# User

I have created new repository on github - URL: https://github.com/jdb130496/code-snippets.git
As per brief steps given on github:

echo "# code-snippets" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/jdb130496/code-snippets.git
git push -u origin main

I am not sure of these. I want to only sync my folder: D:\OneDrive - 0yt2k at this new repository and want to ensure that whenever I add or delete any files (or modify) on my local folder - it sync with that repository.

# Bing

Sure, I can help you with that. Here are the steps to sync your local folder D:\OneDrive - 0yt2k with your GitHub repository:

1. **Initialize Git in your local directory**:
   Open a command prompt, navigate to your local directory D:\OneDrive - 0yt2k, and run the following command:
2. **Add the remote repository**:
   Link your local repository to your GitHub repository using the following command:
3. **Add all files in your local directory to the staging area**:
4. **Commit the changes**:
5. **Push the changes to the GitHub repository**:

Now, whenever you add, delete, or modify any files in your local folder, you can sync these changes with your GitHub repository by repeating steps 3 to 5.

Please note that this will make all the files in your D:\OneDrive - 0yt2k folder public on GitHub. If you have any sensitive information in these files, you might want to consider adding those file names to a .gitignore file in your local directory. This will prevent those files from being tracked by Git and uploaded to GitHub.

Also, remember to replace "Initial commit" with a meaningful message that describes the changes you made. This will help you and others understand what changes were made at each commit.

Let me know if you need further assistance! 😊

# User

When I tried git add . (I am doing it in msys shell), got the error:

$ git add .

error: open(".849C9593-D756-4E56-8D6E-42412F2A707B"): Device or resource busy
error: unable to index file '.849C9593-D756-4E56-8D6E-42412F2A707B'
fatal: adding files failed

# Bing

The error message you're seeing usually occurs when a file is being used by another process, and Git is unable to access it. The file .849C9593-D756-4E56-8D6E-42412F2A707B seems to be the one causing the issue.

Here are a few things you can try:

1. **Close any programs** that might be using the file. If you're not sure which program might be using the file, restarting your computer could help to close any background processes that might be using the file.
2. If the file isn't one you need to track in your Git repository, you can **add it to a .gitignore file**. This tells Git to ignore the file, and it won't try to add it to the repository. You can do this by adding the following line to a file named .gitignore in your repository's root directory:
3. Then, try running git add . again.
4. If you're still having trouble, you can try adding files individually or in smaller groups, rather than using git add .. This might help you avoid the problematic file.

Remember, it's important to understand what a file is for before deciding to ignore it. If .849C9593-D756-4E56-8D6E-42412F2A707B is an important part of your project, you'll need to find a way to add it to the repository without causing errors. Let me know if you need further assistance! 😊

# User

Should I add above file in .gitignore inside .git folder?

# Bing

No, the .gitignore file should not be placed inside the .git folder. It should be in the root directory of your project, at the same level as the .git directory.

Here's how you can create or modify the .gitignore file:

1. Navigate to your project's root directory (in your case, it's D:\OneDrive - 0yt2k).
2. If a .gitignore file doesn't already exist, you can create one using the command touch .gitignore in your terminal.
3. Open the .gitignore file in a text editor.
4. Add the following line to the file:
5. Save and close the .gitignore file.

Now, Git will ignore the .849C9593-D756-4E56-8D6E-42412F2A707B file when you run git add . or git commit.

Remember, the .gitignore file is case sensitive and follows certain [pattern formats](https://git-scm.com/docs/gitignore). Make sure to correctly specify the names of the files or directories you want Git to ignore. Let me know if you need further assistance! 😊

