UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Corso di
FONDAMENTI DI PROGRAMMAZIONE

Package Manager + Virtual Environment

Prof. Marco Mamei

PIP

- Python Package Index (PyPI) è un repository che contiene decine di migliaia di package scritti in Python.
  - GUI, Videogame, Applicazioni Web, Calcolo Scientifico, AI,….

- È possibile accedere ai package del Python Package Index tramite un tool chiamato **pip** (anche integrato in pycharm)

- **pip** è un tool che ci permette di cercare, scaricare ed installare package Python che si trovano sul Python Package Index. pip ci consente inoltre di gestire i package che abbiamo già scaricato, permettendonci di aggiornarli o rimuoverli.

PIP

```
C:\Users\Marco>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pygame
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'pygame'
>>> exit()

C:\Users\Marco>pip install pygame
Collecting pygame
  Using cached https://files.pythonhosted.org/packages/80/2c/3a52e7e9c097229b026b4efbe6711c600f3a
Installing collected packages: pygame
Successfully installed pygame-1.9.6

C:\Users\Marco>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import pygame
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
>>>
```

pygame not availabe

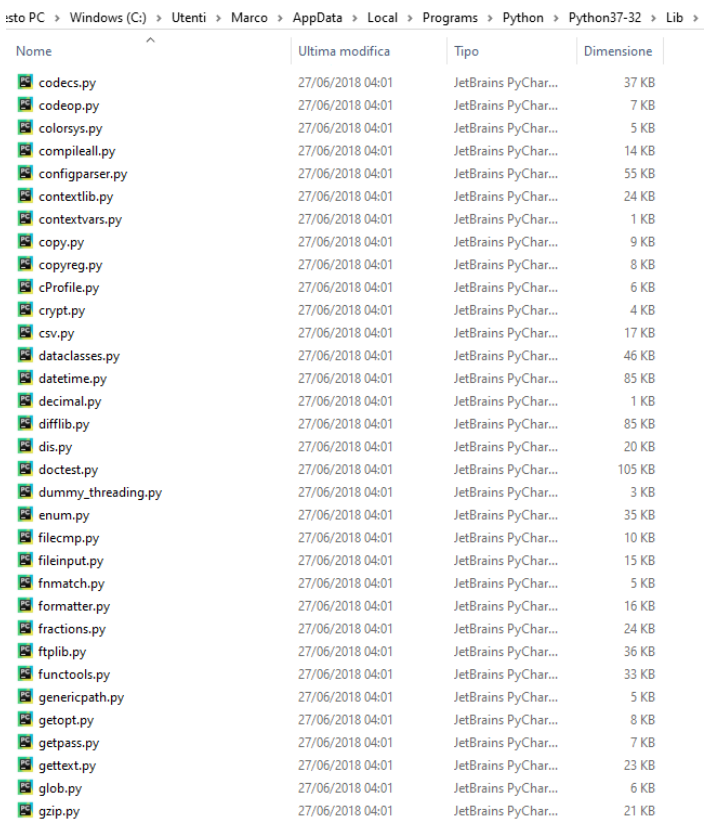
pip install pygame

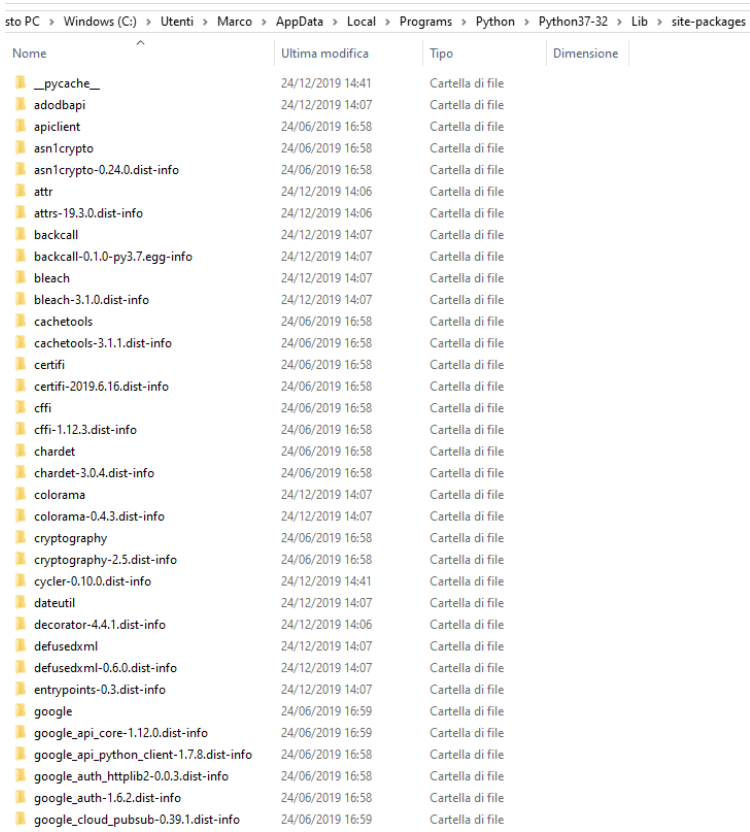
pygame now available

Virtual Environment

- Where does Python installs modules?

```
C:\Users\Marco>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.prefix
'C:\\Users\\Marco\\AppData\\Local\\Programs\\Python\\Python37-32'
>>> import site
>>> site.getsitepackages()
['C:\\Users\\Marco\\AppData\\Local\\Programs\\Python\\Python37-32', 'C:\\Users\\Marco\\AppData\\Local\\Programs\\Python\\Python37-32\\lib\\site-packages']
>>>
```





Virtual Environment

- So, why do all of these little details matter?

- By default, every project on your system will use these same directories to store and retrieve site packages (third party libraries). This is not a big problem for system packages (packages that are part of the standard Python library), but it does matter for site packages.

- Consider the following scenario where you have two projects: ProjectA and ProjectB, both of which have a dependency on the same library, ProjectC. The problem becomes apparent when we start requiring different versions of ProjectC. Maybe ProjectA needs v1.0.0, while ProjectB requires the newer v2.0.0.

- This is a real problem for Python since it can't differentiate between versions in the site-packages directory. So both v1.0.0 and v2.0.0 would reside in the same directory with the same name

