Lab Sheet 1

Installing Git on Windows

Pleasenote:You will needadministrator's rights to install the Git software. If you do not have Administrator's right, please contact your computer support team.

1. Navigate to http://git-scm.com/

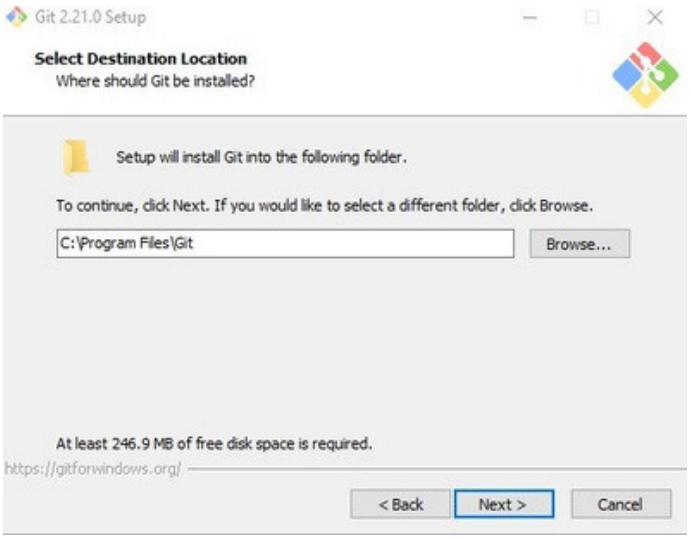
   The webpage will auto-detect what operating system. Look for the computer screen on the right-hand side of the page. Click on Download 2.21.0 for Windows. Your computer should automatically launch an explorer window and ask you where to save the program on your computer.

2. Navigate to where you saved the Git program .exe file. Right-click on the file and choose 'Run as Administrator'. If asked, enter in administrator credentials. This should launch the installer.

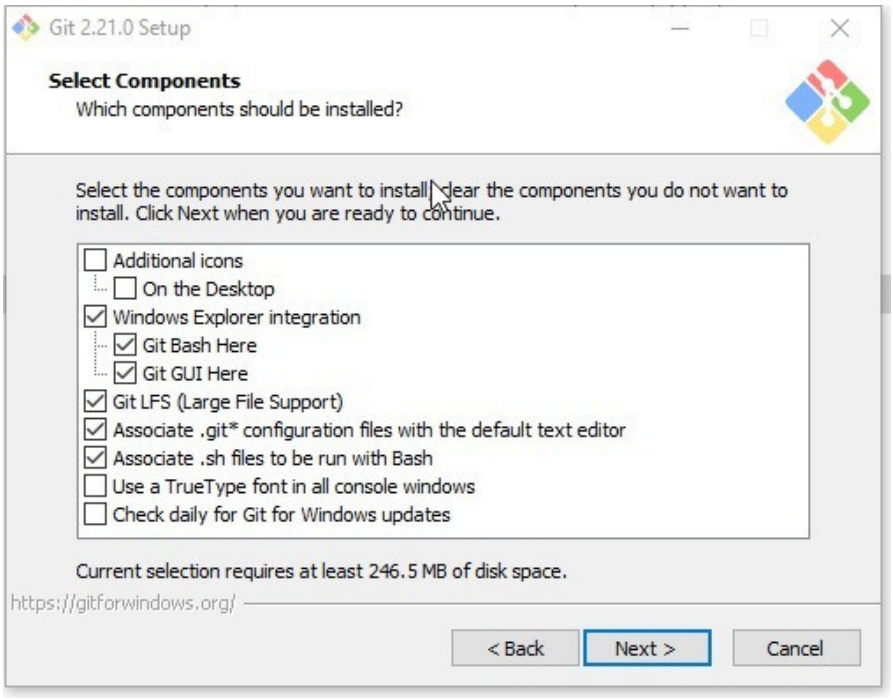
   You may receive a security warning about the software being from an unknown publisher. You can safely ignore this message. Click the Run or Yes button to continue or cancel to exit out of the installation.

3. The Git License will display. Read through and then Click Next to accept the terms of the license.

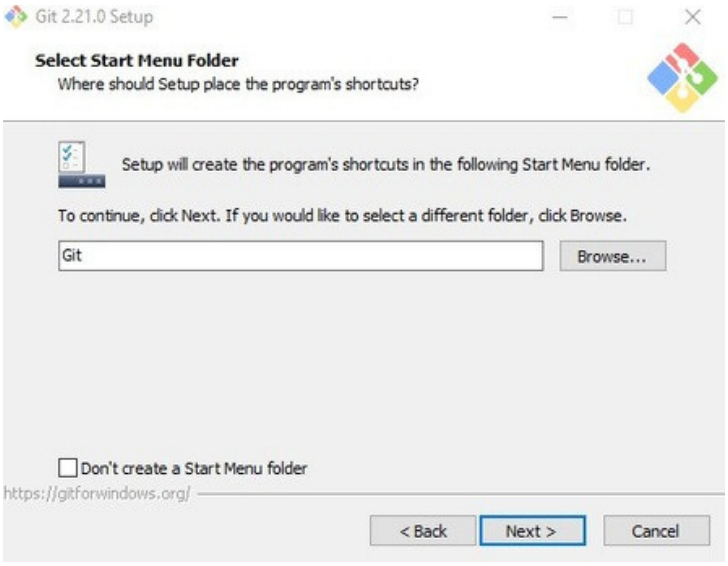
4. Choose where Git will be installed. Click Next to choose the default location provided unless you know you need to install the software somewhere else on your machine.



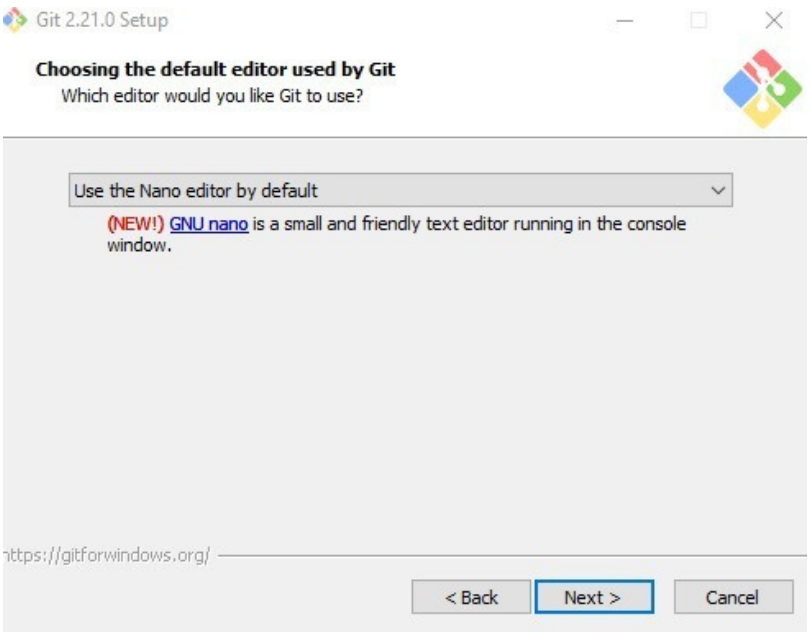
5. Select the components you want installed along with the Git software. There are already some default options checked. Make sure to check: Windows Explorer integration, including Git Bash Here and Git GUI Here. This component will allow you to use the Windows Explorer context menu to access git Bash and Git Gui. All other options are optional. See screenshot below.



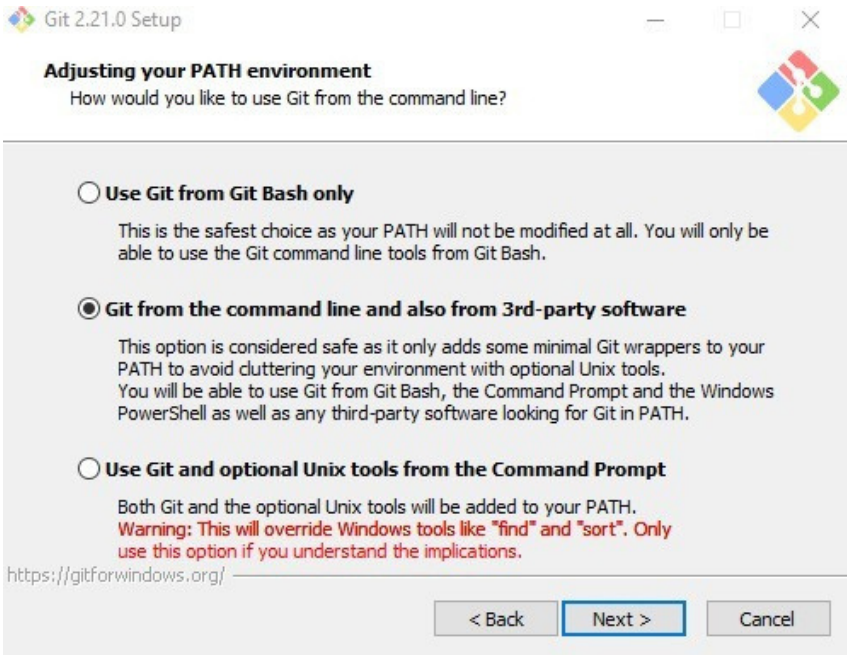
6. Select your Start menu Folder. Choose Next to use the default.



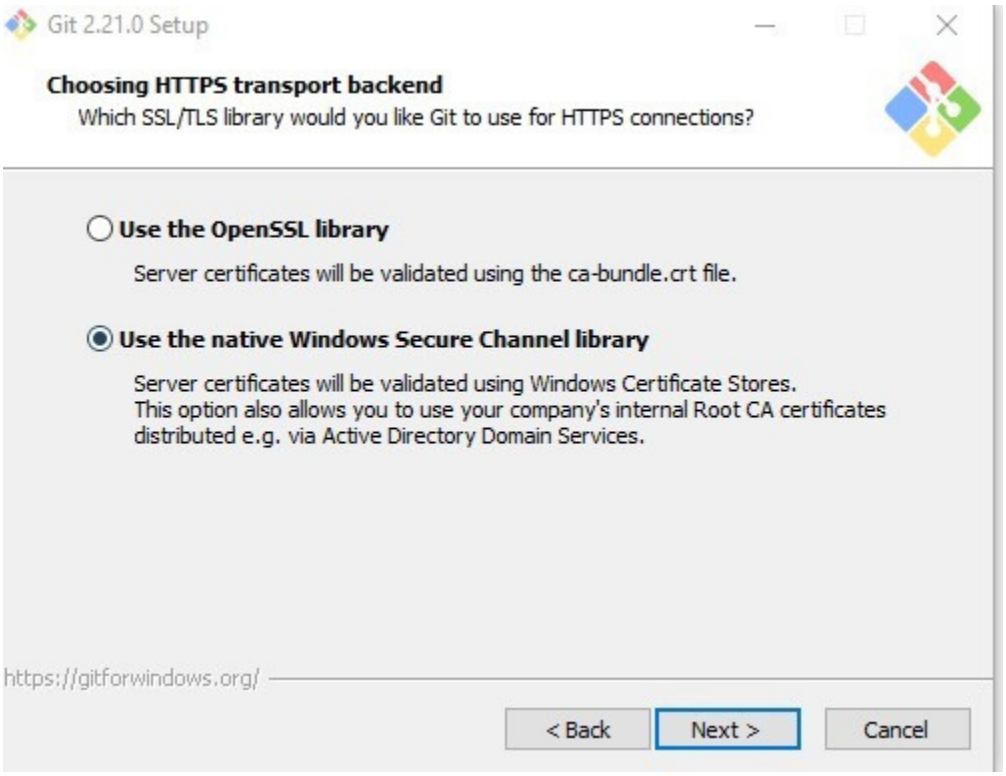
7. Choose which default editor you would like Git to use. For this class, choose ‘Use the Nano editor by default’ and click Next



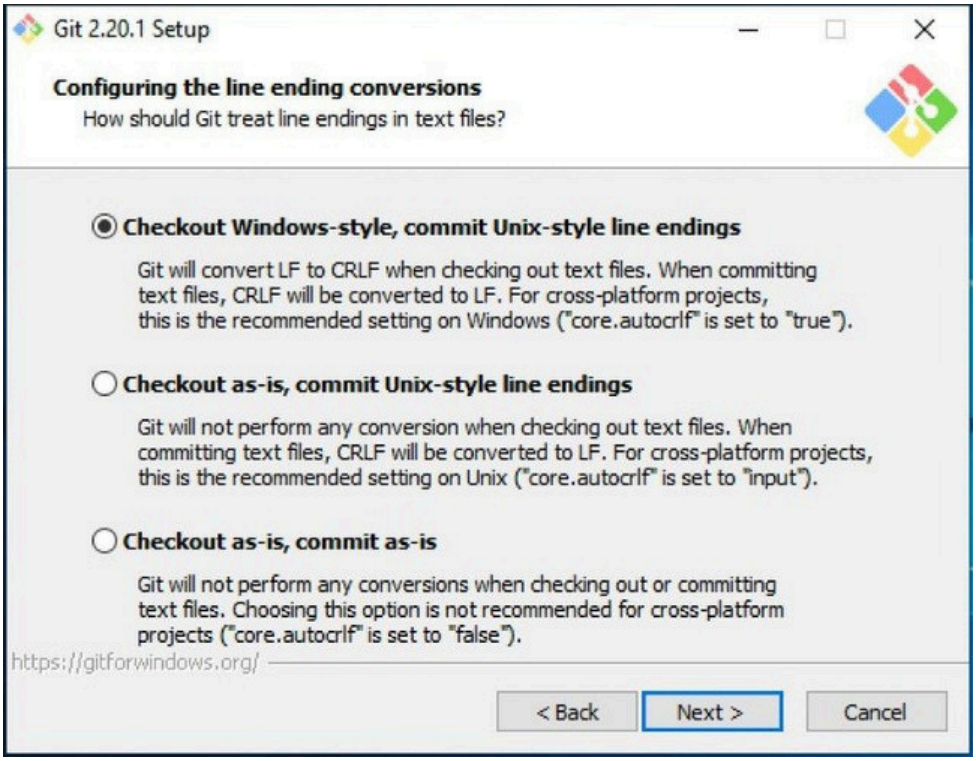
8. Choose how you would like to use Git from the command line. You can choose 'Git from the command line and also from 3rd-party software' or 'Use Git from Git Bash only' See screenshot.



