实验一 Git和Markdown基础

班级： 21计科03

学号： B20210302322

姓名： 董一浩

Github地址：https://github.com/ByL1eng/xuexi

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：

显示效果如下：

```
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：

Python代码

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？

什么是版本控制

答：版本控制是软件配置管理的核心功能。所有置于配置库中的元素都应自动予以版本的标识，并保证版本命名的唯一性。版本在生成过程中，自动依照设定的使用模型自动分支、演进。除了系统自动记录的版本信息以外，为了配合软件开发流程的各个阶段。还需要定义、收集一些元数据来记录版本的辅助信息和规范开发流程，并为今后对软件过程的度量做好准备。当然如果选用的工具支持，这些辅助数据将能直接统计出过程数据，从而方便软件过程改进活动的进行。对于配置库中的各个基线控制项，应该根据其基线的位置和状态来设置相应的访问权限。一般来说，对于基线版本之前的各个版本都应处于被锁定的状态，如需要对它们进行变更，则应按照变更控制的流程来进行操作

使用Git作为版本控制软件有什么优点

答：Git是分布式的版本控制系统，这意味着它不会像CVS、SVN那样使用一个中心服务器来保存所有版本库

2.什么是Git，Git和SVN的区别在哪里

什么是Git

答：Git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。Git是Linux内核项目下使用的版本管理软件，使用Git可以有效、高速地处理从很小到非常大的项目版本管理。

Git和SVN的区别在哪里

答： Git是分布式的，SVN不是

3. 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）

如何使用Git撤销还没有Commit的修改

答：首先，使用git status命令查看当前工作区状态，然后使用git checkout命令撤销修改。

如何使用Git检出（Checkout）已经以前的Commit？（实际操作）

答：首先，使用git status命令查看当前工作区状态，然后使用git checkout命令撤销修改。

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

Git中的HEAD是什么

答： HEAD是一个指针，指向当前分支，HEAD指向的分支就是当前分支。

如何让HEAD处于detached HEAD状态

答： 首先，使用git status命令查看当前工作区状态，然后使用git checkout命令撤销修改。

5. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）

什么是分支（Branch）

答：分支是用来将不同的开发工作分离开来，从而避免开发过程中产生冲突。

如何创建分支

答：使用git branch命令创建分支。

何切换分支？（实际操作）

答：使用git checkout命令切换分支。

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

如何合并分支

答：使用git merge来合并分支。

git merge和git rebase的区别在哪里？（实际操作）

答：区别在于git rebase能够在合并时将分支记录为线性，而git merge不能。

7. 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：在Markdown格式的文本中使用标题、数字列表、无序列表和超链接，可以使用Markdown语法。例如，使用#号表示一级标题，使用1.表示有序列表，使用*表示无序列表，使用表示超链接。

实验总结

在本次实验中我学习了Git的基本操作，包括Git的安装、Git的基本操作、Git中的HEAD、分支、合并等。通过本次实验，我掌握了Git的基本操作，对Git有了更深的理解，还学习了关于markdown