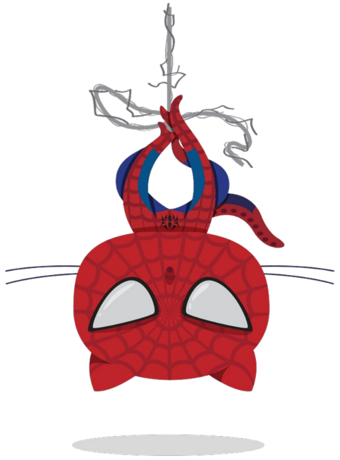
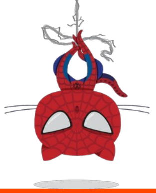
# Version Control
# Day 2



## Day 2 Contents



- Git SSH Key.
- Branching.
- Tagging & Stashing.
- History and Configuration.
- Ignoring files.
- READ.md file.
- Rebasing.
- Pull request.

## Git SSH Key

- **Why use an SSH Key?**

**When working with a GitHub repository, you'll often need to identify yourself to GitHub using your username and password.**

**An SSH key is an alternate way to identify yourself that doesn't require you to enter you username and password every time.**

# Git SSH Key

- SSH keys come in pairs, a public key that gets shared with services like GitHub, and a private key that is stored only on your computer. If the keys match, you're granted access.

- The cryptography behind SSH keys ensures that no one can reverse engineer your private key from the public one.

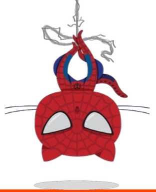
## Git SSH Key

- Generating an SSH key pair.

```
ssh-keygen -t rsa -b 4096 -c “fatma10khaled10@gmail.com”
cat /home/fatma/.ssh/id_rsa.pub
```

copy the public key to github

## Branching Out



- If you need to work on a feature that will take some time, it’s preferred to make a branch

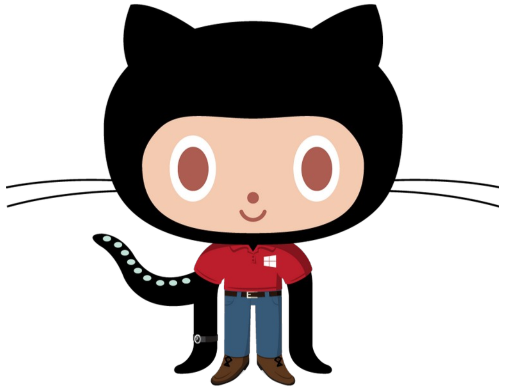
```
git branch newbranch
```

- To list all branches:

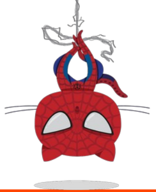
```
git branch
* master
newbranch
```

- To switch to a branch:

```
git checkout newbranch
```



# Creating A Remote Branch



- When you need other people to work on your branch.

  OR

- Any branch that will last more than a day
- Then you have to make you branch availableremotely:

  git checkout -b newbranch

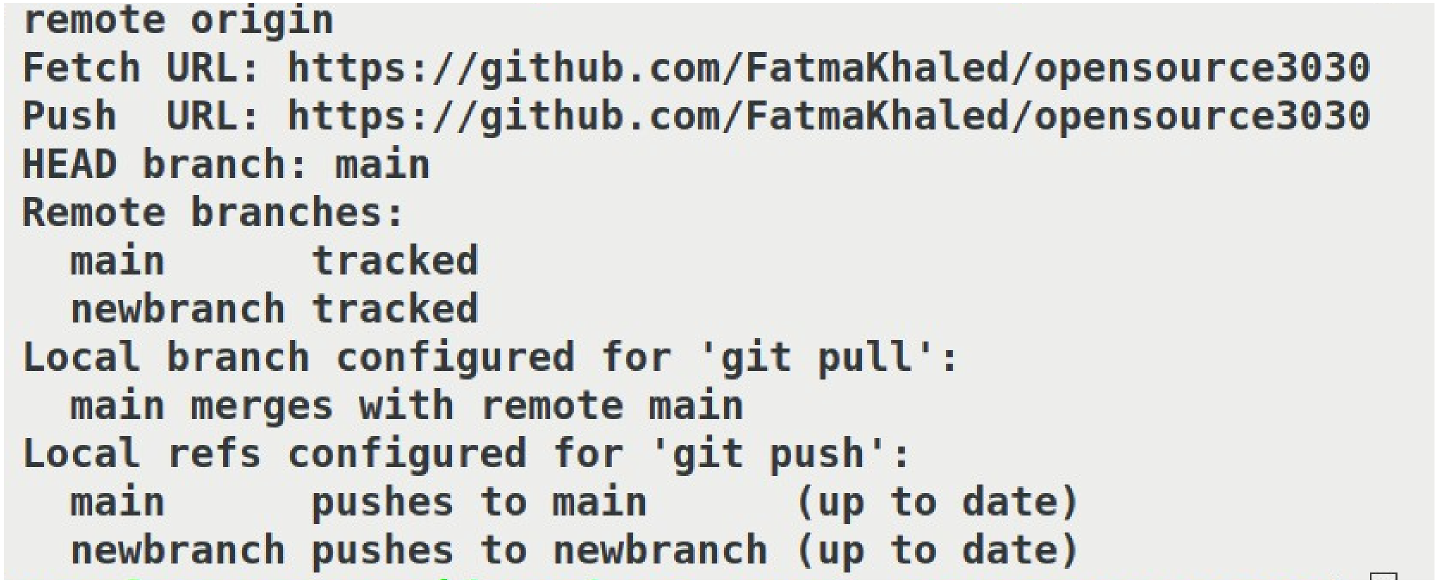
  git push origin newbranch

- To list the remote branches

  git branch -r

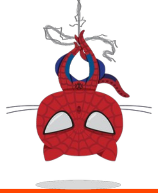
## Remote Show

- **git remote show origin**



```
remote origin
Fetch URL: https://github.com/FatmaKhaled/opensource3030
Push  URL: https://github.com/FatmaKhaled/opensource3030
HEAD branch: main
Remote branches:
  main      tracked
  newbranch tracked
Local branch configured for 'git pull':
  main merges with remote main
Local refs configured for 'git push':
  main      pushes to main      (up to date)
  newbranch pushes to newbranch (up to date)
```