What Is a Virtual Environment?

- The main purpose of Python virtual environments is to create an isolated environment for Python projects. This means that each project can have its own dependencies, regardless of what dependencies every other project has.

```
C:\Users\Marco>python -m venv project1

C:\Users\Marco>project1\Scripts\activate

(project1) C:\Users\Marco>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import sys
>>> sys.prefix
'C:\\Users\\Marco\\project1'
>>> import site
>>> site.getsitepackages()
['C:\\Users\\Marco\\project1', 'C:\\Users\\Marco\\project1\\lib\\site-packages']
```

What Is a Virtual Environment?

```
(project1) C:\Users\Marco>pip install guizero
Collecting guizero
  Using cached https://files.pythonhosted.org/packages/b0/eb/c58693afb94bc1e5f5f77d0f8e6b4e6dc84
Installing collected packages: guizero
Successfully installed guizero-1.1.0
You are using pip version 10.0.1, however version 19.3.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(project1) C:\Users\Marco>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import guizero
>>> exit()

(project1) C:\Users\Marco>project1\Scripts\deactivate
C:\Users\Marco>python -m venv project2

C:\Users\Marco>project2\Scripts\activate

(project2) C:\Users\Marco>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import guizero
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'guizero'
>>> exit()

(project2) C:\Users\Marco>project2\Scripts\deactivate
C:\Users\Marco>
```

Install new package in project1

package is visible in project1

exit from project1 - deactivate

create project2

package is NOT visible in project2
It is only installed in project1

UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Corso di
FONDAMENTI DI PROGRAMMAZIONE

Git e Github

Prof. Marco Mamei

Git

Git is a popular version control system.

It is used for:

- **Tracking code changes**
- **Tracking who made changes**
- **Coding collaboration**

It allows to

- Manage projects with **Repositories**
- **Clone** a project to work on a local copy
- Control and track changes with **Staging** and **Committing**
- **Branch** and **Merge** to allow for work on different parts and versions of a project
- **Pull** the latest version of the project to a local copy
- **Push** local updates to the main project

Working with Git

- Initialize Git on a folder, making it a **Repository**
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered **modified**
- You select the modified files you want to **Stage**
- The **Staged** files are **Committed**, which prompts Git to store a **permanent** snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.

