实验一 Git和Markdown基础

班级： 21计科1

学号：B20210302101

姓名： 金库

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
```

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

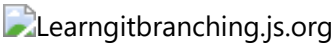
- GitLens
- Git Graph
- Git History
- Markdown All in One
- Markdown Preview Enhanced
- Markdown PDF
- Auto-Open Markdown Preview
- Paste Image
- markdownlint

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。



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：

 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。

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？

   （1）版本控制是一种记录和管理文件或代码变更历史的系统。它可以追踪文件的修改、添加和删除，并允许多个人协同工作在同一个项目上。

   （2）Git作为版本控制软件具有高效性能、分布式（不依赖中央处理器，开发者可以有自己的代码仓库）、强大的分支管理和合并功能，以及数据完整性和安全性等优点

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）

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

   // 这个命令会将指定文件恢复到最近一次提交的状态，丢弃掉所有的修改。

   ```
   git checkout <commit-hash>
   ```

   // 其中 是你要检出的提交的哈希值。这个命令会将工作目录和索引回滚到指定提交的状态，并切换到一个分离头指针（detached HEAD）的状态。在这个状态下，你可以查看以前的提交，但是不能进行新的提交

3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？

   (1)在Git中，HEAD是一个特殊的指针，它指向当前所在的分支或者提交。它可以理解为当前工作目录的快照，指示了你正在工作的分支或者提交。

   当HEAD指向一个分支时，表示当前工作目录处于正常的工作状态，可以进行提交、切换分支等操作。而当HEAD指向一个提交（commit）时，表示当前工作目录处于分离头指针（detached HEAD）的状态。

4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）

分支是指向提交（commit）的可变指针。它可以理解为一个特殊的指针，指向某个提交，表示当前工作目录的状态。

```
git branch <branch-name>//创建分支
```

```
git checkout <branch-name>//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）

合并分支是将一个分支的更改合并到另一个分支中的操作。合并分支可以将一个分支的更改应用到另一个分支，从而保持两个分支之间的同步。

git merge：这是一种常用的合并分支的方法。它将指定分支的更改合并到当前分支中。合并分支时，Git会创建一个新的合并提交，将两个分支的更改合并在一起。以下是具体的操作步骤：

```
a. 切换到接受更改的目标分支
b. 执行合并命令，将指定分支的更改合并到目标分支
```

git rebase：这是另一种合并分支的方法。它将指定分支的更改应用到当前分支上，并且可以使提交历史更加线性。具体操作步骤如下：

```
a. 切换到当前分支
b. 执行变基命令，将指定分支的更改应用到当前分支上
```

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？

在Markdown格式的文本中，可以使用以下语法来使用标题、数字列表、无序列表和超链接：

(1) 标题：使用 # 符号来表示标题的级别，# 的数量表示标题的级别，最多可以使用六个 # 符号。例如：

```
# 一级标题
## 二级标题
### 三级标题
```

(2) 数字列表：使用数字和点号来表示数字列表的项。例如：

```
1. 第一项
2. 第二项
3. 第三项
```

(3) 无序列表：使用 - 或 * 符号来表示无序列表的项。例如：

```
- 无序列表项1
- 无序列表项2
- 无序列表项3
```

(4)超链接：使用 [链接文本](链接地址) 的格式来创建超链接。例如：

```
[百度](https://www.baidu.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。

通过这次实验学到了很多git命令，如下：

```
git branch <分支名>  创建分支

git checkout  <分支名> 切换到分支
```

```
git merge bugFix  把bugFix合并到*所在节点，即目标节点
git rebase main  合并分支，切换到当前分支
```

```
git checkout HEAD~4  //往上返回四级
```

```
git branch -f  main c6
```

命令"git branch -f main c6"用于强制将名为"main"的分支移动到提交"c6"。这意味着名为"main"的分支现在将指向提交"c6"，在"c6"之后的任何先前提交将不再属于该分支。

```
git branch -f bugFix c0
```

```
git reset HEAD~1   //等价撤销^,^^ ;
```

命令git reset HEAD^用于将当前分支的HEAD指针向后移动一个父节点提交，并将工作目录中的更改保留在未提交的状态下。//这个命令的效果是取消最后一次提交，并将更改保留在工作目录中。通过使用HEAD~1，我们将HEAD指针向后移动一个父提交，因此取消了最后一次提交。但是，工作目录中的更改仍然存在，并且处于未提交的状态。您可以对这些更改进行修改、撤销或重新提交。

```
git cherry-pick C2 C4//添加的地方要有*

git rebase -i HEAD~4
```