# Removing A Remote Branch



- To delete a remote branch

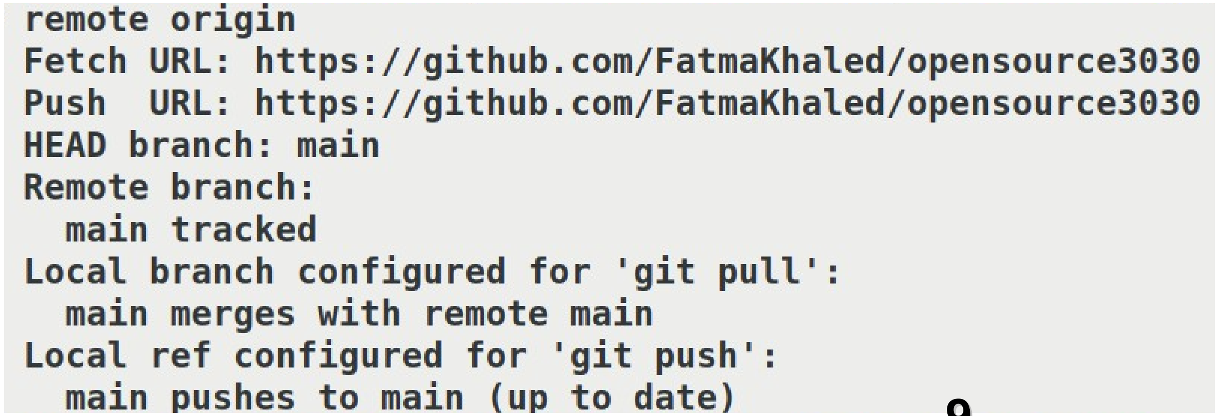
  git push origin :newbranch

- To delete a local branch

  git branch -d newbranch

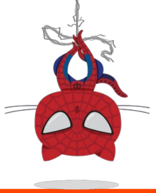
  git branch -D newbranch

- git remote show origin

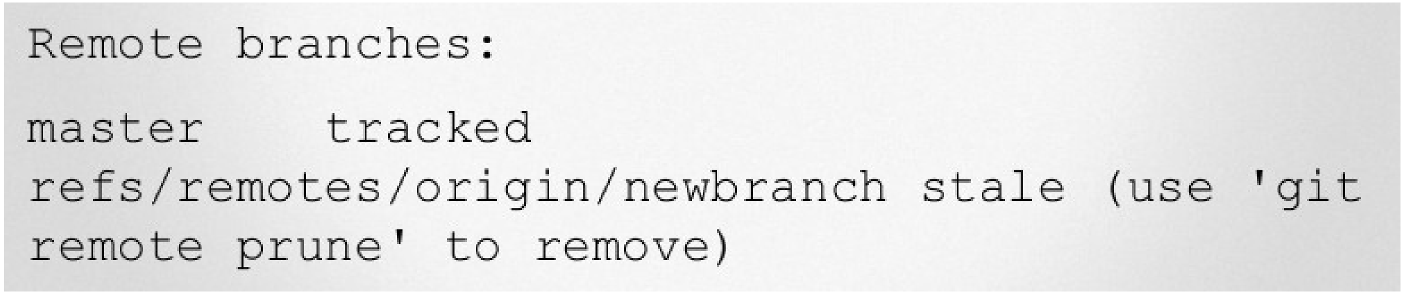


```
remote origin
Fetch URL: https://github.com/FatmaKhaled/opensource3030
Push  URL: https://github.com/FatmaKhaled/opensource3030
HEAD branch: main
Remote branch:
  main tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (up to date)
```

## Removing A Remote Branch



- git remote show origin



```
Remote branches:

master      tracked
refs/remotes/origin/newbranch stale (use 'git
remote prune' to remove)
```

- git remote prune origin

## Merging Branches

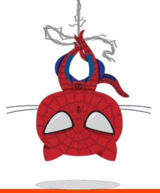
- After finishing your work on the branch you have to merge it with master branch (assuming no changes were made to the master)
- git checkout master
- git merge newbranch

## Merging Branches

- If there were changes made to the master the merge will not run in fast forward mode instead it will run in reclusive mode and the editor will pop up to enter the merge message
- git checkout master
- git merge newbranch

  Merge made by the 'recursive' strategy

# Create / Delete Branch



- When you’re done with a branch, you can safely remove it

```
git branch -d newbranch
```

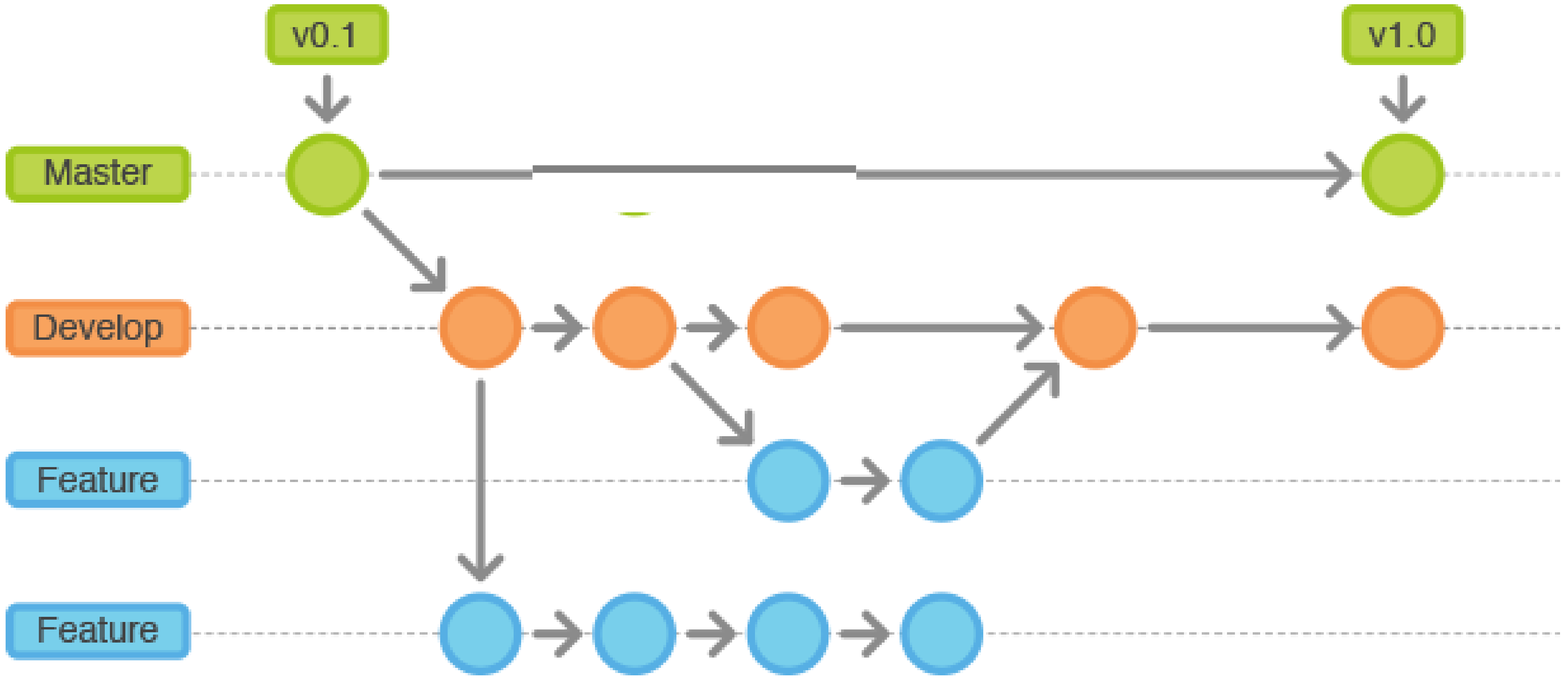
- If there were files that are not merged

```
git branch -D newbrnach
```

