实验一 Git和Markdown基础

班级： 21计科03

学号： B20210302312

姓名： 曾靖

Github地址：https://github.com/flojelr/python-class

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
```

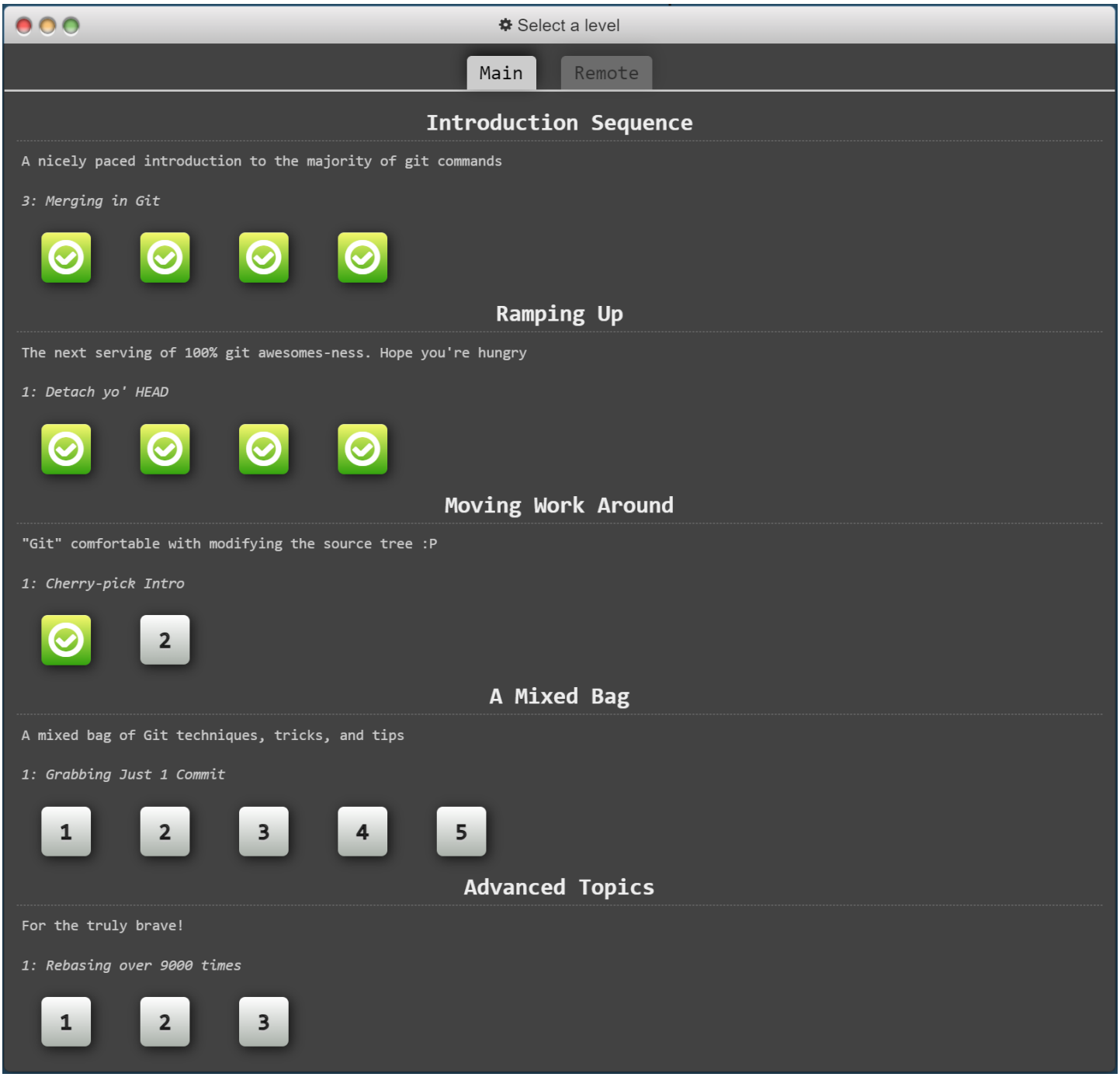
3. 注册Github账号，创建一个新的仓库，用于存放实验报告和实验代码。
4. 安装VScode，下载地址：Visual Studio Code
5. 安装下列VScode插件
    - GitLens
    - Git Graph
    - Git History
    - Markdown All in One
    - Markdown Preview Enhanced