Git Commands

- Installing Git: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git
- Creating Git Folder

```
git --version
mkdir test
cd test
git init
```

- Create new files and check status

```
git status

On branch master
No commits yet

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

        file1.txt
nothing added to commit but untracked files present (use "git add" to track)
```

- Files in your Git repository can be:
  - **Tracked** - files that Git knows about and are added to the repository
  - **Untracked** - files that are in your working directory, but not added to the repository
- To get Git to track them, you need to **stage** them, or add them to the **staging environment**.

Git Commands

- Add to staging

```
git add file1.txt
git add --all
git add .
git status --short
```

.gitignore
I can list in this file all the files and directories that I do not want to enter the staging area (e.g., config and password files)

git status --short
?? - Untracked files
A - Files added to stage
M - Modified files
D - Deleted files

- Commit

```
git commit -m "First Commit"
```

- Once files change

```
git add .
git commit -m "Second Commit"
or
git commit -a -m "Second Commit"
```

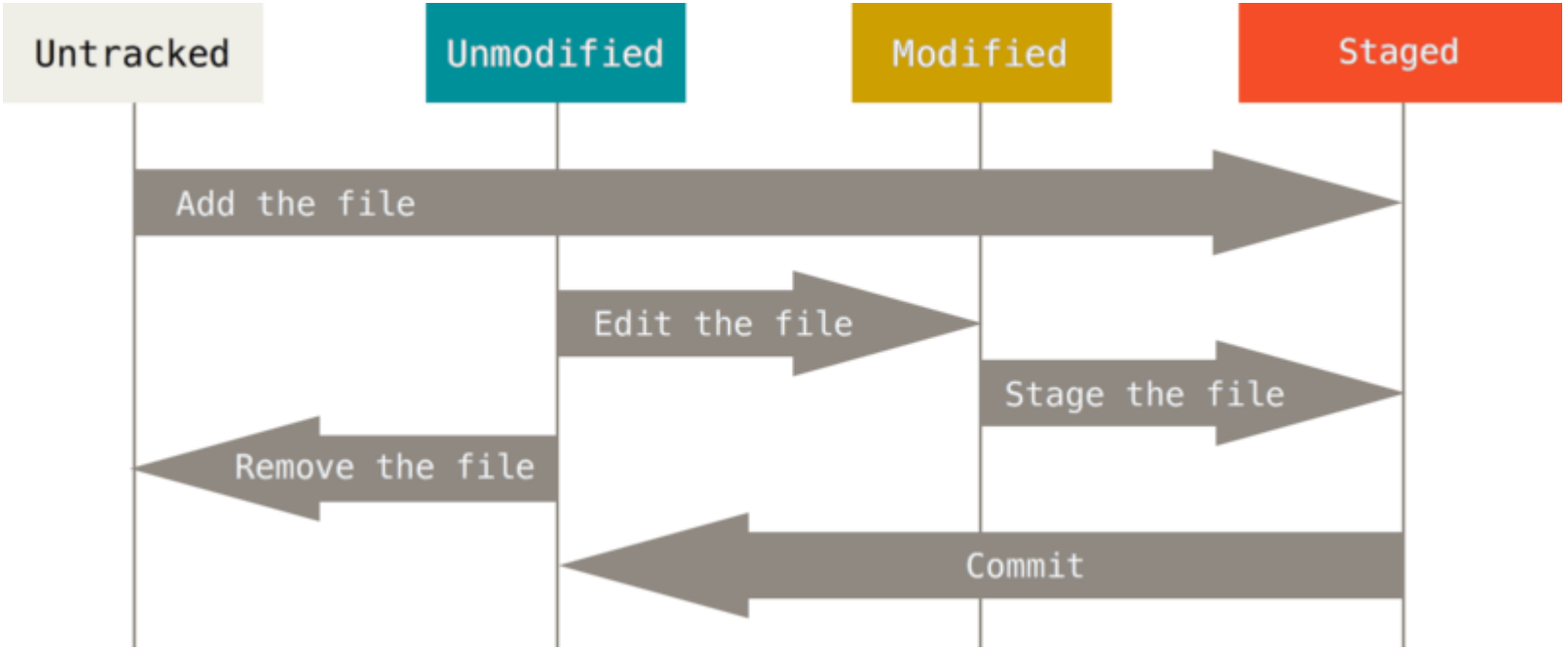
The staging area contains all the changes to be committed. To commit something, you first have to add it to staging