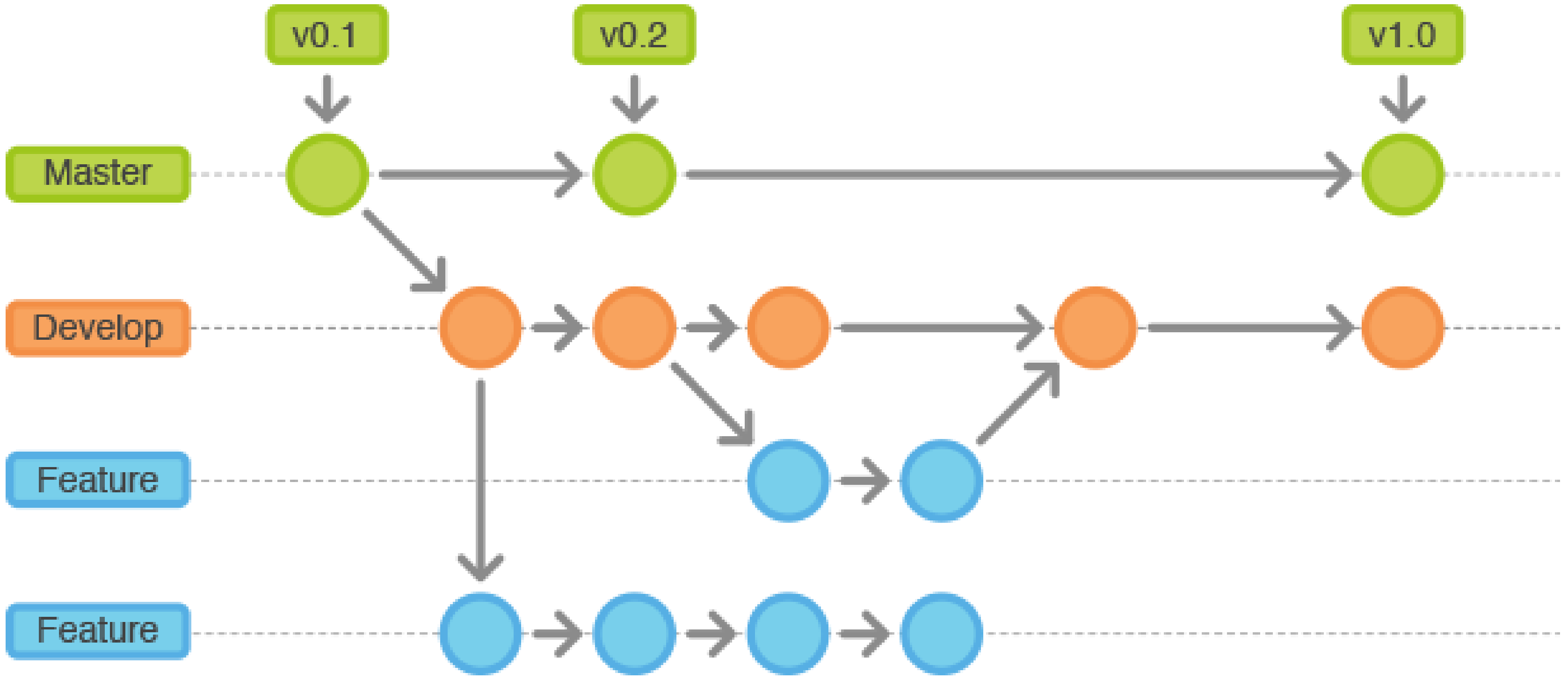
- To create a branch and checkout it in one step

```
git checkout -b newbranch
```

