实验一 Git和Markdown基础

班级： 21计科04

学号： B20210302404

姓名： 刘卓涵

Github地址：https://github.com/dawn1234ar/python-homework

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
```

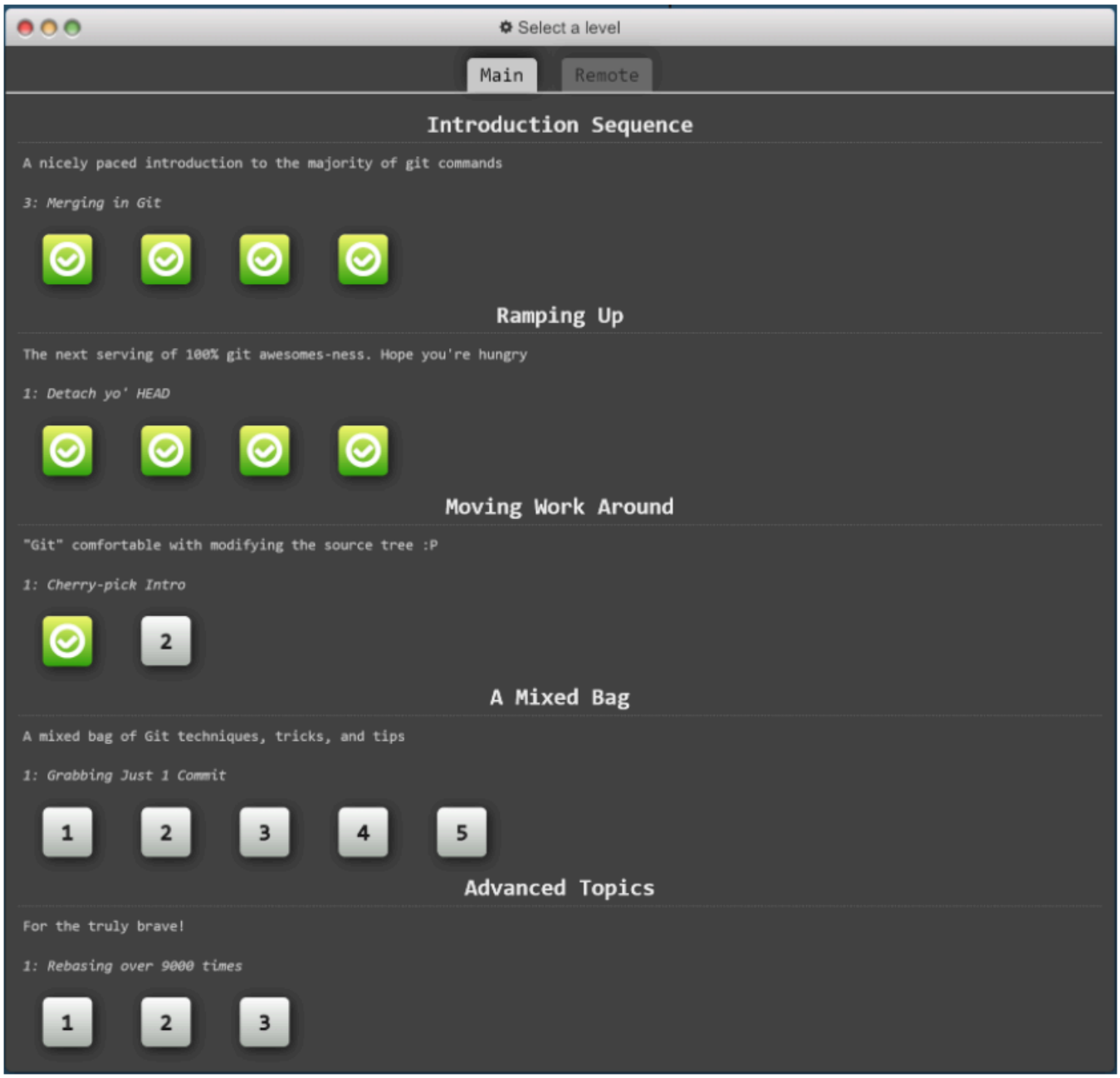
3. 注册Github账号，创建一个新的仓库，用于存放实验报告和实验代码。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件
    - GitLens
    - Git Graph
    - Git History
    - Markdown All in One
    - Markdown Preview Enhanced
    - Markdown PDF
    - Auto-Open Markdown Preview
    - Paste Image
    - markdownlint

第二部分 Git基础

教材《Python编程从入门到实践》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

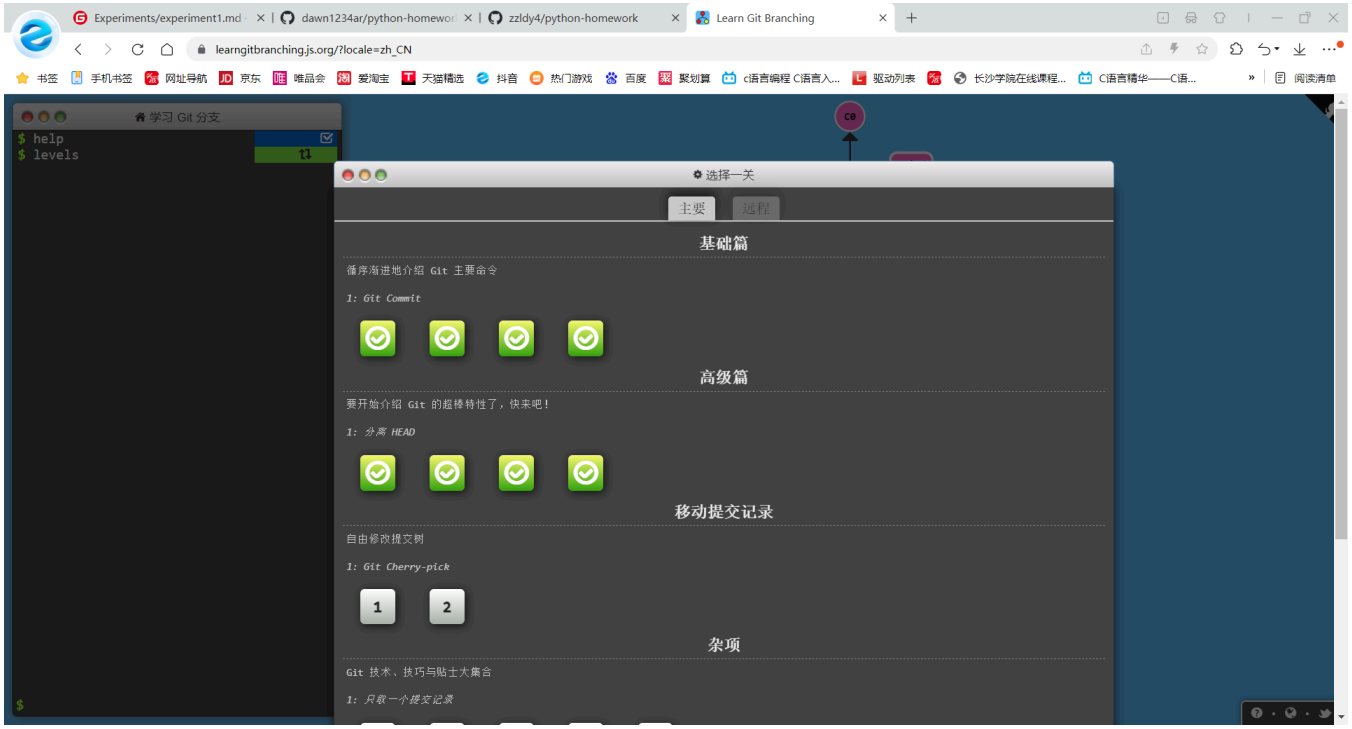
查看Markdown cheat-sheet，学习Markdown的基础语法

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

请将实验过程中编写的代码和运行结果放在这里，注意代码需要使用markdown的代码块格式化，例如Git命令行语句应该使用下面的格式：

Git命令



基础篇

1. 提交