    - Markdown PDF
    - Auto-Open Markdown Preview
    - Paste Image
    - markdownlint

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

第四部分 Markdown基础

查看Markdown cheat-sheet，学习Markdown的基础语法

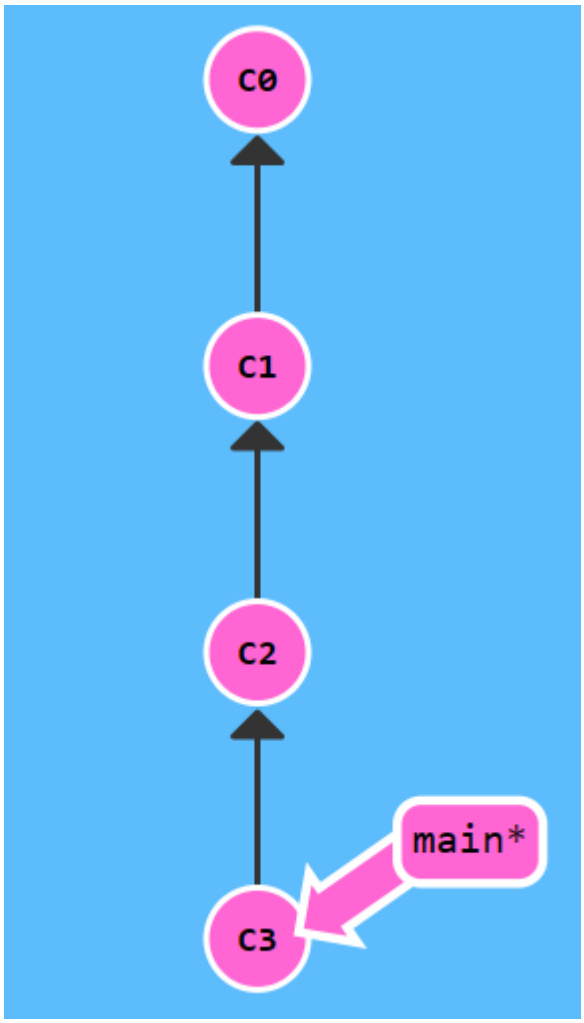
使用Markdown编辑器（例如VScode）编写本次实验的实验报告，包括实验过程与结果、实验考查和实验总结，并将其导出为 **PDF格式** 来提交。

实验过程与结果

一、 关于learngitbranching.js.org的学习

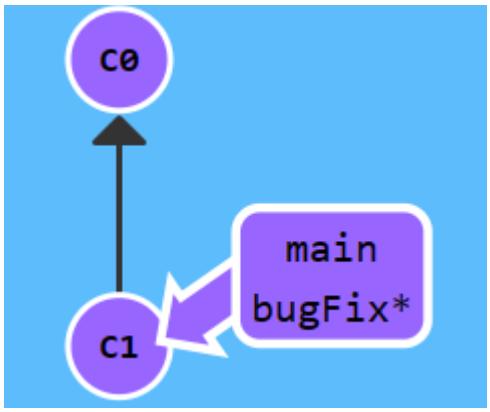
