实验一 Git和Markdown基础

班级： 21计科03班

学号： B20210302326

姓名： 李俊瑜

Github地址：https://gitee.com/YukiIm/python_exp

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://gitee.com/zj204/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。

主要 远程

基础篇

循序渐进地介绍 Git 主要命令

4: Git Rebase

高级篇

要开始介绍 Git 的超棒特性了，快来吧！

1: 分离 HEAD

移动提交记录

自由修改提交树

1: Git Cherry-pick

杂项

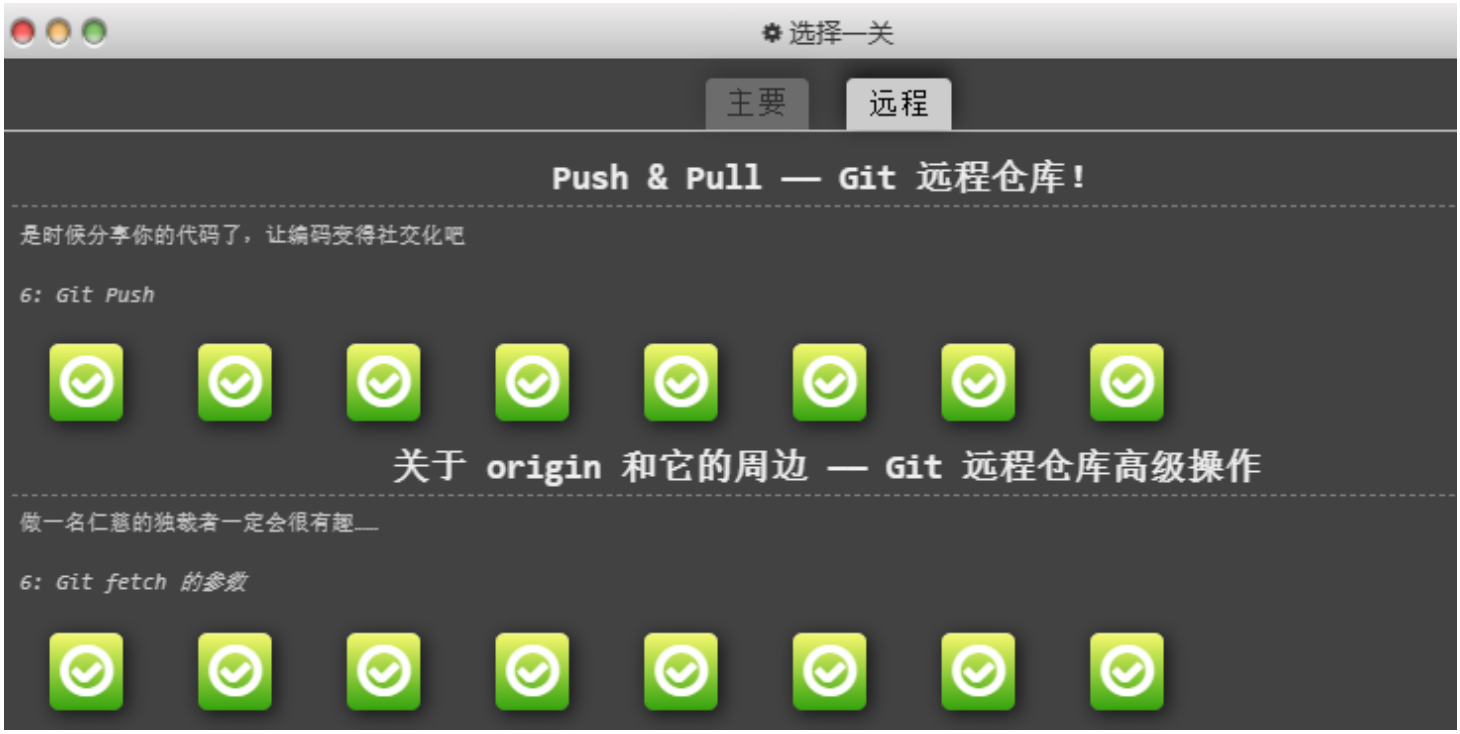
Git 技术、技巧与贴士大集合

1: 只取一个提交记录

高级话题

只为真正的勇士！

1: 多次 Rebase



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：

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

显示效果如下：

```
git init
Initialized empty Git repository in D:/Python/git_practice/.git/
git status
git add .
git status
On branch master
No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   .gitignore
        new file:   hello_git.py

[master (root-commit) 05017a1] Started project.
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 hello_git.py
```

```
git commit -m "Started project."
git log
git log --pretty=oneline
git commit -am "Extended greeting"
git status
git log --pretty=oneline
git status
git restore hello_git.py
git status
git log --pretty=oneline
git checkout 05017a
git switch -
git status
git log --pretty=oneline
git reset --hard 05017a
git log --pretty=oneline
git status
git log --pretty=oneline
git status
del .git
.git, 是否确认(Y/N)? Y
git status
git init
git status
git add .
git status
```

```
On branch master
nothing to commit, working tree clean

On branch master
nothing to commit, working tree clean

commit 05017a108964e94e1dca405443c4570dfd57e494 (HEAD -> master)
Author: Licifer <2813206264@qq.com>
Date:   Mon Sep 18 10:35:49 2023 +0800

    Started project.

D:\Python\git_practice>git log --pretty=oneline
05017a108964e94e1dca405443c4570dfd57e494 (HEAD -> master) Started project.

05017a108964e94e1dca405443c4570dfd57e494 (HEAD -> master) Started project.

D:\Python\git_practice>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello_git.py

no changes added to commit (use "git add" and/or "git commit -a")

[master ac7dc7a] Extended greeting
 1 file changed, 2 insertions(+), 1 deletion(-)

On branch master
nothing to commit, working tree clean

ac7dc7aeede5bb8faf92fbd04afdaea43e2213fb (HEAD -> master) Extended greeting
05017a108964e94e1dca405443c4570dfd57e494 Started project.

On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello_git.py

no changes added to commit (use "git add" and/or "git commit -a")
```

```
