实验一 Git和Markdown基础

班级： 21计科4

学号： B20210302426

姓名： 陈佩儿

Github地址：https://github.com/shaliey/python_course

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
```

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
```

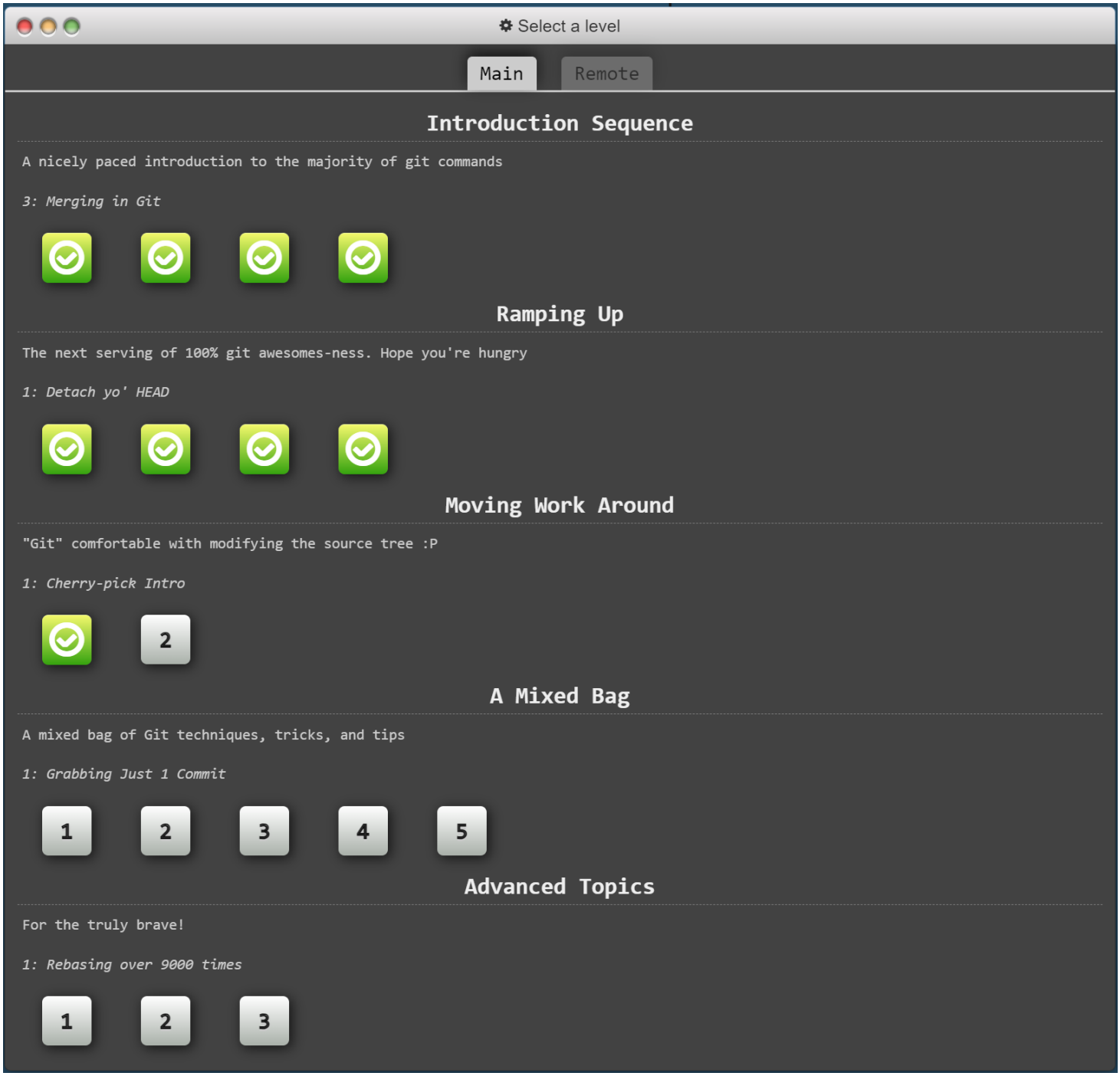
3. 注册Github账号，创建一个新的仓库，用于存放实验报告和实验代码。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件
    - GitLens
    - Git Graph
    - Git History
    - Markdown All in One
    - Markdown Preview Enhanced
    - Markdown PDF
    - Auto-Open Markdown Preview
    - Paste Image
    - markdownlint