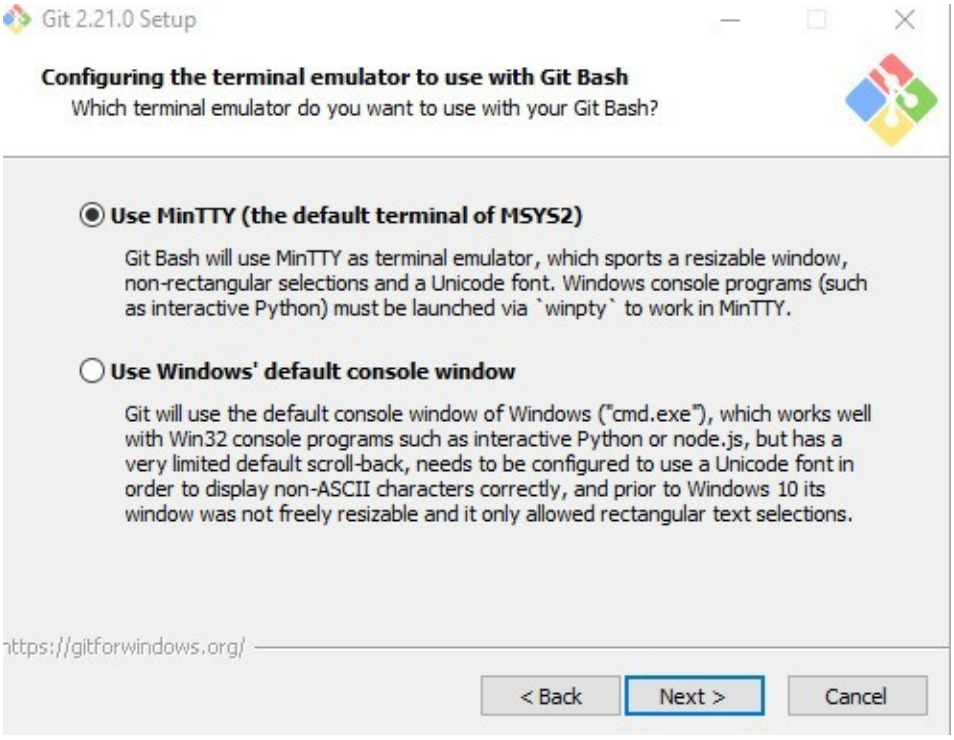
9. Choose a SSL/TLS library. Select 'Use the native Windows Secure Channel library. This allows Git to use the certificates that are native to your machine and may avoid a path problem later on. See screenshot.



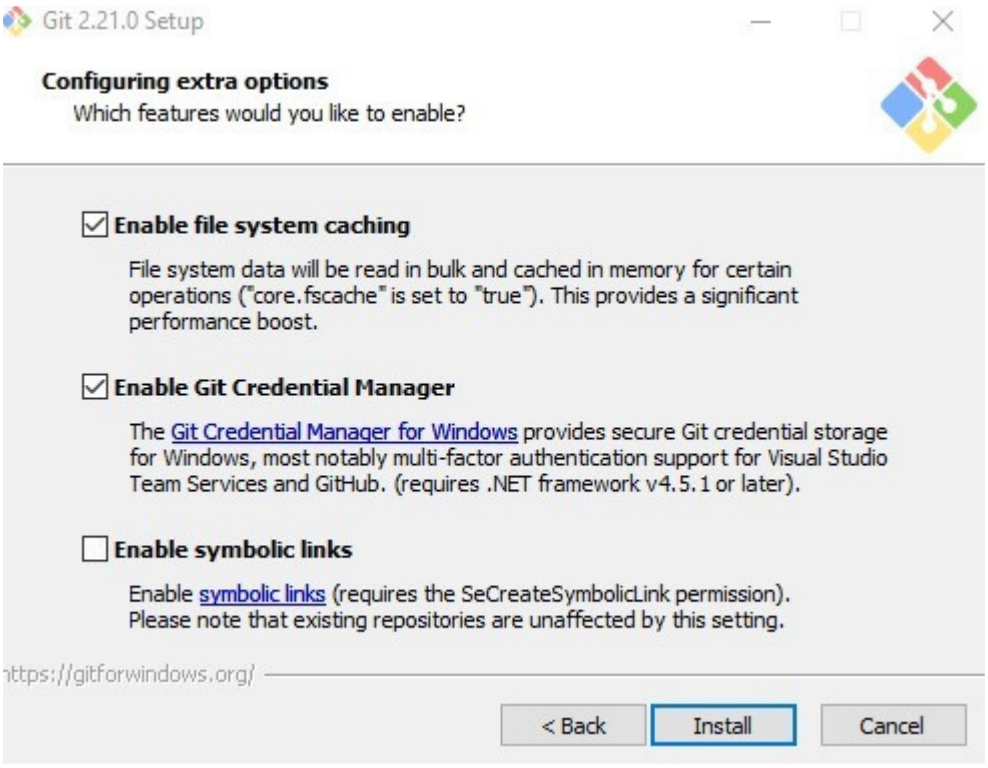
10. Choose an option for how Git should treat line endings in text files. Recommended to keep the default as this is better if you're planning on sharing your project with others who may be using a different operating system. Choose 'Checkout Windows-style, commit Unix-style line endings'. See screenshot.



11. Configure your terminal emulator to use with Git Bash. Use the default 'Use MinTTY'



12. Configure extra options. Check: Enable file system caching and Enable Git Credential Manager



13. Choose Install to start the installation.

14. Check or uncheck the boxes to launch Git Bash or to view the release notes. Click Finish to exit the installer.

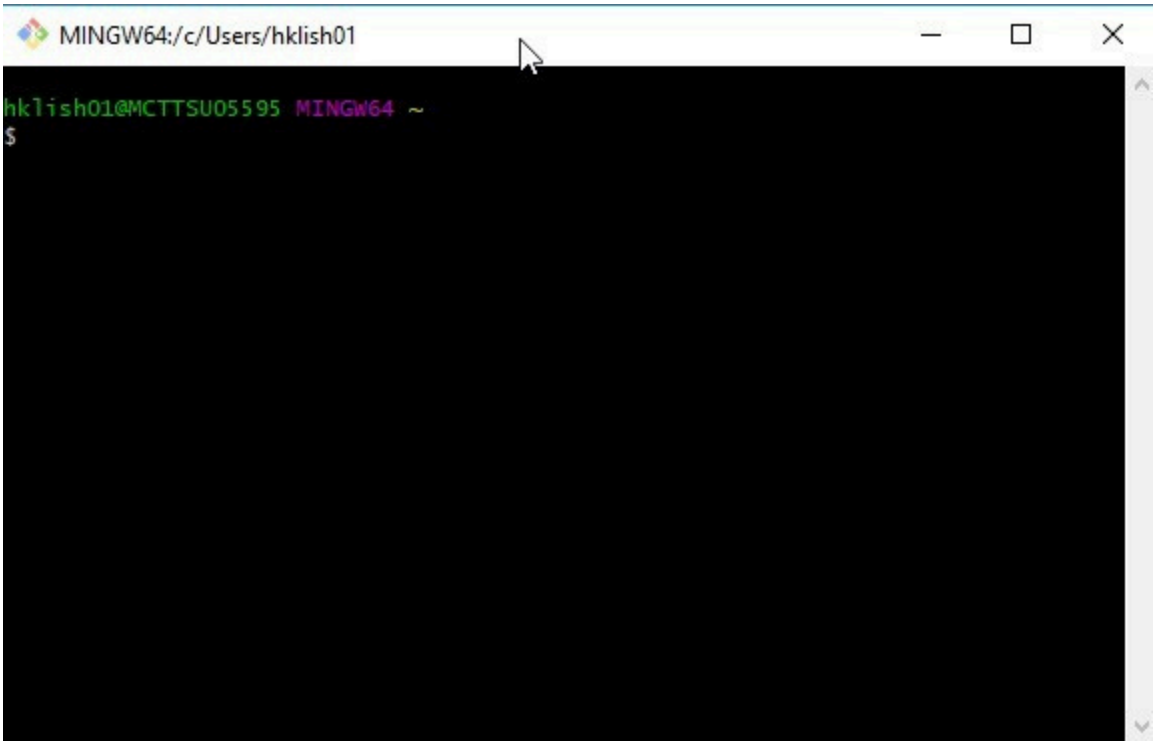
Once installed, you can access Git bash or the Git GUI via two different ways:

1. Open the Start menu, you'll find a new Git entry with the Git Bash icon

2. Best Way: Navigate to a folder on your machine where you want to use Git

    a. Right click inside the folder. You'll notice two new options in your context menu: Git Bash and Git Gui . Choose Git Bash.

Either way, you will see the following when you open Git Bash.



Note that you can't use Git in directories that are from Box Sync or Box Drive.

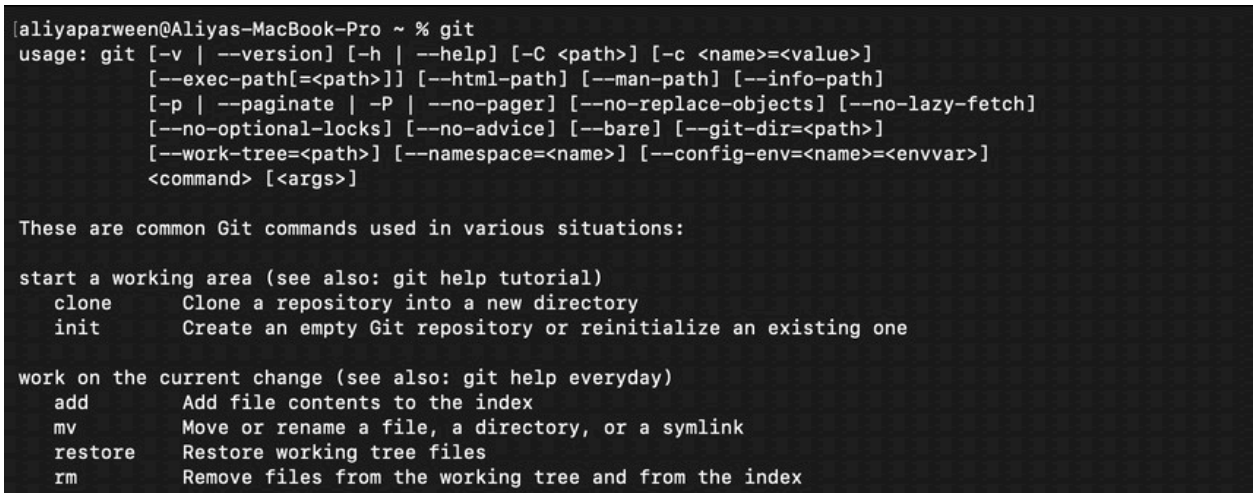
After the class, if you want to change from Nano to using Notepad ++ as your text editor, see these instructions. https://www.toolsqa.com/git/set-up-notepad-for-git-bash/

Lab Sheet 2

Git is not just a tool for versioning files—it was designed with collaboration at its core. Back in 2005, Linus Torvalds needed a lightweight and efficient system to manage the large number of patches submitted to the Linux kernel by countless contributors. His goal was to create a tool that allowed him and hundreds of developers to work together seamlessly without chaos. This practical approach led to the creation of a robust mechanism for sharing repositories across multiple computers, without depending on a central server.

Running our first Git command

Open a prompt and simply type git (or the equivalent, git --help), as shown in the following screenshot



Git Config

Thegit config command sets the git user configuration. This is usually an important and first command to execute once installed git. It set user information such as the author and the email address associated with it. The username and email are very crucial and useful during commits on the repositories.

The git config also helps set other parameters such as the default text editors.

General General syntax

$ git config --global user.name "Username"

$ git config --global user.email “user@someone.com”

Setting up a new Local repository The first step is to set up a new repository (or repo, for short). A repo is a container
for your entire project; every file or subfolder within it belongs to that repository,

in a consistent manner. Physically, a repository is nothing other than a folder that contains a special. git folder.