fatal: you must specify path(s) to restore



On branch master
nothing to commit, working tree clean

ac7dc7aeede5bb8faf92fbd04afdaea43e2213fb (HEAD -> master) Extended greeting
05017a108964e94e1dca405443c4570dfd57e494 Started project.

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 05017a1 Started project.

Previous HEAD position was 05017a1 Started project.
Switched to branch 'master'

On branch master
nothing to commit, working tree clean

ac7dc7aeede5bb8faf92fbd04afdaea43e2213fb (HEAD -> master) Extended greeting
05017a108964e94e1dca405443c4570dfd57e494 Started project.

HEAD is now at 05017a1 Started project.

05017a108964e94e1dca405443c4570dfd57e494 (HEAD -> master) Started project.

On branch master
nothing to commit, working tree clean
```

```
05017a108964e94e1dca405443c4570dfd57e494 (HEAD -> master) Started project.

On branch master
nothing to commit, working tree clean

fatal: not a git repository (or any of the parent directories): .git

Initialized empty Git repository in D:/Study/Git/git_practice/.git/

On branch master

No commits yet

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

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

D:\Python\git_practice>git commit -m "Starting over"
[master (root-commit) 385539f] Starting over
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 hello_git.py

On branch master
nothing to commit, working tree clean
```

如果是Python代码，应该使用下面代码块格式，例如：

```
print("Hello Git World!")
print("Hello everyone.")
print("Oh no, I broke the project!")
```

显示效果如下：

```
Hello Git World!
Hello everyone.
Oh no, I broke the project!
```

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？
   版本控制是对我们开发的软件等编程内容的历史记录技术
   使用Git作为版本控制可以便于输入指令与管理 同时还可以将内容进行上传开源
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 <分支名>
3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）
   HEAD是当前所指向的分支的指针
   git checkout <不是当前所指向的分支>
4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）
   Branch是指向提交的指针
   创建分支：git branch <分支名>
   切换分支：git checkout <分支名>
5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作）
   使用 merge 或者 rebase 指令
   所合并后的结构不同 rebase是一条直线没有合并的结果 merger则有一个合并的结果
   git merge <合并分支名>
   git rebase <合并分支名>
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）
   使用不同个数“#”后接空格表示标题层级：
   一个#:一级标题表示
   两个#:二级标题表示
   三个#:三级标题表示
   使用数字加“. ”表示数字列表：
7. 第一项
8. 第二项
9. 第三项
   使用-、+或*表示无序列表

- 1
- 2

- 3
  使用链接文本的格式表示超链接
  百度

实验总结

通过本次实验，了解了对Git进行版本控制的一些基本命令与操作，知晓了版本控制这一思想，同时掌握了markdown文档的一些基础语法运用。