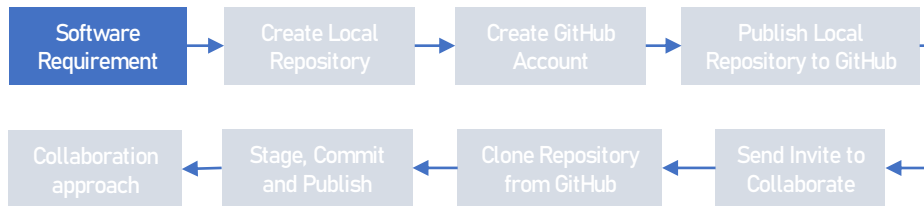




+



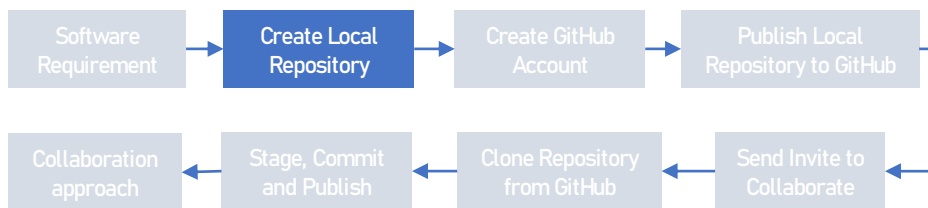
How to collaborate coding with team members



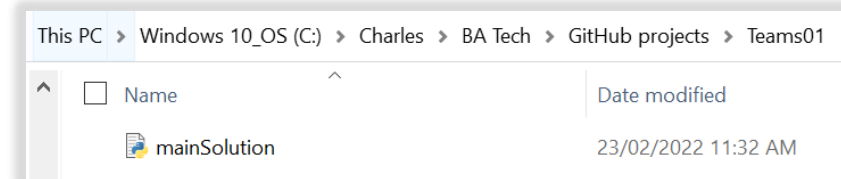
Pre-requisites

Download and install if you do not have following software

1. Git <https://git-scm.com/>
2. Python
<https://www.python.org/downloads/>
3. Visual Studio Code
<https://code.visualstudio.com/download>

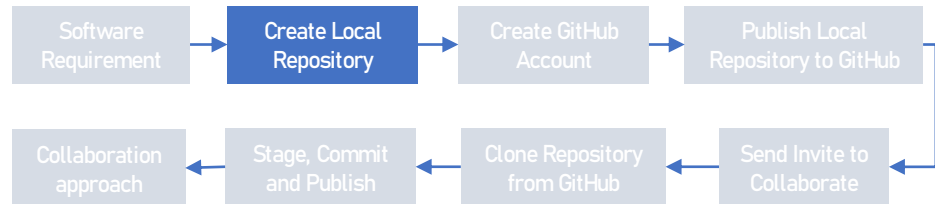


1. Create a project folder and store your files into the folder. In the example below, a folder **Teams01** was created and stores the project python file named **mainSolution**



2. Launch Visual Studio Code and navigate to the folder you have created. In this example, the **mainSolution.py** contains the skeleton of the project, with the all the required functionalities defined (add, delete, multiply)

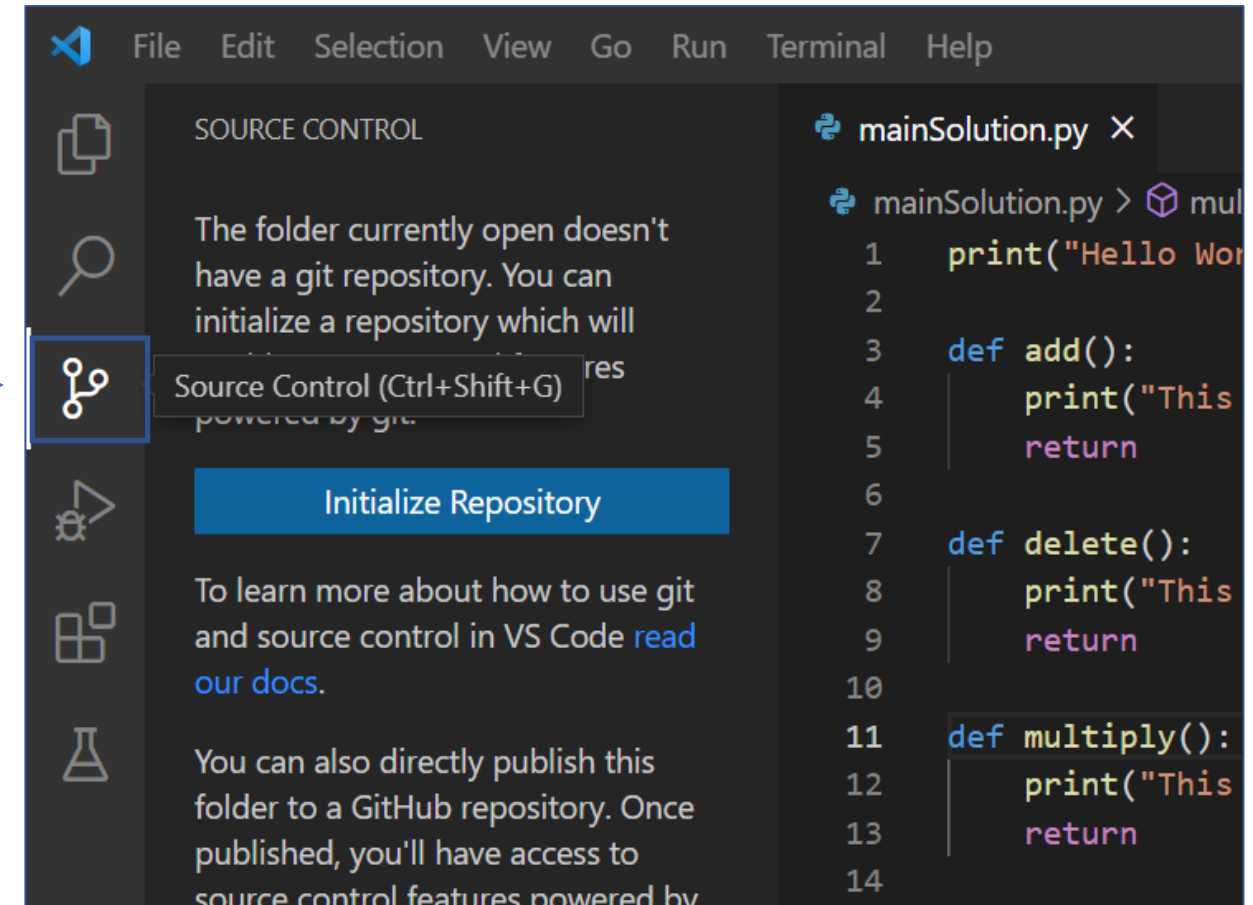
```
File Edit Selection View Go Run Terminal Help mainSolution.py - Teams01 - Visual Studio Code
EXPLORER
TEAMS01
mainSolution.py
mainSolution.py X
mainSolution.py > ...
1 print("Hello World! This is our Team's solution!")
2
3 def add():
4     print("This is function add")
5     return
6
7 def delete():
8     print("This is function delete")
9     return
10
11 def multiply():
12     print("This is function multiply")
13     return
14
15 add()
16 delete()
17 multiply()
18
```

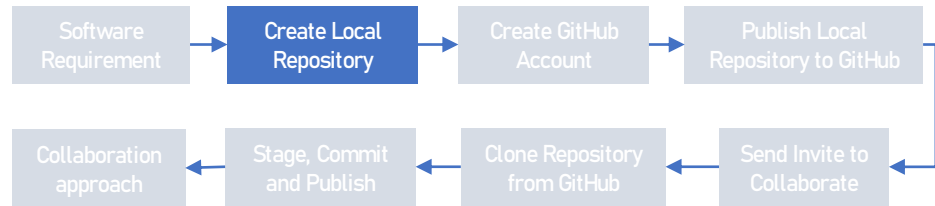


Source Control



3. Select **Source Control** at the left panel or press **Ctrl+Shift+G**, then select **Initialize Repository**



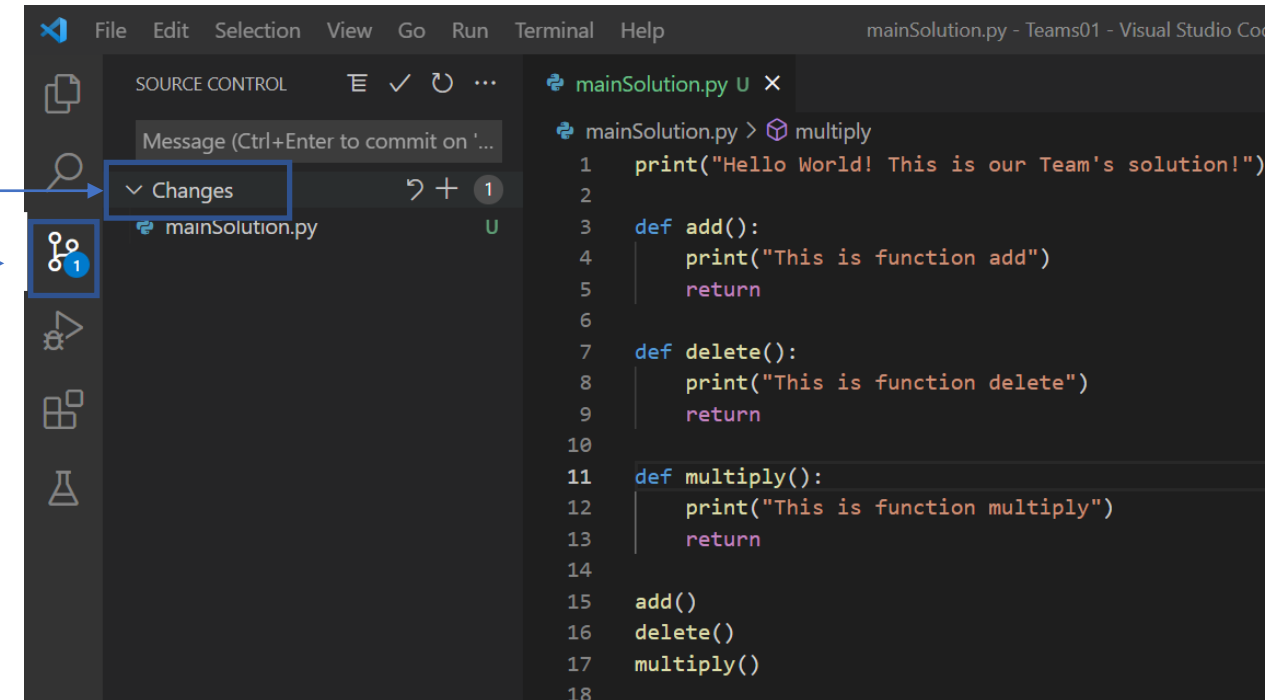


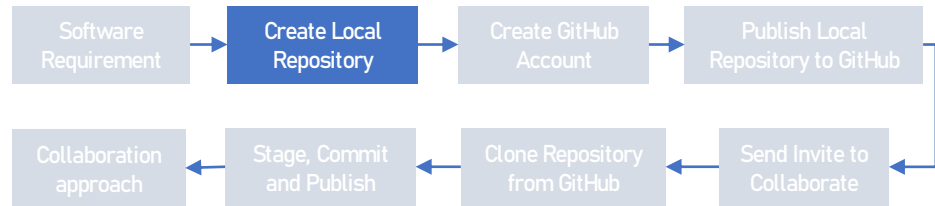
4. After repository is initialized, you will see an expandable list **Changes**.

In our example there is only one file, `mainSolution.py`, inside this list. The letter **u** located on the right-hand side of file `mainSolution.py` indicates that it is untracked or not tracked by Git.

Because this repository was just initialized, the file(s) inside are considered to be changes made to this repository. This number shows that there is **1 pending changes** in our case.

