Git命令整理：

新手教程中使用到的命令：

ls: 查看你目前所定位的文件夹中的文件/文件夹

example:F:/Git/目录下有一个test1文件夹

LI@LI-PC MINGW64 /F/GIT

$ ls

test1/

LI@LI-PC MINGW64 /F/GIT/test1 (master)

$ git add test.txt

下面语句的意思是 给你刚刚上传的文件一个备注，方便查找记忆而已

LI@LI-PC MINGW64 /F/GIT/test1 (master)

$ git commit -m "cc"

[master a547638] cc

1 file changed, 0 insertions(+), 0 deletions(-)

create mode 100644 test.txt

登录github并上传文件

LI@LI-PC MINGW64 /F/GIT/test1 (master)

$ git push origin master

建立并初始化仓库—默认在C:\Users\LI(用户电脑名)\,路径下创建并初始化仓库

$ mkdir git-tutorial

$ cd git-tutorial

$ git init

Initialized empty Git repository in /Users/hirocaster/github/github-book

/git-tutorial/.git/

修改仓库的创建位置

$ cd //返回初始的位置

$ cd /F //设置F盘为当前路径

$ cd /F/test //当前路径F:\test\

//在当前路径重新创建并初始化仓库

$ mkdir git-tutorial

$ cd git-tutorial

$ git init

**git status**命令用于显示Git 仓库的状态

**git add**——向暂存区中添加文件

要想让文件成为Git 仓库的管理对象，就需要用git add命令将其

加入暂存区（Stage 或者Index）中。暂存区是提交之前的一个临时区域

Example:

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git add README.md

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: README.md

README.md 文件显示在Changes to be committed 中.

**git commit**——保存仓库的历史记录,将当前暂存区中的文件实际保存到仓库的历史记录中.

下例只简洁地记述了一行提交信息，如果想要记述得更加详细，不加-m，直接执行git commit命令

在编辑器中记述提交信息的格式如下。

● 第一行：用一行文字简述提交的更改内容

● 第二行：空行

● 第三行以后：记述更改的原因和详细内容

以#（井号）标为注释的行不必删除

-m 参数后的"First commit"称作提交信息，是对这个提交的概述

Example:

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git commit -m "First commit"

[master (root-commit) 254277d] First commit

1 file changed, 2 insertions(+)

create mode 100644 README.md

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git status

On branch master

nothing to commit, working tree clean

执行完commit,当前工作树处于刚刚完成提交的最新状态，所以结果显示没有更改

**git log**——查看提交日志

可以查看以往仓库中提交的日志

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git log

commit 254277d5b0b2cd3ed62209ebd4bf38e08a1b8b00 (HEAD -> master)

Author: lirong <1269471727@qq.com>

Date: Wed Oct 10 21:10:29 2018 +0800

First commit

如果只想让程序显示第一行简述信息，可以在git log命令后加上--pretty=short

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git log --pretty=short

commit 254277d5b0b2cd3ed62209ebd4bf38e08a1b8b00 (HEAD -> master)

Author: lirong <1269471727@qq.com>

First commit

只要在git log命令后加上目录名，便会只显示该目录下的日志。如果加的是文件名，就会只显示与该文件相关的日志

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git log README.md

commit 254277d5b0b2cd3ed62209ebd4bf38e08a1b8b00 (HEAD -> master)

Author: lirong <1269471727@qq.com>

Date: Wed Oct 10 21:10:29 2018 +0800

First commit

如果想查看提交所带来的改动，可以加上-p参数，文件的前后差

别就会显示在提交信息之后

$ git log –p README.md

$ git log –p

**git diff**——查看更改前后的差别

打开README.md，修改首行为#Git教程

//绿色的为差异

。“+”号标出的是新添加的行，被删除的行则用“-”号标出

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git diff

**diff --git a/README.md b/README.md**

**index 6c1c9ad..e8da7aa 100644**

**--- a/README.md**

**+++ b/README.md**

@@ -1,2 +1,2 @@

-# test1

+#Git<BD>̳<CC>

testwork

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git add README.md

LI@LI-PC MINGW64 ~/git-tutorial (master)

$ git diff HEAD

**diff --git a/README.md b/README.md**

**index 6c1c9ad..e8da7aa 100644**

**--- a/README.md**

**+++ b/README.md**

@@ -1,2 +1,2 @@

-# test1

+#Git<BD>̳<CC>

testwork

建议：在执行git commit命令之前先执行，git diff HEAD命令，查看本次提交与上次提交之间有什么差别，等确认完毕后再进行提交。这里的HEAD 是指向当前分支中最新一次提交的指针