## Branching



# Branching



## Tagging

- A tag is a reference to a commit (used mostly for release versioning)
- There are two types of tagging:
  - A lightweight tag is very much like a branch that doesn't change – it's just a pointer to a specific commit.
  - Annotated tags, however, are stored as full objects in the Git database. They're checksummed; contain the tagger name, email, and date; have a tagging message .

## Tagging

- To create a lightweight tag (This is basically the commit checksum stored in a file – no other information is kept).

```
git tag v1.4-lw
```

```
git show v1.4-lw
```

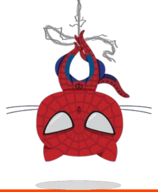
- To switch to a tag

```
git checkout v1.4-lw
```

- To list existing tags

```
git tag
```

## Tagging



- To add a annotated tag

```
git tag -a v0.0.3 -m "version 0.0.3"
git show v0.0.3
```

- To push tags

```
git push origin <tagname>
git push --tags
git push origin --follow-tags
```

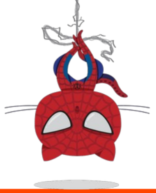
## Tagging Later

- You can also tag commits after you've moved past them.

```
git log

git tag  -a  v1.2  9fceb02 -m "message"
```

## Deleting Tags

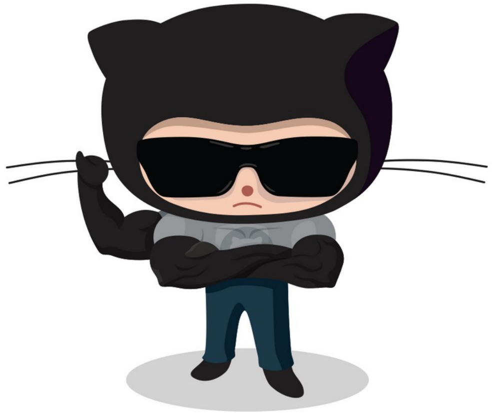


- To delete a remote tag

```
git  push origin  --delete v1.4
git  push origin :refs/tags/v1.4
```

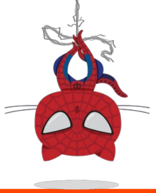
- To delete a local tag

```
git tag -d v1.4
```



## Stashing

- git-stash - Stash the changes in a dirty working directory away.
- git stash
- git stash save “message”
- git stash list
- git stash show
- git stash pop
- git stash apply
- git stash drop
- git stash clear



## History & Configuration

- To show the commits history

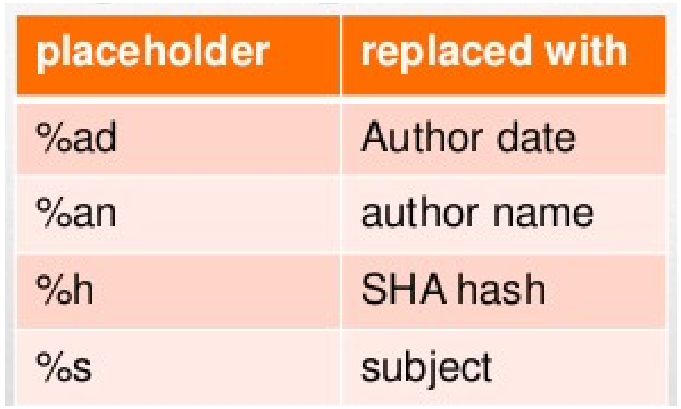
  git log

- To show the history in one line

  git log --pretty=oneline

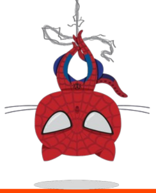
- To format the history log

  git log --pretty=format:"%h %ad- %s [%an]”



| placeholder | replaced with |
|---|---|
| %ad | Author date |
| %an | author name |
| %h | SHA hash |
| %s | subject |