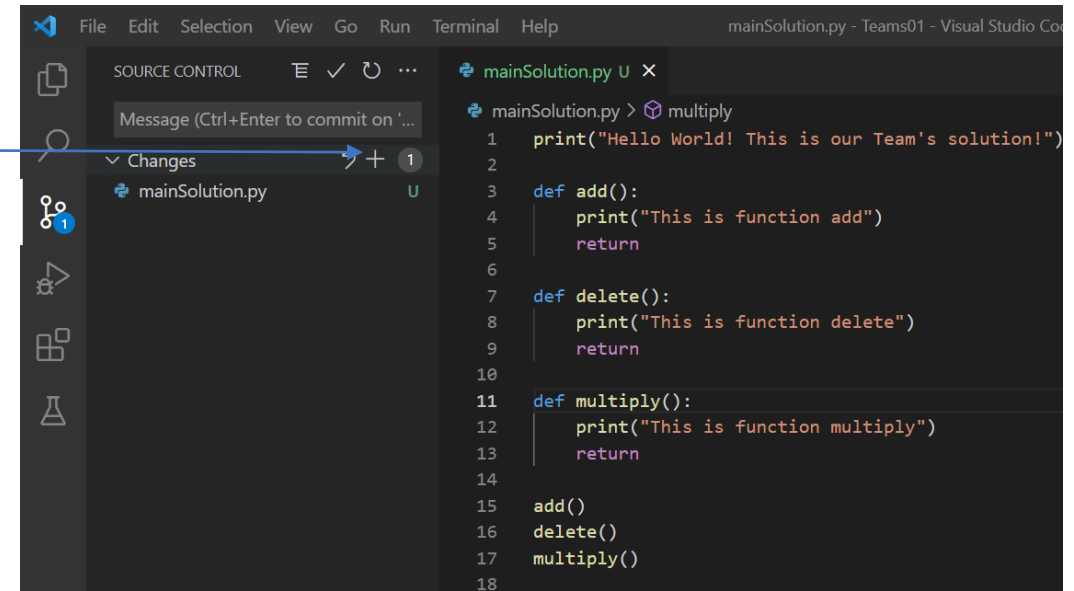
Changes
1 pending changes



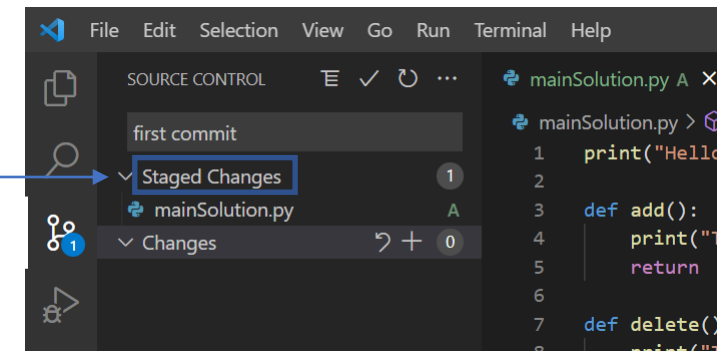


5. Select expandable list **Changes** and click on the + sign to stage all changes.

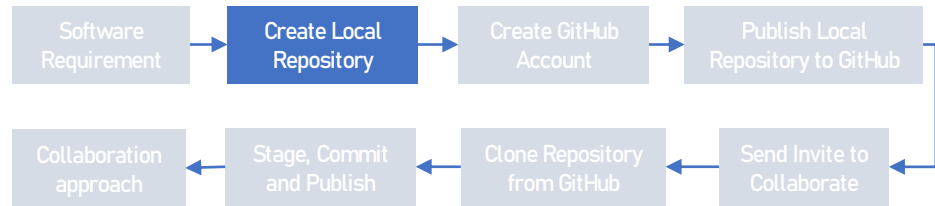
+ sign



6. You should then see that **Staged Changes** appears with the number of files that were staged for changes.

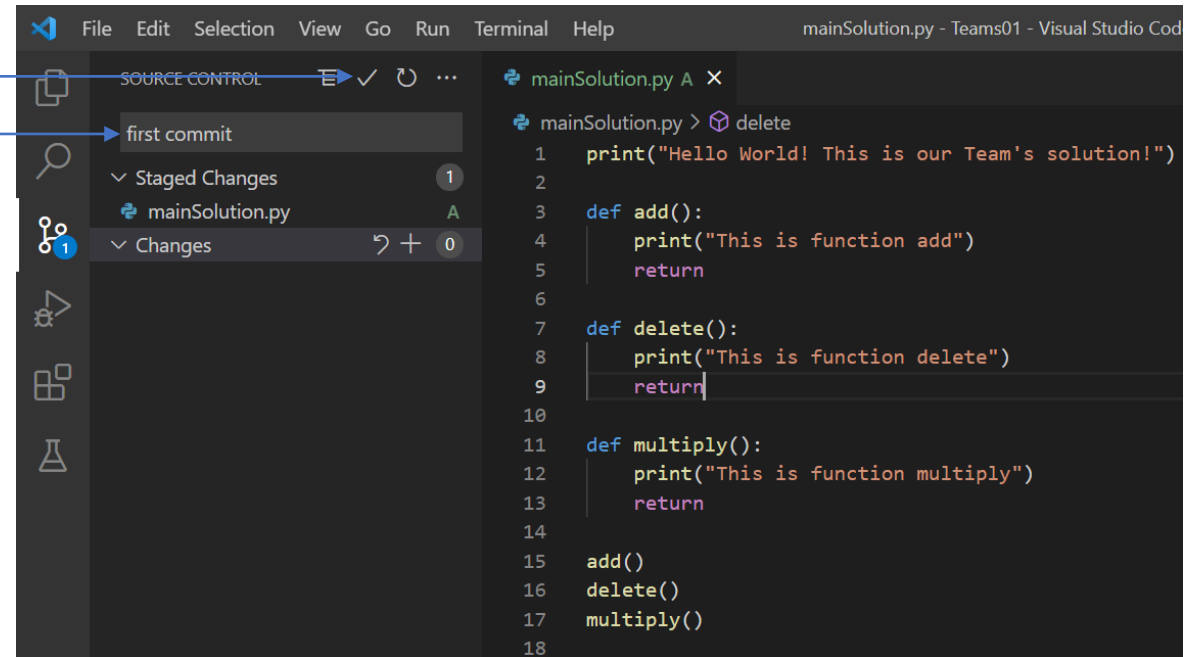


Note: Stage changes means adding all the files that have changes made to be available for Commit.



✓ sign
first commit

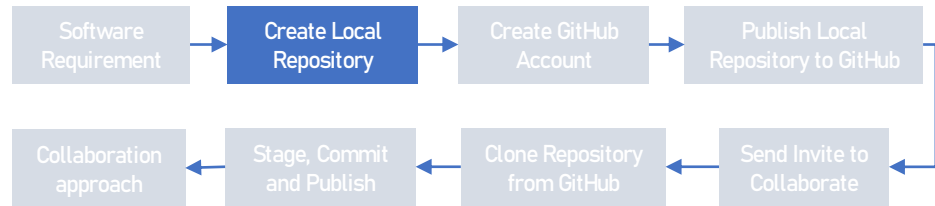
7. Enter a meaningful message inside this box describing what the commit is about. In this example, “**first commit**” was used.



8. Click on the ✓ sign above to commit staged changes.

Note:

Commit captures a snapshot of the project's currently staged changes. Committed snapshots can be thought of as “safe” versions of a project.



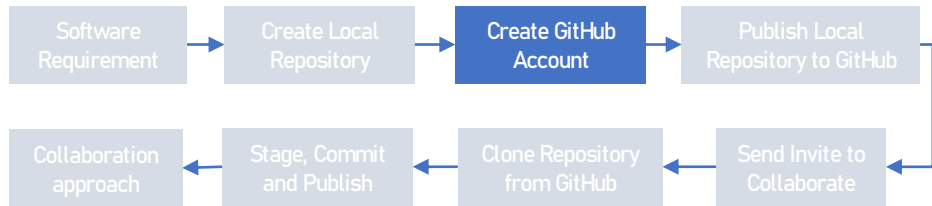
9. After your first commit, an option will be available for you to publish to GitHub.

Follow steps on the next slide to set up username and user email.

DO NOT PUBLISH YET AT THIS POINT AS SOME PREPARATION WORKS ARE REQUIRED

The screenshot shows the Visual Studio Code interface. On the left, the 'Source Control' panel is visible with a 'Publish Branch' button. The main editor displays a Python file named 'mainSolution.py' with the following code:

```
mainSolution.py > multiply
1 print("Hello world! This is our Team's solution!")
2
3 def add():
4     print("This is function add")
5     return
6
7 def delete():
8     print("This is function delete")
9     return
10
11 def multiply():
12     print("This is function multiply")
13     return
14
15 add()
16 delete()
```

1. You need a GitHub account. If you do not have one, sign up at <https://github.com/join>

Join GitHub

Create your account

Username *

teams01member01 ✓

Email address *

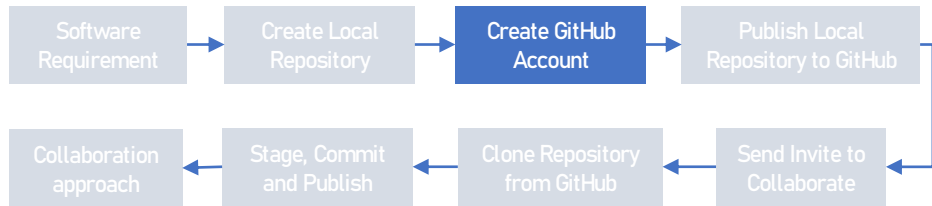
teams01member01@gmail.com ✓

Password *

..... ✓

2. Sign-in to your GitHub account and **remain signed in**.

THIS STEP IS IMPORTANT IF IT IS THE FIRST TIME YOU ARE ACCESSING GITHUB FROM YOUR COMPUTER USING VISUAL STUDIO CODE EDITOR



3. Setup your `user.name` and `user.email` in your local machine where Git was installed. These are the name and email address that was used when creating the GitHub account. At the powershell, enter the following commands to set your `user.name` and `user.email`.

```
git config --global user.name "<username>"
git config --global user.email "<emailaddress>"
git config --global --list
```

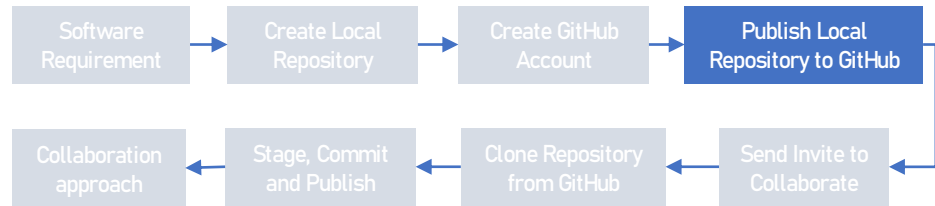
The last command list the `user.name` and `user.email` to verify that the info entered were correctly stored inside Git.



shortcut keys to powershell terminal **CTRL + ~**

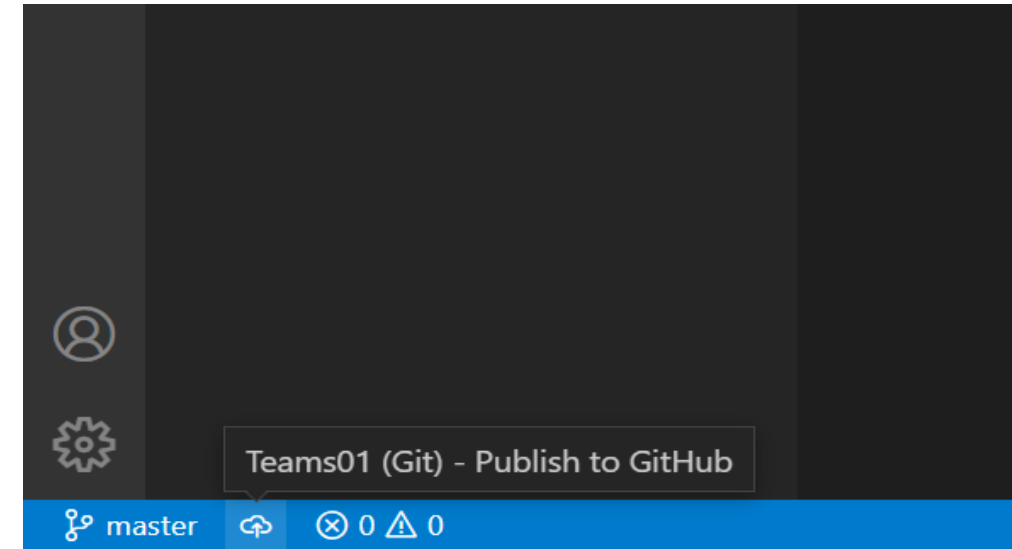
At terminal, setup your `user.name` and `user.email`

