**CSC207 笔记和易错点**

**M0 Git入门**

**什么是版本控制（Version Control）：**

版本控制系统是一种控制我们代码版本的一种方式，它可以追踪我们代码的改动以及帮助我们进行版本复现。

想象一下，在一个基于代码的项目中，你会如何与他人（或跨多台机器与自己）合作：你会四处发送代码的最新版本 .zip 文件吗？你会将修改后的片段发送给拥有主副本的其他人吗？你会使用协作软件（例如 Google Docs）并在同一个地方更新代码吗？

所有提到的可能性都有缺点：要么很难跟踪当前版本或找出每一步的修订版本，要么很难独立工作。如果你犯了错误需要回滚更改，你要么必须找到旧版本或你的代码，要么手动回滚！这太繁琐了！

版本控制没有这些缺点：每个人都在自己独立的版本上工作，版本控制系统允许你根据需要合并更改，并跟踪所做的修订。

-来自教材提供例子

总结，版本控制的目有3：1.保证在多人合作的项目中，每个人的工作不会过多的影响到其他合作者；2.提供一种再多人合作项目下将不同人负责部分整合在一起的方法；3.当程序在一次更新后出现问题后，可以快速的回滚至旧版本

**版本控制如何工作：**

在一个版本控制系统中，将会有一个主储存库（master repository）：这是一个对于所有文件最后一次更新的复制文件。合作者须将这些文档（库）复制一份到他们的本地运行环境上，然后独立完成各自工作。

当合作者完成一些修改并希望将它们提供给其他合作者时，他们需要首先**push**他们的修改。相应的，如果需要获取最新的库存，则需要**pull**这些修改。

虽然各个版本控制系统有不同的特性，但是了解以上基础知识是学习如何使用版本控制系统的前提。

**Git入门**

Git是一种常用的版本控制系统

在git中，主存储库通常被称为**Origin**。我们常使用git clone <url>来获取一份本地库的复制。此命令类似与下载所有本地文档，但是提供了一些特殊的方法（git status）来进行文档修改的追踪。

1 修改初始文档

如果我们希望将一些修改后的文件更新至主存储库中，我们可以使用git add <files>来添加所有我们希望修改的文件。

在完成add命令后，我们需要紧接着使用git commit -m “<message>”来提供我们此次修改的描述（例如“修改了xxx”），方便后续版本回退时进行更快捷的查找。最后，当我们完成这些所有的工作后，使用git push进行最终提交

总结：如果你需要进行一次对主存储库的更新，请按照以下步骤完成：

git status:来查看那些文档被修改了（将会暂时工作路径和暂存区的情况，例如修改和暂存文件）

git add <file>:提供需要被提交的修改的文档（此时文档并没有被正式的更新至主存储库，此行为被称为暂存更改）

git commit -m “<message>”:为你的更新提供描述（将会被永久展示在主存储库的修改提交信息中）

git push:提交目标更改至主存储库

2 拉取新版本

git pull命令常用于获取主存储库的更新（将最新的主存储库内容复制到你的本地环境）

请注意，此操作可能会导致一些文档冲突或者其他错误。你需要通过adding/committing 本地修改或者手动处理冲突（常见于两个合作者对同一个文档进行修改的情况）

3 分支

有时你可能会在两个不同的库存来处理整个功能，此时如果将你的内容提交至主库存可能不符合你的需求（比如你的部分是完成库存中的某个子文件夹内容程序编写，你不应该将这些文档重新提交至主库存，这会改变库存的原始结构，同时对于这些子库存的修改也不应该在主库存中被看到）。

对于这种情况，你可以创建一个新分支，等待你将所有的工作完成后，再将他们整个提交至主库存中。

如果需要完成以上操作，首先使用git checkout -b <branch name>。或者git branch <branch name> + git checkout <branch name>（第一句命令用于创建新分支，第二句命令用于将你的修改传送至分支而不是主库存）。

同理，如果需要提交文档到其他的分支，使用git checkout <branch name>即可。

当完成分支任务，需要将文档融合至主库存时：git merge <branch name>。然而，更好的方法是发出拉取请求，以便人们可以看到和审查您尝试推送的更改。（？）

4 其他资源

可视化理解版本控制：<https://betterexplained.com/articles/a-visual-guide-to-version-control/>

