21计科03-B20210302301杨韬-实验一

实验一 Git和Markdown基础

班级： 21计科03

学号： B20210302301

姓名： 杨韬

Github地址：https://github.com/bairimenglin/yangtao/tree/main/experiment

---

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。

"img/2023-07-28-21-07-40.png" 未创建，点击以创建。

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：

"img/2023-07-26-22-48.png" 未创建，点击以创建。

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：

"img/2023-07-26-22-52-20.png" 未创建，点击以创建。

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

git commit

2.git branch

git branch bugFix

git cheackout bugFix

3.git checkout

git branch bugFix

git checkout bugFix

4.git merge

git branch bugFix

git checkout bugFix

git commit

git checkout main

git commit

git merge bugFix

5.git rebase

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.相对引用2

git branch -f main c6

git checkout bugFix

git branch -f bugFix HEAD~3

git checkout C1

9.撤销变更

git reset HEAD~1

git checkout pushed

git revert pushed

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？

版本控制是指在软件开发过程中对源代码或其他文档的版本进行管理和控制。

简单易用：Git具有简单的命令行界面，使得开发者可以轻松地进行版本控制。它还提供了一些自动化命令，使得开发者可以更快速进行开发。

可靠：Git具有良好的可靠性，可以有效地保存和恢复代码历史。它使用SHA-1哈希值来标识每个版本，这使得每个版本都是唯一的。

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 git reset --hard HEAD^ 撤销还没有Commit的修改。

可以使用 git checkout 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）

HEAD是Git中的一个指针，它指向当前工作目录中暂存区的提交。

可以将HEAD重置为某个提交，而不是指向一个分支。要实现这一点，可以使用git checkout命令并指定提交哈希值.

4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）

分支是指在同一个项目的不同阶段或版本之间创建的命名空间，它允许开发者同时开发多个功能或问题解决方案。

可以使用git branch命令创建分支，使用git checkout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）

可以使用git merge命令将一个分支合并到另一个分支。

git rebase命令将一个分支的修改应用到另一个分支上，从而避免了合并操作产生的冲突。git merge命令将两个分支的更改合并到一个分支。在合并过程中，Git会尝试保留每个分支的历史，并创建一个新的提交来表示合并的结果。

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）

“#”这是一个标题 “##”这是一个二级标题

“1.”这是一个有序列表 “-”这是一个无序列表

“链接名称”这是一个超链接

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。