Python实验报告1：Git和Markdown基础

班级： 21计科1班

学号： 20210302103

姓名： 刘阳阳

实验过程与结果

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
```

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？

    - 版本控制是一种管理项目源代码和文件版本的系统。它允许团队协作、跟踪更改、恢复到以前的状态以及管理代码的历史记录。Git作为版本控制软件有以下优点： 分布式：每个开发者都可以拥有完整的代码仓库，无需依赖中央服务器。 分支管理：容易创建、合并和切换分支，使并行开发和功能隔离更加容易。 版本历史：详细记录每个提交，轻松查看和回滚历史更改。 远程协作：支持多人协作，通过远程仓库共享和同步代码。

  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 -- 撤销单个文件的修改，或使用 git reset HEAD 将文件从暂存区中取消。
    - 使用 git checkout <commit_hash> 来切换到特定提交的状态，或者使用 git checkout <branch_name> 来切换到分支。

  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。

  本次实验通过在learn git branching网站上进行练习了解了Git的基本概念和操作，包括版本控制、提交管理、分支操作，还在vscode上编写了Markdown文本格式。Git和Markdown是开发中不可或缺的工具，通过实际操作可以更好地理解和应用这些概念。