1. git commit 命令的使用
   git commit 命令用于创建下一节点，下面为连续两次使用git commit命令的代码以及结果显示。

```
git commit
git commit
```



2. git branch <分支名>和git checkout <分支名>命令的使用
   git branch <分支名>用于创建分支，git chekout <分支名>用于切换分支，下面为创建一个bugFix分支并切换到bugFix分支的代码以及结果显示。

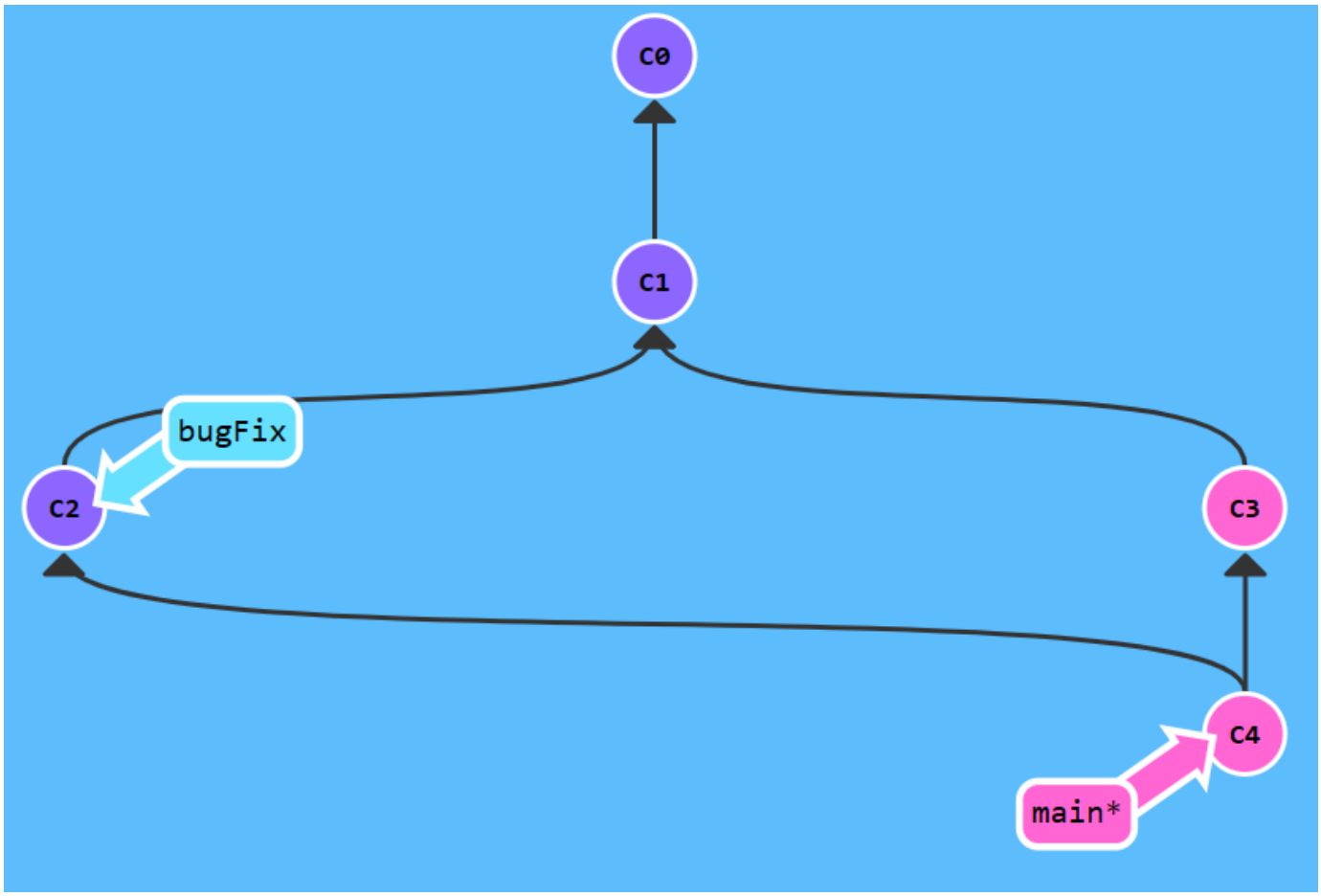
```
git branch bugFix
git checkout bugFix
```



3. git merge<分支名>命令的使用