# User

I got passed that error with your guidance. When I issued command git commit as above, it showed messages like:

create mode 100644 text files - assorted topics/power\_query\_running\_totals.txt
create mode 100644 text files - assorted topics/power\_query\_select\_case\_and\_running\_totals\_custom\_functions.txt
create mode 100644 text files - assorted topics/power\_query\_tr\_cnt\_expense\_wise.txt
create mode 100644 text files - assorted topics/pythonic way of string manipulation in list.txt
create mode 100644 text files - assorted topics/rdrand python package installation instructions.txt
create mode 100644 text files - assorted topics/rdrand\_random\_1600.txt
create mode 100644 text files - assorted topics/vim\_regex\_commands\_to\_convert\_commas\_delimited\_to\_tab\_delimited.txt
create mode 100644 word files for explanations/Excel TAKE function compared with Python and R equivalents.docx
create mode 100644 word files for explanations/Linkedin Article - Linear Programming - Excel Solver Addin.docx
create mode 100644 word files for explanations/Proposed Linkedin Post About xlwings & role of bingai - 19-12-2023.docx
create mode 100644 word files for explanations/explanation about randombetween without duplicate formula.docx

After that when I issued command: git push -u origin main

Errors are:

$ git push -u origin main
error: src refspec main does not match any
error: failed to push some refs to 'https://github.com/jdb130496/code-snippets.git'

I am on msys shell (just to remind you).

# Bing

The error message src refspec main does not match any typically means that the branch you're trying to push (in this case, main) doesn't exist in your local repository.

When you initialized your local repository, Git might have created a default branch with a different name. The default branch name depends on your Git configuration. For many users, the default is master, not main.

You can check the name of your current branch by running the command git branch. The branch you're currently on will be highlighted and marked with an asterisk (\*).

If your current branch is not main, you can rename it using the command git branch -m main. This renames the current branch to main.

After renaming the branch, try pushing again with git push -u origin main.

Here are the commands:

git branch

git branch -m main

git push -u origin main

Let me know if this helps or if you have any other questions! 😊

# User

How I can change master to main?

# Bing

You can change the name of the default branch from master to main using the following commands:

1. First, check out to the master branch:
2. Rename the master branch to main:
3. Push the main branch to your remote repository and set it as the upstream branch:

Please note that after renaming the branch on your local repository, you might also want to update the default branch in your GitHub repository settings to main.

Let me know if you need further assistance! 😊

# User

I did above steps, but was held up at the last step with following messages:

$ git push -u origin main
Username for 'https://github.com':
Password for 'https://jdb130496@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/jdb130496/code-snippets.git/'

# Bing

The error message you're seeing is because GitHub has removed support for password authentication. You'll need to use a personal access token (PAT) or SSH keys for authentication.

Here's how you can create a personal access token on GitHub:

1. Go to your GitHub account settings.
2. Select "Developer settings" from the left sidebar.
3. Click on "Personal access tokens".
4. Click on "Generate new token".
5. Give your token a descriptive name, select the scopes (or permissions) you'd like to grant this token, and click "Generate token".
6. After generating the token, make sure to copy it. You won't be able to see it again!

Now, when you try to push to GitHub from your command line and it asks for a password, use the personal access token instead of your GitHub password.

Please note that you should treat personal access tokens like passwords and keep them secret. If you think your token has been compromised, you should revoke it immediately and generate a new one.

Let me know if you need further assistance! 😊

# User

I cannot trace developer settings but found that I have previously added ssh keys. Will that be useful?

# Bing

Yes, if you have previously added SSH keys to your GitHub account, you can use SSH instead of HTTPS for remote operations. Here's how you can change the remote URL from HTTPS to SSH:

1. First, check the current remote URL:
2. If the URL starts with https://, you can change it to SSH. The SSH URL should look like this: git@github.com:username/repo.git. Replace username with your GitHub username and repo with the name of your repository. Here's the command to change the remote URL:
3. Verify that the remote URL has been changed:

Now, when you push to GitHub, it will use your SSH keys for authentication. Let me know if you need further assistance! 😊

# User

But I want to continue to use https. I can see settings but no "Personal Tokens" etc. Can you search and tell me how to do this personal token thing?