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Charles\BA Tech\GitHub projects\Teams01> git config --global user.name "teams01member01"
PS C:\Charles\BA Tech\GitHub projects\Teams01> git config --global user.email "teams01member01@gmail.com"
PS C:\Charles\BA Tech\GitHub projects\Teams01> git config --global --list
user.name=teams01member01
user.email=teams01member01@gmail.com
PS C:\Charles\BA Tech\GitHub projects\Teams01> |
```

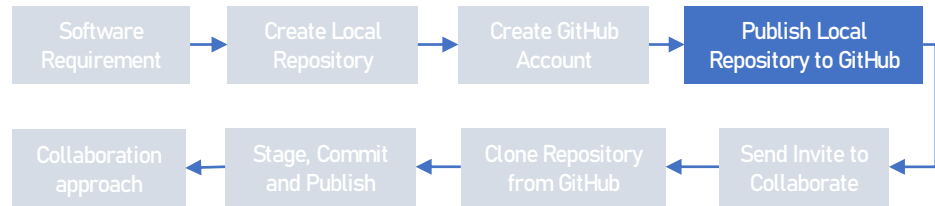


1. At the bottom left-hand side of VS Code, mouse over the **cloud** icon and you should see your project folder name (Teams01 in this example), followed by (Git) – Publish to GitHub.

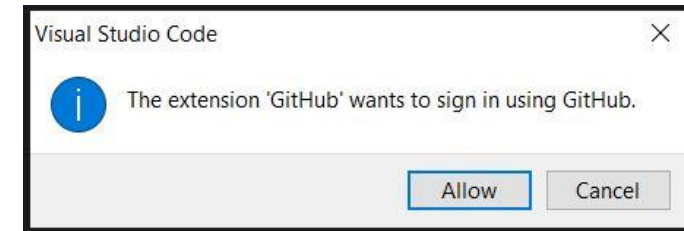
Click to Publish to GitHub.



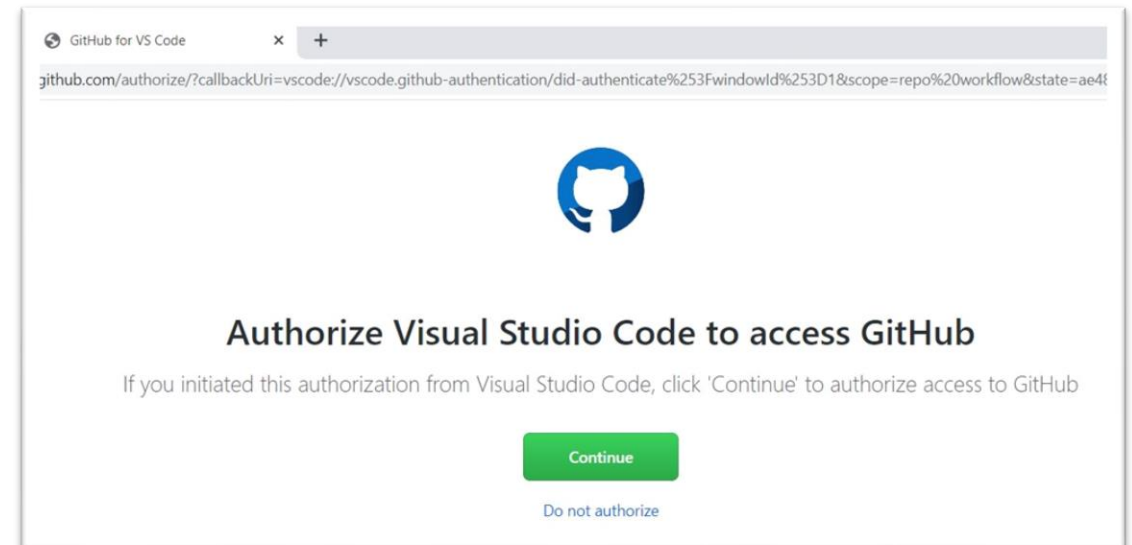
↑
cloud icon

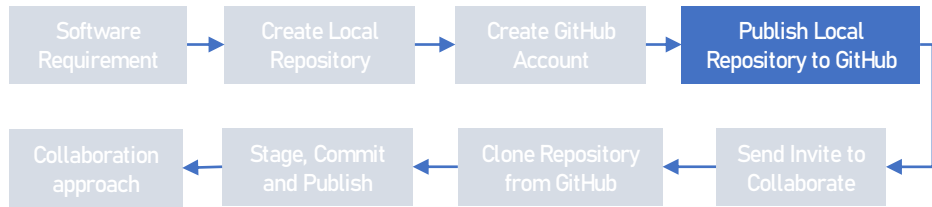


2. This dialog will pop up. Click on “Allow” for extension to sign in using GitHub




3. You will be directed to this website. Click on “Continue” to allow VS Code to access GitHub





4. You should see the following screen that directed you back to VS Code editor.



Success!

Authorization was successful. You will be redirected back to Visual Studio Code

Didn't work?

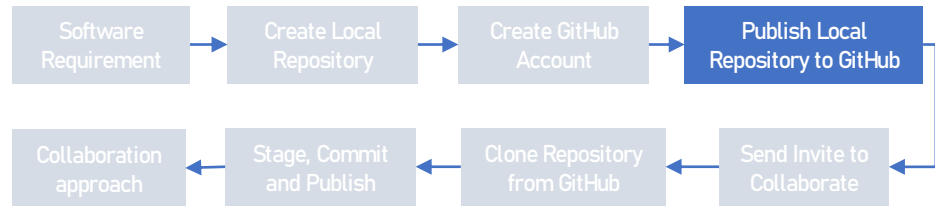
If you aren't redirected, you can add the token manually.

Your authorization token:

```
vscode://vscode.github-authentication/did-authenticate?windowid=1&code=d8d2cfb246d643d6f6458&state=ae48c79e-ba7d-4c01-b549-8d8
```

1. Copy the token.
2. Switch back to VS code.
3. Click **Signing in to github.com...** in the status bar.
4. Paste the token and hit **enter**.

If it didn't work, follow these instructions



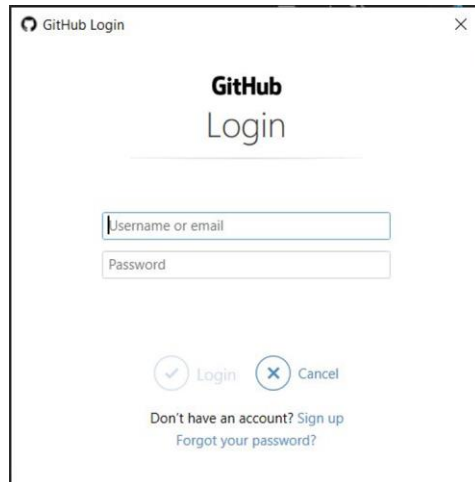
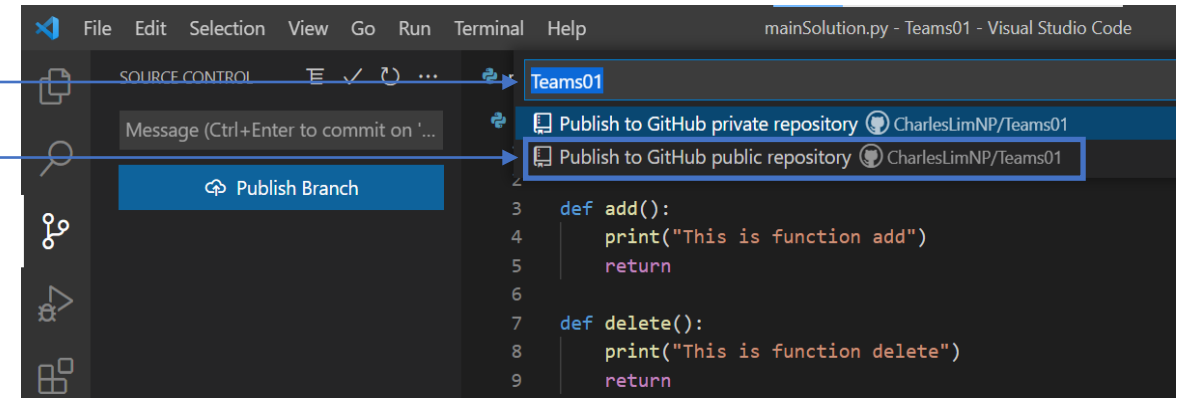
5. The remote repository name, by default, will be the same as your folder name.

`Teams01` appears as in this example which means this repository name is available for use. If a name is already in use, you will have to use another available name.

Select **Publish to GitHub public repository@....**

Repository name Teams01

Publish to public repository

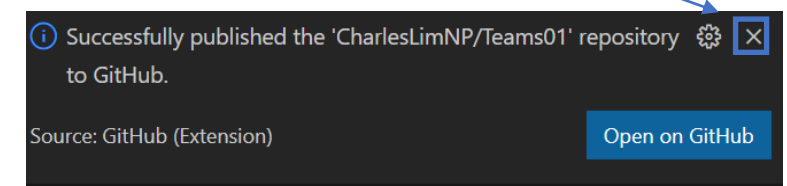


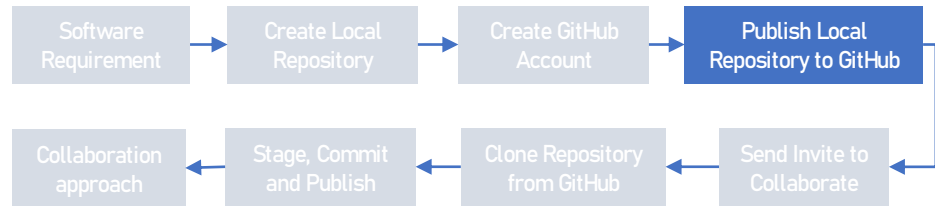
6. If this is the first time you are accessing GitHub using VS Code, this dialog box will appear for you to login and authenticate.

Key in your credentials to login in order to progress with publishing to a remote repository.

7. You should see this dialog box after it was successfully published to the remote repository. Go ahead to **close** it.

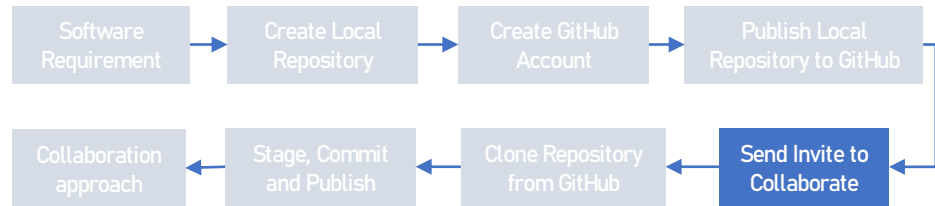
close dialog



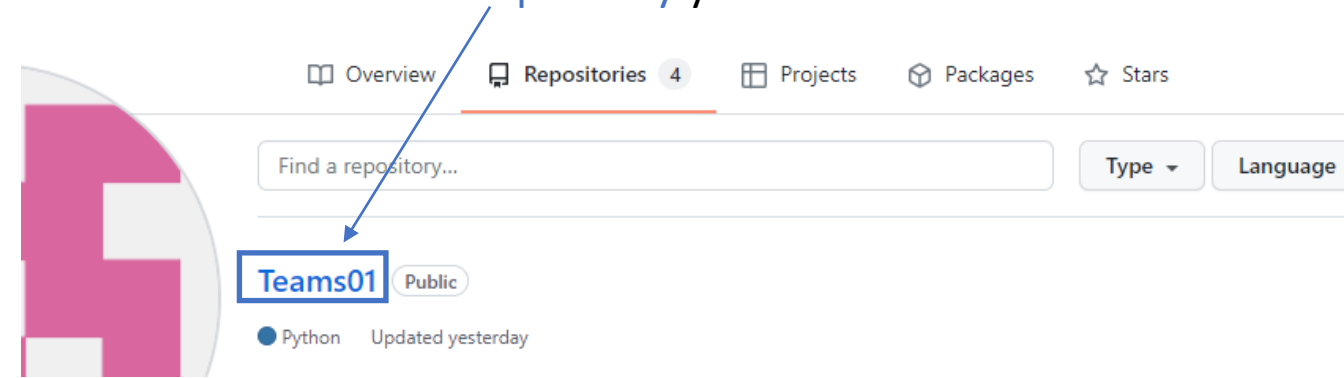


8. Refresh your browser that was signed in to GitHub and you should see that a new public repository **Teams01** was created.

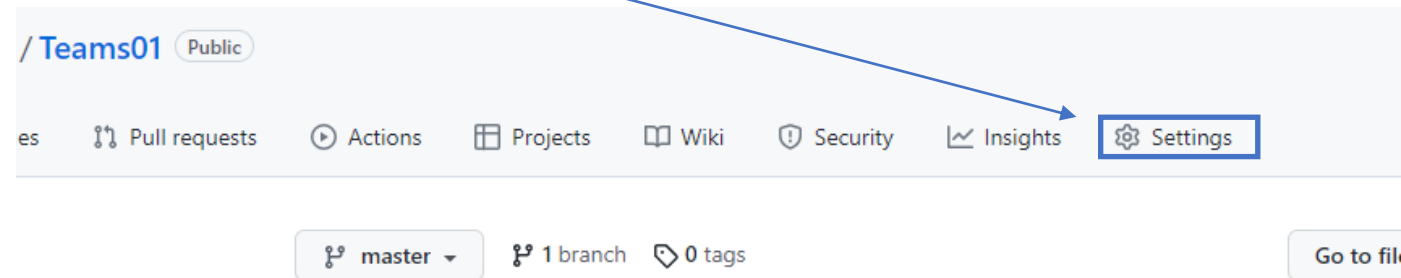
The screenshot shows the GitHub interface. The top navigation bar includes the GitHub logo, a search bar, and links for Pull requests, Issues, Marketplace, and Explore. The user's profile picture and notification bell are on the right. The main content area shows the 'Repositories' tab with a count of 4. A search bar and filters for Type, Language, and Sort are present. A green 'New' button is in the top right. The repository list shows a new repository named 'Teams01' (Public) with a Python icon and a status 'Updated 22 seconds ago'. A blue box highlights the repository name, and a blue arrow points from the text 'new public repository' below to the repository entry.