   git merge <目标分支名>用于合并当前分支和目标分支，下面为创建两个分支后合并分支的代码以及结果显示。

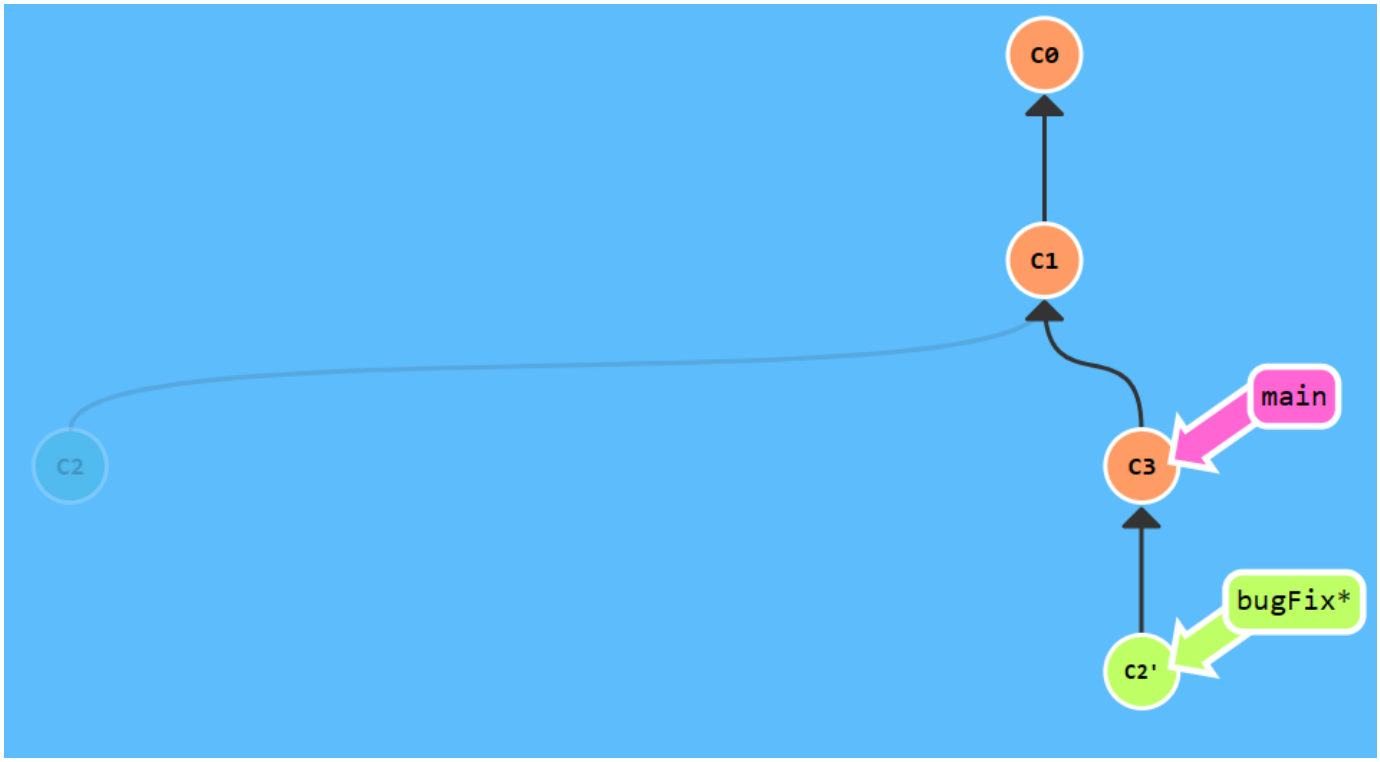
```
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git merge bugFix
```



4. git rebase <分支名>命令的使用

   git rebase <目标分支名>用于将当前分支复制到目标分支下，下面为创建两个分支后将bugFix分支复制到main分支下的代码以及结果显示。

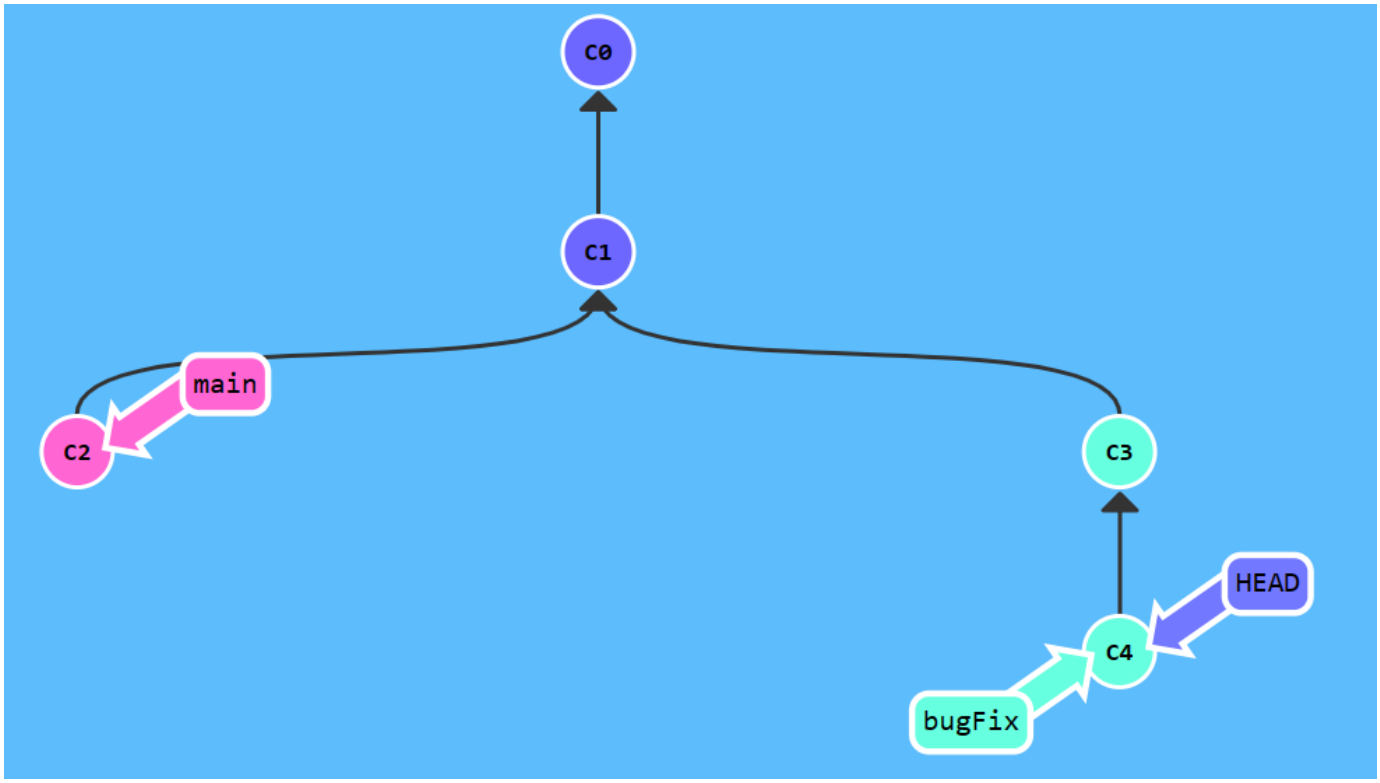
```
git branch bugFix
git checkout bugFix
git commit
git checkout main
git commit
