实验一报告

实验过程和结果

Git基础：

初始化一个新的Git仓库

使用以下命令初始化一个新的Git仓库：

```
git init
```

克隆现有仓库

使用以下命令克隆一个现有的Git仓库到本地：

```
git clone <repository_url>
```

配置Git用户信息

配置Git用户的全局用户名和邮箱地址：

```bash
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
```

查看仓库状态

查看当前Git仓库的状态：

```
git status
```

添加文件到暂存区

将文件添加到Git的暂存区：

```
git add <file_name>
```

提交更改

提交暂存区的更改到仓库：

```
git commit -m "Commit message"
```

查看提交历史

查看仓库的提交历史记录：

```
git log
```

创建分支

创建一个新的分支：

```
git branch <branch_name>
```

切换分支

切换到一个已存在的分支：

```
git checkout <branch_name>
```

合并分支

将一个分支的更改合并到当前分支：

```
git merge <branch_name>
```

拉取远程更改

从远程仓库拉取最新的更改：

```
git pull
```

推送更改到远程仓库

将本地分支的更改推送到远程仓库：

```
git push origin <branch_name>
```

learngitbranching.js.org部分方法：

- Git commit：
  创建提交记录

```
git commit
git commit
```

- Git Brach：
  创建这个bugfix分支，然后选中

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

- Git Merge：
  如果有两个分支main和bugFix,且当前分支为main,那么使用git merge bugFix就会使main分支commit一次，并且commit后的版本会合并bugFix中的新功能

```
git checkout -b bugFix
git commit
git checkout master
git commit
git merge bugFix
```

- Git Rebase：
  创建bugFix分支并切换到该分支，bugFix分支提交一次，切换到main分支，main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

```
git checkout -b bugFix
git commit
git checkout master
git commit
git checkout bugFix
git rebase master
```

- 分离HEAD ： 将HEAD切换到提交记录C4上

```
git checkout c4
```

- 相对引用(^):
  选中bugfix，往上跳一格

```
git checkout bugFix^
```

- 相对引用2(~)：用数字的方式往上移，-f是强制修改分支的-f参数可以使分支指向另一个提交

```
git branch -f master c6
git branch -f bugFix c0
git checkout c1
```

- 撤销变更:
  通过把分支记录回退几个提交记录来实现撤销改动

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

Markdown基础

当您想要学习Markdown文档的基础操作时，以下是一些常用的Markdown语法示例:

1. 标题

Markdown中可以使用 # 符号来表示标题，标题的级别通过 # 的数量来确定，例如:

```
#  这是一级标题
<p class="crossnote-header " id="这是一级标题"></p>

##  这是二级标题
<p class="crossnote-header " id="这是二级标题"></p>

###  这是三级标题
<p class="crossnote-header " id="这是三级标题"></p>
```

2. 列表

Markdown支持有序和无序列表，使用 `*` 或 `+` 或 `-` 表示无序列表，使用数字和点表示有序列表，例如：

无序列表：

```
- 项目1
- 项目2
- 项目3
```

或

```
* 项目1
* 项目2
* 项目3
```

有序列表：

```
1. 第一项
2. 第二项
3. 第三项
```

3. 链接与图片

插入链接和图片的语法如下：

链接：

```
[Github](https://github.com/)
```

图片：

```
![图片描述](链接到图片的URL)
```

4. 引用块

使用 `>` 符号表示引用块，例如：

```
> 这是一个引用块。
```

5. 代码块

使用三个反引号（```)来创建代码块，还可以指定代码的语言，例如：

```
```python
def hello_world():
    print("Hello, world!")
```

6. 粗体与斜体

使用 `**` 或 `__` 包围文本可以将其设置为粗体，使用 `*` 或 `_` 包围文本可以将其设置为斜体，例如：

```
**这是粗体文本**
*这是斜体文本*
```

7. 表格

创建表格的语法如下：

```
| 表头1 | 表头2 |
|-------|-------|
| 单元格1 | 单元格2 |
| 单元格3 | 单元格4 |
```
```

这些是Markdown文档的基础操作示例，可以根据需要将它们组合使用以创建格式清晰的文档和报告。Markdown是一种简单而强大的文本标记语言，适用于许多不同的文档编辑任务。

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?
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）
3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）
4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）
5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作）
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）

•

1. 版本控制是一种管理文件和代码变更历史的方法，它允许开发者跟踪文件的修改、恢复到先前的状态、合并多个开发者的工作，以及记录和比较不同版本的变更。Git是一种分布式版本控制系统，具有以下优点：
   - 分布式架构：每个开发者都可以拥有完整的代码仓库，不需要依赖中央服务器。
   - 高效性能：Git对大型项目和快速提交有很好的性能支持。
   - 强大的分支管理：Git支持轻松创建和合并分支，使并行开发变得容易。
   - 安全性：Git使用SHA-1哈希值来确保数据完整性，防止损坏或篡改。
   - 灵活性：Git允许用户自定义工作流程和操作。
   - 社区支持：Git具有广泛的社区支持和丰富的文档资源。
2. 若要撤销尚未提交（uncommitted）的修改，可以使用以下命令：
   - 撤销所有未提交的修改，恢复到上次提交的状态：

     ```
     git reset --hard HEAD
     ```

   若要检出以前的提交，可以使用以下命令：
   - 检出特定提交的SHA-1哈希值，创建一个分离头（detached HEAD）状态：

     ```
     git checkout <commit_SHA-1>
     ```

3. Git中的HEAD是一个指向当前工作树（working tree）或分支（branch）的指针。要使HEAD处于分离头状态，可以使用以下命令：
   - 检出特定提交的SHA-1哈希值，而不是分支名称，会导致HEAD处于分离头状态：

```
git checkout <commit_SHA-1>
```

4. 分支是Git中的一个重要概念，它允许开发者在不影响主要代码线的情况下并行开发或测试新功能。要创建新分支并切换到它，可以使用以下命令：
   - 创建新分支：

     ```
     git branch <branch_name>
     ```

   - 切换到新分支：

     ```
     git checkout <branch_name>
     ```

5. 合并分支的方法有两种，分别是git merge和git rebase：
   - 使用git merge合并分支会创建一个新的合并提交，保留分支的历史，合并后的分支树更清晰。

     ```
     git checkout <target_branch>
     git merge <source_branch>
     ```

   - 使用git rebase合并分支会将源分支的提交放在目标分支的最后，形成线性历史，但可能导致冲突更难处理。

     ```
     git checkout <source_branch>
     git rebase <target_branch>
     ```

6. 在Markdown格式的文本中使用以下元素：
   - 标题：使用 `#` 符号，例如 `# 这是一个标题` 。
   - 数字列表：使用数字后跟句点和空格，例如 `1. 项目1` 。
   - 无序列表：使用减号、加号或星号后跟空格，例如 `- 项目1` 。
   - 超链接：使用方括号包裹链接文本，后跟圆括号包裹链接URL，例如 `[文本](URL)` 。例如：Github,长沙学院官网

实验总结

经过这次实验课的学习，我学习到了有关Git和Markdown的相关知识并初步了解了Git的一些操作例如：如何初始化Git仓库、添加文件、提交更改、创建分支、合并分支等，这些知识可以有效的进行版本控制。同时了解了一些Markdown文档的编写基础：如Markdown语法，包括标题、列表、链接、图片插入、引用块、代码块、粗体和斜体，以及表格的创建。Markdown是一种简单而强大的文档编辑工具，可以用于编写格式清晰的文档和报告。这两类工具的使用，能更好的帮助我在接下来的学习中更好的掌握Python这门课程的知识。