## Dates Ranges



- git log --until=1.minute.ago
- git log --since=1.day.ago
- git log --since=1.hour.ago
- git log --since=1.month.ago --until=2.weeks.ago

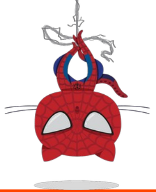
## Ignoring Files

- Often, you’ll have a class of files that you don’t want Git to automatically add or even show you as being untracked.

- In such cases, you can create a file listing patterns to match them named .gitignore.

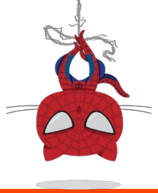
```
cat .gitignore
cache/
logs/*.log
```

# README.md

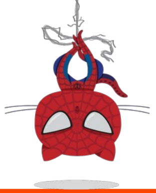


- A README is a text file that introduces and explains a project. It contains information that is commonly required to understand what the project is about.
- Markdown is a simple way to format text that looks great on any device.
- It doesn’t do anything fancy like change the font size, color, or type — just the essentials, using keyboard symbols you already know.
- https://www.makeareadme.com/
- https://commonmark.org/help/

## Issues



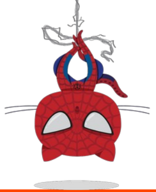
- https://docs.github.com/en/free-pro-team@latest/github/managing-your-work-on-github/creating-an-issue



## Lab 2

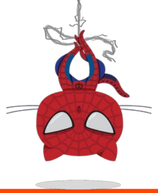
- Create a new project on your local machine then push it to your remote repo.
- create two branches (dev and test) then create two files in dev branch and push this changes to remote repo.
- merge this changes on main branch and then push it to your remote main branch.
- Tell me how to remove them locally and remotly.
- Send me invitation on my mail.