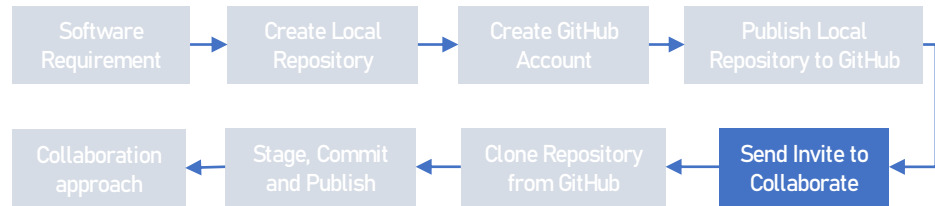


1. Click to select the **repository** you wish to invite collaborators



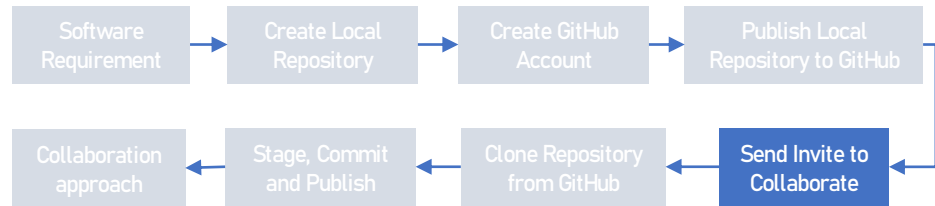
2. Click on **Settings**



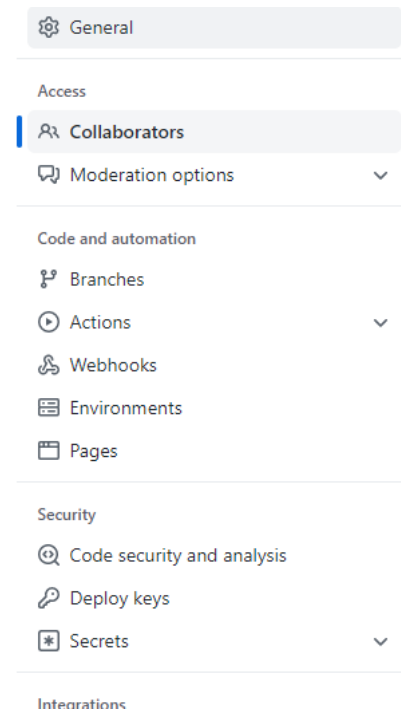


3. Click on Collaborators

A screenshot of the GitHub repository settings page. The 'Settings' tab is selected in the top navigation bar. The left sidebar shows the 'General' section, with the 'Collaborators' option highlighted by a blue box and a blue arrow pointing to it from the text '3. Click on Collaborators'. The main content area shows the 'General' settings, including the 'Repository name' field with the value 'Teams01' and a 'Rename' button. Below this is the 'Template repository' checkbox, which is unchecked. The 'Social preview' section is also visible, with a description and a 'Download template' link.



4. Select **Add people**



Who has access

PUBLIC REPOSITORY

This repository is public and visible to anyone.

[Manage](#)

DIRECT ACCESS

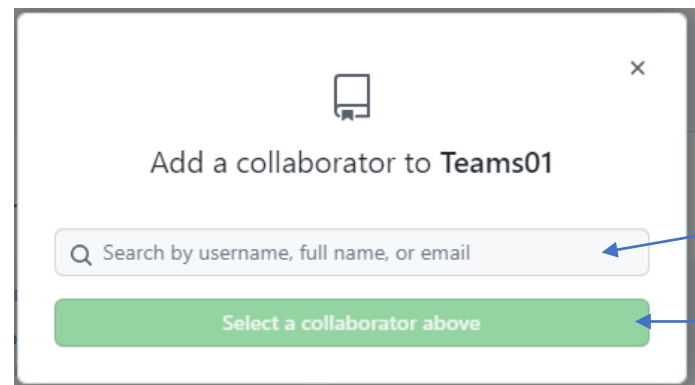
0 collaborators have access to this repository. Only you can contribute to this repository.

Manage access



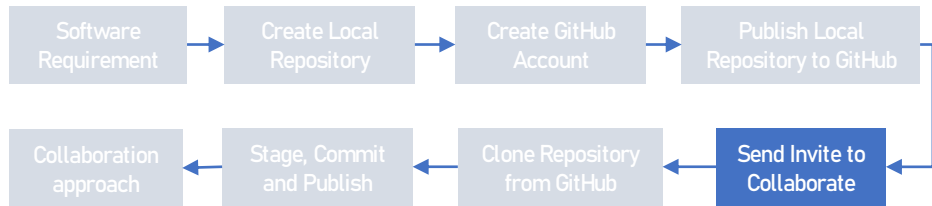
You haven't invited any collaborators yet

[Add people](#)



5. Enter the **username** or **email address** inside this box.

6. Click on **Select a collaborator above** to continue. This will send an invitation notification to the user.



7. An invitation was sent, pending user's acceptance within 7 days, otherwise it will expire.

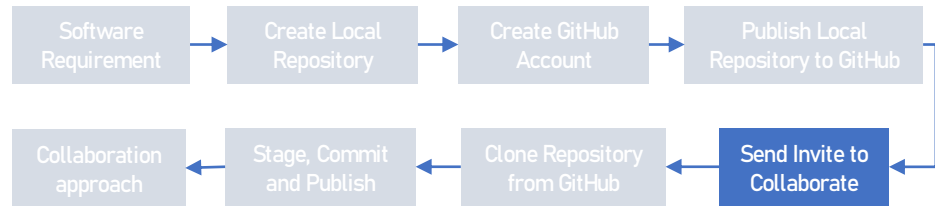
Who has access

Collaborators

Manage access

realbigbear88
Awaiting realbigbear88's response

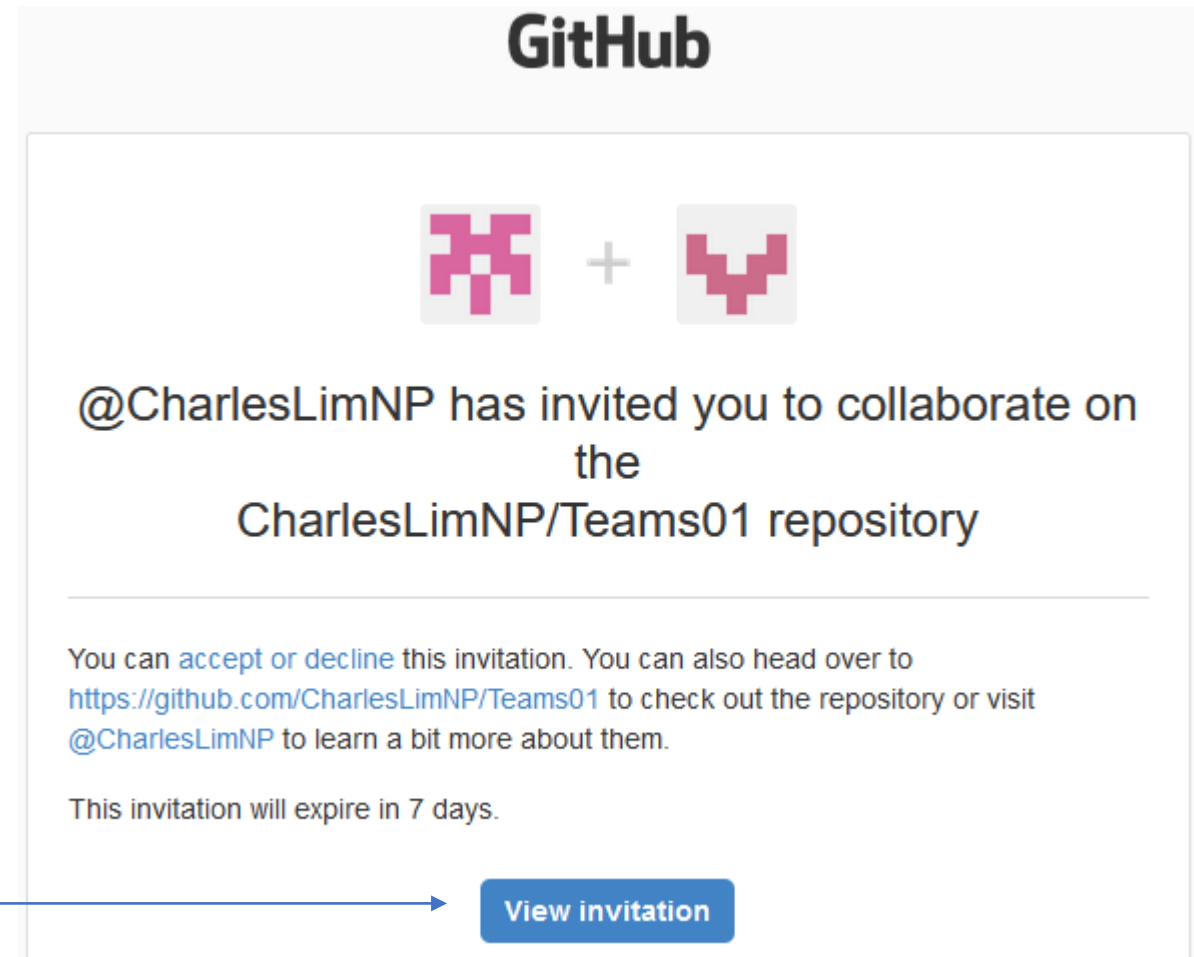
Pending Invite

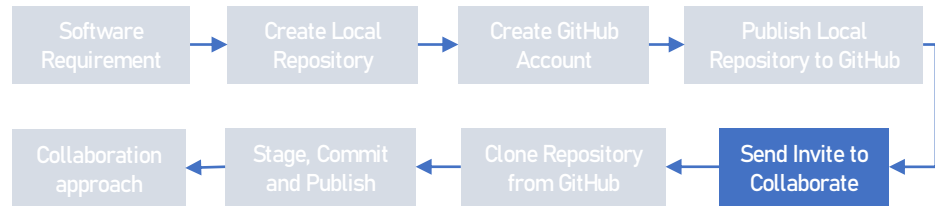


8. Collaborator should receive an invitation via email that looks like this.

Click on [View invitation](#) to proceed

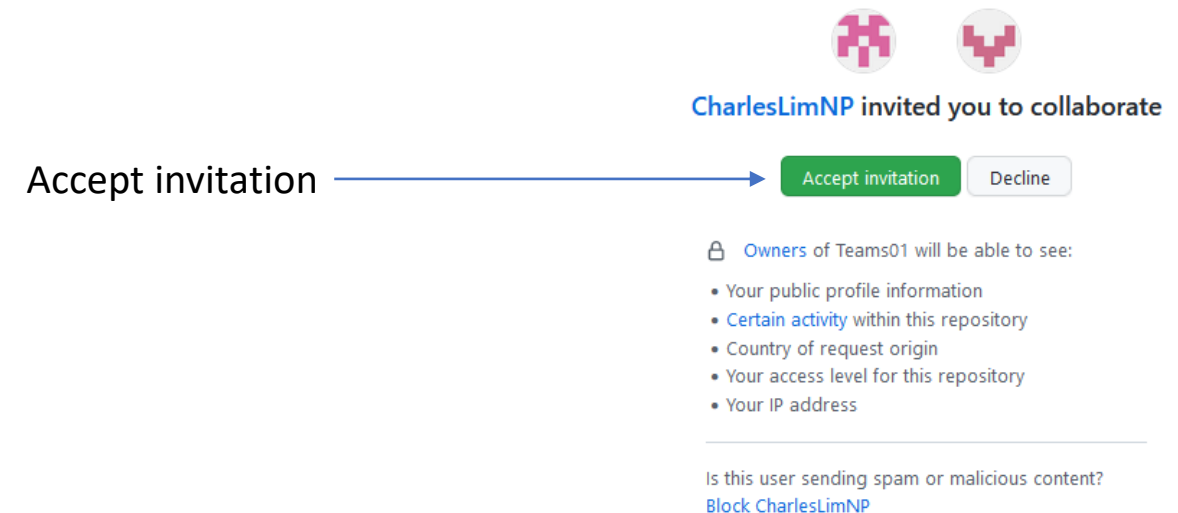
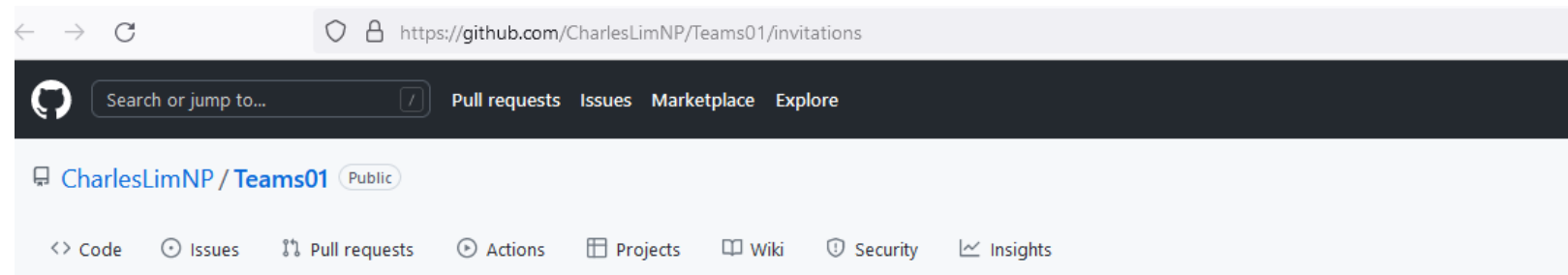
View invitation

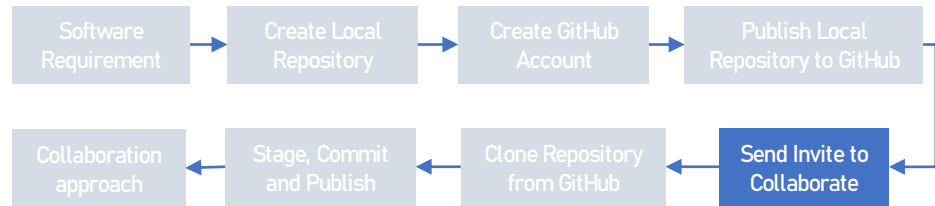




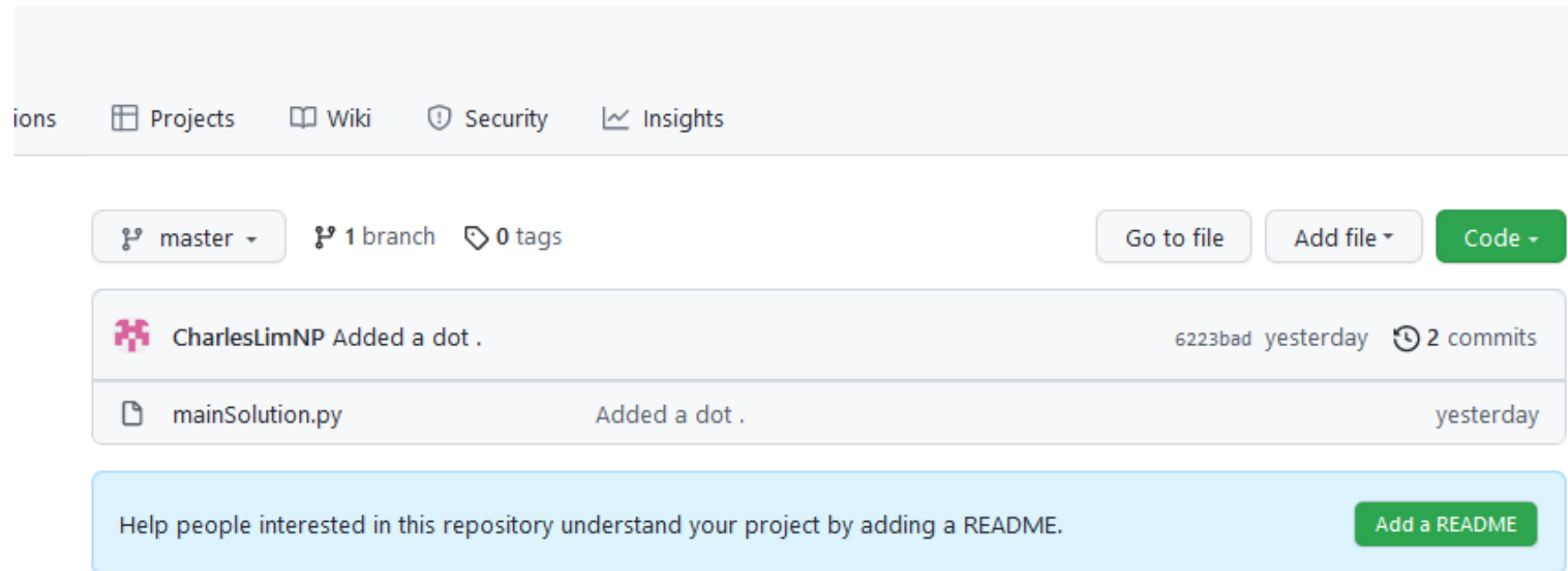
