实验一 Git和Markdown基础

班级： 21计科1

学号： B20210302106

姓名： 彭浩 Github地址：https://github.com/234519402/pythonpeng

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
```

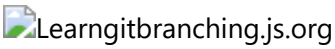
3. 注册Github账号或者Gitee帐号，创建一个新的仓库，使用上面同样的方法将该仓库clone到本地，用于存放实验报告和实验代码，使用`git pull`和`git push`命令保持远程仓库和本地仓库的同步。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件
    - GitLens
    - Git Graph
    - Git History
    - Markdown All in One
    - Markdown Preview Enhanced
    - Markdown PDF
    - Auto-Open 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基础的学习
D.1配置Git
ming2@rang MINGW64 ~/Desktop/GIT基础
$ git config --global user.name “itcast”

ming2@rang MINGW64 ~/Desktop/GIT基础
$ git config --global user.email “usename@example.com”

ming2@rang MINGW64 ~/Desktop/GIT基础
$ git config --global user.name
“itcast”

ming2@rang MINGW64 ~/Desktop/GIT基础
$ git config --global user.email
“usename@example.com”

D.4初始化仓库
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice
$ git init
Initialized empty Git repository in C:/Users/ming2/Desktop/GIT基
础/git_practice/.git/

D.5检查状态
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main

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.6将文件加入仓库
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git add .
```

```
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main

No commits yet

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

D.7执行提交

```
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git commit -m "Started project."
[main (root-commit) 5e5ecc6] Started project.
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 hello_git.py

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
nothing to commit, working tree clean
```

D.8查看提交历史

```
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git log
commit 5e5ecc6f3efd3a678fe422a265af6ccae6423ace (HEAD -> main)
Author: “itcast” <“usename@example.com”>
Date:   Fri Oct 6 20:11:58 2023 +0800

    Started project.

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git log --pretty=oneline
5e5ecc6f3efd3a678fe422a265af6ccae6423ace (HEAD -> main) Started project.
```

D.9第二次提交

```
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
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")

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git commit -am "Extended greeting."
[main d03ceb3] Extended greeting.
 1 file changed, 2 insertions(+), 1 deletion(-)

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
```

```
$ git status
On branch main
nothing to commit, working tree clean

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git log --pretty=oneline
d03ceb3f179f8634c74db1555f49eef006953ce0 (HEAD -> main) Extended greeting.
5e5ecc6f3efd3a678fe422a265af6ccae6423ace Started project.

D.10放弃修改
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
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")

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git restore .

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
nothing to commit, working tree clean

D.11检出以前的提交
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git log --pretty=oneline
d03ceb3f179f8634c74db1555f49eef006953ce0 (HEAD -> main) Extended greeting.
5e5ecc6f3efd3a678fe422a265af6ccae6423ace Started project.

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git checkout 5e5ecc
Note: switching to '5e5ecc'.

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 5e5ecc6 Started project.
```

```
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice ((5e5ecc6...))
$ git switch -
Previous HEAD position was 5e5ecc6 Started project.
Switched to branch 'main'

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
nothing to commit, working tree clean

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git log --pretty=oneline
d03ceb3f179f8634c74db1555f49eef006953ce0 (HEAD -> main) Extended greeting.
5e5ecc6f3efd3a678fe422a265af6ccae6423ace Started project.

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git reset --hard 5e5ecc
HEAD is now at 5e5ecc6 Started project.

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
nothing to commit, working tree clean

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git log --pretty=oneline
5e5ecc6f3efd3a678fe422a265af6ccae6423ace (HEAD -> main) Started project.

D.12删除仓库
ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
nothing to commit, working tree clean

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ rm -rf .git/

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice
$ git status
fatal: not a git repository (or any of the parent directories): .git

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice
$ git init
Initialized empty Git repository in C:/Users/ming2/Desktop/GIT基
础/git_practice/.git/

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main

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)

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git add .

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git commit -m "Starting over."
[main (root-commit) c41f20b] Starting over.
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 hello_git.py

ming2@rang MINGW64 ~/Desktop/GIT基础/git_practice (main)
$ git status
On branch main
nothing to commit, working tree clean
```

```
print("hello git world!")
```

新建分支的git命令语法为，git branch <name>，<name>为新建分支的名字。
切换分支的命令语法为，git checkout <name>。<name>为切换进的分支名。
新建并切换分支的命令语法为，git checkout -b <name>。即，在切换分支的命令基础上加上-b参数。
合并分支的语法为，git merge <name>。将名为<name> 的分支合如当前所在的分支。
删除分支的语法形式有两种：
git branch -d <name> git branch -D <name>
其中 -d 和 -D都代表delete，不同之处在于-d参数用来删除已经合入到本分支的分支。既然已经合入进本分支，那就代表本分支拥有要删除分支的所有提交记录，所以删除分支毫无压力。
而-D参数表示强制删除，可以删除没有合入进本分支的分支。
查看分支git branch 不加任何参数就能展示当前所有分支的清单。

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文件和代码变化的系统，它跟踪文件的修改历史，允许多人协作，回滚到先前的版本，以及解决冲突。它有助于团队更好地管理项目，确保代码的稳定性和可维护性。 使用Git作为版本控制软件有以下优点：1. 分布式版本控制2. 强大的分支管理3. 快速和高效4.社区支持和广泛采用
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） 撤销修改:git checkout 检出已经以前的Commit:git log git checkout

3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作） HEAD是一个特殊的指针，它指向当前所在的分支或Commit git branch git log git checkout
4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作） Branch是一个独立的开发路径，它允许您在项目中并行进行工作，而不会影响主要的代码线。 创建分支git branch 新分支名 切换分支git checkout 新分支名
5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作） 使用git merge合并分支：git checkout 目标分支 git merge 要合并的分支 git merge会创建一个新的合并提交，保留了分支的完整历史。这意味着您可以清晰地看到分支的历史和合并点。但它可能会导致分支历史变得复杂。 git rebase将重新设置分支的提交，将它们应用到目标分支的顶部，使得历史线看起来更加线性。这可以保持分支历史的整洁，但会改写提交历史，因此在共享分支时要小心使用。
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 # 符号来创建标题，# 的数量表示标题级别（从 1 到 6） 要创建数字列表，只需在每个项目前加上数字和点号（.） 要创建无序列表，可以使用 *、+ 或 - 符号 要创建超链接，可以使用 链接文本 的格式

实验总结

在完成Git基础学习、Markdown基础语法学习以及将Markdown文件转换为PDF格式的实验后，我获得了以下关键知识和技能： Git基础： 了解Git是分布式版本控制系统，用于跟踪和管理项目的代码变化。 掌握常用Git命令，如git init、git clone、git add、git commit、git branch、git merge等。 理解Git的分支管理、版本历史记录、协作和远程仓库等功能。 Markdown基础语法： 了解Markdown是一种轻量级标记语言，用于创建格式化文档。 学会使用基本Markdown语法，如#（标题）、*（无序列表）、1.（有序列表）、链接文本（超链接）等。 通过这些实验，我能够更好地管理项目、创建格式化文档，并将Markdown文档转换为可分享和打印的PDF文件。这些技能对于日常的软件开发、文档编写和协作非常有用，可以提高工作效率和产出质量。