实验一 Git和Markdown基础

班级： 21计科3班

学号： B20210302302

姓名： 蒋俊杰

Github地址：https://github.com/jiangjunjie666/python_study

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
```

3. 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。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件

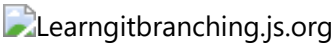
- GitLens
- Git Graph
- Git History
- Markdown All in One
- Markdown Preview Enhanced
- Markdown PDF
- Auto-Open Markdown Preview
- Paste Image
- markdownlint

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。


Learngitbranching.js.org

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：


Git命令

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：

 Python代码

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 练习

提交名为：c1,c2 代码

```
git commit -m "c1"
git commit -m "c2"
```

2.git branch 练习

创建一个bugFix的分支，并切换到bugFix分支

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

3.git merge 练习

创建新分支 bugFix 切换到该分支 提交一次c1代码 切换回main分支 提交一次c2代码 将bugFix分支合并到main分支

```
git branch bugFix
git checkout bugFix
git commit -m "c1"
git checkout main
git commit -m "c2"
git merge bugFix
```

4.git rebase 练习

新建并切换到bugFix分支 提交一次 切换回main分支提交一次 再次切换到bugFix分支，并执行rebase操作到main上

```
git branch bugFix
git checkout bugFix
git commit -m "c2"
git checkout main
git commit -m "c3"
git checkout bugFix
git rebase main
```

5.Head 练习

从bugFix分支中分离出HEAD并让其指向一个提交记录 通过哈希值指定提交记录，每个提交记录的哈希值显示在代表提交记录的圆圈中

```
git checkout c4
```

6.相对引用 ^

切换到bugFix的perent节点

```
git checkout c4
git checkout HEAD^
```

7.相对引用~

```
git branch -f main c6
git checkout c1
git branch -f bugFix HEAD^
```

8.撤销变更

```
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和管理项目中代码和文件变化的系统。使用Git作为版本控制软件的优点包括版本历史记录、并行开发和合并、错误恢复和回退、分支和标签、以及协作和团队合作的能力。Git作为一个分布式版本控制系统，具有速度和性能高、分布式架构、强大的分支和合并、完整性和鲁棒性、以及广泛的社区支持等优点。

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） 要使用 Git 撤销尚未提交的修改，可以使用以下命令：

```
git checkout -- <file>
```

撤销对所有文件的修改：

```
git checkout -- .
```

要使用 Git 检出（Checkout）已经以前的 Commit，可以使用以下命令：

```
git checkout <commit>
```

检出以前的提交并创建新分支：

```
git checkout -b <new-branch> <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 是一个特殊的指针，它指向当前所在的分支或提交。它可以用来引用最新的提交或当前所在的分支。

让HEAD处于detached HEAD状态： 首先，使用 git log 命令查看提交历史，并找到要检出的提交的哈希值：

```
git log
```

复制要检出的提交的哈希值。 运行以下命令，将 HEAD 设置为所选提交的哈希值： 将 commit 替换为要检出的提交的哈希值

```
git checkout <commit>
```

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（Branch）是在 Git 中用于并行开发的重要概念。它允许你在项目中创建独立的线条，每个线条上可以独立进行修改和提交，而不会影响主分支或其他分支的代码。 创建分支

```
git branch <branch-name>
```

切换分支

```
git checkout <branch-name>
```

创建并切换到此分支

```
git checkout -b <branch-name>
```

5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作） 这二条命令的作用都是合并分支

```
git merge <branch-name>
git rebase <branch-name>
```

git merge: 将指定分支的更改合并到当前分支中。这会创建一个新的合并提交，并保留每个分支的提交历史。适合用于合并长期并行开发的分支或团队合作的分支。

git rebase: 将当前分支的更改应用于指定分支的顶部。这会将当前分支的更改重新基于目标分支的最新提交，形成一条线性的提交历史。适合用于保持干净的提交历史，消除分支间的间隙，并使提交历史更易读。

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）

使用标题：

```
# 一级标题
## 二级标题
### 三级标题
```

数字列表：

```
1. 第一项
2. 第二项
3. 第三项
```

无序列表：

```
- 第一项
- 第二项
- 第三项
```

超链接：

```
[链接文本](链接URL)
```

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。

这次实验我学会的vscode这个编程工具的使用，以及进一步学习了git的一些命令，并尝试了在本地创建一个git仓库，并提交了代码，收获很大。