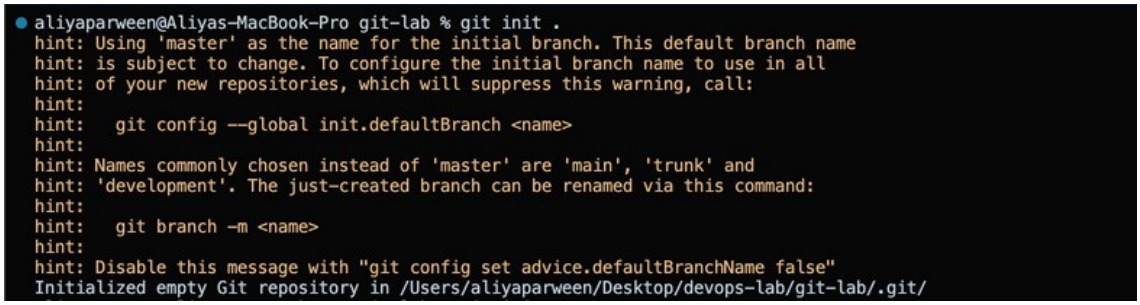
Open the Visual Studio Code and upload folder

Open the terminal, select command prompt and set git repo using the command

Git Init

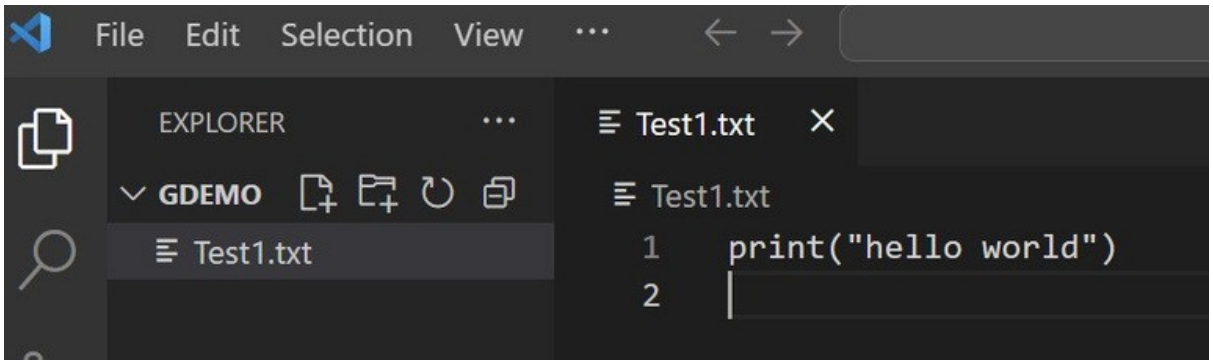
The git init command initializes a directory into a git local repository. General General syntax

$git init.



Git Add

Create a file in local repo



Check the status of file test1.txt

```
aliyaparween@Aliyas-MacBook-Pro git-lab % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        test1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

The git add command adds all the changed files into a staging or indexing area before committing.

General syntax

$ git add Filename

```
aliyaparween@Aliyas-MacBook-Pro git-lab % git add test1.txt
aliyaparween@Aliyas-MacBook-Pro git-lab % git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test1.txt
```

To add multiple files or all files within the directory.

$ git add *

Git Commit

The Git commit command is useful in two main areas.

General syntax $

git commit -m

git commit -m “message”

It records the file permanently in the version history with a specific message.

The -m flag requires a string argument as the message.

```
aliyaparween@Aliyas-MacBook-Pro git-lab % git commit -m "file added"
[master (root-commit) 6c53162] file added
 1 file changed, 1 insertion(+)
 create mode 100644 test1.txt
```

Git Status

The git status command displays the current state of the working directory in the staging area. It enables any changes moved to the staging area, those that have changed but are not in staging, and those files not being tracked by Git.

```
aliyaparween@Aliyas-MacBook-Pro git-lab % git status
On branch master
nothing to commit, working tree clean
```

Git Log

The git status command does not, however, give information about previous commits within the project history. This requires the use of the git log command.

```
aliyaparween@Aliyas-MacBook-Pro git-lab % git log
commit 6c53162eff4f8f08205c6225626cad17a09b4eb1 (HEAD -> master)
Author: aliya-parween <aliyaparween1205@gmail.com>
Date:   Wed Nov 19 16:21:49 2025 +0530

    file added
```

Git Push

Git push is another useful command within the git environment. It uploads a local repo to the remote or master repository.

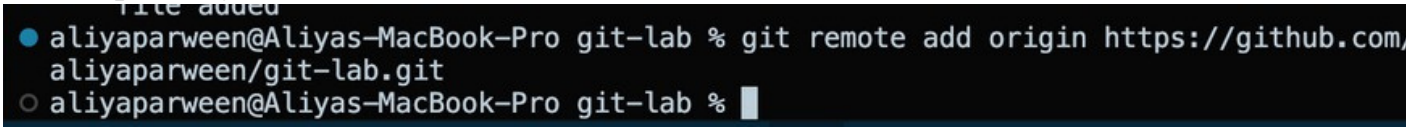
General syntax

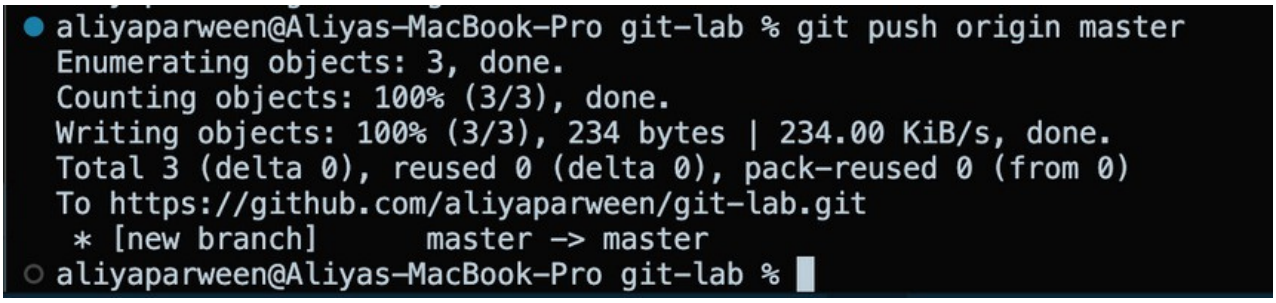
$ git push master

The above command pushes the changes made to the master branch in the remote repository.

$ git push -all

This command pushes all the changes within the entire repository to the remote repository.





Git Pull

Thegit pull command downloads or receives data from remote git hosting services and merges it to the local working directory.

General syntax

$ git pull URL

Git Commands – Remote Repositories

InGit,a ***remote*** refersto another computerthat hosts the same repository as local repo. Any computer within a shared network containing that repository can serve as a remote for others.

So, a remote Git repository is nothing other than a remote copy of the same Git repository created locally. This repository can be accessed via common protocols such as SSH, HTTPS or the custom git.

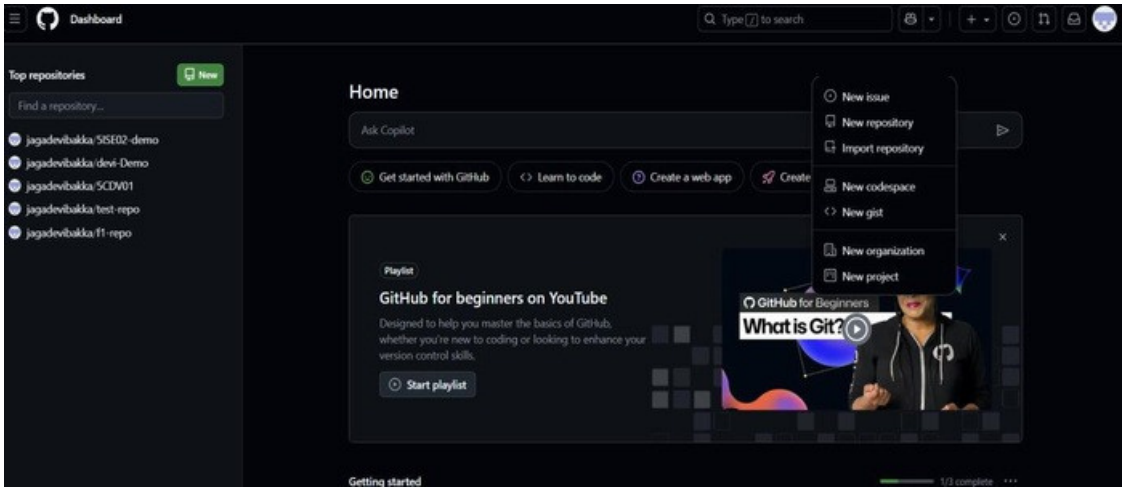
To start working with a remote

1.Setting up a new GitHub account

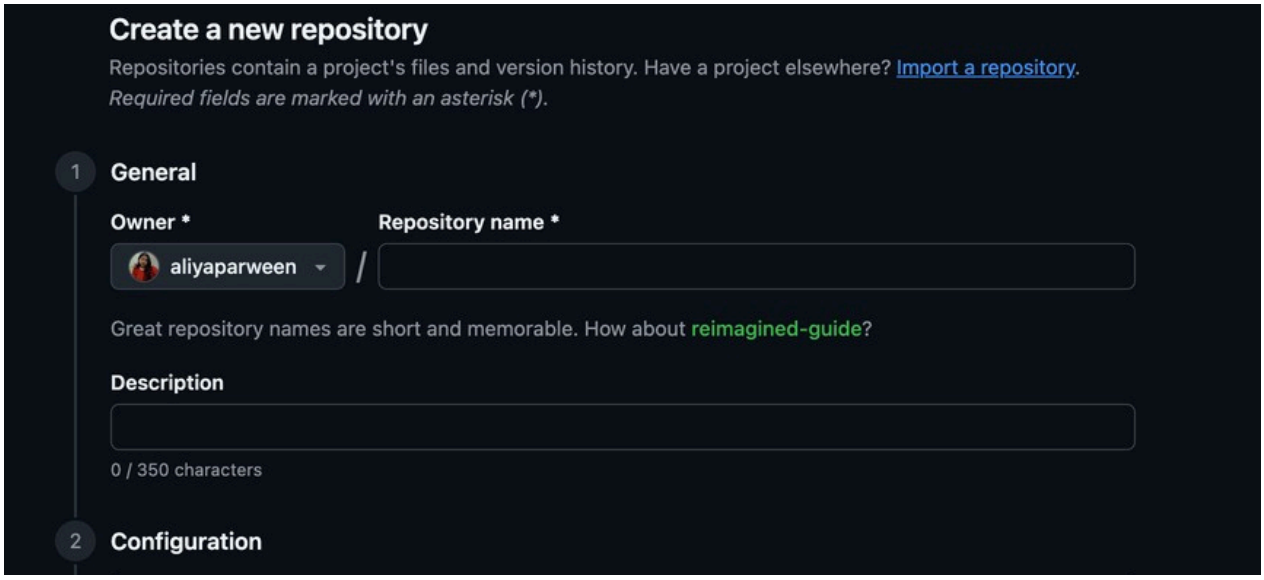
GitHub offers unlimited free public repositories. Create a new account using the following steps:

1. Go to https://github.com.
2. Sign up using your e-mail.

2.Create a new repository

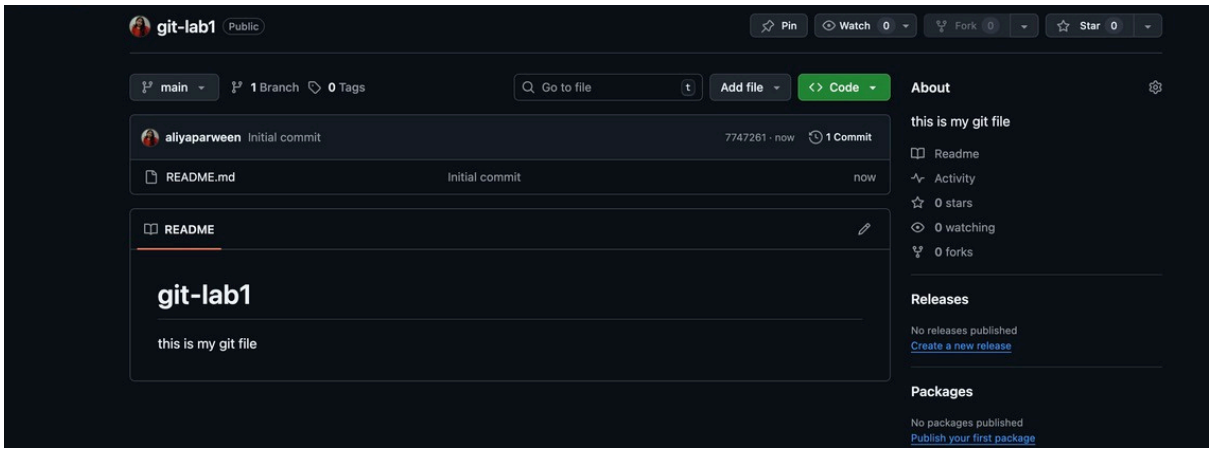


Add repository name, Then, write a description for repository. (This is useful to allow people who come to visit your profile to better understand what your project is intended for). create repository as public Then, initialize it with a README file or upload the file from local drive. Choosing this GitHub makes a first commit for us, initializing the repository that is now ready to use.



3.Cloning a repository

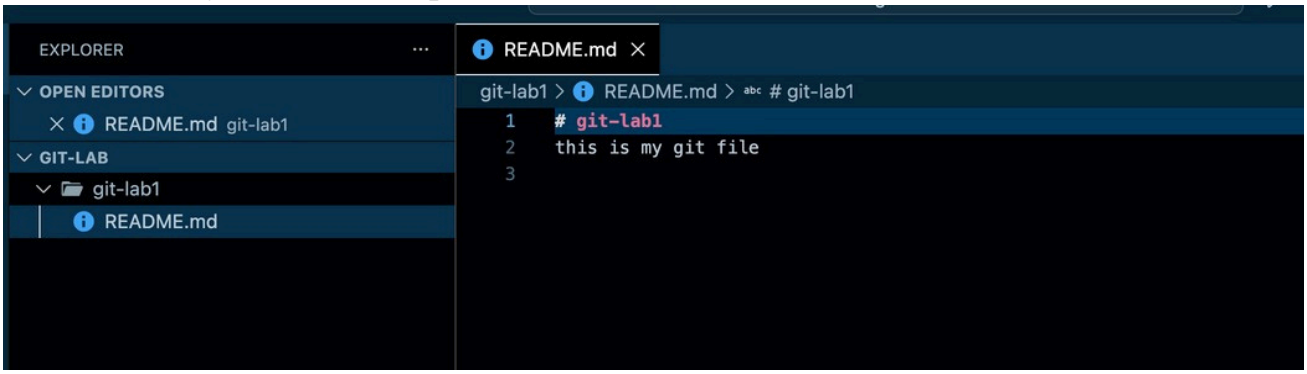
Git provides the git clone command. copy the URL, to copy the URL,click on the clipboard button on the right-hand side of the textbox.



1. Go to the local root folder, C:\Repos, for the repositories.
2. Open command line prompt.
3. Type git clone https://github.com/fsantacroce/Cookbook.git.