git checkout bugFix
git rebase main
```



5. 分离HEAD

从当前给出的分支中将bugFix和HEAD分离开，下面为代码和结果展示，其中分支已创建好。

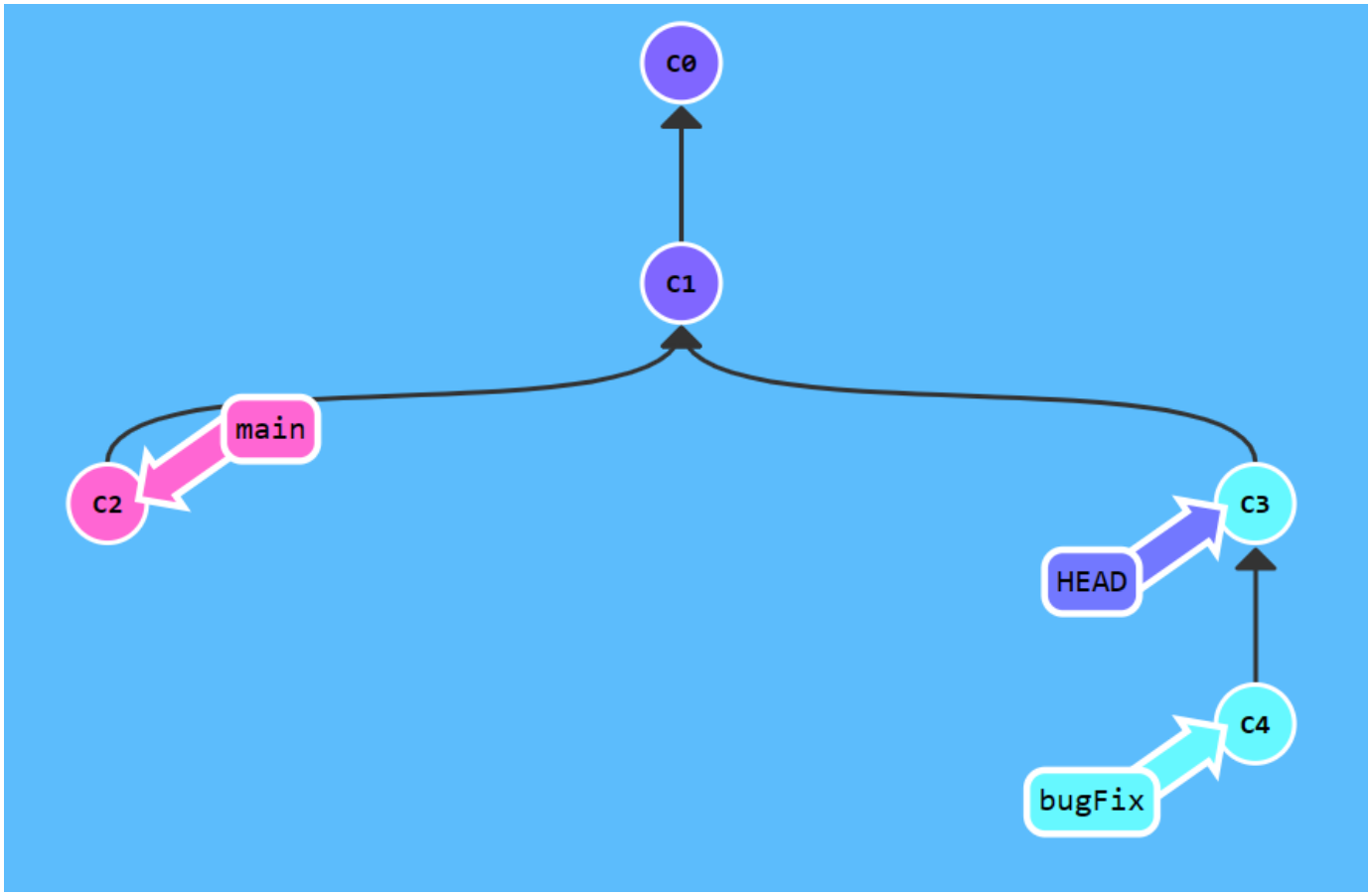
```
git checkout c4
```



6. 相对引用（^）

   git checkout <分支名>^命令可使HEAD分离到分支名的父节点，下面为代码和结果展示，其中分支已创建好。

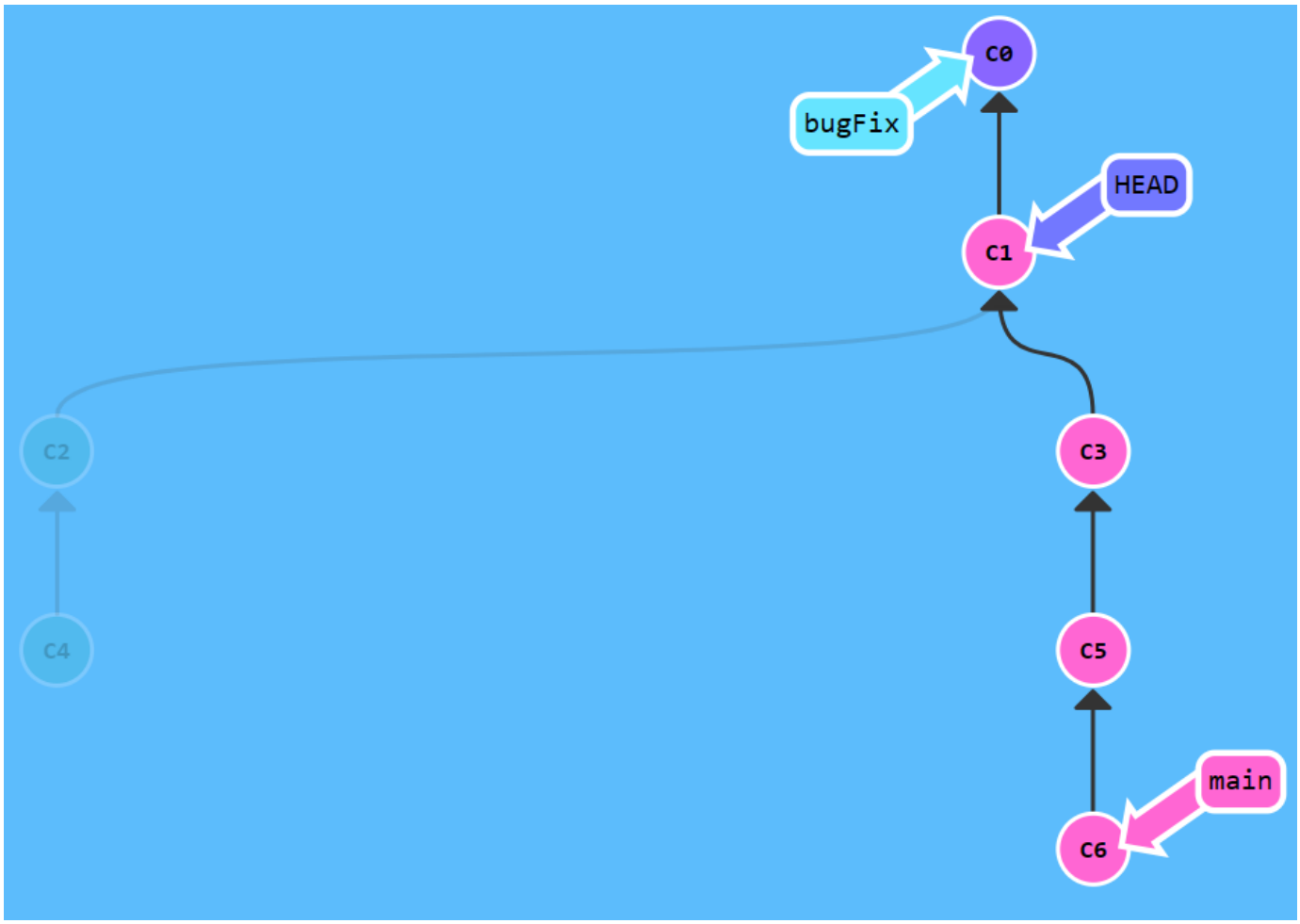
```
git checkout bugFix^
```



7. 相对引用(~)

   git checkout <分支名>~n 命令可使HEAD分离到分支名往上第n个父节点，下面为代码和结果展示，其中分支已创建好。

