实验一 Git和Markdown基础

班级： 21计科1班

学号： B20210102113 姓名： 谭志峰

Github地址： https://github.com/bahdksnxja/python_tasks.git

实验过程与结果

Git基本操作总结

练习网址 **1.1 git commit**

- git commit用于将已添加到git暂存区的更改保存为一个新的提交
- 代码：

```
git commit
git commit
```

1.2 git branch|git checkout

- git branch <branch_name>用于创建分支
- git checkout (-b) <branch_name>(创建并)切换分支
- 代码：

```
  git checkout -b bugFix
```

1.3 git merge

- git merge <branch_name>将当前分支合并更改到目标分支
- 代码：

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

1.4 git rebase

- git rebase <branch_name>将当前分支重新基于目标分支
- 代码：

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

2.1分离HEAD

- HEAD 是一个对当前所在分支的符号引用 —— 也就是指向你正在其基础上进行工作的提交记录。HEAD 总是指向当前分支上最近一次提交记录。大多数修改提交树的 Git 命令都是从改变 HEAD 的指向开始的。HEAD 通常情况下是指向分支名的
- 代码：

```
git checkout C4
```

2.2 ^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
- 代码：

```
git checkout C3
```

2.3 ~相对引用

- ~该操作符后面可以跟一个数字（可选，不跟数字时与 ^ 相同，向上移动一次），指定向上移动多少次。
- 用于移动分支：git branch -f main HEAD~3
- 代码：

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

上面的命令会将 main 分支强制指向 HEAD 的第 3 级 parent 提交。

2.4 撤销变更

- git reset通过把分支记录回退几个提交记录来实现撤销改动。你可以将这想象成“改写历史”。git reset 向上移动分支，原来指向的提交记录就跟从来没有提交过一样。
- 虽然在本地分支中使用git reset很方便，但是这种“改写历史”的方法对大家一起使用的远程分支是无效的，为了撤销更改并分享给别人，我们需要使用git revert。
- 代码：

```
git reset HEAD~1
git checkout pushed
git revert HEAD
```

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？

> 版本控制是一种记录文件或代码在不同时间点的变化，并允许多人协同工作的系统。它可以追踪文件的修改、添加和删除，并提供了回滚到之前版本的功能。 Git是一种分布式版本控制系统。 它具有以下**优点**： 分布式：每个开发者都可以在本地拥有完整的代码仓库，不需要依赖中央服务器，可以离线工作。 高效：Git使用快照来存储文件，只保存文件的变化部分，因此占用更少的存储空间。 分支管理：Git支持轻松创建和合并分支，使得并行开发和团队协作更加容易。 安全性：Git使用SHA-1哈希值来标识文件内容，确保文件的完整性和安全性。 可扩展性：Git具有丰富的插件生态系统，可以根据需要进行定制和扩展。

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）

> (1) git checkout -- ：撤销对指定文件的修改，将文件恢复到最近一次Commit或者Add的状态。 (2) git checkout .：撤销对所有文件的修改，将所有文件恢复到最近一次Commit或者Add的状态。 要检出已经以前的Commit，可以使用以下命令： (1)git log：查看提交历史，获取要检出的Commit的哈希值。 (2)git checkout ： 检出指定的Commit，将工作目录和暂存区恢复到该Commit的状态。可以使用Commit的哈希值、分支名或者标签名来指定Commit。

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

> HEAD 是一个特殊的指针，它指向当前所在的分支或提交。 让HEAD处于detached HEAD状态：使用 git checkout <commit_hash> 来直接切换到一个特定的提交，而不是分支。这将使HEAD分离，并在此提交上工作。

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中的开发线程，允许并行开发不同的功能或修复。 创建分支：使用 git branch <branch_name> 命令创建一个新分支。 切换分支：使用 git checkout <branch_name> 切换到特定分支。

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：使用 git merge <branch_name> 命令将指定分支的更改合并到当前分支。 区别： git merge 创建一个新的合并提交，保留分支的完整历史。 git rebase 将当前分支的更改在目标分支上重新应用，创建一个线性的提交历史，看起来像是在目标分支上开发的。

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）实验总结

> 标题使用 #，例如：# 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 -，例如：* 项目1。 超链接使用 [显示名称] (链接URL)，例如：练习网址。

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。

通过这次Python实验，了解和学习了Python编程语言的一些基本语法和函数的使用，同时也学习了VS code这一编程工具的是使用，将其中各种插件结合起来使用，提高了开发效率，其次也很方便。其次也学习到了Git这一·分布式版本控制系统的使用，以及一些基本的版本控制操作。什么是版本控制？ 版本控制是一种记录一个或若干文件内容变化，以便将来查阅特定版本修订情况的系统。在编写实验报告时也掌握了Markdown的一些操作，关于一些标题，字体，图片插入，分割线，引用等等一些操作。