```
aliyaparween@Aliyas-MacBook-Pro git-lab % git clone https://github.com/aliyaparween
/git-lab1.git
Cloning into 'git-lab1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
aliyaparween@Aliyas-MacBook-Pro git-lab %
```

4.Verify the remote repo is cloned



5.add new file or change content of remote repo

```
aliyaparween@Aliyas-MacBook-Pro git-lab1 % git status
On branch main
Your branch is up to date with 'origin/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:   README.md

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        text2.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

6.add the modified file to git

```
aliyaparween@Aliyas-MacBook-Pro git-lab1 % git add text2.txt
aliyaparween@Aliyas-MacBook-Pro git-lab1 % git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   text2.txt
```

7.Commit the changes of file

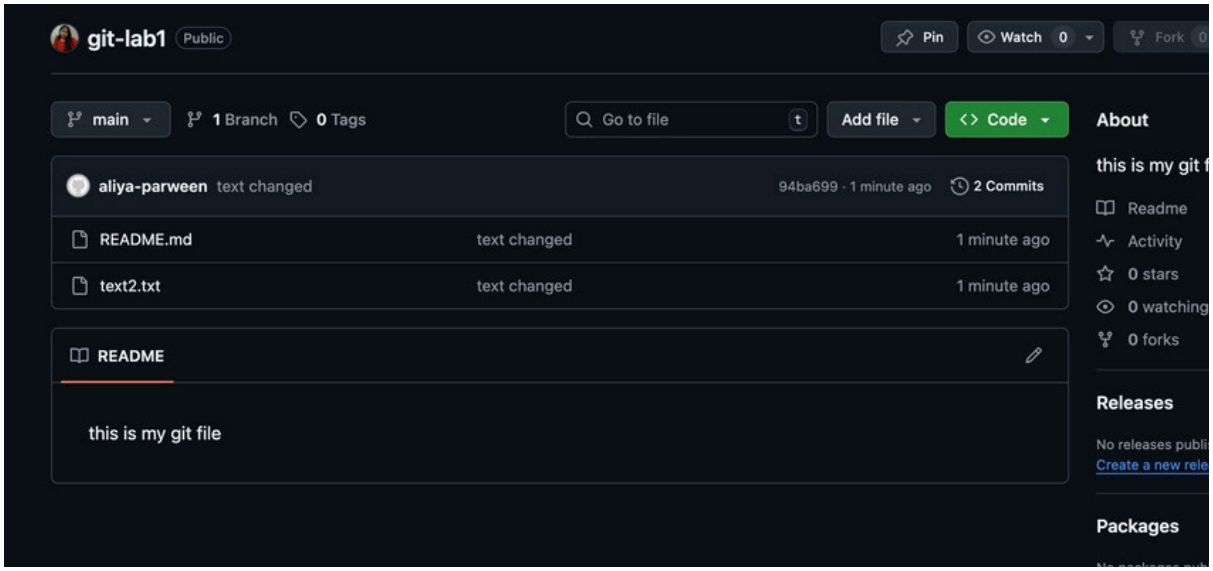
```
aliyaparween@Aliyas-MacBook-Pro git-lab1 % git commit -m "text changed"
[main 94ba699] text changed
 2 files changed, 1 insertion(+), 1 deletion(-)
 create mode 100644 text2.txt
aliyaparween@Aliyas-MacBook-Pro git-lab1 % git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

8.push the local repo content to remote repo

```
aliyaparween@Aliyas-MacBook-Pro git-lab1 % git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 10 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 310 bytes | 310.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/aliyaparween/git-lab1.git
   7747261..94ba699  main -> main
```

9.Verify the remote repository



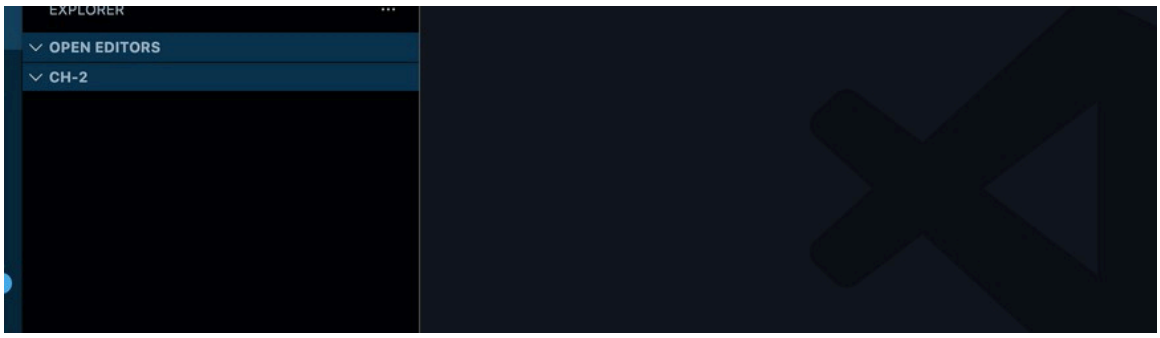
Lab Sheet 3

How Git can handle automatically file modifications when they are not related to the same lines of text.

Level 1: You are in a new repository located in C:\Repos\Exercises\Ch2-1. You have a master branch with two previous commits: the first commit with a file1.txt file and the second commit with a file2.txt file.

Level 2: After the second commit, you created a new branch called File2Split. You realized that file2.txt is too big, and you want to split its content by creating a new file2a.txt file. Do it, and then commit the modifications.

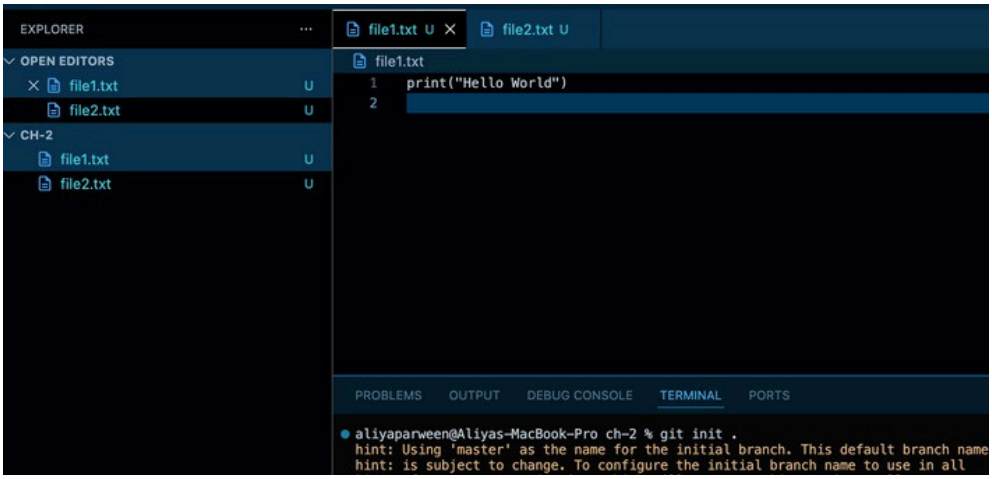
Step 1 — Create repo ch2-1



Initialize git repository

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git init .
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
hint:
hint: Disable this message with "git config set advice.defaultBranchName false"
Initialized empty Git repository in /Users/aliyaparween/Desktop/devops-lab/ch-2/.git/
```

Create file1.txt and file2.txt



```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1.txt
        file2.txt

nothing added to commit but untracked files present (use "git add" to track)
aliyaparween@Aliyas-MacBook-Pro ch-2 %
```

Add both files to git local repo

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git add file1.txt file2.txt
```

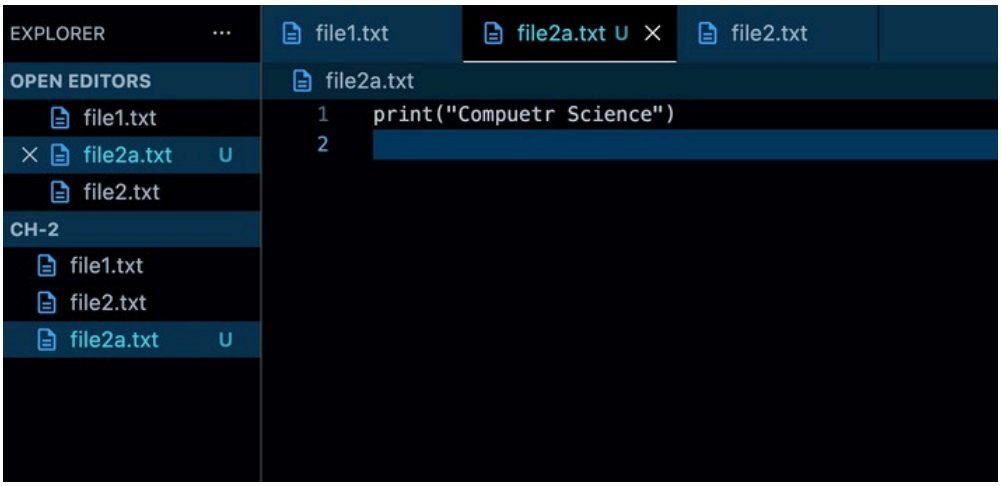
Commit file1 .txt and file2.txt

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git add file1.txt file2.txt
aliyaparween@Aliyas-MacBook-Pro ch-2 % git commit -m "text added"
[master (root-commit) 0227c22] text added
 2 files changed, 3 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
aliyaparween@Aliyas-MacBook-Pro ch-2 % git status
On branch master
nothing to commit, working tree clean
aliyaparween@Aliyas-MacBook-Pro ch-2 %
```

Step 2: create Branch to split file2

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git checkout -b file2split
Switched to a new branch 'file2split'
```

Create file2a in branch file2split



Add and commit file2a.txt in file2split

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git add file2a.txt
```

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git commit -m "text added"
[file2split f507e62] text added
 1 file changed, 1 insertion(+)
 create mode 100644 file2a.txt
aliyaparween@Aliyas-MacBook-Pro ch-2 %
```

Switch to master branch and merge

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git branch
* file2split
  master
aliyaparween@Aliyas-MacBook-Pro ch-2 % git checkout master
Switched to branch 'master'
aliyaparween@Aliyas-MacBook-Pro ch-2 %
```

```
aliyaparween@Aliyas-MacBook-Pro ch-2 % git merge file2split
Updating 0227c22..f507e62
Fast-forward
 file2a.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 file2a.txt
```

Lab Sheet 4

How to resolve conflicts when Git cannot merge files automatically.

Level 1: Create and commit the repository with two two files file1.txt and file2.txt, C:\Repos\Exercises\Ch2-1. On the master branch, you add the file3.txt file and commit it.

Level 2a: Then, you realize that it is better to create a new branch to work on file3.txt, so you create the File3Work branch. You move in this branch, and you start to work on it, committing modifications.

Level 2b: The day after, you accidentally move to the master branch and make some modifications on the file3.txt file, committing it. Then, you try to merge it.