第二部分 Git基础

教材《Python编程从入门到实践》P436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

- 第三部分 learngitbranching.js.org

1.
```
git commit
git commit
```

2.
```
git branch bugFix
git checkout bugFix
```

3.
```
delay 2000
show goal
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git merge bugFix
```

4.
```
delay 2000
show goal
git branch bugFix
git checkout bugFix
git commit
git checkout main
git checkout main
git commit
git rebase main
```
分支已经是最新啦

5.
```
delay 2000
show goal
git checkout C4
objective
git checkout bugFix
git rebase main
```

```
6.
delay 2000
show goal
git checkout c4
git checkout HEAD^


7.
delay 2000
show goal
show solution
reset --forSolution
git branch -f main C6
git checkout HEAD~1
git branch -f bugFix HEAD~1


8.
delay 2000
show goal
git recet head
The command "git recet head" isn't supported, sorry!
show solution
reset --forSolution
git reset HEAD~1
git checkout pushed
git revert HEAD
```

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?**
   版本控制是一种管理文件和代码变更历史的系统。它跟踪文件的修改、添加和删除，使团队能够协同工作，追踪变更历史，解决冲突，并轻松恢复到先前的状态。Git是一个广泛使用的版本控制工具，它的优点包括：
   - 分布式：每个开发者都有完整的代码仓库，可以在本地工作，不需要持续连接到中央服务器。
   - 高效：Git在本地存储完整的历史记录，因此操作速度快。
   - 强大的分支管理：Git允许创建、合并和管理分支，使并行开发变得简单。
   - 安全性：Git使用哈希值来确保数据完整性，防止数据损坏。
   - 社区支持：Git有一个庞大的用户社区和丰富的文档资源。
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）**

- 撤销还没有Commit的修改：

```
git checkout -- <文件名>
```

- 检出已经以前的Commit：

```
git checkout <Commit哈希>
```

3. **Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）**
   - HEAD是Git中的特殊指针，它表示当前工作目录所在的位置，通常指向最新的Commit。
   - 让HEAD处于detached HEAD状态：

```
git checkout <Commit哈希>
```

4. **什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）**
   - 分支是Git中的一个独立的开发线，允许并行开发不同的功能或修复不同的Bug。
   - 创建分支：

```
git branch <分支名>
```

   - 切换分支：

```
git checkout <分支名>
```

5. **如何合并分支？git merge和git rebase的区别在哪里？（实际操作）**
   - 合并分支：

```
git merge <目标分支>
```

   - 区别：
     - `git merge` 将目标分支的更改合并到当前分支，并创建一个新的合并Commit。这会保留分支历史。
     - `git rebase` 将当前分支的更改“挪动”到目标分支的最后，看起来像是线性的提交历史。这可以保持更干净的历史记录，但要小心使用，因为它会改变Commit的哈希值。
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）**
   - 标题使用 `#` 符号，例如：`# 这是一个标题`。
   - 数字列表使用数字和句点，例如：`1. 第一项`。
   - 无序列表使用 `-` 或 `*` 符号，例如：`- 无序项`。
   - 超链接使用 `[链接文本](链接URL)`，例如：`[点击这里](https://www.example.com)`。

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。

在这次实验中，我学习和使用了许多与编程和计算机科学相关的知识和技能。以下是我在这次实验中学到和使用到的主要方面：

1. **版本控制和Git**：我学习了版本控制的基本概念，以及如何使用Git来跟踪和管理代码的变更历史。我了解了Git的优点，如分布式特性、分支管理、高效性等，并学会了使用Git进行撤销、检出以前的Commit、创建分支、切换分支和合并分支。
2. **命令行操作**：在使用Git和其他编程工具时，我积累了更多的命令行操作经验。这包括文件和目录操作、Git命令、以及通过命令行界面进行各种任务。
3. **Markdown语法**：我学会了使用Markdown语法来格式化文本，包括创建标题、数字列表、无序列表和超链接。Markdown是写文档和博客文章的常用工具。