# Bing

Sure, here are the steps to generate a personal access token on GitHub:

1. Click on the profile image of your GitHub account in the upper-right corner of any page[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).
2. Click on "Settings"[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).
3. Click on "Developer settings"[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).
4. Click on "Personal access tokens"[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).
5. Click on "Generate new token"[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).
6. Add a note for your token. This could be something like "Token for command line operations" to help you remember what the token is for[**2**](https://www.geeksforgeeks.org/how-to-generate-personal-access-token-in-github/).
7. Set an expiration for your token. You can choose a preset option or set a custom date[**2**](https://www.geeksforgeeks.org/how-to-generate-personal-access-token-in-github/).
8. Select the scopes for your token. The scopes you select determine what actions the token can perform[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens). For basic operations like pushing and pulling from repositories, you can select the "repo" scope[**2**](https://www.geeksforgeeks.org/how-to-generate-personal-access-token-in-github/).
9. Click "Generate token" at the bottom of the page[**2**](https://www.geeksforgeeks.org/how-to-generate-personal-access-token-in-github/).

After generating the token, make sure to copy it. You won't be able to see it again[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens). Now, when you try to push to GitHub from your command line and it asks for a password, use the personal access token instead of your GitHub password[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).

Please note that you should treat personal access tokens like passwords and keep them secret. If you think your token has been compromised, you should revoke it immediately and generate a new one[**1**](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens).

Let me know if you need further assistance! 😊

Source(s)

1. [Managing your personal access tokens - GitHub Docs](https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens)

2. [How to Generate Personal Access Token in GitHub?](https://www.geeksforgeeks.org/how-to-generate-personal-access-token-in-github/)

3. [Managing your personal access tokens - GitHub Docs](https://docs.github.com/en/enterprise-server@3.9/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens)

# User

Okay I generated token. Let's continue with the steps to sync my folder on local drive with repository.

# Bing

Great! Now that you have your personal access token, you can use it to authenticate with GitHub when pushing changes. Here are the steps to continue:

1. **Add all files in your local directory to the staging area**:
2. **Commit the changes**:
3. **Push the changes to the GitHub repository**:

When you run the git push command, it will prompt you for your username and password. For the username, enter your GitHub username. For the password, paste the personal access token you just generated.

Now, whenever you add, delete, or modify any files in your local folder, you can sync these changes with your GitHub repository by repeating steps 1 to 3.

Remember to replace "Initial commit" with a meaningful message that describes the changes you made. This will help you and others understand what changes were made at each commit.

Let me know if you need further assistance! 😊

# User

See errors below:

$ git push -u origin main
Username for 'https://github.com':
Password for 'https://jdb130496@github.com':
remote: Support for password authentication was removed on August 13, 2021.
remote: Please see https://docs.github.com/en/get-started/getting-started-with-git/about-remote-repositories#cloning-with-https-urls for information on currently recommended modes of authentication.
fatal: Authentication failed for 'https://github.com/jdb130496/code-snippets.git/'

I have used personal token in place of password. Is there any change in command to use personal token in place of git push as above?

# Bing

The command to push to your repository remains the same, even when using a personal access token. The error message you're seeing might be due to one of the following reasons:

1. **Incorrect token**: Ensure that you've copied the entire token correctly. If you're unsure, you can generate a new token and try again.
2. **Insufficient permissions**: When generating the token, make sure you've given it the necessary permissions. For pushing to a repository, the token should have the repo scope.
3. **Credential helper interference**: If you've previously used a credential helper, it might be automatically providing your old credentials. You can unset the credential helper using the following command:
4. After running this command, try pushing again. When asked for your password, enter your personal access token.
5. **Two-factor authentication (2FA)**: If you have 2FA enabled on your GitHub account, make sure you're entering your personal access token, not your GitHub password, when prompted for a password.

If you're still having trouble, you might want to consider using SSH keys for authentication, which is another secure method and doesn't require entering your credentials each time you push.

Let me know if you need further assistance! 😊

# User

Finally it was success! See the messages below:

$ git push -u origin main
Username for 'https://github.com':
Password for 'https://jdb130496@github.com':
Enumerating objects: 851, done.
Counting objects: 100% (851/851), done.
Delta compression using up to 4 threads
Compressing objects: 100% (769/769), done.
Writing objects: 100% (851/851), 45.72 MiB | 4.15 MiB/s, done.
Total 851 (delta 234), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (234/234), done.
To https://github.com/jdb130496/code-snippets.git

* [new branch] main -> main
  branch 'main' set up to track 'origin/main'.

What next?