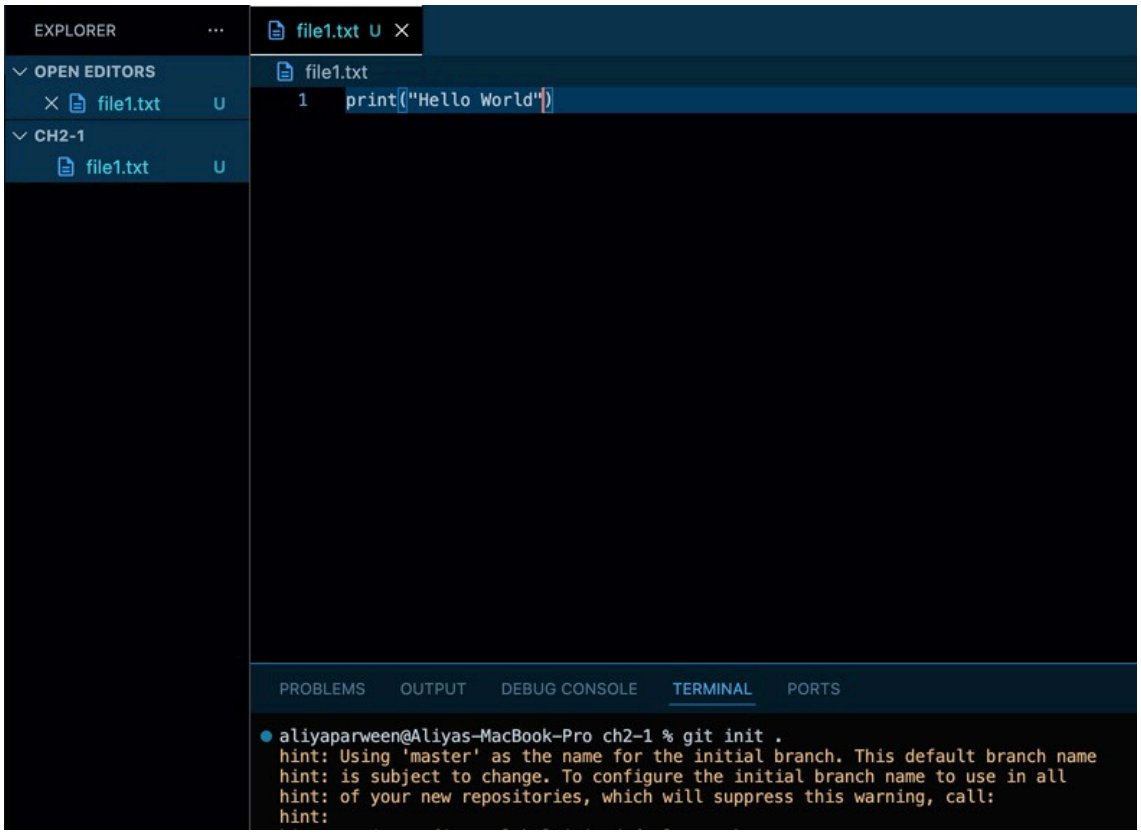
How Git can handle automatically file modifications when they are not related to the same lines of text.

Level 1: You are in a new repository located in C:\Repos\Exercises\Ch2-1. You have a master branch with two previous commits: the first commit with a file1.txt file and the second commit with a file2.txt file. Level 2: After the second commit, you created a new branch called File2Split. You realized that
file2.txt is too big, and you want to split its content by creating a new file2a.txt file. Do it, and then commit the modifications.

Step 1 — Create repo ch2-1

Initialize git repository

```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git init .
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
hint:
hint: Disable this message with "git config set advice.defaultBranchName false"
Initialized empty Git repository in /Users/aliyaparween/Desktop/devops-lab/ch2-1/.git/
```



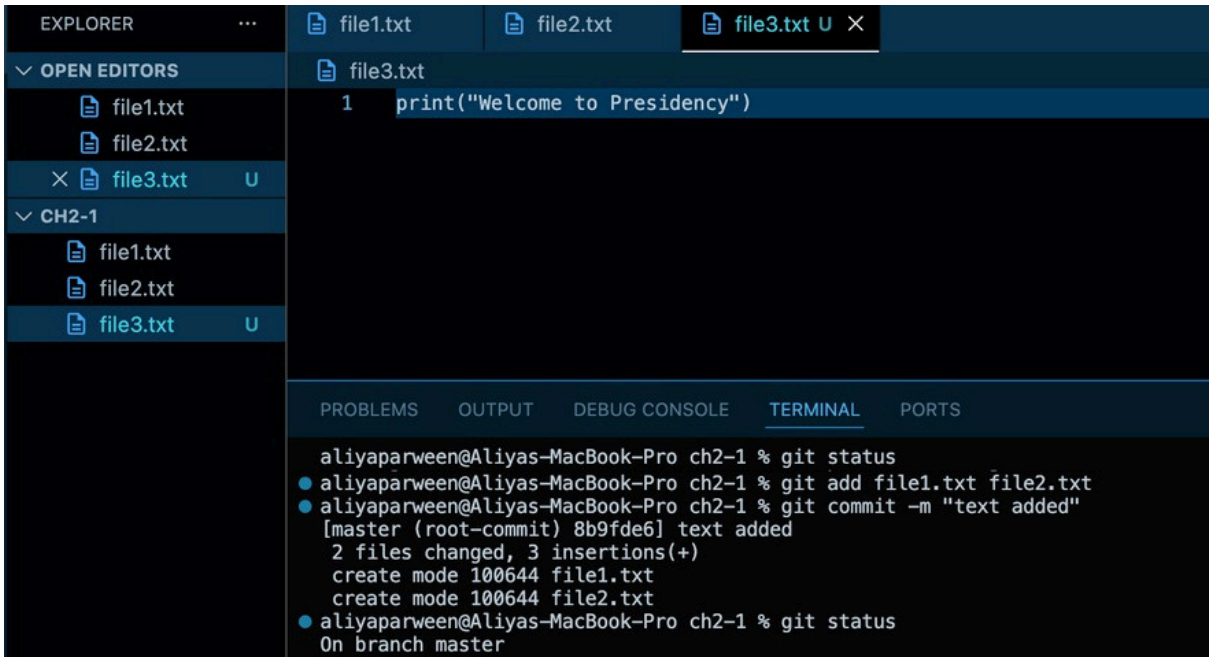
Add both files to git local repo

```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git add file1.txt file2.txt
```

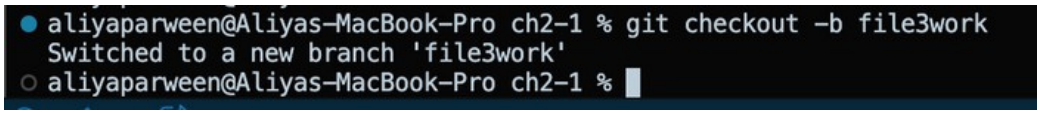
Commit file1 .txt and file2.txt

```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git add file1.txt file2.txt
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git commit -m "text added"
[master (root-commit) 8b9fde6] text added
 2 files changed, 3 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git status
On branch master
nothing to commit, working tree clean
aliyaparween@Aliyas-MacBook-Pro ch2-1 %
```

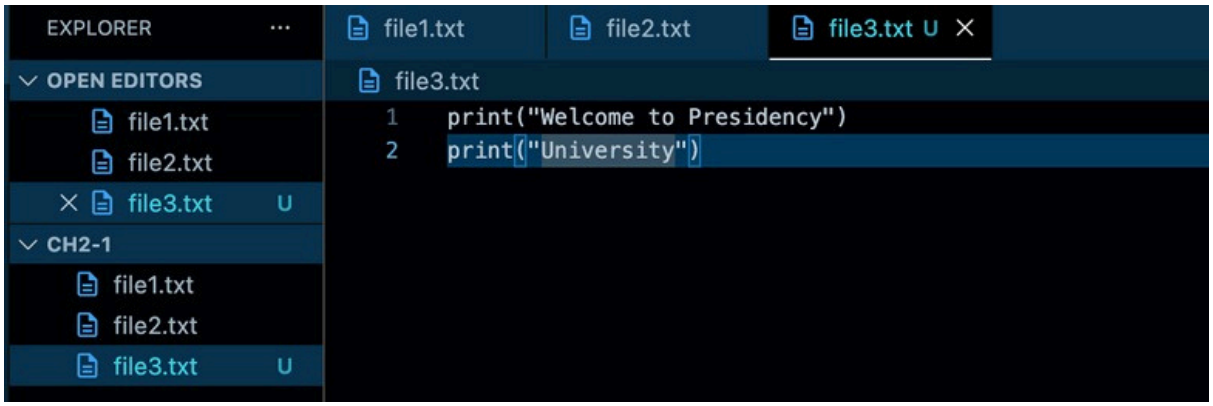
Step 2: create file3 on master

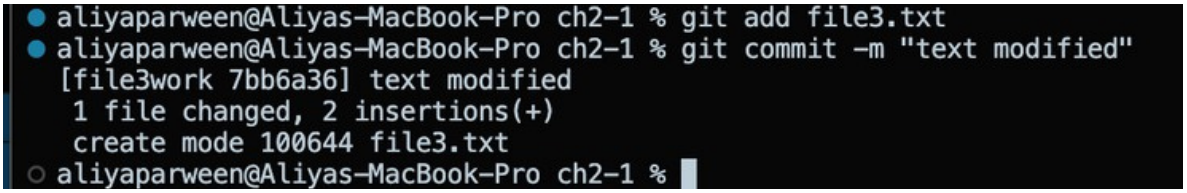


Step 3:Create Branch fil3work



Modify the file3

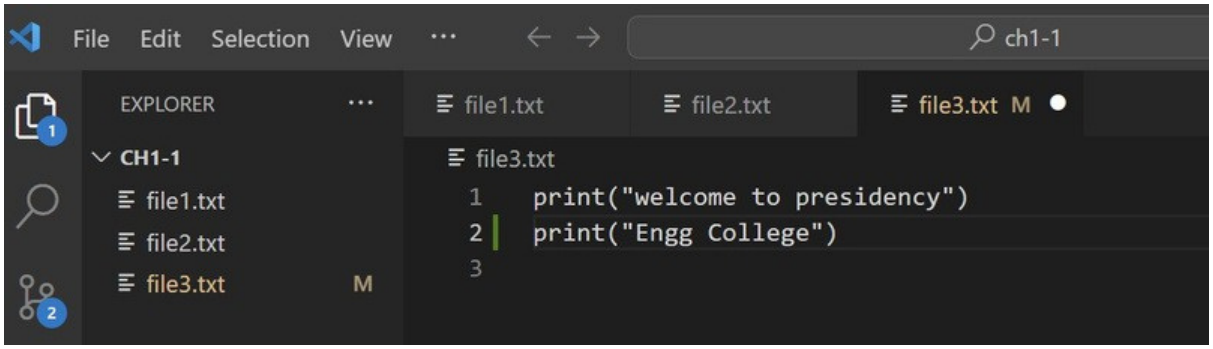




Switch to master branch

```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git checkout master
Switched to branch 'master'
```

Modify the file3 content



Add and commit file3 on master

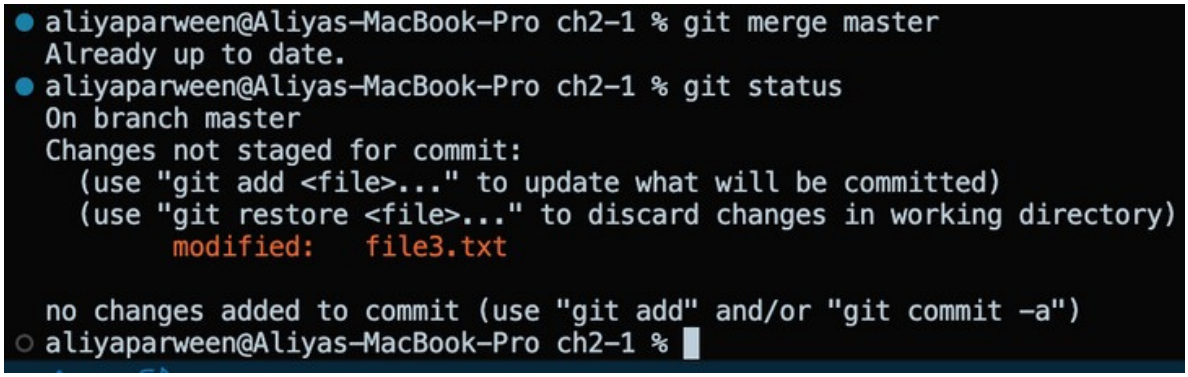
```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git add file3.txt
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git commit -m "text added"
[master aa2510e] text added
 1 file changed, 2 insertions(+)
 create mode 100644 file3.txt
aliyaparween@Aliyas-MacBook-Pro ch2-1 %
```

Step 3: Merge branch and master

```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git merge file3work
Auto-merging file3.txt
CONFLICT (add/add): Merge conflict in file3.txt
Automatic merge failed; fix conflicts and then commit the result.
aliyaparween@Aliyas-MacBook-Pro ch2-1 %
```