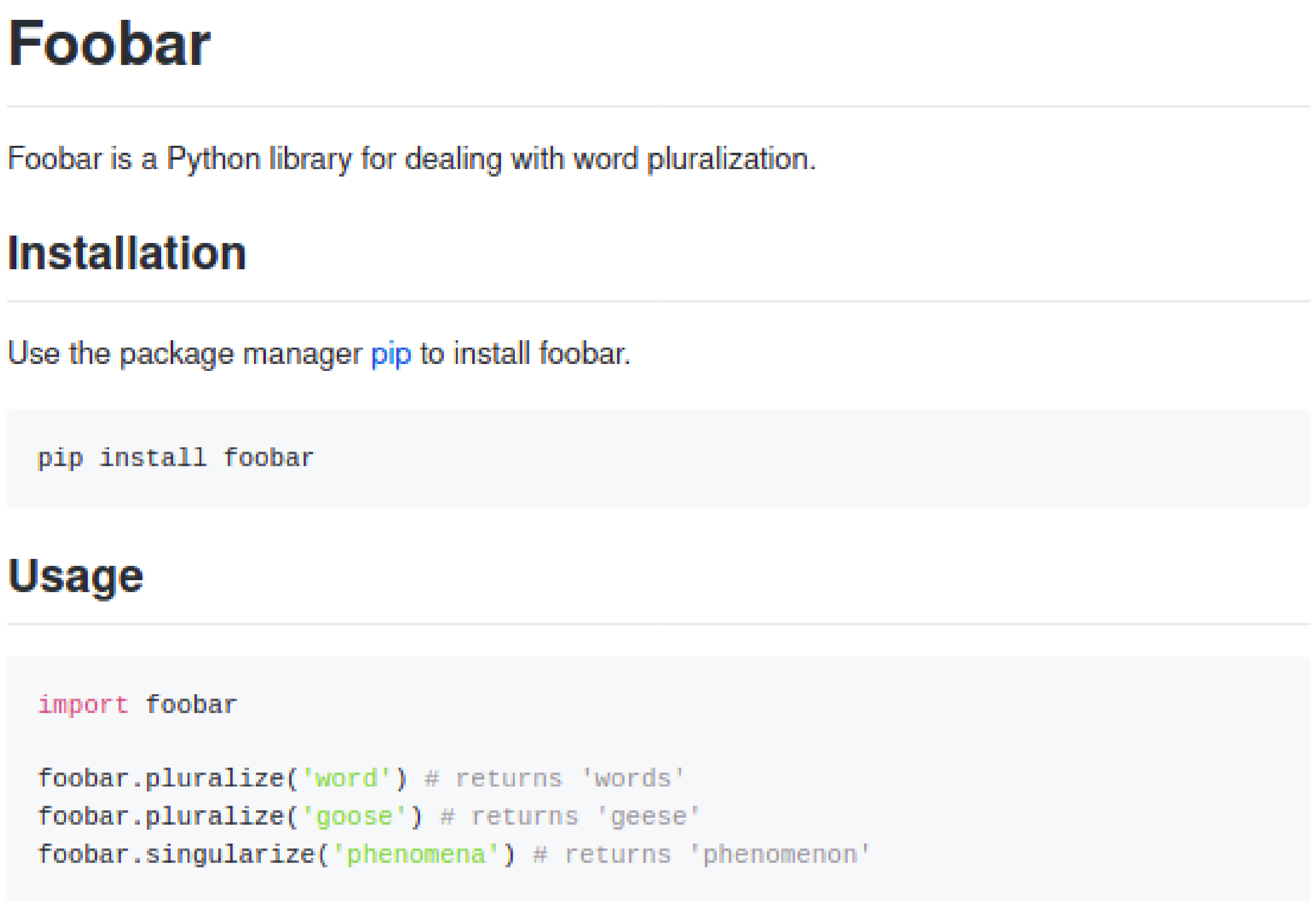
## Lab 2



- create an annoted tag with tagname v1.4.
- push it to remote server.
- tell me how to list tags locally.
- tell me how to delete tag locally and remotely.
- What is git rebase? Give me an example.
- pull request .
- README.md file.   Add image at the end of README.md file

# Lab 2





## Foobar

Foobar is a Python library for dealing with word pluralization.

### Installation

Use the package manager pip to install foobar.

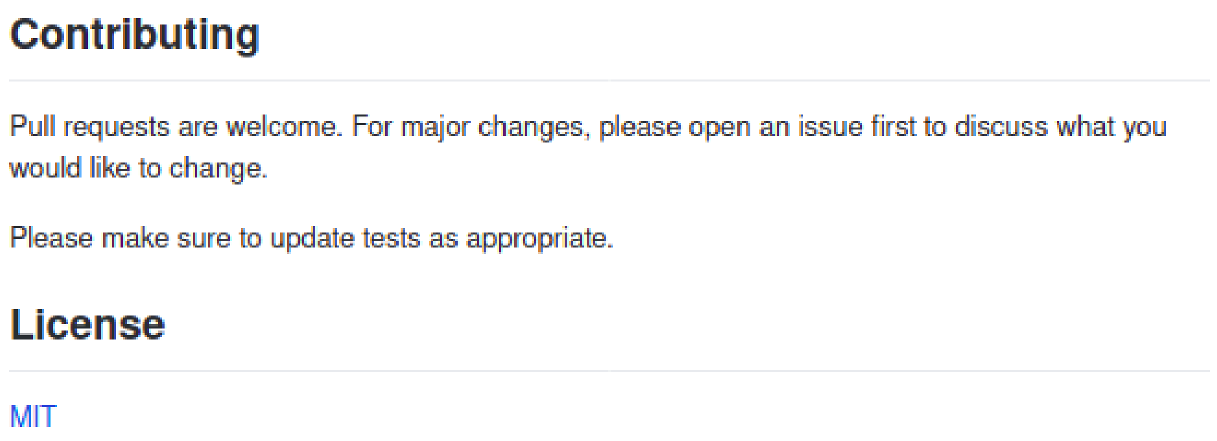
```
pip install foobar
```

### Usage

```python
import foobar

foobar.pluralize('word') # returns 'words'
foobar.pluralize('goose') # returns 'geese'
foobar.singularize('phenomena') # returns 'phenomenon'
```

## Lab 2



### Contributing

Pull requests are welcome. For major changes, please open an issue first to discuss what you would like to change.

Please make sure to update tests as appropriate.

### License

MIT