# Bing

Congratulations on successfully pushing your local changes to GitHub! 🎉

Now that your local repository is linked to your GitHub repository, here's what you can do next:

1. **Keep your repository updated**: Whenever you make changes to your files locally, remember to commit and push those changes. This will keep your GitHub repository up-to-date with your local changes. Here are the commands you'll need:
2. Replace "Your commit message" with a brief description of the changes you made.
3. **Pull changes from GitHub**: If you make changes directly on GitHub (like editing a file or README), you can pull those changes to your local repository using the command:
4. **Explore more Git features**: Git has many powerful features that can help you manage and track changes to your project. Some of these include branching, merging, and resolving conflicts. You might find these useful as your project grows.

Remember, every time you're asked for a password when pushing to GitHub, use the personal access token you generated.

Let me know if you need help with anything else! 😊

# User

But how can I make changes directly on github? Because I cannot access apps like gcc and others (including excel) from github.

# Bing

When I mentioned making changes directly on GitHub, I was referring to editing text files, markdown files, or code files that are part of your repository. GitHub provides an interface for editing these types of files in your browser.

Here's how you can edit files directly on GitHub:

1. Navigate to your repository on GitHub.
2. Browse to the file you want to edit.
3. Click on the file to view it.
4. Click on the pencil icon in the upper right corner of the file view to edit the file.
5. Make your changes in the editor.
6. Below the editor, type a commit message that describes your changes.
7. Click "Commit changes".

Please note that GitHub is not an integrated development environment (IDE), so you can't run applications or compile code directly on GitHub. For tasks like running a gcc compiler or using Excel, you would need to do that on your local machine, and then push any changes to your GitHub repository.

Let me know if you need further assistance! 😊

# User

I have uploaded files but later found that I want to avoid 2-3 files. How I can change? Can I simply delete them or is there a mechanism?

# Bing

Yes, you can remove files from your repository. Here's how you can do it:

1. **Remove the file locally**:
   Use the git rm command followed by the name of the file you want to remove. For example, if you want to remove a file named example.txt, you would use the following command:
2. Repeat this step for each file you want to remove.
3. **Commit the changes**:
   After removing the files, you need to commit the changes. You can do this with the git commit command. Remember to include a message that describes what you did. For example:
4. **Push the changes to GitHub**:
   Finally, you need to push your changes to GitHub. You can do this with the git push command:

This will remove the files from your GitHub repository. Please note that this will permanently remove these files from the history of your project. If you think you might need these files in the future, consider moving them to a different location instead of deleting them.

Let me know if you need further assistance! 😊

# User

Is there any way to keep the files in local folder but tell github to avoid them when syncing via git push or git pull?

# Bing

Yes, you can tell Git to ignore certain files in your local directory by using a .gitignore file. A .gitignore file is a text file that tells Git which files or folders to ignore in a project.

Here's how you can create a .gitignore file:

1. In your project's root directory, create a new file named .gitignore.
2. Open the .gitignore file in a text editor.
3. Each new line should contain a pattern that matches the file or folder that you want Git to ignore. For example, if you want Git to ignore a file named example.txt, you would add the following line to .gitignore:
4. If you want Git to ignore all files of a certain type, you can use a wildcard (\*). For example, to ignore all .log files, you would add the following line to .gitignore:
5. If you want Git to ignore a whole directory, you can include the directory name. For example, to ignore a directory named test/, you would add the following line to .gitignore:
6. Save and close the .gitignore file.

Now, Git will ignore the files or folders that match the patterns in your .gitignore file. These files won't show up when you run git status, and they won't be included when you run git add ..

Remember to commit and push the .gitignore file to your GitHub repository. This ensures that the ignore rules are shared with any other people working on the project.

Let me know if you need further assistance! 😊

# User

I want to ignore folder "D:\OneDrive - 0yt2k\powershell scripts" Tell me how to include in .gitignore?

# Bing

To ignore the entire "powershell scripts" folder, you can add its path relative to the repository root in the .gitignore file.