Step 4:resolve the conflicts

```
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git add file3.txt
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git commit -m "text added
[master 440d48a] text added
aliyaparween@Aliyas-MacBook-Pro ch2-1 %
```


aliyaparween@Aliyas-MacBook-Pro ch2-1 % git merge master
Already up to date.
aliyaparween@Aliyas-MacBook-Pro ch2-1 % git status
On branch master
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: file3.txt
no changes added to commit (use "git add" and/or "git commit -a")
aliyaparween@Aliyas-MacBook-Pro ch2-1 %

Labsheet 5

Level 1: Installation of Ansible on Windows

Ansible is a Linux-based tool, so it does not run natively on Windows. Instead, we install it on Windows using Windows Subsystem for Linux (WSL) or through Python pip inside WSL**.**

Step 1: Install Ansible

1.1 Install WSL on Windows:

- Open Command Line Prompt
- wsl --install

This installs Ubuntu as the default Linux distribution. Restart your system if required.

```
:\Users\vaiji>wsl --install
ownloading: Ubuntu
nstalling: Ubuntu
istribution successfully installed. It can be launched via 'wsl.exe -d Ubuntu'
aunching Ubuntu...
rovisioning the new WSL instance Ubuntu
his might take a while...
reate a default Unix user account: vaiji
ew password:
etype new password:
asswd: password updated successfully
o run a command as administrator (user "root"), use "sudo <command>".
ee "man sudo_root" for details.
```

1. **. Launch Ubuntu (WSL):**
   - Search for **Ubuntu** in the Start menu and launch it.
2. **.Update Ubuntu packages:**

- sudo apt update && sudo apt upgrade -y

```
root@cc6d99c71de9:/# apt update
Get:1 http://deb.debian.org/debian bookworm InRelease [151 kB]
Get:2 http://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:3 http://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:4 http://deb.debian.org/debian bookworm/main arm64 Packages [8693 kB]
Get:5 http://deb.debian.org/debian bookworm-updates/main arm64 Packages [6936 B]
Get:6 http://deb.debian.org/debian-security bookworm-security/main arm64 Packages [283 kB]
Fetched 9237 kB in 10s (914 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
```

1. **. Install Ansible inside WSL (Ubuntu):**

- sudo apt install software-properties-common -y
- sudo add-apt-repository --yes --update ppa:ansible/ansible
- sudo apt install ansible -y

```
root@cc6d99c71de9:/# apt install software-properties-common -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  appstream dirmngr distro-info-data gir1.2-glib-2.0 gir1.2-packagekitglib-1.0 gnupg
  libappstream4 libassuan0 libbrotli1 libcap2-bin libcurl3-gnutls libduktape207 libdw
  libicu72 libksba8 libldap-2.5-0 libldap-common libnghttp2-14 libnpth0 libpackagekit
  libsasl2-modules libsasl2-modules-db libssh2-1 libstemmer0d libunwind8 libxml2 libx
  python-apt-common python3-apt python3-blinker python3-cffi-backend python3-cryptogr
  python3-lazr.uri python3-oauthlib python3-pkg-resources python3-pyparsing python3-s
Suggested packages:
  apt-config-icons pinentry-gnome3 tor parcimonie xloadimage scdaemon isoquery low-me
  libsasl2-modules-ldap libsasl2-modules-otp libsasl2-modules-sql pinentry-doc polkit
  python-dbus-doc python3-crypto python3-setuptools python-pyparsing-doc sgml-base-do
The following NEW packages will be installed:
  appstream dirmngr distro-info-data gir1.2-glib-2.0 gir1.2-packagekitglib-1.0 gnupg
  libappstream4 libassuan0 libbrotli1 libcap2-bin libcurl3-gnutls libduktape207 libdw
  libicu72 libksba8 libldap-2.5-0 libldap-common libnghttp2-14 libnpth0 libpackagekit
  libsasl2-modules libsasl2-modules-db libssh2-1 libstemmer0d libunwind8 libxml2 libx
  python-apt-common python3-apt python3-blinker python3-cffi-backend python3-cryptogr
  python3-lazr.uri python3-oauthlib python3-pkg-resources python3-pyparsing python3-s
```

```
root@cc6d99c71de9:/# apt install ansible -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ansible-core ieee-data libpsl5 libyaml-0-2 lsb-release netbase publicsuffix python-babel-localedata python3-anyio pyth
  python3-cffi-backend python3-chardet python3-charset-normalizer python3-click python3-colorama python3-cryptography py
  python3-hpack python3-httpcore python3-httplib2 python3-httpx python3-hyperframe python3-idna python3-jinja2 python3-j
  python3-lockfile python3-markdown-it python3-markupsafe python3-mdurl python3-netaddr python3-ntlm-auth python3-packag
  python3-pyparsing python3-requests python3-requests-kerberos python3-requests-ntlm python3-requests-toolbelt python3-r
  python3-simplejson python3-six python3-sniffio python3-tz python3-urllib3 python3-winrm python3-xmltodict python3-yaml
Suggested packages:
  cowsay sshpass python-cryptography-doc python3-cryptography-vectors python3-trio python3-aioquic python-jinja2-doc pyt
  python-pygments-doc ttf-bitstream-vera python-pyparsing-doc python3-openssl python3-socks python-requests-doc python3-
The following NEW packages will be installed:
  ansible ansible-core ieee-data libpsl5 libyaml-0-2 lsb-release netbase publicsuffix python-babel-localedata python3-an
```

1. **.Verify installation:**

- ansible –version

```
aliyaparween@Aliyas-MacBook-Pro ~ % ansible --version
ansible [core 2.19.2]
  config file = None
  configured module search path = ['/Users/aliyaparween/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packages/ansible
  ansible collection location = /Users/aliyaparween/.ansible/collections:/usr/share/ansible/collections
  executable location = /Library/Frameworks/Python.framework/Versions/3.12/bin/ansible
  python version = 3.12.2 (v3.12.2:6abddd9f6a, Feb  6 2024, 17:02:06) [Clang 13.0.0 (clang-1300.0.29.30)] (/Library/Frameworks/Python.framework/Versions/3.12/bin/python3.12)
  jinja version = 3.1.5
  pyyaml version = 6.0.2 (with libyaml v0.2.5)
```

Step 2: Get a Server to Manage

Option A (Best for learning): Create a free Linux VM on the cloud (AWS, Azure, GCP, Oracle Cloud free tier).

Option B (Local test): Install Debian, VirtualBox + Ubuntu VM on your laptop and connect via SSH.

2.1. Install a second WSL distribution

wsl --install -d Debian

2.2 : List distributions

wsl--list –verbose

2.3 : start debian

wsl-d Debian

Step 3: Set Up SSH Access

3.1: Set Up SSH for Debian

sudo apt update

sudo apt install openssh-server -y

3.2 .Start SSH service

sudo service ssh start

3.3.Confirm it's running

ps-ef|grep ssh

3.4: Find Debian's network address

ipaddrshow eth0

Like this:inet 172.22.40.89/20

Step 4:Generate SSH key for Obuntu

ssh-keygen -t rsa -b 2048

Copy the SSH key to yourserver:

ssh-copy-id ubuntu@192.168.1.10

Test login manually:

ssh ubuntu@192.168.1.10

Level 2a: Create a Basic Inventory File

Step 1:Create an inventory.ini

file to define managed hosts.

nano inventory.ini

Example content:

[managed_nodes]

172.22.40.89 ansible_user=user ansible_ssh_private_key_file=~/.ssh/id_rsa

Step 2: Save & exit

Press CTRL+O → Enter → CTRL+X

Level 2b: Running Your First Ad-Hoc Ansible Command

ansible all -i inventory.ini -m ping

Expected Output: 172.22.40.89 | SUCCESS => { "ping": "pong" }

Lab Sheet 7

Shell Module & First Playbooks

This guide covers the use of the ansible.builtin.shell module for ad-hoc commands and how to write your first playbooks to create files and directories.

All commands are run from your ~/ansible_lab directory on your **Ubuntu (control) machine**.

Set 1: Ansible Shell Module (Ad-hoc)

The ansible.builtin.shell module is used for running commands that require shell features like pipes (|), redirection (>), or shell variables ($HOME).

Level 2a: Execute a Single Command

This command runs ls -l on the /tmp directory of your Debian host.

ansible -i inventory debian_servers -m ansible.builtin.shell -a "ls -l /tmp/"

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible -i inventory debian_servers -m ansible.builtin.shell -a "ls -l /tmp/"

[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/bin/python3.13', but future installation of anothe
 See https://docs.ansible.com/ansible-core/2.19/reference_appendices/interpreter_discovery.html for more information.
localhost | CHANGED | rc=0 >>
total 32
srwx------@ 1 aliyaparween  wheel     0 19 Nov 20:02 2f093fe9-a235-5d1c-9a62-3fae2cdf7eb1
drwxr-xr-x  2 aliyaparween  wheel    64 19 Nov 11:58 3CE90355-69AE-473A-9A49-D80695A71AD9
drwxr-xr-x  2 aliyaparween  wheel    64 19 Nov 11:58 405423D5-5B43-4CFE-806B-6FF633694B35
prw-rw-rw-  1 aliyaparween  wheel     0 17 Nov 11:29 5F4A18C6-EB47-412E-B240-9618F53053B3_IN
prw-rw-rw-  1 aliyaparween  wheel     0 17 Nov 11:29 5F4A18C6-EB47-412E-B240-9618F53053B3_OUT
drwxr-xr-x  2 aliyaparween  wheel    64 16 Nov 11:28 62CBFD4D-C26C-4620-BFDA-AC532F4C13BB
srwx------@ 1 aliyaparween  wheel     0 19 Nov 20:02 7c3338c5-8eec-50ae-bc4a-1f6969419936
prw-rw-rw-  1 aliyaparween  wheel     0 18 Nov 11:56 83AFDE7B-A094-4B16-B84B-06E1B08A8B40_IN
prw-rw-rw-  1 aliyaparween  wheel     0 18 Nov 11:55 83AFDE7B-A094-4B16-B84B-06E1B08A8B40_OUT
drwxr-xr-x  2 aliyaparween  wheel    64 19 Nov 11:58 A7CED39C-4B5B-4731-9C69-F457B88BD799
-rw-rw-rw-  1 aliyaparween  wheel  4835 16 Nov 11:28 adobegc.log
-rw-r--r--  1 root          wheel     0 16 Nov 11:27 AlTest1.err
-rw-r--r--  1 root          wheel     0 16 Nov 11:27 AlTest1.out
prw-rw-rw-  1 aliyaparween  wheel     0 16 Nov 11:28 C9ABCA45-39A0-4DF9-B97E-EF2BB413BD17_IN
prw-rw-rw-  1 aliyaparween  wheel     0 16 Nov 11:28 C9ABCA45-39A0-4DF9-B97E-EF2BB413BD17_OUT
srwx------  1 aliyaparween  wheel     0 16 Nov 11:28 com.adobe.AdobeIPCBroker.ctrl-aliyaparween
drwx------  3 aliyaparween  wheel    96 16 Nov 11:27 com.apple.launchd.30ax9qyQdK
d----w--w-  2 root          wheel    64 16 Nov 11:27 devio_semaphore_logi_hpp_OptionsPlus_A7D4B139-F9F9-44A6-9F5D-7BC0A9B8B80F
-rw-rw-rw-  1 aliyaparween  wheel     0 16 Nov 11:28 ExmanProcessMutex
drwxr-xr-x  2 aliyaparween  wheel    64 19 Nov 11:58 F9C6B81B-CBEE-4049-B0E2-4CA613ED9EF2
prw-rw-rw-  1 aliyaparween  wheel     0 19 Nov 11:58 FEE0CBF1-D4FA-4D34-A197-4C0B545B1095_IN
prw-rw-rw-  1 aliyaparween  wheel     0 19 Nov 11:58 FEE0CBF1-D4FA-4D34-A197-4C0B545B1095_OUT
-rw-r--r--  1 root          wheel     0 16 Nov 11:27 logi.optionsplus.updater.log
srwxrwxrwx@ 1 aliyaparween  wheel     0 19 Nov 20:02 logitech_kiros_agent-c757e6630ac370b4d2e18b099bb830fc
srwxrwxrwx@ 1 root          wheel     0 16 Nov 11:27 logitech_kiros_updater
srwxrwxrwx  1 _mysql        wheel     0 16 Nov 11:27 mysql.sock
-rw-------  1 _mysql        wheel     4 16 Nov 11:27 mysql.sock.lock
srwxrwxrwx  1 _mysql        wheel     0 16 Nov 11:27 mysqlx.sock
-rw-------  1 _mysql        wheel     4 16 Nov 11:27 mysqlx.sock.lock
drwxr-xr-x  2 root          wheel    64 16 Nov 11:27 powerlog
aliyaparween@Aliyas-MacBook-Pro ansible_lab %
```

Level 2b: Execute with Pipe and Redirection

1. Example with Redirection (>):

Create a new file on the Debian host by redirecting the output of echo.

ansible -i inventory debian_servers -m ansible.builtin.shell -a "echo 'Hello from the shell module' > /tmp/shell_output.txt"

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible -i inventory debian_servers -m ansible.
builtin.shell -a "echo 'Hello from the shell module' > /tmp/shell_output.txt"

[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/bin/
python3.13', but future installation of another Python interpreter could cause a different in
terpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference_appendic
es/interpreter_discovery.html for more information.
localhost | CHANGED | rc=0 >>

aliyaparween@Aliyas-MacBook-Pro ansible_lab %
```

- **Verify it worked** by reading the file's content:ansible -i inventory debian_servers -m ansible.builtin.shell -a "cat /tmp/shell_output.txt"

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible -i inventory debian_servers -m ansible.
builtin.shell -a "cat /tmp/shell_output.txt"

[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/bin/
python3.13', but future installation of another Python interpreter could cause a different in
terpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference_appendic
es/interpreter_discovery.html for more information.
localhost | CHANGED | rc=0 >>
Hello from the shell module
aliyaparween@Aliyas-MacBook-Pro ansible_lab %
```

2. Example with Pipe (|):

List files in /tmp and use grep to find only the .zip file.

ansible -i inventory debian_servers -m ansible.builtin.shell -a "ls -l /tmp/ | grep .zip"

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible -i inventory debian_servers -m ansible.
builtin.shell -a "ls -l /tmp/ | grep -E '\.zip$' || true"

[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/bin/
python3.13', but future installation of another Python interpreter could cause a different in
terpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference_appendic
es/interpreter_discovery.html for more information.
localhost | CHANGED | rc=0 >>
```

Lab Sheet 8

Writing andRunningYourFirstPlaybooks

Update Your Inventory

Your lab exercises refer to a webservers group. We must add this to our inventory file.