9. You will be directed to GitHub.com to view the invitation. Proceed to [Accept invitation](#).

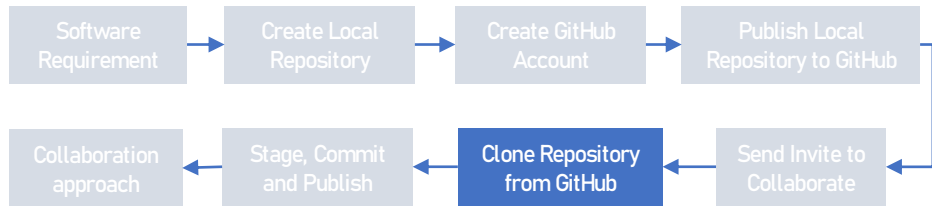
If you are not already logged in to GitHub, you will be prompted for user name and password to proceed.





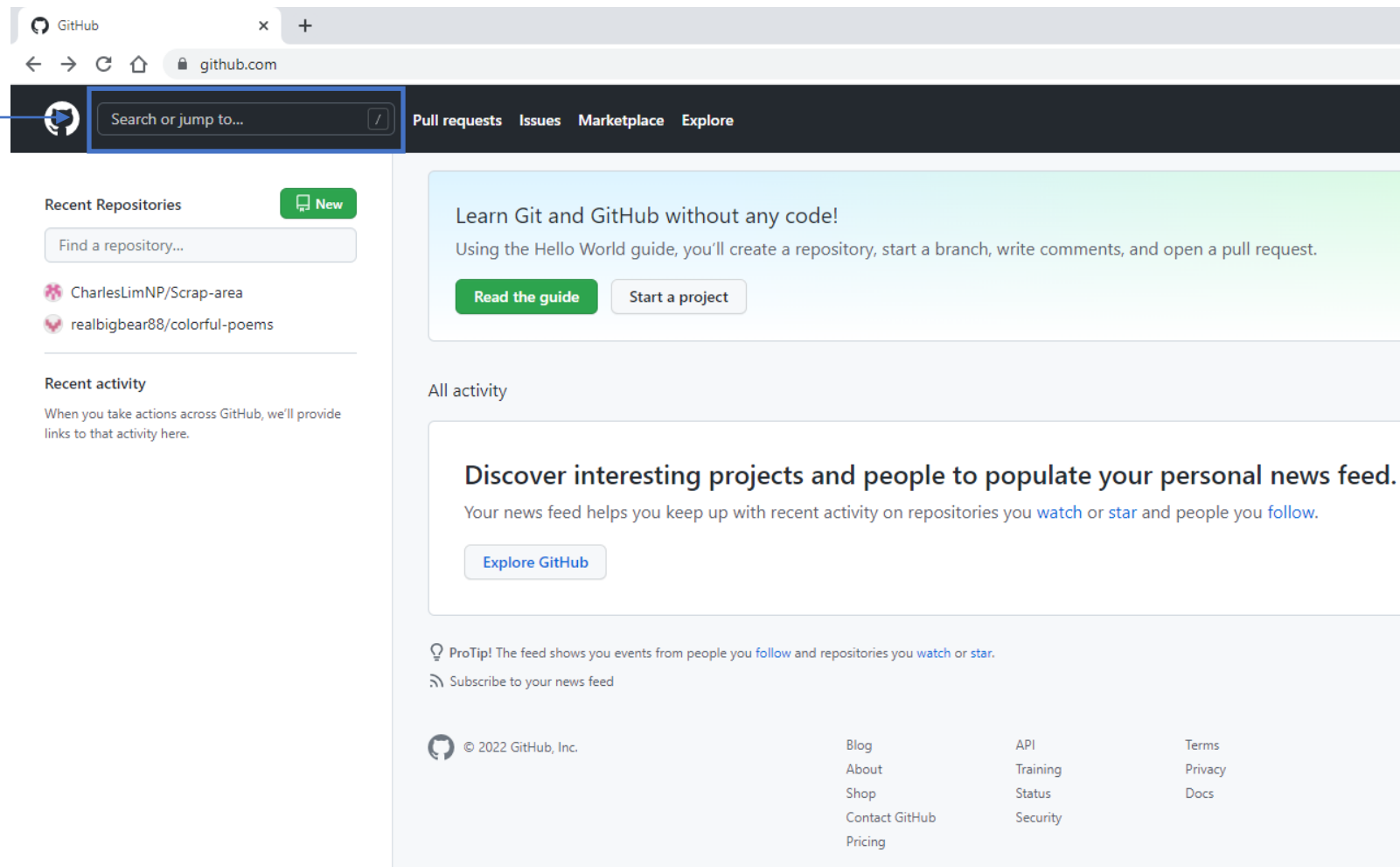
10. You should be able to access all the files and any folders that that are available inside this repository that was shared with you.

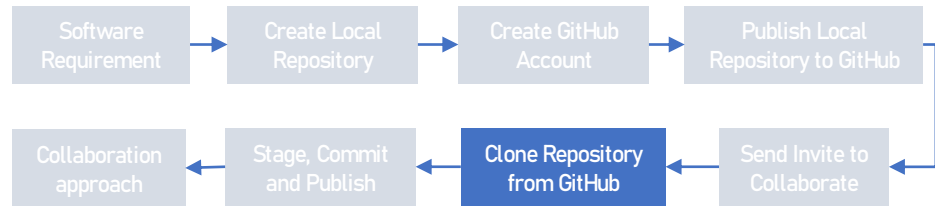




1. Sign in to GitHub and type the name inside here to **search for the repository** that was shared with you.

Search for a repository by typing in the name of repository here





2. In our example, [Teams01](#) was the targeted repository. Click on [Teams01](#) to open repository.

Search · teams01

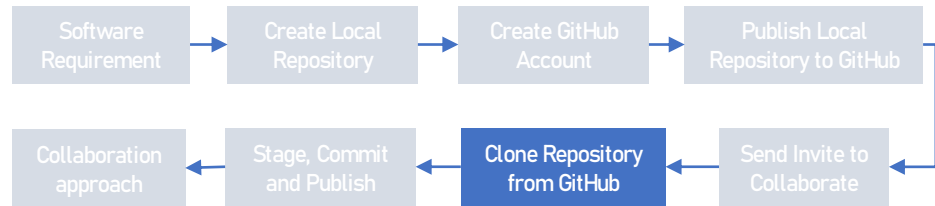
github.com/search?q=teams01

teams01 Pull requests Issues Marketplace Explore

Repositories	1
Code	111
Commits	6
Issues	1
Discussions	0
Packages	0
Marketplace	0
Topics	0

1 repository result

[CharlesLimNP/Teams01](#)
Python Updated 2 days ago



3. Select the **Code** dropdown list and click here to copy the URL link to this repository.

Projects Wiki Security Insights

master 1 branch 0 tags

Go to file Add file Code

CharlesLimNP Added a dot .

mainSolution.py Added a dot .

Help people interested in this repository understand your project by adding

Clone

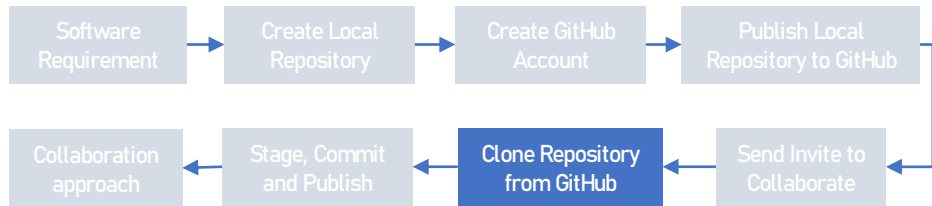
HTTPS SSH GitHub CLI

<https://github.com/CharlesLimNP/Teams01.git>

Use Git or checkout with SVN using the web URL.

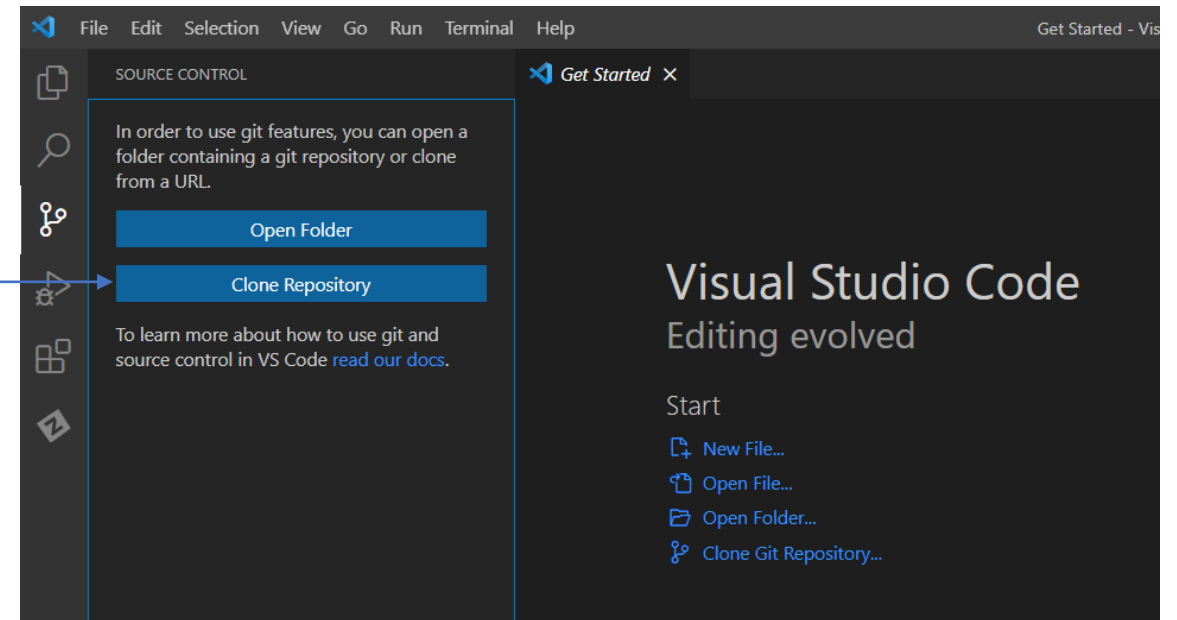
Open with GitHub Desktop

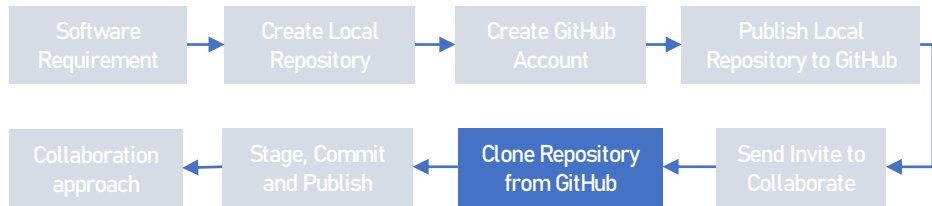
Download ZIP



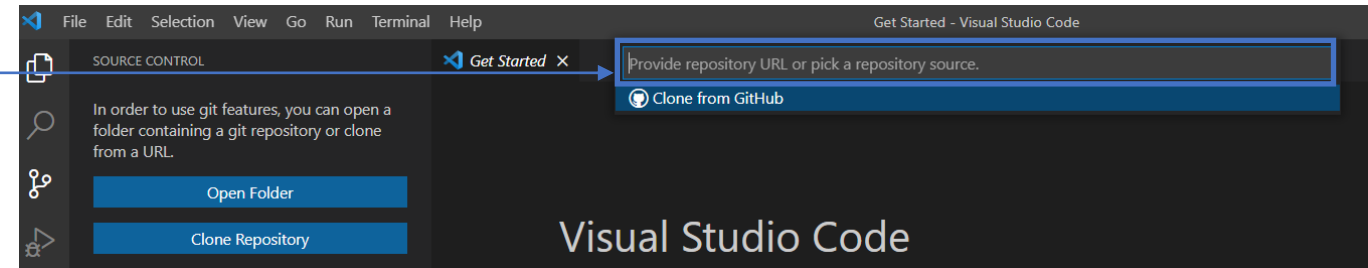
4. Launch Visual Studio Code. If there is already a folder opened, close the folder. Click on Clone Repository

5. If there is already a folder opened, close the folder. Click on **Clone Repository**

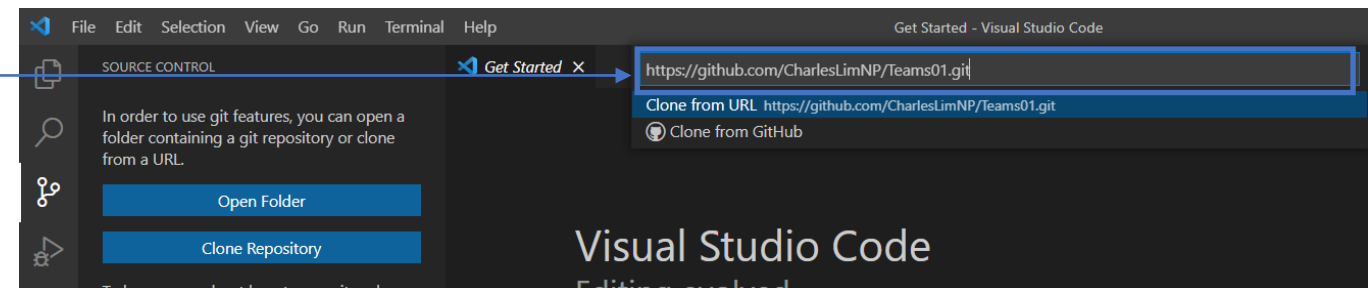


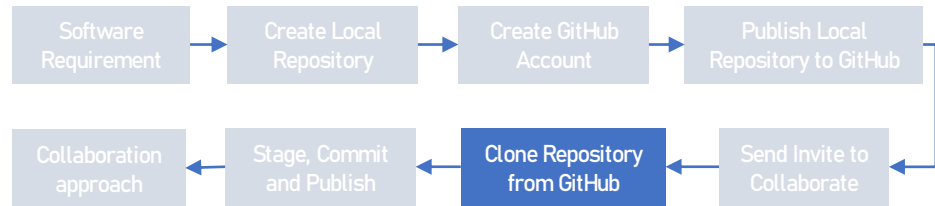


6. An input box will appear for you to provide repository URL or pick a repository source.



