实验一 Git和Markdown基础

班级： 21计科04

学号： B20210502235

姓名： 高楷皓

Github地址: https://github.com/Sgran777/Python_Resource

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，使用上面同样的方法将该仓库clone到本地，用于存放实验报告和实验代码，使用`git pull`和`git push`命令保持远程仓库和本地仓库的同步。
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。

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

Git

克隆git仓库

```
git clone https://github.com/Lxiunneg/my_python_course
Cloning into 'my_python_course'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

新建文件

```
touch test.txt
```

检查本地仓库状态

```
git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test.txt

nothing added to commit but untracked files present (use "git add" to track)
```

添加全部文件

```
git add .
```

再次检查本地仓库状态

```
git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   test.txt
```

提交

```
git commit -m "这是一次测试提交"
[main 4f97ee5] 这是一次测试提交
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
```

推送

```
git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 309 bytes | 309.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Lxiunneg/my_python_course
   0e8f00b..4f97ee5  main -> main
```

更新本地仓库

```
git pull
Already up to date.
```

创建一个新分支newBranch

```
git branch newBranch
```

切换至newBranch分支

```
git checkout newBranch
Switched to branch 'newBranch'
```

也可创建分支后直接跳转到新分支

```
git checkout -b newBranch
```

合并分支

```
git merge main
Already up to date.
```

也可以使用rebase

```
git rebase main
git checkout main
git rebase newBranch
```

Level分离HEAD

```
# git checkout C1(Hash Value)
git checkout 4f97ee5
Note: switching to '4f97ee5'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false
```

```
HEAD is now at 4f97ee5 这是一次测试提交
```

查看日志

```
git log
commit 4f97ee523c24e996d2f8a73058c3cdc02f92fc09 (HEAD -> main, origin/main,
origin/HEAD, newBranch)
Author: Lxiunneg <1127576980@qq.com>
Date:   Thu Sep 21 09:04:14 2023 +0800

    这是一次测试提交

commit 0e8f00b9dc6e16491eddb4ff536172e4cdbd47b6
Author: Lxiunneg <129406506+Lxiunneg@users.noreply.github.com>
Date:   Thu Sep 21 08:53:17 2023 +0800

    Initial commit
```

相对引用^ 单次移动

使用^可以向当前节点的父节点跳转

```
git checkout newBranch^
Note: switching to 'newBranch^'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 0e8f00b Initial commit
```

相对引用~ 多次移动

```
git checkout newBranch~1
HEAD is now at 0e8f00b Initial commit
```

撤销变更

- 撤销本地变更(`reset`)

  ```
  git reset HEAD~1
  ```

- 撤销远程变更(`revert`)

  ```
  git revert HEAD
  ```

整理分支信息-复制(`cherry_pick`)

```
git cherry_pick C1 C2 ...
```

将提交信息:`C1`,`C2`,...按顺序复制到当前的`HEAD`之下

交互式`rebase`

```
git rebase -i OverHere
```

如果你不清楚你想要的提交记录的哈希值,可以利用交互式的 `rebase` —— 如果你想从一系列的提交记录中找到想要的记录, 这就是最好的方法

交互式 rebase 指的是使用带参数 `--interactive` 的 `rebase` 命令, 简写为 `-i`

标签

```
git tag c1[node] v1[tag]
```

描述

```
git describe
```

```
<tag>_<numCommits>_g<hash>
```

`tag` 表示的是离 `ref` 最近的标签， `numCommits` 是表示这个 `ref` 与 `tag` 相差有多少个提交记录， `hash` 表示的是你所给定的 `ref` 所表示的提交记录哈希值的前几位。

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？

   版本控制是一种用于管理文件变更的系统。它可帮助团队协作开发软件或其他项目，并追踪文件的修改历史记录。

   1.在使用Git进行版本控制时，每个开发者都具有本地的完整代码仓库副本，可以在本地进行代码修改和提交，而不需要即时连接到中央服务器。

   2.Git的分支功能非常强大，它可以轻松地创建、合并和删除分支，使得团队成员能够并行开展工作而不会相互干扰。

   3.Git保留了每次提交的完整历史记录，包括提交者、提交时间和提交的具体内容等信息。

2. 如何使用Git撤销还没有Commit的修改？如何使用Git检出（Checkout）已经提交的Commit？（实际操作）

   - 还没有`commit`的撤销

     1.

     ```
     git checkout <filename>
     ```

     2.

     ```
     git cheeckout .(通配)
     ```

   - 已经提交的`commit`

     1. git checkout <提交的hash>

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提交的指针。它标识了当前工作树的状态。
   - 查看所有的提交记录，找到要切出的提交的提交ID。

     ```
     git log
     ```

切换到特定的提交：

```
git checkout --detach <提交ID>
```

4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）

   - 在Git中，分支（`Branch`）是指代码库的一个独立线条，用于开发不同的功能、解决问题或并行开发。每个分支都代表了代码库的不同状态或版本。
   - 创建分支

     ```
     git branch <newBreachName>
     ```

   - 切换分支

     ```
     git checkout <branchName>
     ```

   - 创建并切换

     ```
     git checkout -b <newBreachName>
     ```

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`命令或`git rebase`命令来实现分支合并。
   - `git merge`:

     确保在要合并的目标分支上：

     ```
     git checkout <目标分支>
     ```

     执行合并命令来将源分支的修改合并到目标分支：

     ```
     git merge <源分支>
     ```

   - `git rebase`:

     确保在要合并的分支上：

     ```
     
     ```

```
git checkout <目标分支>
```

执行rebase命令来将源分支的修改合并到目标分支：

```
git rebase <源分支>
```

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：使用#，多少个#，代表是多少级标题。
    - 数字列表：使用<number>. ，来生成数字列表
    - 无序列表： 使用- ，来生成无序列表
    - 超链接：[nikename](hyperlink)

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。