1. Open your inventory file for editing:nano inventory
2. Add the [webservers] group and make your debian_wsl host a member of it. The file content should look like this:[debian_servers]debian_wsl ansible_host=localhost ansible_port=2222 ansible_user=debian_user[webservers]debian_wsl*(Thisreusesthedebian_wslhostdefinition from the first group).*
3. Save and exit (Ctrl+X, then Y, then Enter).

Level 1 & 2a: Run Playbook to Create a File

1. **Create the Playbook File:** nano create_file.yml
2. **Add this YAML code** to the file. This playbook targets the webservers group and uses the ansible.builtin.file module.---- name: Create a file on web servers hosts: webservers gather_facts: no tasks: - name: Create an empty test file ansible.builtin.file: path: /tmp/webserver_test_file.txt state: touch mode: '0644'
3. Save and exit(Ctrl+X,then Y, thenEnter).
4. **Run the playbook** (This is the "Level 1" step):ansible-playbook -i inventory create_file.yml*You should see a "PLAY RECAP" indicating the task was successful.*

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible-playbook -i inventory create_file.yml


PLAY [Create a file on web servers] ********************************************************

TASK [Create an empty test file] ***********************************************************
[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/bin
/python3.13', but future installation of another Python interpreter could cause a different
interpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference_appen
dices/interpreter_discovery.html for more information.
changed: [localhost]

PLAY RECAP *********************************************************************************
localhost                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0

aliyaparween@Aliyas-MacBook-Pro ansible_lab %
```

Level 2b: Create Multiple Directories with a Loop

1. **Create a New Playbook File:**nano create_dirs.yml
2. **Add this YAML code.** This playbook uses a loop to run the ansible.builtin.file task for each item in the list.---- name: Create multiple directories hosts: webservers gather_facts: no tasks: - name: Create new app directories ansible.builtin.file: path: "/tmp/{{ item }}" state: directory mode: '0755' loop: - app_dir1 - app_dir2 - app_dir3
3. Save and exit (Ctrl+X, then Y, then Enter).
4. **Run the playbook:**ansible-playbook -i inventory create_dirs.yml

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible-playbook -i inventory create_dirs.ym
l


PLAY [Create multiple directories] ***************************************************
***

TASK [Create new app directories] ****************************************************
***
[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/b
in/python3.13', but future installation of another Python interpreter could cause a differ
ent interpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference
_appendices/interpreter_discovery.html for more information.
changed: [localhost] => (item=app_dir1)
changed: [localhost] => (item=app_dir2)
changed: [localhost] => (item=app_dir3)

PLAY RECAP ***************************************************************************
***
localhost                  : ok=1    changed=1    unreachable=0    failed=0    skipped=0
  rescued=0    ignored=0

aliyaparween@Aliyas-MacBook-Pro ansible_lab %
```

5.Verify it worked using an ad-hoc command:ansible -i inventory webservers -m ansible.builtin.shell -a "ls -ld /tmp/app_dir*"*The output should list the three new directories: app_dir1, app_dir2, and app_dir3.*

```
aliyaparween@Aliyas-MacBook-Pro ansible_lab % ansible -i inventory webservers -m ansible.b
uiltin.shell -a "ls -ld /tmp/app_dir*"

[WARNING]: Host 'localhost' is using the discovered Python interpreter at '/opt/homebrew/b
in/python3.13', but future installation of another Python interpreter could cause a differ
ent interpreter to be discovered. See https://docs.ansible.com/ansible-core/2.19/reference
_appendices/interpreter_discovery.html for more information.
localhost | CHANGED | rc=0 >>
drwxr-xr-x@ 2 aliyaparween  wheel  64 19 Nov 21:12 /tmp/app_dir1
drwxr-xr-x@ 2 aliyaparween  wheel  64 19 Nov 21:12 /tmp/app_dir2
drwxr-xr-x@ 2 aliyaparween  wheel  64 19 Nov 21:12 /tmp/app_dir3
aliyaparween@Aliyas-MacBook-Pro ansible_lab %
```

Lab Sheet 9

- **Selenium IDE:** A browser-based tool for **record-and-playback**. It's great for beginners, quick tests, and learning Selenium commands. user don't need to write code.
- **Selenium WebDriver:** A powerful library (API) for automating browsers **with code**. This is what's used for robust, large-scale automation. User will write scriptsina language like Python, Java, or C#.

1. Selenium IDE: Download and Install

Selenium IDE is a browser extension. Installation is simple:

1. **Open your browser:** It's available for **Chrome**, **Firefox**, and **Edge**.
2. **Go to the Add-on/Extension Store:**
    - o **For Chrome:** Go to the Chrome Web Store.
    - o **For Firefox:** Go to the Firefox Add-ons site.
    - o **For Edge:** Go to the Edge Add-ons store.
3. **Search:** Type "Selenium IDE" in the search bar.
4. **Install:** Find the official extension (it will be by "Selenium") and click "Add to Chrome" (or Firefox/Edge).
5. **Confirm:** A popup will ask for permissions. Accept it.
6. **Access:** You will now see the Selenium IDE icon (a white 'Se' on a red/blue background) in your browser's toolbar. Click it to open the IDE.

2. Selenium IDE: First Test Case (Login Test)

Let'screateatestforacommondummyloginpage.

Steps to Record Your First Test

1. **Open Selenium IDE:** Click the extension icon in your toolbar.
2. **Create a New Project:** Adialog will pop up. Select "**Create a new project**".
3. **Name Your Project:** Giveit a name, like "LoginTests".
4. **Set the Base URL:** TheIDE will ask for your project's Base URL. This is the starting page foryour tests.Forthisexample, let's use a practice site.
    - o Enter: https://practicetestautomation.com/practice-test-login/
    - o Click "**START RECORDING**".
5. **Record the Actions:**
    - o A new browserwindow will open to the URL you provided.

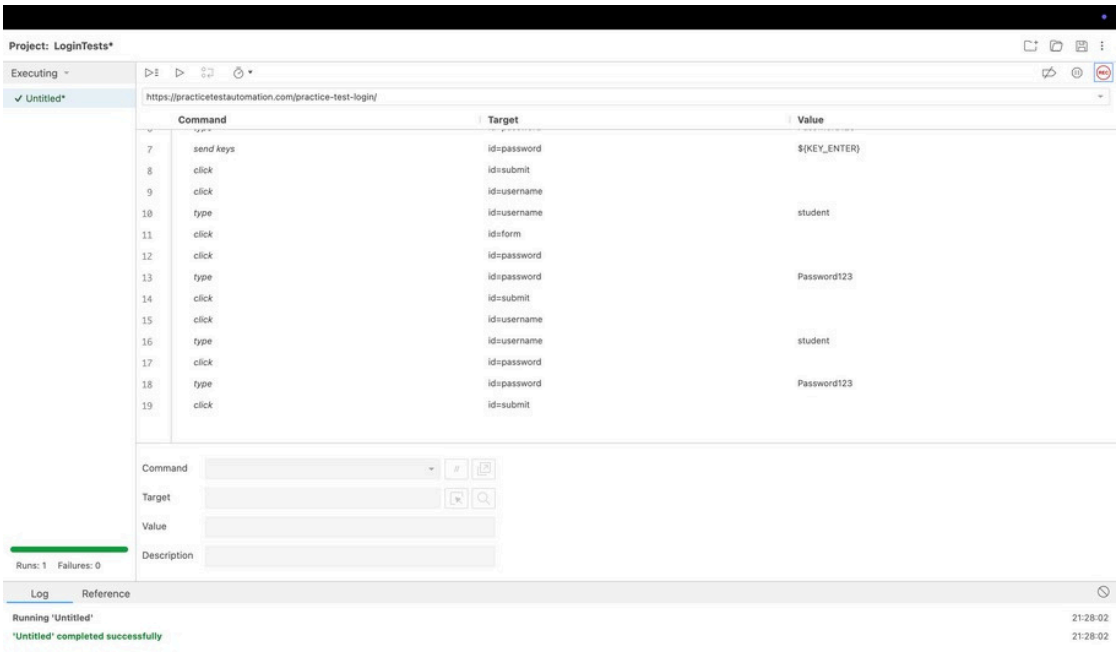
- In the **Username** field, typestudent.
- In the **Password** field, typePassword123.
- Click the "**Submit** " button.

6. **Add a Verification (Assertion):**
   - The page will load andshowa"Logged In Successfully" message.
   - **Right-click** on the text"Congratulations" or "Logged in successfully".
   - In the context menu, goto**Selenium IDE > Assert > text**. This tells the IDE to *fail* thetest if this textisn'tpresent, confirming the login worked.
7. **Stop Recording:**
   - Go backtothe Selenium IDE window.
   - Click the**Stop Recording**button (red circle) in the top-right corner.
8. **NametheTest:** You'll be prompted to name this test case. Call it "SuccessfulLogin".
9. **Playback:**
   - With the "SuccessfulLogin" test selected, click the **"Run current test"** button (looks like a 'Play' icon) at the top.
   - Watch as the IDE automatically repeats all your steps in the browser. If everything passes, the test case will turn green.

Common Commands

- **Command:** open
  - **Target:** /practice-test-login/
  - **Whatitdoes:** Opens the specified URL (relative to your Base URL).
- **Command:** type
  - **Target:** id=username (or name=username, etc. This is the **selector**).
  - **Value:** student
  - **Whatitdoes:** Types text into an input field.
- **Command:** click
  - **Target:** id=submit
  - **Whatitdoes:** Clicks on an element (like a button or link).
- **Command:** assert text
  - **Target:** css=h1.post-title (the selector for the text)
  - **Value:** Logged In Successfully

- **What it does: Asserts** that the text is present. If it's not, the test **stops and fails**. (A verify command would log the failure but *continue* the test).



Lab sheet 10

Script to Open google.co.in (WebDriver)

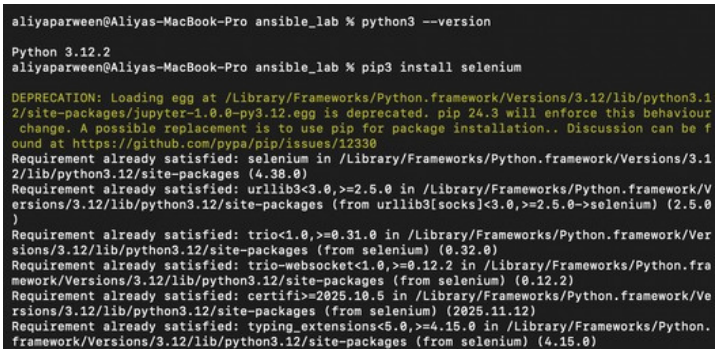
This task requires **Selenium WebDriver**,notthe IDE. Here are the steps and the Python script.

Setup (One-time)

Install Python: Ifyou don't have it, download it from python.org.

Install Selenium: Open your terminal or command prompt and run:

1. 2. Bash pip install selenium



3. **Get ChromeDriver:** Thisisthebridgebetweenyourscriptandthe Chrome browser.
   - o **Good news:** AsofSelenium4.6.0+,theselenium-manager is bundled, which **automatically downloads the correct driver for you** .You usually don't need todoanything else.

Python Script

1. Create a new Python file (e.g., open_google.py).
2. Copy this code into the file:

Python

```
from selenium import webdriver

import time

# 1. Initialize the Chrome driver
    # Selenium Manager will automatically find and manage
    # the correct ChromeDriver for your browser version.
    driver = webdriver.Chrome()


# 2. Open the desired URL
    print("Opening google.co.in...")
    driver.get("https://www.google.co.in")
# 3. Print the title to confirm
    print(f"Page title is: {driver.title}")
# 4. Wait for 5 seconds so you can see the browser
    print("Waiting for 5 seconds...")
    time.sleep(5)
# 5. Close the browser
```

```
print("Closing the browser.")
driver.quit()
```

Run the script: Open your terminal in the same directory and run:

Bash

```
python open_google.py
```

```
aliyaparween@Aliyas-MacBook-Pro selenium_lab % nano open_google.py

aliyaparween@Aliyas-MacBook-Pro selenium_lab % python3 open_google.py

Opening google.co.in...
Page title is: Google
Closing the browser.
```

Script to Verify Title and Redirection (WebDriver)

This script builds on the previous one, adding verification steps.