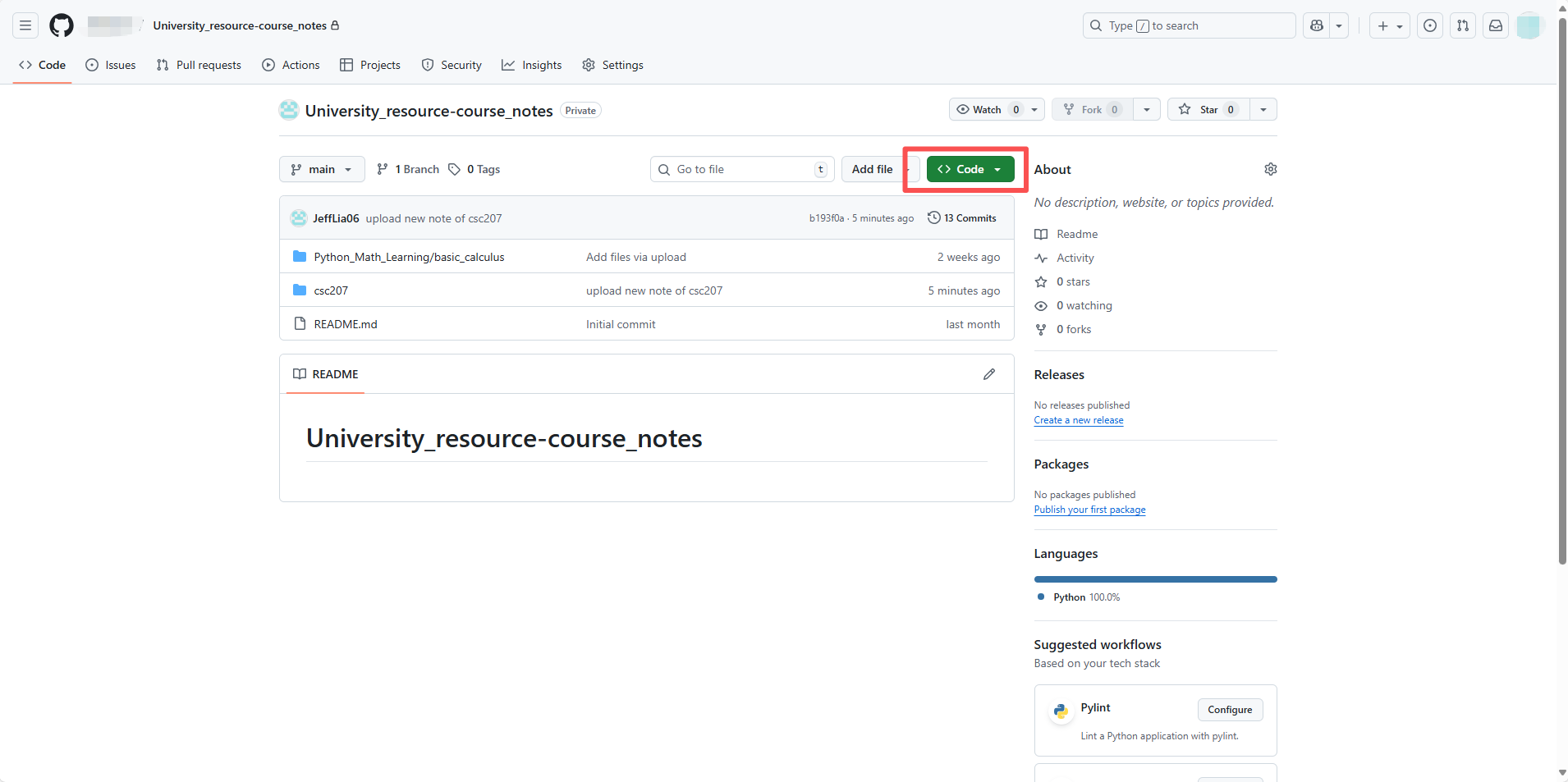
Github入门：<https://docs.github.com/en/get-started/start-your-journey>

