

## How to find and utilize Github website resources

1. github trending  
<https://github.com/trending>
2. Following are tips for searching in below github explorer:  
<https://github.com/search>

- 2.1 find wiki about a topic  
"awesome XXX" in code search box
- 2.2 find an example of a topic  
"XXX example" in code search box
- 2.3 find skeleton of a project for own use  
"XXX starter"/"XXX boilerplate" in code search box
- 2.4 find tutorial  
"XXX tutorial" in code search box

---

## Github Basics in working environment

1. "Gitize" a project if downloaded without git clone (no hidden .git folder):  
install git -> right click the folder -> "Git Bash Here" -> (in the git terminal) \$ git init (a .git hidden folder will be created)
2. integrated "Git Bash terminal" in VS Code:  
install git in default location (C drive) -> ctrl+shift+p open command palette in VS Code -> type Terminal: Select Default Profile -> choose Bash
3. open a .git folder in VS Code & view files' status:  
The most left "Files" icon will list changed files as "green" color  
  
The most left "branch" icon will show the changing info of the files -> All changes are gathered under the "Changes" divider
- [4. "git add (<fileNames>) -A" will add the (selected) changed files in the cache area.
5. "git commit -m "<commit message for human to read>" " will commit the added files in the branch of the repository (if successful, green will disappear) ]
6. GUI git commit: steps 4 & 5 can be replaced by operations on the left explorer of VS Code under "branch" icon (GUI method).  
you can choose which files to "git add" in the explorer, and "git commit" the added files with above text box. Type human readable message in the above box and hit "Ctrl + Enter" to commit. (Note that commit is only a local action)
- NOTE: "git commit" does NOT change the online Repo yet. You need to finally "git push" to ensure synchronization of all changes.
- [7. find a historical commit hash value: "git log --stat"]
8. GUI git features: install a VS Code extension "Gitlens"(important) -> under "branch" icon -> "commits" divider -> find the commit subdivider -> now you can look at the commits tree
- [9. "git checkout": if not commit yet, this command can help go back to the prior-change status.
10. "git HEAD^1": if commit, this command can help cancel the commit. ]

11. Gui "git checkout": click the relevant button under "branch" explorer.

Gui "git HEAD^": right click on the commit under "COMMITTS" divider (Gitlens needed), and choose "Undo Commit".

12. If long time later, we want to find back the historical codes (GUI method):

under "branch" icon "FILE HISTORY" -> click the file name -> copy the codes from the editor window and paste in another new file for editing -> commit the new file (old codes) to avoid complicated issues.

13. Branch creation & switch "git checkout -b <branchName>"

Branch swith "git checkout <branchName>"

14. Branch merging "git checkout main" -> "git merge <branchName(non-main)>"

15. merging conflicts: choose from "Accept Current Change"/"Accept Incoming Change"/"Accept Both Change"

we need a new process of commit for the conflict

\*\*if unsure about the conflicts, abort this merging by "git merge --abort"

16. List all branches "git branch". After merging, we usually need to delete unnecessary branch(es) "git branch -D <branchName>"

17. There are also GUI for branches operations.

18. "git push -u origin main": Cooperation across different computers need "Github". Create an empty Github repository first if there isn't. Follow the webpage instruction and paste the command to terminal for "git remote add origin https://github.com/rui-min/<RepoName>.git";

Next we switch to local main branch "git branch -M main"; Finally we can use "git push -u origin main" to the online repo (login in needed).

(we can use Github Desktop)

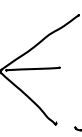
19. "git pull": we need to pull from online Repo to update colleague's changes (VS Code will warn you to pull before push).



**Tip1: do commit TWICE before any merge to have a "revert to latest node"**

**Tip1': if tip1 is not followed, best do merging with a cloned branch, instead of working branch**

## Markdown Syntax

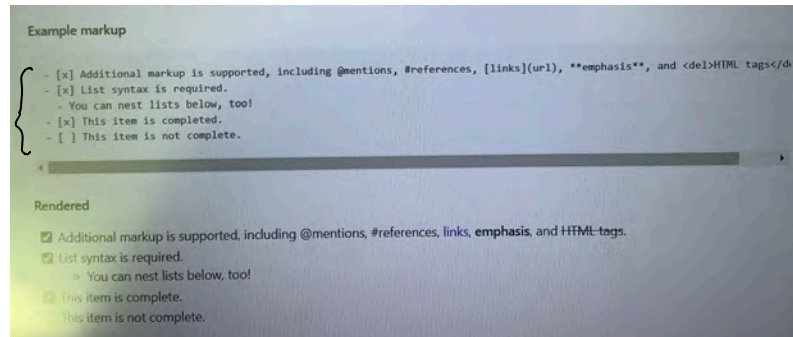
Can be used in  "issues", "pull requests".  
Files with ".md", ".markdown"  
Sharing snippets of text in "Gists"