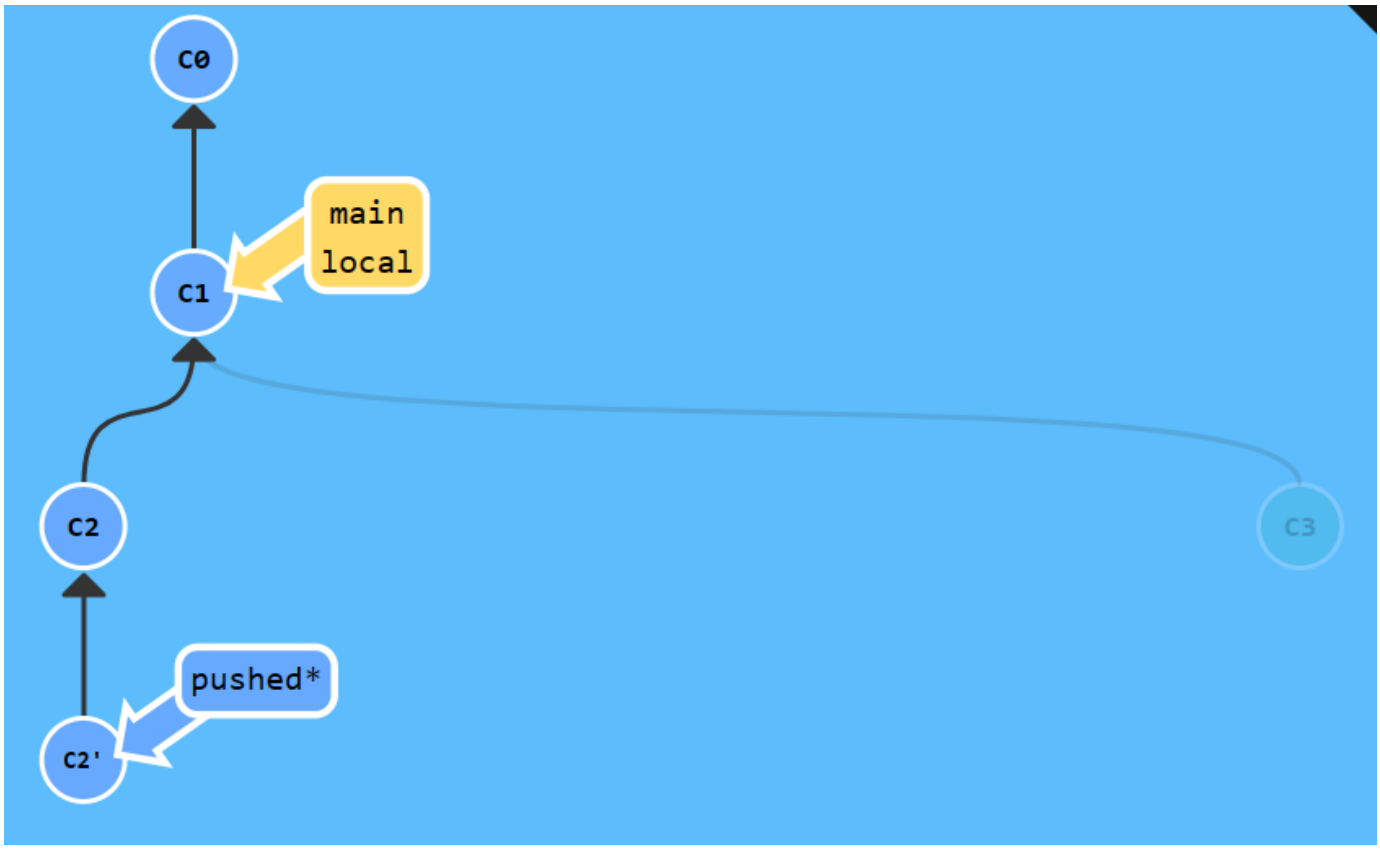
```
git checkout HEAD^
git branch -f main c6
git branch -f bugFix HEAD~1
```



8. 撤销变更

   撤销方式有两种命令分别是git revert <撤回目的分支名> 和git reset <要撤回的当前分支名>，下面为代码和结果展示，其中分支已创建好。

```
git reset c1
git checkout pubshed
git revert pubshed
```



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？
   答：1.Git有能力高效管理类似Linux内核一样的超大规模项目； 2.Git实现了离线开发、代码审核特性，解决了跨地域协同开发中代码质量和编码协同的问题； 3.分支管理功能强大，便于查询和追溯分支间的提交历史。
