实验一 Git和Markdown基础

班级： 21计科03

学号： 20210302304

姓名： 潘振宇

Github地址：https://github.com/yourusername/python_course

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
```

1. 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。
2. 安装VScode，下载地址：Visual Studio Code
3. 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Gt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learngitbranchingjs.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：

显示效果如下：

```
def add_binary(a,b):
    return bin(a+b)[2:]
```

实验总结

请将实验过程中总结的git命令和操作记录放在这里，包括：

- git init
- git add .

代码运行结果的文本可以直接粘贴在这里。

1.git commit

2.git branch

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

3.git checkout

4.git merge

创建新分支 bugFix

用 git checkout bugFix 命令切换到该分支
提交一次

用 git checkout main 切换回 main
再提交一次
用 git merge 把 bugFix 合并到 main

```
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git merge bugFix
```

5.git rebase

新建并切换到 bugFix 分支
提交一次

切换回 main 分支再提交一次
再次切换到 bugFix 分支，rebase 到 main 上

```
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git checkout bugFix
git rebase main
```

6.HEAD

```
git checkout HEAD
```

7.相对引用

```
git branch -f mian c6
git branch -f bugFix HEAD~2
git checkout c1
```

8.gitreset,gjitrevert

```
git reset HEAD^
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?

   这意味着每个开发者都可以拥有完整的项目仓库的副本。 这有助于离线工作，减少了对中央服务器的依赖，提高了安全性。g侑存储速度快、 安全性高、 容易使用等优 点。并且还有分支管理和 查看历史提交记录的功能。

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）
   可以使用gtreset--hardHEAD^ 撤销还没有commit的修改。
   可以使用gitcheckout命令检出已经以前的commit。

3. Git中的HEAD是什么？ 如何让HEAD处于detached HEAD状态？ （实际操作）
   在Git中， HEAD是一个引用，它指向当前分支的提交。
   当检出到 一个不包含任何文件的commit时，Git将进入detachedHEAD状态。所以只要使用gtcheckout命令检测出 一个不包含任何文件的commit ，Git就会进入detachedHEAD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）
   分支是-种在Git中记录更改的方法， 它可以在不同的时间点保留代码更改。
   可以使用gitbranch命令创建分支。 使用gtcheckout命令切换分支。

5. 如何合并分支？ git merge和git rebase的区别在哪里？ （实际操作）
   rebase会把当前分支的 commit 放到公共分支的最后面,所以叫变基。就好像从公共分支又重新拉出来这个分支一样。merge会把公共分支和你当前的commit 合并在一起，形成一个新的 commit 提交 。采用merge和rebase后，git log的区别，merge命令不会保留merge的分支的commit。

实验总结

本次实验用到了git的基本用法，使用了git、git bash、markdown、vscode等工具，熟悉了git的基本操作，包括：创建仓库、克 隆仓库、提交修改、查看提交历史、创建分支、切换分支、合并分支等。