

实验一 Git和Markdown基础

班级： 21计科02

学号： 20210302221

姓名： 王日晖

Github地址： <https://github.com/wrrh>

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](https://git-flight-rules.com/)

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：

![Git命令]

显示效果如下：

```
git init
git add .
git status
git commit -m "first commit"
```

代码运行结果的文本可以直接粘贴在这里。

1.[Git Commit]

显示效果如下：

```
git commit
git commit
```

2.[Git Branch]

显示效果如下：

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

3.[Git Merge]

显示效果如下：

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

4.[Git Rebase]

显示效果如下：

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

5.[分离 HEAD]

显示效果如下：

```
git checkout C4
```

6.[相对引用 (^)]

显示效果如下：

```
git checkout C4  
git checkout HEAD^
```

7.[相对引用2 (~)]

显示效果如下：

```
git branch -f main C6  
git checkout C1  
git branch -f bugFix C0
```

8.[撤销变更]

显示效果如下：

```
git reset C3^  
git checkout pushed  
git revert C2
```

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文件变化的系统，可以追踪文件的修改、删除和新增，并且可以轻松地回退到之前的版本。使用Git作为版本控制软件的优点包括：

分布式版本控制：每个开发者都可以拥有完整的代码仓库，便于离线工作和并行开发。

高效管理分支：Git具有强大的分支管理功能，可以轻松创建、切换、合并和删除分支。

快速和轻量级：Git的设计目标是速度和简单性，因此它非常快速和轻量级。

安全性：Git使用SHA-1哈希算法来存储文件内容，确保数据的完整性和安全性。

丰富的功能：Git提供了强大的命令行工具和图形界面，支持各种版本控制操作和 workflow。

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）

要撤销还没有提交的修改，可以使用以下命令：

`git checkout - <文件名>`：撤销对指定文件的修改。

`git reset HEAD <文件名>`：将指定文件从暂存区移回工作区。

要检出以前的提交，可以使用以下命令：

`git log`：查看提交历史，找到要检出的提交的哈希值。

`git checkout <commit的哈希值>`：将HEAD指向指定的提交，并将文件恢复到该提交的状态。

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是指向当前所在分支的指针。要将HEAD置于detached HEAD状态，可以使用以下命令：

`git checkout <commit的哈希值>`：将HEAD指向指定的提交，不再指向任何分支。

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Git中用于并行开发的重要概念，它是一条独立的提交链。要创建分支，可以使用以下命令：

`git`

`git branch <分支名>`：创建一个新的分支。

`git checkout <分支名>`：切换到指定的分支。

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

要合并分支，可以使用以下命令：

`git merge <要合并的分支名>`：将指定分支的修改合并到当前分支。

`git merge`和`git rebase`的区别在于合并的方式不同：

`git merge`会将指定分支的修改合并到当前分支的最新提交中，形成一个新的合并提交。

`git rebase`会将当前分支的修改放在指定分支的最新提交之后，形成一个线性的提交历史。

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）

在Markdown格式的文本中使用标题、数字列表、无序列表和超链接可以按照以下方式进行：

标题：使用#符号，#后面加空格和标题内容，#越多表示标题级别越低。

数字列表：使用数字和句点，例如"1. "，后面加上列表项内容。

无序列表：使用星号、加号或减号，后面加上列表项内容。

超链接：使用方括号和圆括号，方括号中是链接的显示文本，圆括号中是链接的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。

在学习Git命令和使用过程中，我们涉及到了一些与编程相关的知识和技能。总结如下：

1. 版本控制：学习了版本控制的概念，了解了Git作为一种强大的版本控制系统的优点和使用方法。
2. Git基础操作：学习了Git的基本命令，包括初始化仓库、添加文件、提交修改、撤销修改、查看提交历史、切换分支、合并分支等操作。
3. 分支管理：了解了分支的概念和作用，学习了创建、切换、合并和删除分支的方法。
4. Markdown语法：学习了如何在Markdown格式的文本中使用标题、数字列表、无序列表和超链接。

在学习过程中，我们也触碰到了一些编程的相关知识和技巧：

- 编程工具的使用：使用Git命令行工具进行版本控制操作。
- 数据结构：Git中的提交历史可以看作是一种树形结构的数据结构。
- 程序语言的语法：Git命令行工具使用的是命令行语法，需要熟悉基本的命令用法。
- 算法：Git使用哈希算法来存储文件内容，确保数据的完整性。
- 编程技巧：学习了一些Git的常用命令和操作技巧，例如撤销修改、合并分支等。
- 编程思想：学习了分支管理和版本控制的思想，了解了并行开发和代码合并的概念。
-

通过学习Git命令和使用到的知识，我们不仅可以更好地管理和追踪代码的变化，还能够更高效地进行团队协作和并行开发。