实验一 Git和Markdown基础

班级： 21计科2

学号： B20210302224

姓名： 莫杭程

Github地址：https://github.com/berlincun

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"
```

或者运行下面的命令:

```
git config --global http.sslVerify false
```

如果遇到错误：`error setting certificate file`，请运行下面的命令重新指定git的安全证书：

```
git config --global --unset http.sslCAInfo
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
```

在本地的仓库内容有更新后，可以运行下面的命令，将本地仓库的内容和远程仓库的内容同步：

```
git push origin main
```

3. 注册Github账号或者Gitee帐号，创建一个新的仓库，例如：https://gitee.com/zj204/python_task.git，使用下面的命令将新建的仓库clone到本地：

```
git clone https://gitee.com/zj204/python_task.git
```

如果已经关联了远程仓库，显示结果如下：

```
origin  https://github.com/zhoujing204/python_course.git (fetch)
origin  https://github.com/zhoujing204/python_course.git (push)
```

如果还没有关联远程仓库，可以使用你创建的远程仓库的地址和下面的命令，添加你要关联的远程仓库：

```
git remote add gitee https://gitee.com/zj204/python_task.git
```

接下来准备好你的远程仓库账号的邮箱地址和密码，使用下面的命令下载远程仓库的内容更新本地仓库：

```
git pull gitee main
```

运行下面的命令，将本地仓库的内容同步到远程仓库：

```
git push gitee main
```

4. 安装VScode，下载地址：Visual Studio Code
5. 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 Git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 learngitbranching.js.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。

如何将markdown文件转换为pdf格式的文件?

- 安装vscode插件Markdown PDF，安装后重启vscode，打开markdown文件，按下`Ctrl+Shift+P`，输入`Markdown PDF: Export (pdf)`，回车即可导出pdf文件。
- 使用Google Chrome浏览器，在Github网站或者Gitee网站打开你的仓库，浏览你的markdown文件，按下`Ctrl+P`，选择打印，选择目标打印机为另存为PDF，点击保存即可导出pdf文件。

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:]
```

代码运行结果的文本可以直接粘贴在这里。

```
1.
git commit
git commit
```

```
2.
git branch bugfix
git checkout bugfix
```

```
3.
git branch bugfix
git checkout bugfix
git commit
git checkout main
git commit
git merge bugfix
```

```
4.
git checkout -b bugfix
git commit
git checkout main
git commit
git checkout bugfix
git rebase main
```

```
5.
git checkout c4
```

```
6.
git checkout bugfix
git checkout HEAD^
```

```
7.
git branch -f bugFix c0
git branch -f main c6
git checkout c1
```

```
8.
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？ 版本控制是一种管理和跟踪文件和代码变更的系统。它有助于团队协作，记录变更历史，并允许开发者在不同版本之间进行切换和比较。 Git 是一个广泛使用的版本控制软件，具有以下优点： 分布式版本控制：Git 使用分布式模型，每个开发者都有完整的代码仓库的副本。这意味着即使没有网络连接，你仍然可以继续工作，而且更容易进行协作。 速度和性能：Git 非常快，因为它在本地存储了一个索引，只需与远程仓库进行有限的通信。这使得提交、分支切换和合并操作非常高效。 强大的分支支持：Git 鼓励分支开发，你可以轻松地创建、合并和删除分支。这使得并行开发和功能开发更容易管理。 完整的历史记录：Git 记录每个提交的详细信息，包括作者、日期和变更内容。这有助于了解项目的演变，解决问题和跟踪贡献者的工作。 灵活性：Git 支持多种工作流程，包括集中式和分布式开发模型。你可以根据项目需求选择适合的工作流程。 社区支持和生态系统：Git 是一个开源项目，有庞大的社区支持。有许多第三方工具和服务可以与 Git 集成，以扩展其功能。 安全性：Git 使用加密来保护数据的完整性，确保提交历史不会被篡改。
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）

```
git reset --hard HEAD
git checkout <提交的哈希值>
```

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是一个特殊的指针，它代表当前所在的本地代码库的最新提交（最顶部的提交）。当HEAD处于"detached HEAD"状态时，意味着它不再指向分支，而是直接指向某个具体的提交。这通常发生在你切换到一个特定的提交（commit）或标签（tag）时。

```
git branch
git checkout <commit-SHA>
```

4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作） 分支是版本控制系统中的一个重要概念，它允许开发者在项目中创建并独立开发不同的代码线，以便在不影响主代码线的情况下进行新功能开发、bug修复等工作.

```
git branch 分支名
git checkout 分支名
```

5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操

```
git rebase 分支名
git merge 分支名
```

git merge 会创建合并提交，保留分叉点，而 git rebase 会将提交历史线性化，看起来更整洁。 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. 使用 # 符号来表示标题，数量的 # 符号决定了标题的级别

```
# 这是一级标题
## 这是二级标题
### 这是三级标题
```

2. 使用数字和点号来创建有序列表

```
1. 第一项
2. 第二项
3. 第三项
```

3. 无序列表使用星号、加号或减号加上空格来表示列表项

```
- 项目1
- 项目2
- 项目3
```

4. 使用方括号 [] 来表示链接文本，紧接着使用圆括号 () 来表示链接的URL

```
[点击这里查看更多信息](https://www.example.com)
```

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。

1. 学会了基本的git命令
2. 掌握了Markdown基础，使用Markdown进行文档编辑