If the "powershell scripts" folder is in the root of your repository, you would add the following line to your .gitignore file:

powershell scripts/

However, if the "powershell scripts" folder is inside another folder, you need to include the path to the folder. For example, if it's inside a folder named "scripts" in the root of your repository, you would add:

scripts/powershell scripts/

Remember, the path should be relative to the root of your Git repository, not the absolute path on your file system. Also, note that spaces in directory or file names are perfectly valid for a .gitignore file.

After you've made the changes, don't forget to commit and push the .gitignore file to your repository. This ensures that the ignore rules are shared with any other people working on the project.

Let me know if you need further assistance! 😊

# User

After I add above directory what are the next steps to do both delete allready uploaded folder on github and keep the folder in locally?

# Bing

To remove the already uploaded folder from GitHub while keeping it in your local directory, you can follow these steps:

1. **Add the directory to your `.gitignore` file**:
   As discussed earlier, add the following line to your .gitignore file:
2. This tells Git to ignore the "powershell scripts" directory in future commits.
3. **Remove the directory from the repository**:
   Use the git rm command followed by --cached to remove the directory from the repository without deleting it from your local file system. For example:
4. The -r option tells Git to remove the directory recursively.
5. **Commit the changes**:
   Commit the changes to your repository with a message indicating that you've removed the directory. For example:
6. **Push the changes to GitHub**:
   Finally, push your changes to GitHub:

Now, the "powershell scripts" directory will be removed from your GitHub repository but will still exist in your local directory. Future changes to this directory will be ignored by Git.

Let me know if you need further assistance! 😊

# User

I noticed that whenever I delete any file or folder on local drive, then commit and push, girhub still shows that deleted file when I click on "difference" after clicking commits link. How to delete the files permanently from repository so that they don't show up the contents even in difference link.

# Bing

When you delete a file and commit that change, the file is removed from the latest version of your repository, but it still exists in the history. This is a fundamental aspect of Git: it's a version control system, so it keeps a history of all changes. When you view the difference between commits, you're seeing this history.

If you want to completely remove a file from your Git repository, including from all historical commits, you'll need to use a command called git filter-branch or the [BFG Repo-Cleaner](https://rtyley.github.io/bfg-repo-cleaner/), a simpler, faster alternative. However, be aware that these are advanced operations and can permanently delete data:

Here's an example of how you can use git filter-branch to permanently remove a file:

git filter-branch --force --index-filter \

"git rm --cached --ignore-unmatch PATH-TO-YOUR-FILE" \

--prune-empty --tag-name-filter cat -- --all

Replace PATH-TO-YOUR-FILE with the path to the file you want to remove from your repository.

After running this command, all references to the file will be removed from your Git history. The file will no longer appear in any commits, and when you push these changes to GitHub, the file will be completely removed from your repository.

**Please be careful** when using this command, as it permanently removes data. Make sure you have a backup of your repository and the files you're removing, just in case.

Let me know if you need further assistance! 😊

# User

What if I want to delete all commits which included many deleted files and folders. My repository is: https://github.com/jdb130496/code-snippets

# Bing

If you want to delete all commits from your repository, you can do so by creating a new initial commit. This will effectively erase your commit history. Here's how you can do it:

1. Check out to a new branch:
2. Add all the files to the new branch:
3. Commit the changes:
4. Delete the old branch:
5. Rename the new branch to main:
6. Finally, force push to GitHub:

Please be aware that this will erase your commit history and cannot be undone. Make sure you have a backup of your repository before proceeding.

Also, note that this will not remove the files from the GitHub repository if they are present in the latest commit. If you want to remove files or folders, you should delete them before committing on the new branch.

Let me know if you need further assistance! 😊

# User

Can I not define newbranch name as main at the first step instead of renaming later?

# Bing

Yes, you can directly use main as the branch name in the first step. However, since a branch named main already exists in your repository, Git will not allow you to create another branch with the same name. That's why we first create a new branch with a different name, delete the old main branch, and then rename the new branch to main.

If you try to create a new branch named main while it already exists, you'll get an error message saying "fatal: A branch named 'main' already exists."

So, the process of creating a new branch, deleting the old one, and renaming the new one is necessary to overwrite the main branch while avoiding naming conflicts. Let me know if you need further assistance! 😊