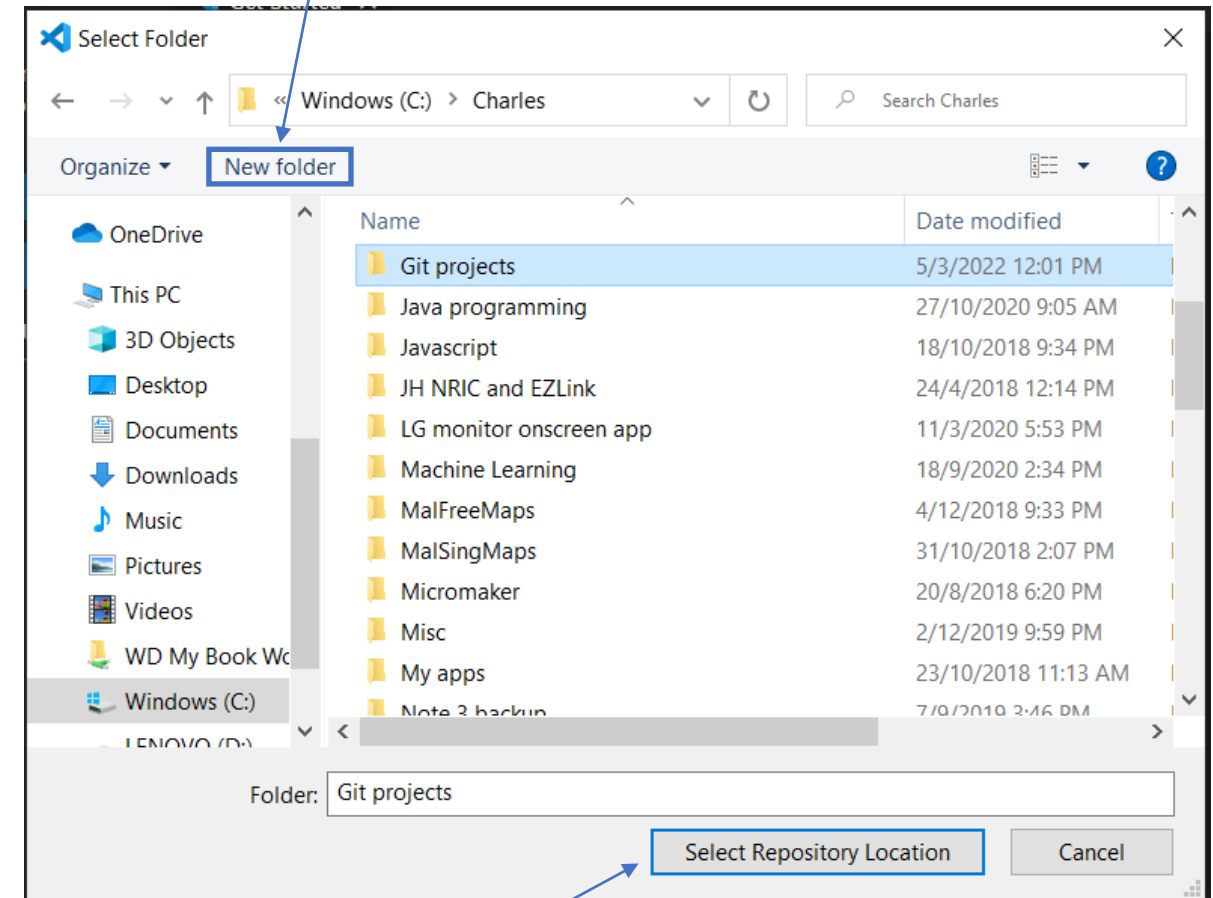
7. Paste the URL of the repository you have copied into the box and hit enter.



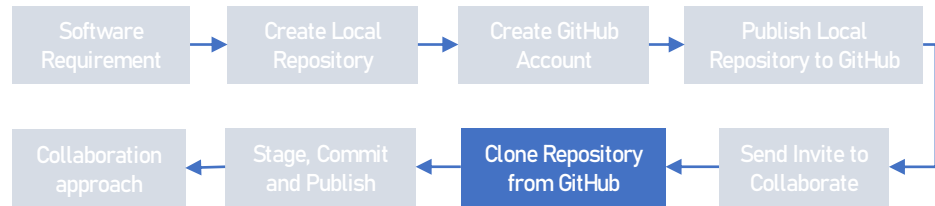


NOTE: During cloning, a new folder will be created with the name **Teams01** which is similar with the remote repository's name.

8. You will be prompted to select a folder as destination to clone the repository. If required, you could create a **New folder**.



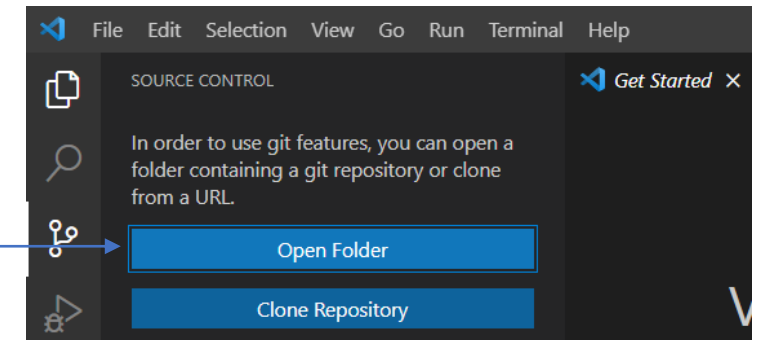
9. Click **Select Repository Location** to proceed with cloning.

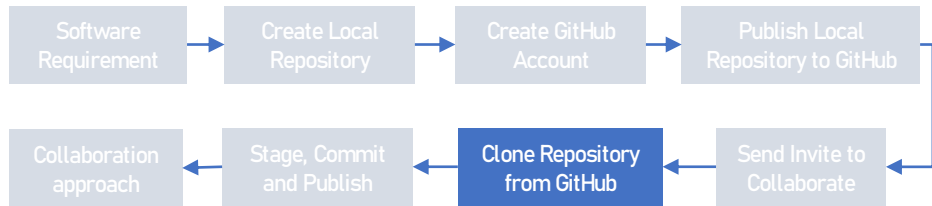


10. To check if cloning was successful, mouse over the group of icons at the bottom left corner of VS Code editor – you should see a **No Problems**



11. Click on **Open Folder** to access the cloned repository.

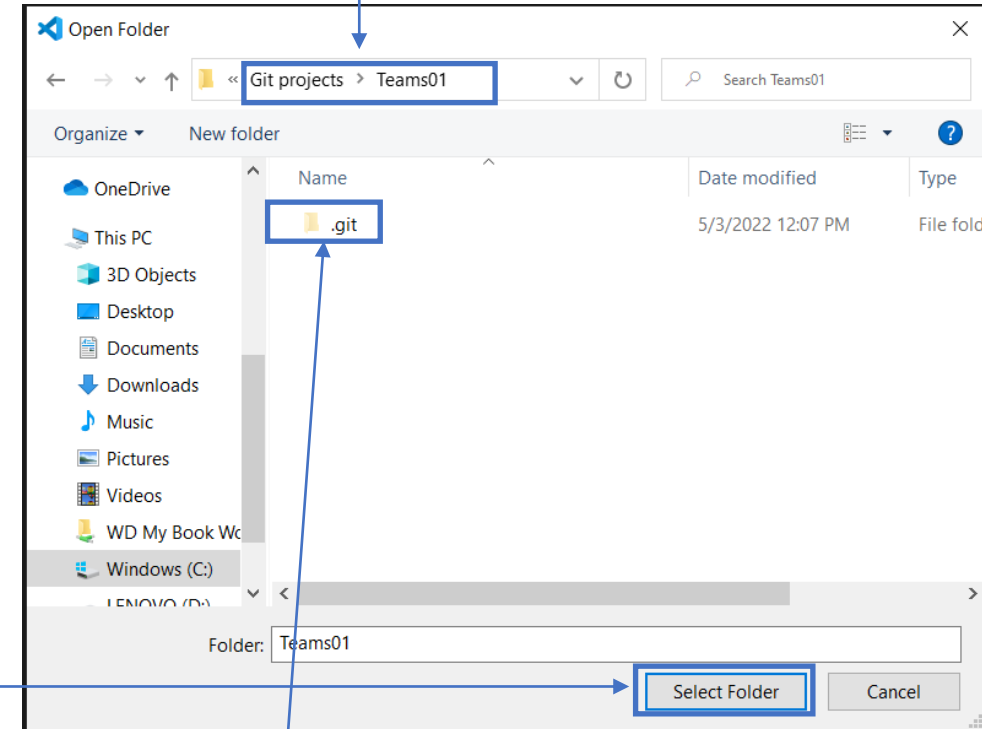




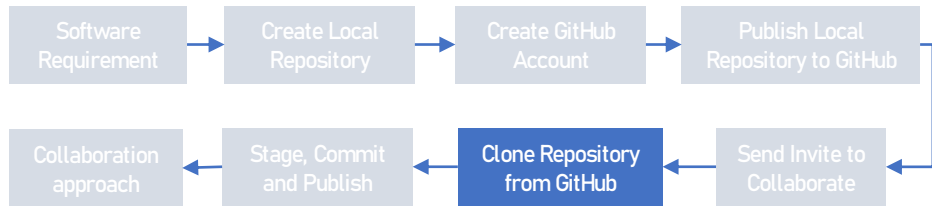
12. Navigate to the folder that was created during cloning (in our example it was **Teams01**) which is located inside the repository location (in our example was **Git projects**) and click on **Select Folder**.

Select the folder where the repository was created

This is the folder created during the cloning of repository in our example.

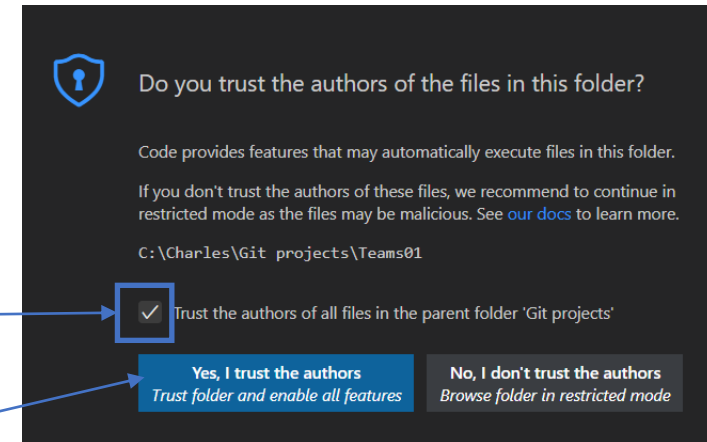


NOTE: The **.git** folder is a hidden folder. It shows that there is a Git repository in this folder. If you are unable to see it, checkout these [instructions](#) on how to view hidden files and folders in Windows.



13. Tick on the checkbox **Trust the authors of all files in the parent folder...**

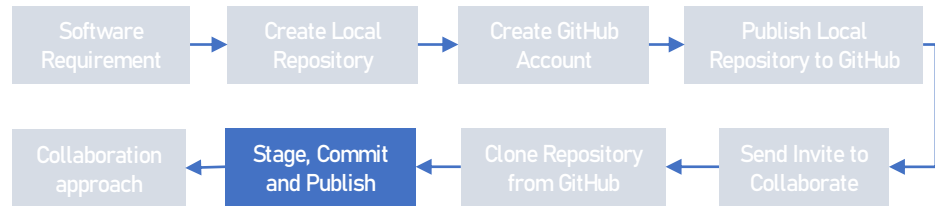
14. Click on **Yes, I trust the authors** to open the cloned repository.



Below is the resulting repository that was cloned.

A screenshot of the Visual Studio Code interface showing a cloned repository. The Explorer sidebar on the left shows the file structure with 'mainSolution.py' selected. The main editor area displays the code for 'mainSolution.py', which includes a print statement and three function definitions: 'add()', 'delete()', and 'multiply()'. The code is as follows:

```
1 print("Hello world! This is our Team's solution!")
2
3 def add():
4     print("This is function add")
5     return
6
7 def delete():
8     print("This is function delete")
9     return
10
11 def multiply():
12     print("This is function multiply.")
13     return
14
15
16 add()
17 delete()
18 multiply()
```



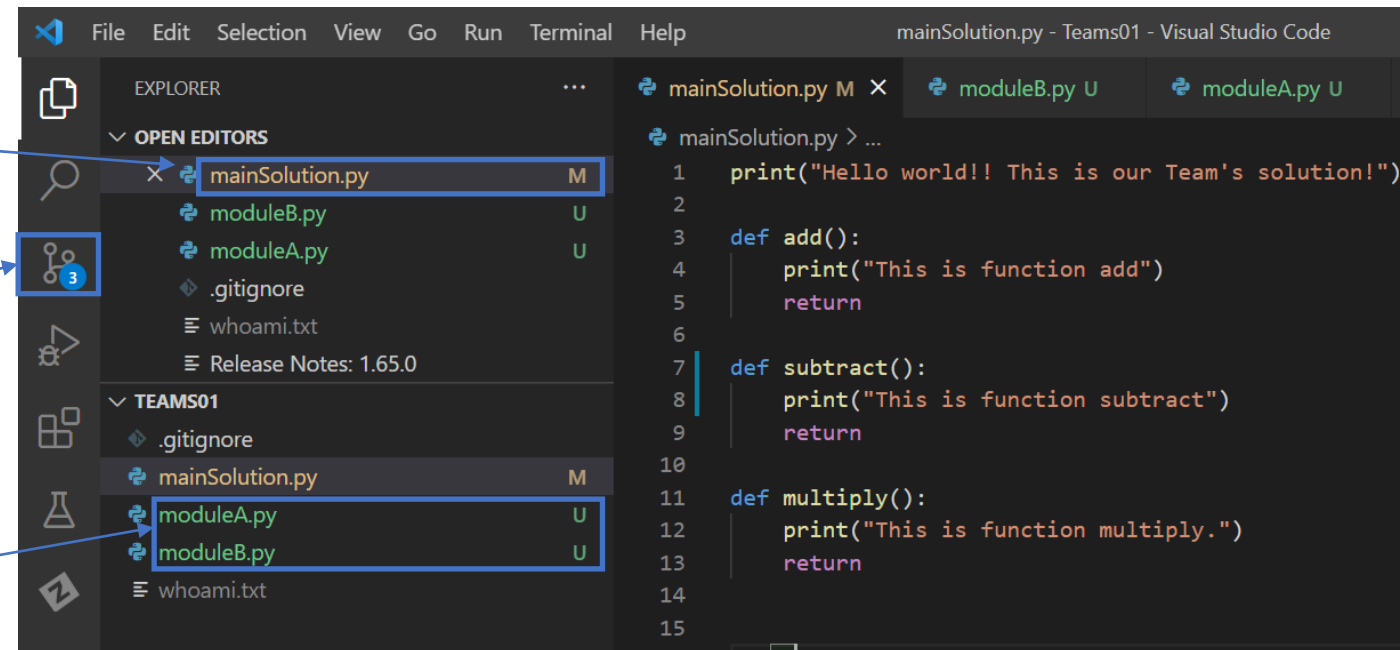
In this example, minor changes were made to the codes inside `mainSolution.py`.

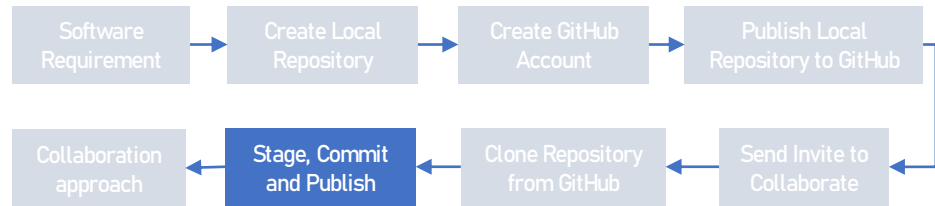
The letter **M** on the right indicates that this file was **Modified**.

The number **3** indicates that there are 3 pending changes.

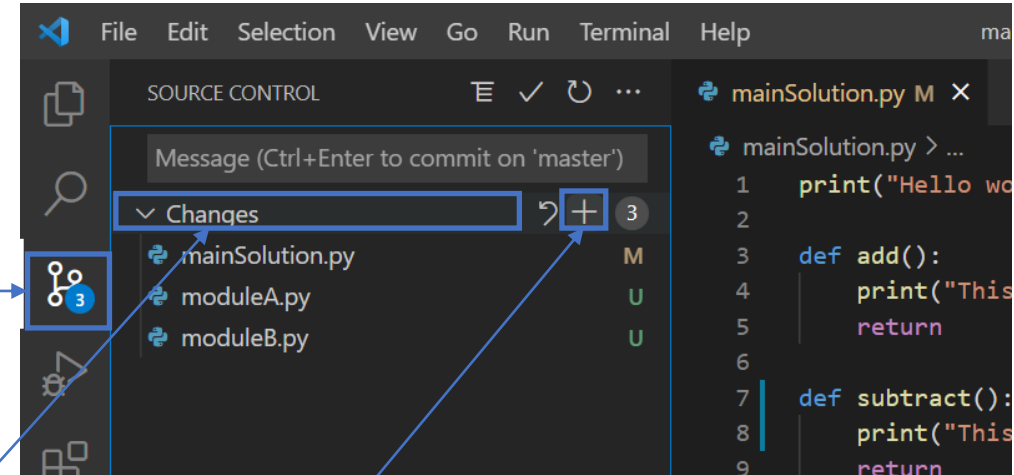
2 new files, `moduleA.py` and `moduleB.py` were added to this depository.

The letter **U** on their right shows that the changes on these files are **Untracked** by Git.



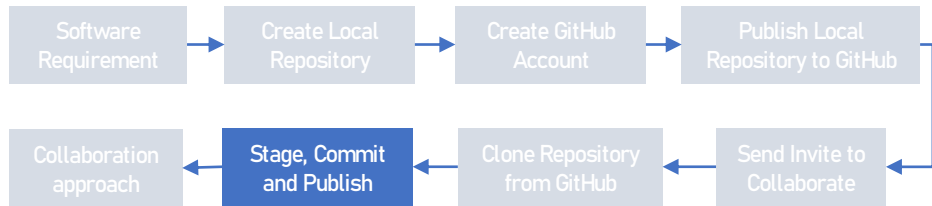


1. Select Source Control



2. Mouse over the expandable list **Changes** until you see the + sign and then click on it to Stage All Changes.

NOTE: When you mouse over each individual file, the + sign you see will only Stage Changes for that selected file.



3. After you have Stage All Changes, enter a meaningful **commit message** describing the changes made for this commit.

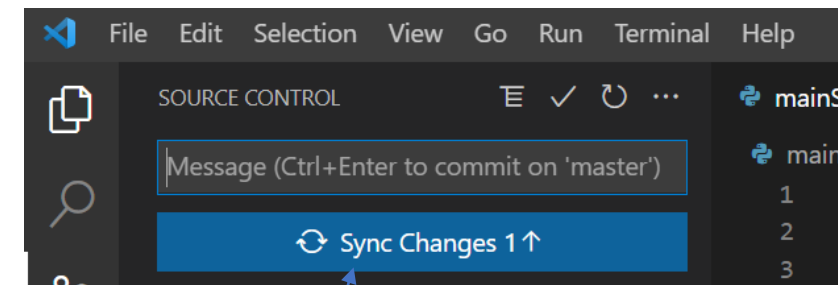
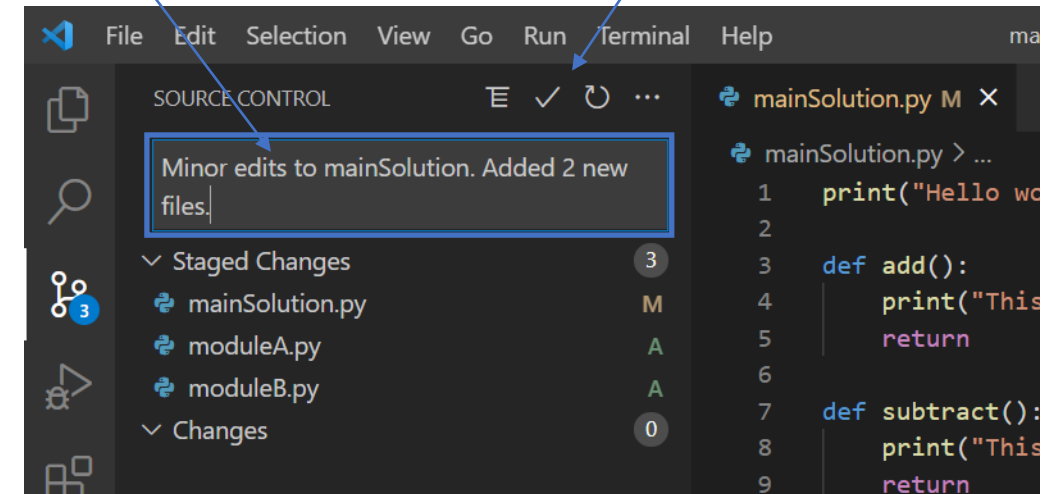
4. Once you are done, press **CTRL+Enter** to commit. Alternatively, you can click on the ✓ located above to commit the changes.

Optional reading: [Art of writing a good commit message](#)

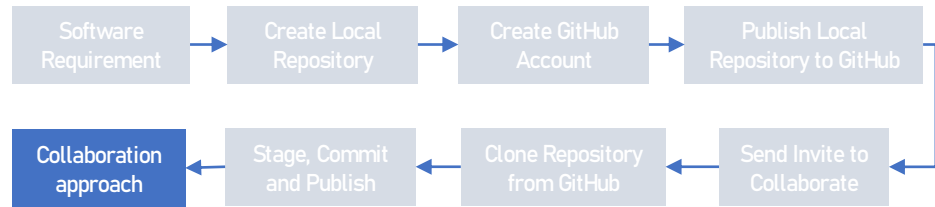
5. After committing the changes, the option to **Sync Changes** will be available to push/pull committed changes to/from remote repository. Click on **Sync Changes** to synchronize changes with the remote repository.

Commit message

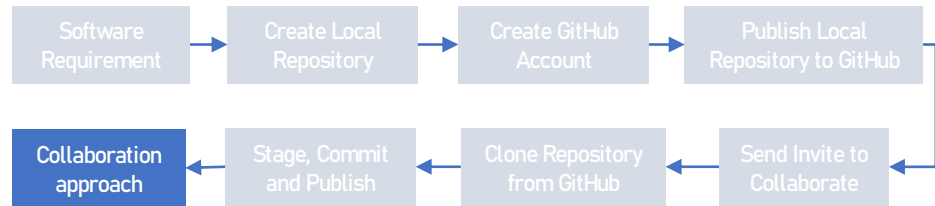
✓ to commit changes



Sync Changes



- Main solution consists of individual modules to be developed by members.
- One or more members will be responsible for designing the main flow of the solution, the functionalities and specifications of the required functions of each module. The same member(s) will be responsible for importing these modules into the main solution for implementation and testing.
- Other members will be responsible for implementing and testing specific functions in each of the modules they are responsible for. In summary, one member will work on one module with specific function(s) that do not overlap with other modules.
- Communication is key to collaboration. Changes, updates and bug fixes should be communicated among members.

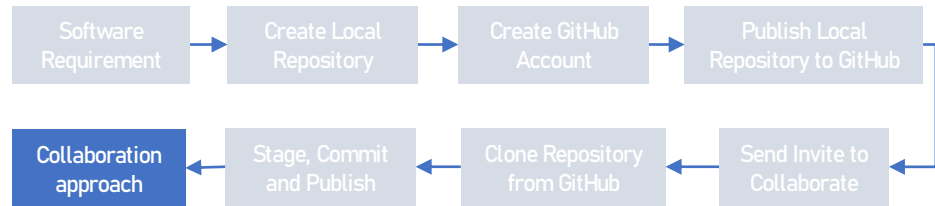


Upon joining the project team, a new project member will need to clone the remote repository which contains all the updated project files and folder, if any.

The screenshot shows the Visual Studio Code interface. The Explorer pane on the left displays the file structure with 'mainSolution.py' selected. The Editor pane shows the content of 'mainSolution.py', which includes a print statement and three functions: 'add()', 'delete()', and 'multiply()'. The Terminal pane at the bottom shows the output of the 'git config --global --list' command, displaying the user's email and name.

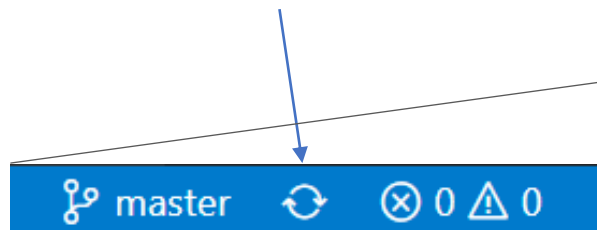
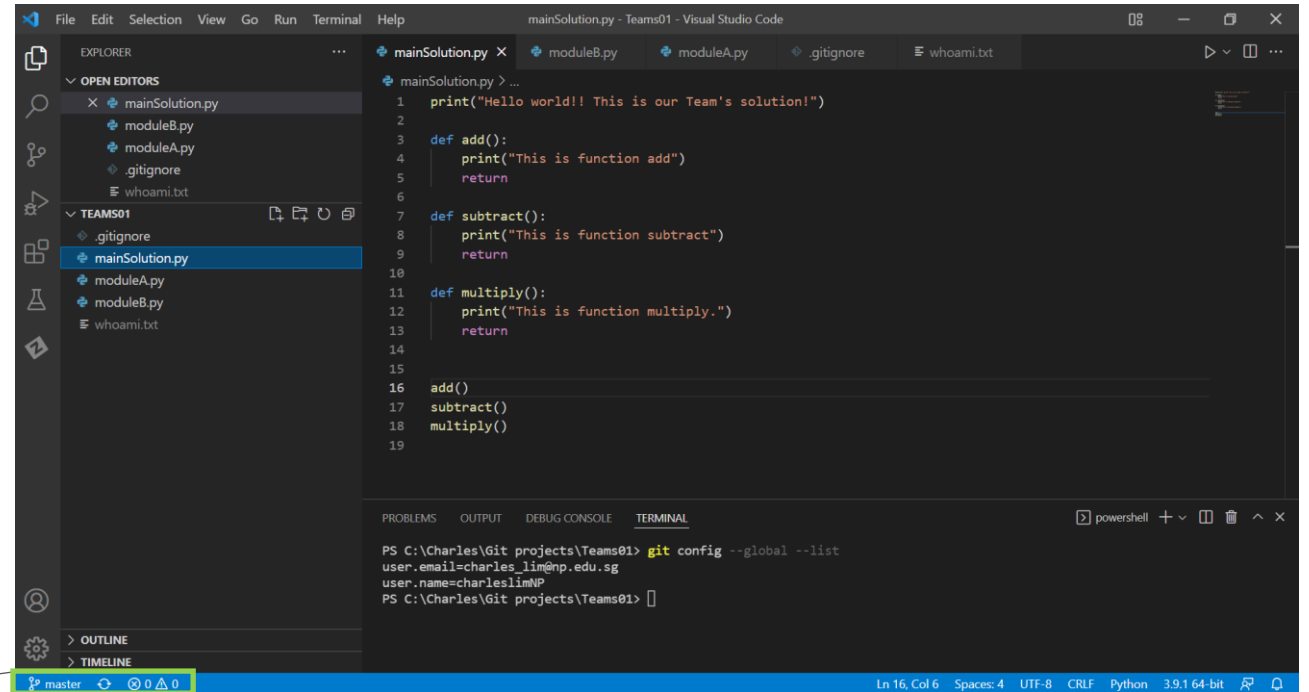
```
mainSolution.py
1 print("Hello world!! This is our Team's solution!")
2
3 def add():
4     print("This is function add")
5     return
6
7 def delete():
8     print("This is function delete")
9     return
10
11 def multiply():
12     print("This is function multiply.")
13     return
14
15
16 add()
17 delete()
18 multiply()
```

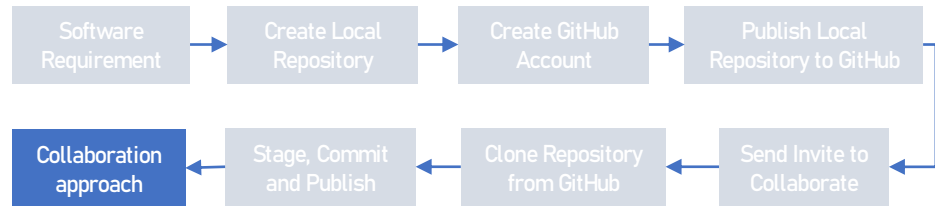
```
PS C:\Charles\Git projects\Realbigbear88\Teams01> git config --global --list
user.email=realbigbear@gmail.com
user.name=realbigbear88
PS C:\Charles\Git projects\Realbigbear88\Teams01>
```



Another project team member working on a separate computer made changes to a file, added 2 files and then **Sync Changes** to remote repository.

Sync Changes will push/pull committed changes to/from remote repository.

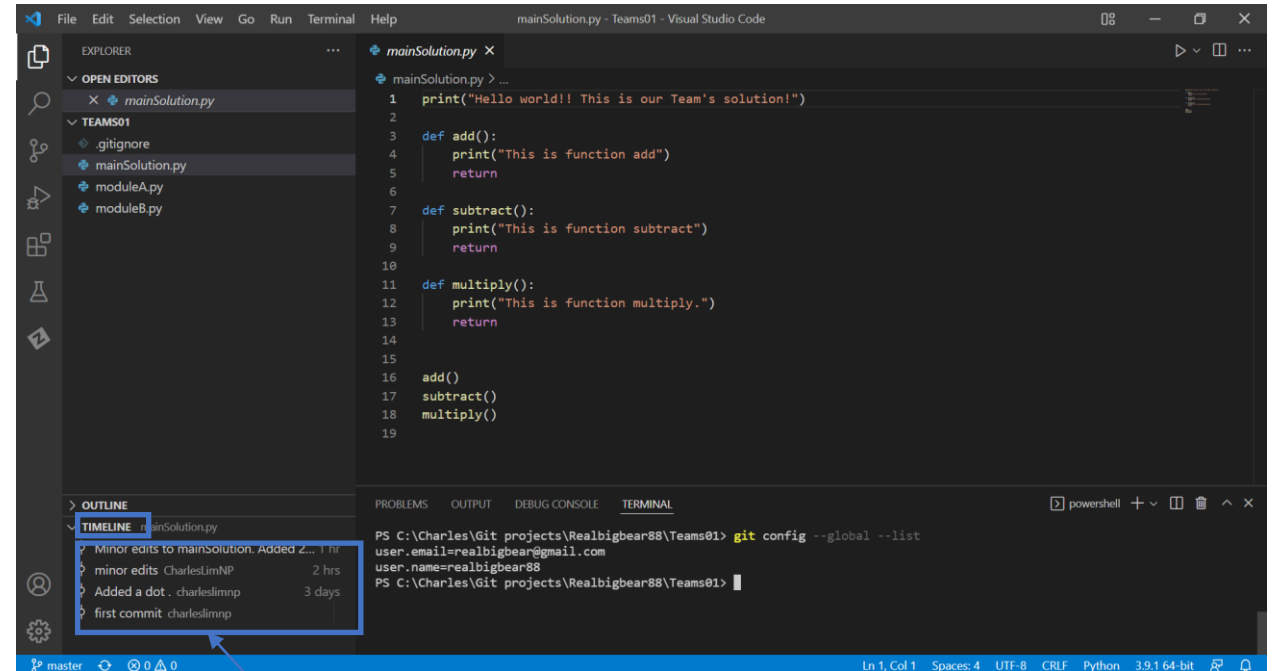




Every time before a member starts working on the project, he should **Sync Changes** as the first step. This is to ensure that changes made by other members will be updated in the local repository.

In this example, the changes committed that has been pushed to the repository by other team members will be downloaded to the local repository after Sync Changes.

In the same manner, this member should sync any changes to the remote repository for collaboration at the end of the day.



Note that the **commits** (items inside expandable list **TIMELINE**) made by other team members are also available with their commit messages and will be refreshed each time Sync Changes is performed.

Practice

Select a **team leader** to –

- Create a local folder with the following Python files:
 - A main solution file `mainSolution.py` that will be used to call functions from imported modules;
 - 4 empty module files: `add.py`, `subtract.py`, `multiply.py` and `divide.py`.
- Assign one of the module files to each team member for implementing the respective functions add, subtract, multiply and divide
- Initialise repo to GitHub with the folder
- Publish to GitHub
- Invite members to collaborate

Each **team members** to –

- Clone the repo from invitation
- Implement the function to the assigned file(s) in repo
 - For example, member 1 to write an add function to `add.py`

```
def add(x, y):  
    return x + y
```
- Stage, commit and publish the changes

Expected output of `mainSolution.py`

```
Enter a number: 144  
Enter another number: 12  
Adding 144 to 12 yields 156  
Multiplying 144 with 12 yields 1728  
Subtracting 144 from 12 yields -132  
Dividing 144 with 12 yields 12.0
```