实验过程与结果

```
git commit//用来提交一个版本，并且会创建一个节点，并且会有一个parent节点
git branch <分支名>//创建一个分支
git checkout <分支名或者哈希值>//将main切换到分支，也可以分离head
git merge <分支名>//将两个分支合并到一起去(第一种方法)
git rebase <分支名>//将两个分支合并到一起去(第二种方法)
git checkout <分支名>^(n)//可以于分支名把head移动到相对于分支名（n）个parent的位置。
git checkout <分支名>~+(n)// 可以于分支名把head移动到相对于分支名（n）个parent的位置。
git branch -f <分支名> <分支名>~+(n)//强制修改分支位置
git reset HEAD~+(n)//在本地撤销更改，撤销后的更改和没有发生一样
git revert HEAD+~(n)//新创建一个提交，此提交和上级提交一样，且可以远程分享给别人。
```

实验考察

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：版本控制是指对软件开发过程中各种程序代码、配置文件及说明文档等文件变更的管理，是软件配置管理的核心思想之一。git属于分布式版本控制系统，有分布式管理控制系统的优点如：版本库本地化、支持离线提交、分支切换快速高效等。同时git数据相对来说比较安全，支持自建本地版本库和分支且有能力高效管理超大规模项目。
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：使用git reset HEAD~+(n)或者git revert HEAD+~(n)撤销。使用git checkout <分支名>来检出以前的Commit。
3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）
   答：HEAD是当前分支引用的指针，可以使用 git checkout<哈希值>或者 git checkout <分支名或HEAD>~+(n)的相对引用来让HEAD处于 detached HEAD状态。
4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）
   答：分支是指向更改快照的指针，一个分支代表一条独立的开发线。使用git branch <分支名>来创建一个分支，使用git checkout <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：使用git merge <分支名>或者git rebase <分支名>。rebase可以让项目提交历史变得非常干净整洁，rebase会造就一个线性的项目提交历史——也就是说你可以从feature分支的顶部开始向下查找到分支的起始点，而不会碰到任何历史分叉。这在使用git log,git bisect以及gitk等命令时更简

单，但是安全性和可塑性有问题，而merge合并操作很友好，因为它没有破坏性。现存的分支历史不会发生什么改变。这一特性避免了rebase操作的所有缺陷。

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：标题使用#，#数量代表了标题等级。数字列表在每个列表项前添加数字并紧跟一个英文句点。数字不必按数学顺序排列，但是列表应当以数字 1 起始。无序列表在每个列表项前面添加破折号 (-)、星号 (*) 或加号 (+) 。超链接的链接文本放在中括号内，链接地址放在后面的括号中，链接title可选。

实验总结

主要学习了git和markdown的使用，git工具用来管理python项目会很方便，其中掌握了一些基本的命令关于提交、分支、合并、分离HEAD、相对引用已经撤销变更。以后在python项目中也需要尽量使用git。而markdown主要是用来方便快捷生成PDF文件，掌握对markdown的基本操作能很直观的写出实验报告。