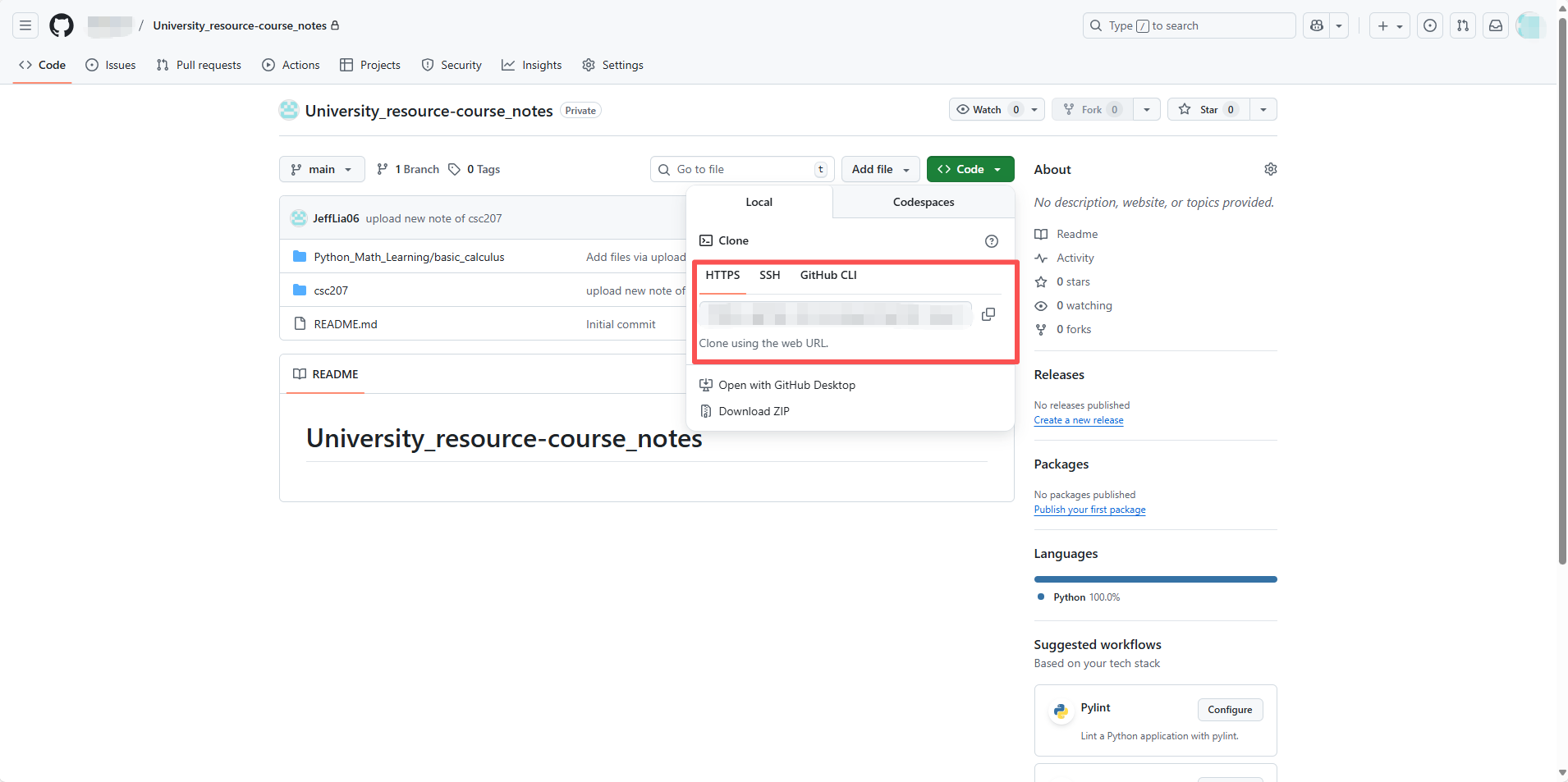
5 实战使用技巧

首先使用github来进行版本控制时，有几个额外步骤：

复制目标库存（若第一次使用库存）：git clone <url>：

url在此处：





完成库存复制后，之后所有的关于此库存的修改须cd至此文件夹进行。

若是此电脑第一次使用，需要通过网页端或者代码进行登录。同时在尝试commit时需要提供一个提交者名字和邮箱，具体代码如下：（此操作为个人电脑私人环境）

git config --global user.name "你的名字"

git config --global user.email "你的邮箱"

同时，git只能检测到主文件夹的更新，它无法对新创建的子文件夹的内容进行检查，须使用git add进行操作（首先使用git status查看具体哪个子文件夹未被添加，然后手动添加git add “文件夹”和commit + push）

