初始化仓库

```
git_practice$ git init
```

五：检查状态

```
git_practice$ git status
```

六：将文件加入仓库

```
git_practice$ git add .
git_practice$ git status
```

七：执行提交

```
git_practice$ git commit -m "Started project."
git status
```

八：查看提交历史

```
git_practice$ git log
```

每次提交，git都会生成一个独一无二的ID，长度为40个字符，因此git提供了一个选项，让我们可以打印提交历史条目的简单版本：

```
git_practice$ git log --pretty=oneline
```

九：第二次提交

在hello_git.py中再添加一行代码

```
print("Hello Git world!")
print("Hello everyone.")
```

```
git_practice$ git status
```

然后提交所做的修改，并再次查看状态

```
git_practice$ git commit -am "Extended greeting."
git_practice$ git status
git_practice$ git log --pretty=oneline
```

标志-am让git将仓库中所有修改了的文件都加入当前提交
标志-m让git在提交历史中记录一条消息

十：放弃修改

先在hello_git.py中再添加一行代码

```
print("Hello Git world!")
print("Hello everyone.")
print("Oh no,I broke project!")
```

然后保存并运行这个文件

```
git_practice$ git status
```

命令git restore filename能够放弃最后一次提交后对指定文件所做出的所有修改

```
git_practice$ git restore .
git_practice$ git status
```

然后返回hello_git.py发现被修改成这样：

```
print("Hello Git world!")
print("Hello everyone.")
```

十一：检出以前的提交

```
git_practice$ git log --pretty=oneline
git_practice$ git checkout ******
```

(******是ID的前六位)
要回到分支main，可以执行这个

```
git_practice$ git switch -
```

这样就回到分支main了
如果参与项目开发的只有自己，又想放弃最近的所有提交并恢复到以前的状态，也可以将项目重置到以前的提交。为此，可在处于分支main上的情况下，执行以下命令

```
git_practice$ git status
git_practice$ git log --pretty=oneline
git_practice$ git re7set --hard ******
git_practice$ git status
git_practice$ git log --pretty=oneline
```

十二：删除仓库

```
git_practice$ git status
git_practice$ rm -rf .git/
git_practice$ git status
git_practice$ git init
git_practice$ git status
git_practice$ git add .
git_practice$ git commit -m "Starting over."
git_practice$ git status
```

在学习过程中可能遇到很多问题，可以多尝试几次，仔细谨慎看看是不是自己粗心导致的，实在不行可以向老师和同学寻求帮助。

learngitbranching.js.org的学习

第一题：学习 git commit

```
git commit
git commit
```

执行两次 git commit 语句即可通关

第二题：学习分支 git branch

```
git branch bugFix
git checkout bugFix
```

先创建分支再切换分支

第三题：学习分支合并 git merge

```
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git merge bugFix
```

创建bugFix分支并切换到该分支，提交一次再切换到main分支，再提交一次，把bugFix分支修改合入main分支

第四题：学习第二种合并分支的方法

```
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git checkout bugFix
git rebase main
```

先创建bugFix分支，切换到该分支，提交一次，切换到main分支，分支提交一次，切换到bugFix分支，将main分支的当前提交作为父节点，并合并修改点到bugFix分支

第五题：学习提交树上的移动

```
git checkout c4
```

将HEAD切换到提交记录c4上

第六题：学习相对引用^

```
git checkout bugFix^
```

第七题：学习相对引用~

```
git branch -f main c6
git checkout HEAD^
git branch -f bugFix HEAD^
```

强制修改分支的-f参数可以使分支指向另一个提交

第八题：学习撤销变更

```
git rest HEAD~1
git checkout pushed
git revert HEAD
```

git reset通过把分支记录回退几个提交记录来实现撤销改动--“改写历史”；为了撤销更改并分享给别人，我们需要使用git revert

注意：不要使用截图，Markdown文档转换为Pdf格式后，截图可能会无法显示。

实验考查

请使用自己的语言回答下面的问题，这些问题将在实验检查时用于提问和答辩，并要求进行实际的操作。

1. 什么是版本控制？使用Git作为版本控制软件有什么优点?

答：版本控制是一种在软件开发过程中用于管理对文件、目录或工程等内容的修改历史，方便查看更改历史记录，备份以便恢复以前的版本的软件工程技术。

Git会跟踪我们对文件所做的更改，我们可以利用他记录已完成的工作，并且在需要时恢复到特定或以前的版本。Git还使多人协作变得更加容易，允许将多个人的更改全部合并到一个源中。

2. 如何使用Git撤销还没有Commit的修改？如何使用Git检出（Checkout）已经以前的Commit？（实际操作）

答：命令git restore filename可以放弃最后一次提交后对指定文件所做的修改，命令git restore则放弃最后一次提交后对所有文件所做的修改，从而将项目恢复到最后一次提交的状态。
检出以前的提交，可以使用命令checkout，并指定该提交的引用ID前6个字符。即

```
git log --pretty=oneline
git checkout ******
```

（******）为引用ID的前6个字符。

3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）
答：HEAD是一个对当前所在分支的符号引用，也就是指向你正在其基础上进行工作的提交记录。
HEAD 总是指向当前分支上最近一次提交记录。
detached HEAD状态（游离状态）是指当前的HEAD指针指向的不是一个分支，而是一个单独的提交。如果让HEAD文件指向一个commit id，那就变成了detached HEAD。使用git checkout可以达到这个效果。

4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）
答：在版本控制的过程中，使用多条线同时推进多个任务。多条线就是分支》
使用命令

```
git branch 分支名称     //创建分支
```

使用命令

```
git checkout 要切换的分支名    //切换分支
```

5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作）
答：1.使用merge命令合并分支。2.使用rebase命令合并分支。
区别：merge就是合并代码，这种方式合并代码后，合并的结果会产生一个新的commit。Rebase（变基）实际上就是取出一系列的提交记录，“复制”它们，然后在另外一个地方逐个的放下去。Rebase 的优势就是可以创造更线性的提交历史。
merge：会产生一个新的节点，与之间的提交分开显示；
rebase：操作不会产生新的节点，是将两个分支融合成一个线性的提交。

6. 如何在Markdown格式的文本中使用标题、数字列表、无序列表和超链接？（实际操作）
答：1.标题：语法：“#”+“空格”+“标题内容”
2.数字列表：数字+.
3.无序列表：语法：连接符+空格
4.超链接：语法：[名字] (地址) 例：百度--------注：方括号与括号之间没有空格。

实验总结

总结一下这次实验你学习和使用到的知识，例如：编程工具的使用、数据结构、程序语言的语法、算法、编程技巧、编程思想。
答：在此次学习过程中，我了解到了版本控制软件git的基础用法，学会了如何安装gity，以及如何使用它来对当前开发的程序进行版本控制，了解了一些基础的操作。同时学习了markdown基础，学会使用markdown进行文档编辑。接下来还会继续学习。