Python Script

1. Create a new Python file (e.g., verify_google.py).
2. Copy this code into the file:

Python

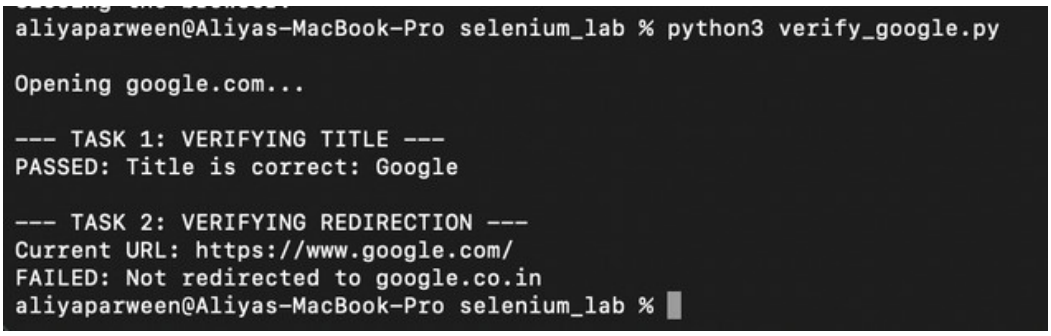
```
from selenium import webdriver
import time
# 1. Initialize the driver
    driver = webdriver.Chrome()
# 2. Open google.com
    print("Opening google.com...")
    driver.get("https://www.google.com")
# 3. Task 1: Verify the title
    print("--- Verifying Title ---")
    expected_title = "Google"
    actual_title = driver.title
    if actual_title == expected_title:
      print(f" PASSED: Title is correct. Actual: '{actual_title}'")
    else:
       print(f" FAILED: Title is incorrect. Expected: '{expected_title}', Actual:
'{actual_title}'")
# 4. Task 2: Verify redirection to google.co.in
print("\n--- Verifying Redirection ---")
expected_url_fragment = "google.co.in"
```

```
        current_url = driver.current_url
        print(f"Current URL is: {current_url}")
        # We use 'in' because the URL might have other things
        # like 'https://www.google.co.in/?gws_rd=ssl'
        if expected_url_fragment in current_url:
          print(f"PASSED: URL contains the fragment '{expected_url_fragment}'.")
        else:
           print(f" FAILED: URL does not contain '{expected_url_fragment}'.")
# 5. Wait for 3 seconds to see the results
        time.sleep(3)
# 6. Close the browser
        print("\nClosing the browser.")
    driver.quit()
```

3. **Run the script:**

Bash

python verify_google.py



Labsheet 11

Level 1: Write a script to open google.co.in using internet explorer (InternetExplorerDriver).

```
from selenium import webdriver
from selenium.webdriver.ie.service import Service as IEService
import time


#IMPORTANT SETUP NOTE:
#1. You must have Internet Explorer installed.
#2. You must download the 'IEDriverServer.exe' from the Selenium website.
#3. Replace the placeholder path below with the actual path to your IEDriverServer.exe.
IE_DRIVER_PATH = "./drivers/IEDriverServer.exe"
def run_ie_test():
    """
    Opens google.co.in using Internet Explorer.
    """
    driver = None
    print("--- Level 1: Internet Explorer Test ---")
    try:
        # 1. Set up the service using the path to the IEDriverServer executable
        ie_service = IEService(executable_path=IE_DRIVER_PATH)
         # 2. Initialize the Internet Explorer WebDriver
        # Note: Use webdriver.Ie() or webdriver.InternetExplorer() in modern Selenium.
        driver = webdriver.Ie(service=ie_service)
        # 3. Set an implicit wait (good practice)
        driver.implicitly_wait(10)
        # 4. Open the target URL
        url = "https://www.google.co.in"
        print(f"Navigating to: {url}")
```

```
driver.get(url)


#PROOF:Print the page title
print(f"Successfully navigated. Page Title: {driver.title}")
  #5.Waitbriefly for visual proof
time.sleep(3)
exceptException as e:
print(f"Anerror occurred during IE execution. Check your driver path and IE settings.")
print(f"Error details: {e}")
finally:
#6.Safelyclose the browser session
if driver:
    print("Closing Internet Explorer browser.")
    driver.quit()
if__name__ == "__main__":
 run_ie_test()
```

Output

Internet Explorer Test --- Navigating to: https://www.google.co.in Successfully navigated. Page Title: Google Closing Internet Explorer browser.

Level 2: Write a script to create browser instance based on browser name.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from selenium.webdriver.firefox.service import Service as FirefoxService
from selenium.webdriver.ie.service import Service as IEService
import time
import sys
# --- CONFIGURATION (UPDATE THESE PATHS) ---
# You must download these driver executables and update the paths accordingly.
CHROME_DRIVER_PATH = "./drivers/chromedriver.exe"
FIREFOX_DRIVER_PATH = "./drivers/geckodriver.exe"
IE_DRIVER_PATH = "./drivers/IEDriverServer.exe"
def get_browser_instance(browser_name: str) -> webdriver.Remote:
    """
    Creates and returns a WebDriver instance based on the provided browser name.
        Args:
        browser_name (str): The name of the browser ('chrome', 'firefox', or 'ie').
         Returns:
        webdriver.Remote: The initialized browser instance.
         Raises:
        ValueError: If an unsupported browser name is provided.
    """
    browser_name = browser_name.lower()
        if browser_name == 'chrome':
        print("Factory creating Chrome instance...")
        service = ChromeService(executable_path=CHROME_DRIVER_PATH)
        driver = webdriver.Chrome(service=service)
         elif browser_name == 'firefox':
        print("Factory creating Firefox instance...")
        service = FirefoxService(executable_path=FIREFOX_DRIVER_PATH)
```

```
        driver = webdriver.Firefox(service=service)


    elifbrowser_name == 'ie':
        print("Factory creating Internet Explorer instance...")
        service = IEService(executable_path=IE_DRIVER_PATH)
        #Note: IE requires specific security settings to be configured.
        driver = webdriver.Ie(service=service)
         else:
        raise ValueError(f"Unsupported browser: '{browser_name}'. Supported are 'chrome',
'firefox', 'ie'.")
    return driver
def run_test(driver_instance: webdriver.Remote):
    """Runs a simple test using the given driver instance."""
    if not driver_instance:
        return
    try:
        driver_instance.implicitly_wait(10)
        test_url = "https://www.google.com"
        print(f"Testing URL: {test_url}")
        driver_instance.get(test_url)
        print(f"Current Title: {driver_instance.title}")
        time.sleep(3) # Wait for visual confirmation
        except Exception as e:
        print(f"An error occurred during test execution: {e}", file=sys.stderr)
         finally:
        print("Closing the browser.")
        driver_instance.quit()
if __name__ == "__main__":
    print("--- Level 2: Dynamic Browser Factory Test ---")
        # --- PROOF 1: Using the factory to get a Chrome instance ---
    try:
```

```
        chrome_driver = get_browser_instance('Chrome')
        run_test(chrome_driver)
    except Exception as e:
        print(f"\n[Chrome Test Failed]: Please ensure the path to chromedriver.exe is correct.", file=sys.stderr)
    # --- PROOF 2: Using the factory to get an IE instance (or attempting to) ---
    # Note: This section will likely fail if IE is not correctly set up.
    try:
        ie_driver = get_browser_instance('ie')
        run_test(ie_driver)
    except Exception as e:
        print(f"\n[IE Test Failed]: Ensure IEDriverServer.exe path is correct and IE is configured.", file=sys.stderr)
    #---PROOF 3: Testing an invalid browser name (Error Handling) ---
    try:
        get_browser_instance('safari')
    except ValueError as e:
        print(f"\nSuccessfully caught expected error for unsupported browser: {e}")
    except Exception as e:
        print(f"\nCaught unexpected error for unsupported browser: {e}", file=sys.stderr)
```

Output

Dynamic Browser Factory Test --- Factory creating Chrome instance... Testing URL: https://www.google.com Current Title: Google Closing the browser. Factory creating Internet Explorer instance... Testing URL: https://www.google.com Current Title: Google Closing the browser.

Labsheet 12

Level 1: Write a script to close all the browsers without using quit() method.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from selenium.webdriver.common.by import By
import time
# --- CONFIGURATION (Ensure this path is correct) ---
CHROME_DRIVER_PATH = "./drivers/chromedriver.exe"
defclose_all_browsers_without_quit():
"""
Opens multiple browser windows and closes them one by one using close(),
demonstrating the difference from quit().
"""
driver = None
try:

print("--- Lab Sheet 12, Level 1: Closing All Browsers without quit() ---")

#1. Setup Driver
service = ChromeService(executable_path=CHROME_DRIVER_PATH)
driver = webdriver.Chrome(service=service)
driver.implicitly_wait(10)



#2. Open Main Window (Handle 1)
driver.get("https://www.google.com")
print(f"Opened main window. Title: {driver.title}")
```

```
    main_handle = driver.current_window_handle

    #3.Open a New Tab/Window (Handle 2)
    #Execute script to open a new blank window/tab
    driver.execute_script("window.open('https://www.wikipedia.org/', '_blank');")
    print("Opened second window/tab to Wikipedia.")
    time.sleep(1)

    #4.Getall window handles
    all_handles = driver.window_handles

    #5.Iterate and close each window/tab
    print(f"Total windows detected: {len(all_handles)}")
    fori,handle in enumerate(all_handles):
        driver.switch_to.window(handle)
        print(f"Closing window {i+1} with title: {driver.title}")
        driver.close()
        time.sleep(0.5)

    #Note:At this point, all browser windows are closed.
    #ThePython script is still running, and the driver session is technically orphaned
    #untilthe Python process ends, but the browser UI is gone.

  exceptException as e:
    print(f"An error occurred: {e}")

  finally:
    #Anecessary cleanup step for an orphaned driver session is usually quit(),
    #butsince the prompt explicitly forbids it, we conclude the script.
    print("\nScript finished. All browser windows were closed using driver.close() on each
handle.")
```

```
if__name__ == "__main__":
    close_all_browsers_without_quit()
```

Output:

```
Closing All Browsers without quit()
--- Opened main window. Title: Google
Opened second window/tab to Wikipedia.
Total windows detected: 2
Closing window 1 with title: Google
 Closing window 2 with title: Wikipedia, the free encyclopedia
Script finished. All browser windows were closed using driver.close() on each handle.
```

Level 2: Write a script to search for specified option in the listbox.

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service as ChromeService
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select # Essential for listbox interaction
import time
import os
# --- CONFIGURATION (Ensure this path is correct) ---
CHROME_DRIVER_PATH = "./drivers/chromedriver.exe"
# Determine the absolute path for the HTML test file
HTML_FILE_NAME = "listbox_test_page.html"
# This assumes the HTML file is in the same directory as this Python script
listbox_test_url = 'file:///' + os.path.abspath(HTML_FILE_NAME).replace('\\', '/')


defsearch_and_select_listbox_option(option_text: str):
    """
    Automates selecting a specific option from a <select> element (listbox).

    Args:
        option_text (str): The visible text of the option to select.
    """
    driver = None
    print("--- Lab Sheet 12, Level 2: Listbox Search and Select ---")
    print(f"Attempting to select option: '{option_text}'")
```

```
try:
    #1. Setup Driver
    service = ChromeService(executable_path=CHROME_DRIVER_PATH)
    driver = webdriver.Chrome(service=service)
    driver.implicitly_wait(10)

    #2. Open the local HTML file
    driver.get(listbox_test_url)
    print(f"Opened local test page: {HTML_FILE_NAME}")

    #3. Locate the listbox element
    listbox_element = driver.find_element(By.ID, "course_list")
     #4. Wrap the element in the Selenium Select class
    select = Select(listbox_element)
     #5. Search for and select the specified option by visible text
    #This is the core action requested in the problem.
    select.select_by_visible_text(option_text)
     #PROOF: Verify the selected option
    selected_option = select.first_selected_option
    print(f"\nSUCCESS: Selected text is '{selected_option.text}'")
    print(f"SUCCESS: Selected value is '{selected_option.get_attribute('value')}'")

    time.sleep(3) # Pause for visual confirmation

except Exception as e:
    print(f"\nFAILURE: Could not select option '{option_text}'. Error: {e}")

finally:
    ifdriver:
        driver.quit()
```

```
if__name__ == "__main__":
    # Test case: Search for and select "Computer Science Fundamentals"
    option_to_find = "Computer Science Fundamentals"
    search_and_select_listbox_option(option_to_find)
```

Output

```
Listbox Search and Select --- Attempting to select option: 'Computer Science Fundamentals'
Opened local test page: listbox_test_page.html SUCCESS: Selected text is 'Computer
Science Fundamentals' SUCCESS: Selected value is 'cs'
```