```
- git commit
- git commit
```

2. 创建分支

```
- git branch bugFix  //创建分支
- git commit
- git checkout bugFix  //切换分支
- git commit
- git checkout -b bugFix  //创建并切换
```

3. 合并方法一 git merge

```
git checkout -b bugFix //创建并切换
git commit
git checkout main //切换main分支
git commit
git merge bugFix 合并到main分支
```

4. 合并方法二 git rebase

```
git checkout -b bugFix
git commit
git checkout main //切换main分支
git commit
git checkout bugFix
git rebase main //合并到main
```

高级篇

1. 分离head

```
git checkout c4 //head指向c4
```

2. 相对引用

- 使用 ^ 向上移动 1 个提交记录
- 使用 ~ 向上移动多个提交记录，如 ~3

```
git checkout bugFix^
```

3. 相对引用2

```
git branch -f main C6
git checkout C1
git branch -f bugFiX HEAD^
```

4. 撤销变更

两种方法用来撤销变更:
1.git reset(本地)，2.git revert(远程)
pushed 是远程分支，local 是本地分支

```
git reset HEAD
git checkout pushed
git revert HEAD
```

显示效果如下:
```bash
git init
git add .
git status
git commit -m "first commit"
```

如果是Python代码，应该使用下面代码块格式，例如：

显示效果如下：

```
def add_binary(a,b):
    return bin(a+b)[2:]
```

代码运行结果的文本可以直接粘贴在这里。

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？

```
一个文件修改一次后就是一个新的版本，而版本控制是记录和查看每个版本，并能恢复到任
意一个版本。优点：git采用分布式版本控制，能实时的在本地计算机上更新代码，不需要
等待他人批准和中央服务器响应。
```

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）

```
使用git checkout +文件名.后缀指令能将文件回退到最近一次提交（未提交的修改）；
git checkout example.txt //撤销单个文件
git checkou .//撤销全部
git checkout abc123// commit的哈希值
```

3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）

```
HEAD是当前分支引用的指针，它总是指向某次提交。分离HEAD是让其指向某个具体的提交而不是分支。
分离HEAD：git checkout +Commit 的哈希值
```

4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）

```
branch（分支）是指开发者在Git仓库上创建的一个副本，用于并行开发不同的功能或版本，
从而避免直接修改主分支带来的风险。
git branch +分支名，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）

```
使用git merge +分支名，或git rebase+分支名；git merge 将源分支的所有更改合并为
一个新的提交，这个新提交有两个父提交，分别指向合并之前的两个分支。git rebase 将
源分支的提交逐个应用在目标分支的顶部，使它们看起来像是在目标分支上连续提交的。
git merge <branch_name>
```

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

```
一个#代表一级（最多6个），一级是最大的
# 一级标题 {#一级标题 }
## 二级标题 {#二级标题 }
### 三级标题 {#三级标题 }
```

- 数字列表

```
1. 第一项
2. 第二项
3. 第三项
```

- 无序列表

```
- 第一项
- 第二项
- 第三项
```

- 超链接

```
[链接文字](链接地址)
```

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。

- 本次实验我学习了git的基本指令和写Markdown格式的文本，并对vscode的使用也有了初步了解。