Steps &

Syntax: ① Task List

Issue → Comment

② Turn on github pages



③ Add headers: #    ##    #####    ← file: \_includes/01-name.md  
                  ↑            ↑            ↑  
                  largest <h1>    <h2>    <h6> smallest

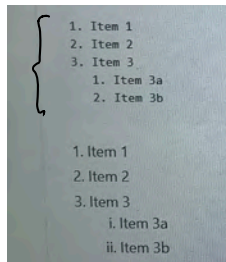
④ Add an image: e.g.: ![Image of Yaktocat](http://octodex.github.com/images/yaktocat.png)  
                  ↑  
                  file: \_includes/02-image.md

⑤ Add a portfolio link: e.g.: [Github](http://github.com)    ← file: \_includes/03-links.md

⑥ Emoji: Syntax → :XXX:    e.g.: :heart:    :+1:    :smile:  
                  cheat sheet → <https://gist.github.com/rxaviers/7360908>

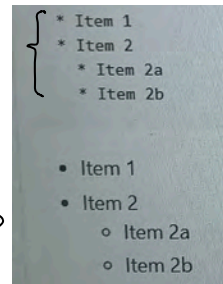
⑦ Ordered Lists Syntax →  
                  ↑  
                  file: \_includes/04-lists.md

Output →

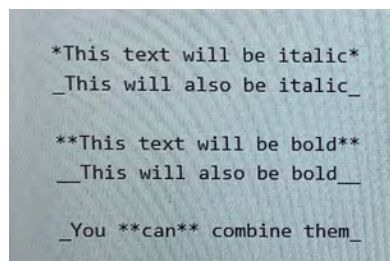


⑧ Unordered Lists: Syntax →

Output →



⑨ Emphasis - bold & italic  
                  ↑            ↑            ↑  
                  \*\*    \*    \*    \*    \*  
                  - - - - -  
                  file: \_includes/05-emphasis.md



⑩ @mention

Syntax → @username

## ⑪ Cross Links

Additionally, references to issues and pull requests are automatically converted to shortened links to the issue or pull request. For example,

Reference type	Raw reference	Short link
Issue or pull request URL	<code>https://github.com/desktop/desktop/pull/3602</code>	<code>#3602</code>
# and issue or pull request number	<code>#3602</code>	<code>#3602</code>
GH- and issue or pull request number	<code>GH-3602</code>	<code>GH-3602</code>
Username/Repository# and issue or pull request number	<code>desktop/desktop#3602</code>	<code>desktop/desktop#3602</code>

For more information, see "Autolinked references and URLs" in the *GitHub Help*.

## Linking Specific Commits

References to a commit's ID (commonly called a SHA or hash) are automatically converted into shortened links to the commit on GitHub. For example,

Reference type	Raw reference	Short link
Commit URL	<code>desktop/desktop@8304e9c</code>	<code>8304e9c</code>
SHA	<code>8304e9c271a5e5ab4fda797304cd7bcca7158c87</code>	<code>8304e9c</code>
User@SHA	<code>desktop@8304e9c271a5e5ab4fda797304cd7bcca7158c87</code>	<code>desktop@8304e9c</code>
Username/Repository@SHA	<code>User/Repository@SHA: desktop/desktop@8304e9c</code>	<code>desktop/desktop@8304e9c</code>

## ⑫ Quotes

Syntax: `> XXX`

e.g.: `> Pardon my French`

## ⑬ Tables

(hyphens & pipes)

Syntax →

```

---
First Header | Second Header
---|-----
Content from cell 1 | Content from cell 2
Content in the first column | Content in the second column
...

```

Output →

First Header	Second Header
Content from cell 1	Content from cell 2
Content in the first column	Content in the second column

## ⑭ Inline Code Blocks

### Inline Code Blocks

Certain words and phrases need to be formatted in monospace fonts, especially when writing about code. As you've seen throughout this lab, words can be distinguished in markdown with inline code blocks.

Inline code is just one `"` character on either side of the text, and can be used within paragraphs, headers, or other Markdown.

`"inline code is just one backtick"`

inline code is just one backtick

## ⑮ Separate Code Blocks: Syntax →

```

"""
xxx...
"""

```

## ⑯ Summary dropdown:

Syntax → `<details>`  
`<summary> Title </summary>`  
 Content here  
`</details>`