- See commits' log

```
git log
git log --oneline
```

Git Commands

- The lifecycle of the status of your files



Git Commands

- Revert (undo) last commit

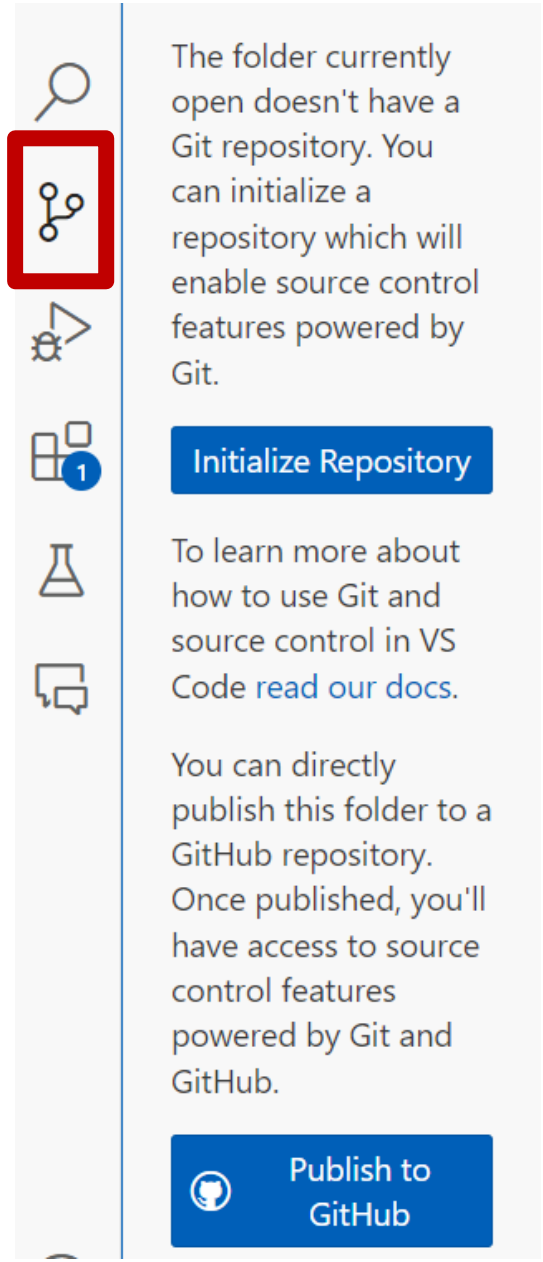
```
git revert HEAD --no-edit
git revert HEAD~x --no-edit
```

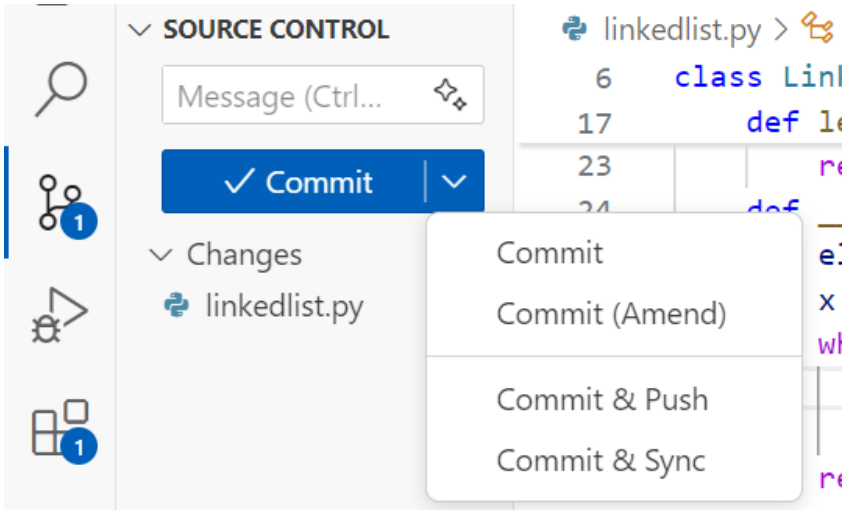
If you have to revert more than one commit, you have to revert all the commits until that point