需要注意一点，由于不适用git pull的话github不会自动更新本地库存，从而导致本地库存和云端有冲突文件。如果遇到了冲突文件的情况，则需要更具具体情况进行文件融合的工作。为了防止此类情况的发生，建议每次提交之前使用git pull对本地文档进行更新，从而避免单独处理冲突。

M1 Java 入门

从对象化软件设计的角度来讲，Java对比Python有以下几个优势：

1. Java现在依然是最广泛使用的计算机语言在工业中
2. 相比于Python的对象化编程，Java包含了更深入的对象化编程的概念，利于初学者
3. Java作为一个学习难度处于C和Python之间的语言，学习Java可以帮助理解其他更加复杂的语言

如何学习Java

1 Java如何运行一个程序

对于计算机来说，他们无法直接理解我们写的代码，通常需要将代码转化为一种机器语言从而进行运行。

这种转化过程通常来说有两种主要方法：

解释（Interpretation）：程序将被一行一行的进行阅读和运行（例如Python）

编译（Compilation）：整个程序会先被转化机械代码通过编译器，然后转化为一个可运行程序

对于Java来说，采取了一种混合的方法。Java的编译器（javac）会首先将源代码转化为字节码。然后Java虚拟机（Java Virtual Machine，JVM）将会解释和优化字节码在运行时。（在终端中，例如假设你有个程序 test.java，你可以使用javac test.java对它进行编译，生成一个test.class文件，随后就可以使用java test.java来运行代码）

为了运行一个Java程序，你首先需要编译它。

但是对于一个更大的项目，通常包含上百个类，使用javac进行一个个编译会变得非常麻烦。因此，现代Java程序通常会使用一些构建系统（Build System）例如Maven/Gradle。这些工具会帮助管理编译，依赖，和运行；从而让程序员可以专注于代码的写作上（这些功能通常已经被集成在现代的IDE上并自动运行）。但是作为计算机科学的学生，了解这些底层逻辑是必要的，并且可以有效的辅助后期C的学习。

2 Java的计算机架构介绍

为例理解一个Java程序是如何运行的，我们有必要了解一下计算机的架构：

应用（Application）

操作系统（Operation System，OS）

硬件（Hardware）

对于像C一类的语言，它让程序直接在OS上运行，因此代码并不是可移值的，这意味着每一次在不同的OS上都需要对程序进行重新编译。

而对于Java，则采取了一个虚拟机架构（Virtual Machine Architecture）：

一个虚拟机（Virtual Machine，VM）被置于应用和操作系统之间，充当一个固定的运行环境，而不是让程序直接在OS上运行。因此，使用VMs的编程语言所写的程序是可移值的，只要合适的Java VM被安装。

应用

虚拟机

操作系统

硬件

通过这个方法，Java的程序可以不用考虑操作系统进行运行。但是，Java的VM则是一种在OS上运行的一个程序，因此OS的一些特征依然会影响Java程序的运行。这些细节依然会被Java的程序自动处理，因此编程者无需担心这些问题。

Java 基础

1 定义类

2 定义方法

3 打印

Java的变量和类型

1 对比Python的区别

2 声明类型

3 声明与赋值

4 Java如何追踪变量

5 常见报错

Java的引用类型和原始类型

1 常用Java数据类型

2 引用类型与原始类型的区别

String类

1 String Pool/String Interning

2 String是不可变的（immutable）

3 String的常用方法

4 创建可变String的方法：StringBuilder

5 单字符String：char

6 String对比StringBuilder

Java的类

1 类初始化

2 APIs

3 调用方法

4 类方法

5 访问数据成员

6 高效管理对象

Arrays（数组）

1 数组的声明

2 构造数组

3 Java数组的长度是不可变的

4 访问数组所有元素

5 为什么Java的数组会有诸多限制

6 二维/高维数组定义

Aliases（别名）

1 设置别名和它的作用（定义）

2 通过引用来创建别名

3 别名无法用于原始类型

4 别名的副作用

5 别名副作用的解决技巧

6 浅拷贝和深拷贝

7 其他的常见副作用

控制结构（Control Structure）

1 if判断结构组

2 for循环组

3 while循环组

4 do…while循环组

参数

1 通过参数传递一个原始变量