红色：新增的文件/修改原文件

绿色：暂存区

分支：git branch：查看分支
git branch dev：新建分支
git checkout dev：切换分支，切换分支前先在当前分支提交
git branch –d/–D dev：删除/强制删除分支
git merge dev：合并分支，merge时，会把提交记录一起merge过来
git branch dev commitid：新建分支并且使分支指向对应的提交对象
git checkout –b dev：创建并切换分支

git存储：
git stash：进行存储
git stash list：查看存储
git stash apply stash@{0}：取出存储
git stash drop stash@{0}：删除存储
git stash pop stash@{0}：取出并删除存储

远程仓库：
1. git clone 远程仓库地址：默认只能克隆主分支，如果需要克隆其它分支使用git pull origin 分支名，然后切换分支git checkout 分支名
2. git remote add origin 远程仓库地址：本地仓库和远程仓库关联
3. git push –u origin main：把本地main分支推送到远程main分支
4. git pull orign main：拉取代码
5. git push origin :dev：删除远程分支
6. git branch –M main：修改master分支为main分支
7. git fetch origin main：从远程分支下载所有代码，但是不会将代码和当前分支自动合并，需要手动操作
8. git merge origin/main：合并远程分支，需要加上origin/main
9. git pull：相当于7 + 8，拉取代码并自动合并

变基1：
git rebase –i ID：输入命令后，会进入到vim，s代表合并，pick代表留下，留下最上面一个，其余都是s，:wq后，会进入到vim，提交注释
注意：一定不要rebase已经push到远程仓库的ID

变基2：
在主分支，创建其它分支，在其它分支开发，然后在其它分支先rebase主分支，在切换到主分支，merge其它分支

变基3：
小明在公司的其它分支开发，git add . + git commit –m ''，但是忘记push了。小明回到家，继续在其它分支开发，git add , + git commit –m '' + git push
这次push了。第二天，小明去公司继续开发其它分支，小明应该先git pull把远程代码拉取下来，但是会造成分叉，所以可以使用git fetch + git rebase

变基4：
变基遇到冲突怎么办？莫慌，如果遇到冲突，先解决冲突，然后git add . + git rebase ––continue