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）
   答：执行git restore指令，可撤销未Commit的修改；执行git checkout < commit-hash> 指令,可检出已经以前的Commit。
3. Git中的HEAD是什么？如何让HEAD处于detached HEAD状态？（实际操作）
   答：HEAD 是当前分支引用的指针，可以通过git checkout <分支名>使HEAD处于detached HEAD状态。
4. 什么是分支（Branch）？如何创建分支？如何切换分支？（实际操作）
   答：分支是指向快照的一个指针，创建分支使用git branch <分支名>指令，切换分支使用git checkout <分支名>指令。
5. 如何合并分支？git merge和git rebase的区别在哪里？（实际操作）
   答：使用git merge或git rebase指令，git merge指令是将两个分支结合到下一节点，而gie rebase指令则是将当前分支复制到目的分支下。
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）

答：标题：用井号进行分级；数字列表：在每个列表前添加数字并在其后紧跟一个英文句点 . 即可，注意在句点后留一个空格；无序列表：需要在每个列表前面添加减号 - 或星号 * 或加号 +；超链接：链接文本放在中括号内，链接地址放在后面的括号中，链接title可选。

实验总结

对于新工具的使用要敢于尝试，同时也不能认死理在进入思路的死胡同时也要借助其他方式对工具的使用进行了解，别人的经验会让我们少走弯路。