

实验一 Git和Markdown基础

班级：21计科1

学号：202302200000

姓名：张三

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

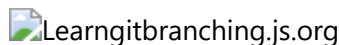
- GitLens
- Git Graph
- Git History
- Markdown All in One
- Markdown Preview Enhanced
- Markdown PDF
- Auto-Open Markdown Preview
- Paste Image
- markdownlint

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](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。

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：



基础篇

1. 提交

```
- git commit  
- git commit
```

2. 创建分支

```
- git branch bugFix //创建分支
- git commit
- git checkout bugFix //切换分支
- git commit
- git checkout -b bugFix //创建并切换
```

3. 合并方法一 git merge

```
git checkout -b bugFix //创建并切换
git commit
git checkout main //切换main分支
git commit
git merge bugFix 合并到main分支
```

4. 合并方法二 git rebase

```
git checkout -b bugFix
git commit
git checkout main //切换main分支
git commit
git checkout bugFix
git rebase main //合并到main
```

高级篇

1. 分离head

```
git checkout c4 //head指向c4
```

2. 相对引用

- 使用 ^ 向上移动 1 个提交记录
- 使用 ~ 向上移动多个提交记录, 如 ~3

```
git checkout bugFix^
```

3. 相对引用2

```
git branch -f main C6
git checkout C1
git branch -f bugFix HEAD^
```

4. 撤销变更

两种方法用来撤销变更: 1.git reset(本地), 2.git revert(远程) pushed 是远程分支, local 是本地分支

```
git reset HEAD
git checkout pushed
git revert HEAD
```

显示效果如下：

```
```bash
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？

版本控制是一种管理和跟踪软件开发过程中不同版本的系统。它允许开发团队记录每个版本的变化，并且可以轻松地从以前的版本、比较不同版本之间的差异、合并不同开发者的工作等。Git 是一个分布式版本控制系统，它具有以下优点：

分布式：每个开发者都拥有完整的版本库，可以在没有网络连接的情况下工作，并且可以直接从其他开发者的库中获取更新。

高效：Git 的设计使得它能够快速处理大型项目和大量的历史记录。

强大的分支和合并功能：Git 的分支模型非常灵活，可以轻松地创建和切换分支，然后将分支合并回主线。

完整性和一致性：Git 使用哈希值来标识文件和目录，以确保数据的完整性和一致性。

## 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 example.txt //撤销单个文件
git checkou .//撤销全部
git checkout abc123// commit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 是指向当前所在分支或 Commit 的指针。通常，HEAD 指向最新的提交，表示当前所在的工作状态。

要将 HEAD 处于 detached HEAD 状态，可以使用 `git checkout` 命令加上某个 Commit 的哈希值

## 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 Git 中用来管理不同版本的代码的一种方式。通过创建分支，可以在同一个代码仓库中并行开发不同的功能或修复不同的 bug。

创建分支：`git branch a`//创建a分支  
切换分支：`git checkout a`//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）

```
git merge <branch_name>
```

`git rebase` 是另一种合并分支的方式。它将某个分支的更改先保存为临时文件，然后再将这些更改应用到目标分支上。相比于 `git merge`，它可以产生更整洁的提交历史，但是比较复杂，也可能导致冲突

## 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）

- 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

[链接文字](链接地址)

## 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。

- 学习了git和Markdown的基础用法,另外还学习了Vscode一些插件用法