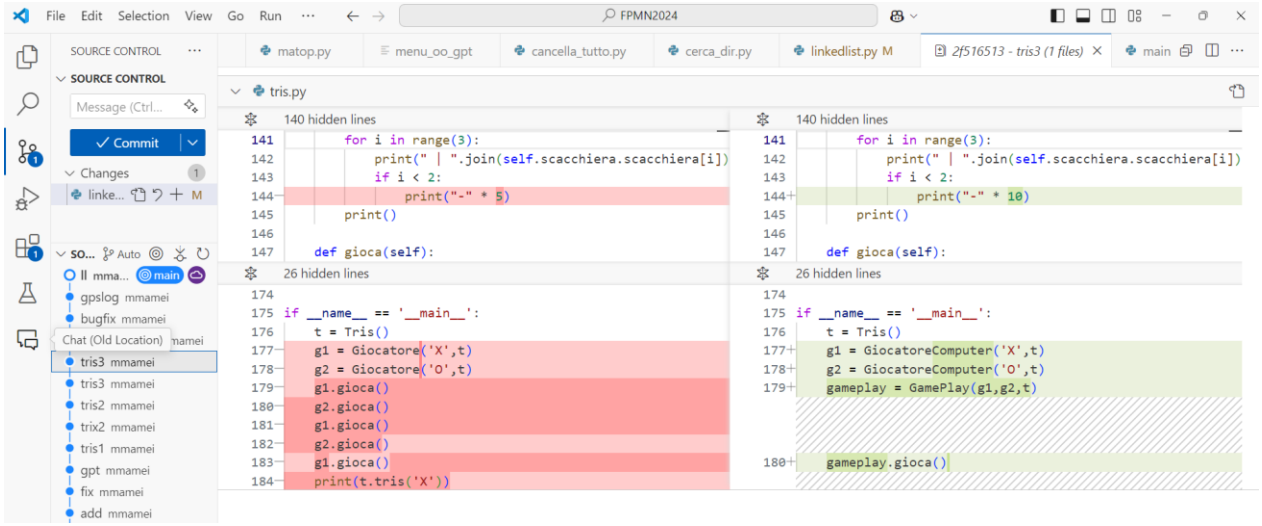
- Delete last commits (change history)

```
git reset --hard <HASH_OLD>
```

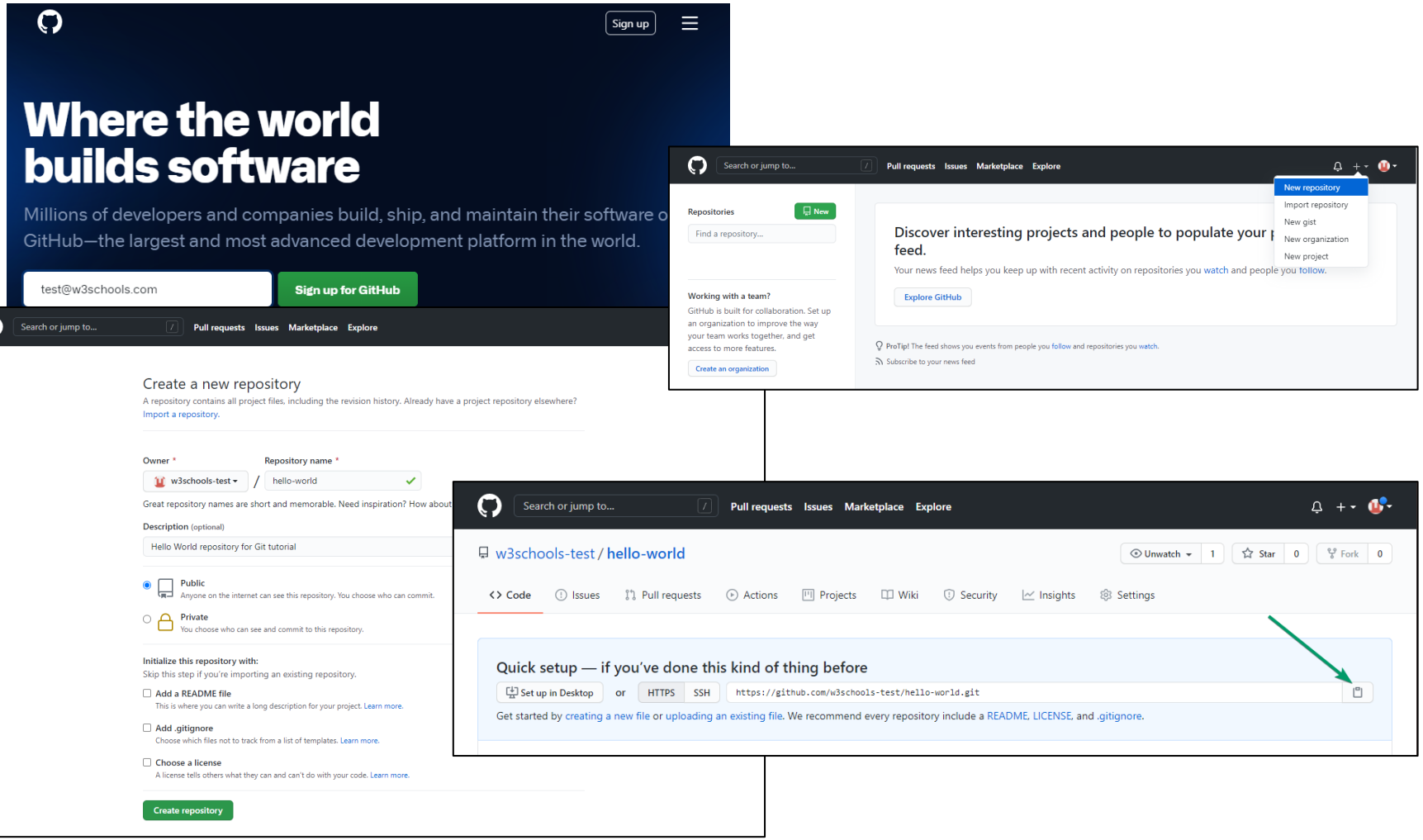
Git Commands (VSCode)







GitHub


Sign up
Where the world builds software
Millions of developers and companies build, ship, and maintain their software o
GitHub—the largest and most advanced development platform in the world.
test@w3schools.com
Sign up for GitHub
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Repositories
New
Find a repository...
Working with a team?
GitHub is built for collaboration. Set up an organization to improve the way your team works together, and get access to more features.
Create an organization
Discover interesting projects and people to populate your feed.
Your news feed helps you keep up with recent activity on repositories you watch and people you follow.
Explore GitHub
ProTip! The feed shows you events from people you follow and repositories you watch.
Subscribe to your news feed
New repository
Import repository
New gist
New organization
New project
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Owner *
Repository name *
w3schools-test
hello-world
Great repository names are short and memorable. Need inspiration? How about
Description (optional)
Hello World repository for Git tutorial
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Skip this step if you're importing an existing repository.
Add a README file
This is where you can write a long description for your project. Learn more.
Add .gitignore
Choose which files not to track from a list of templates. Learn more.
Choose a license
A license tells others what they can and can't do with your code. Learn more.
Create repository
w3schools-test / hello-world
Unwatch
1
Star
0
Fork
0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/w3schools-test/hello-world.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

GitHub Commands

- Push a repository to GitHub

```
git remote add origin https://github.com/mmamei/test.git
git push --set-upstream origin master
```

```
git remote -v
origin becomes an alias of
https://github.com/mmamei/test.git
```

- Clone a repository from GitHub

```
git clone https://github.com/mmamei/test.git
git clone https://github.com/mmamei/test.git dir
```

Creates the project directory in the current folder
Or
Into dir folder

- Push to GitHub

```
git push origin
```

- Pull from GitHub

```
git pull origin
```

Further Resorces

- https://git-scm.com/book/en/v2