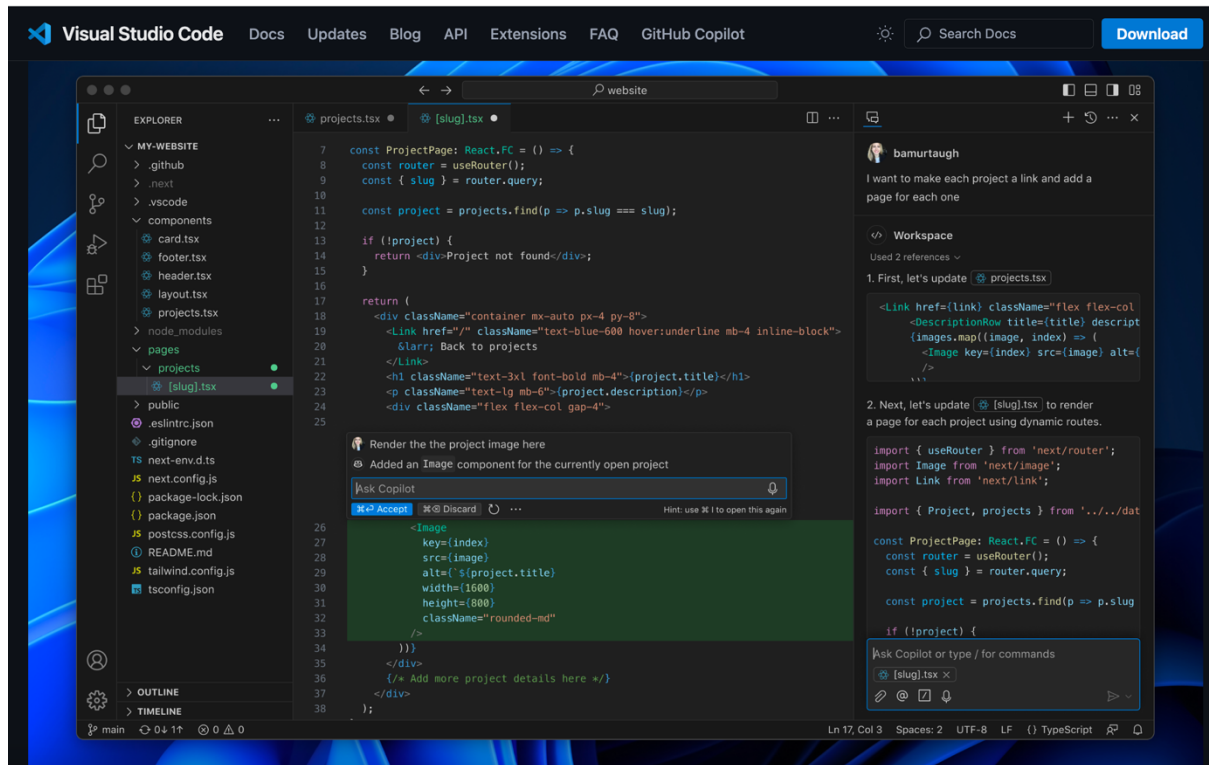


# Visual Studio Code (VS Code)

- È un editor di codice sorgente sviluppato da Microsoft. È gratuito, open-source e multiplatforma (disponibile per Windows, macOS e Linux)



# Strumenti: l'IDE (Integrated Development Environment)

E' un sistema che integra in modo integrato File Manager, Editor del codice, la Shell, il sistema di versionamento e altre funzionalità come il debugger.

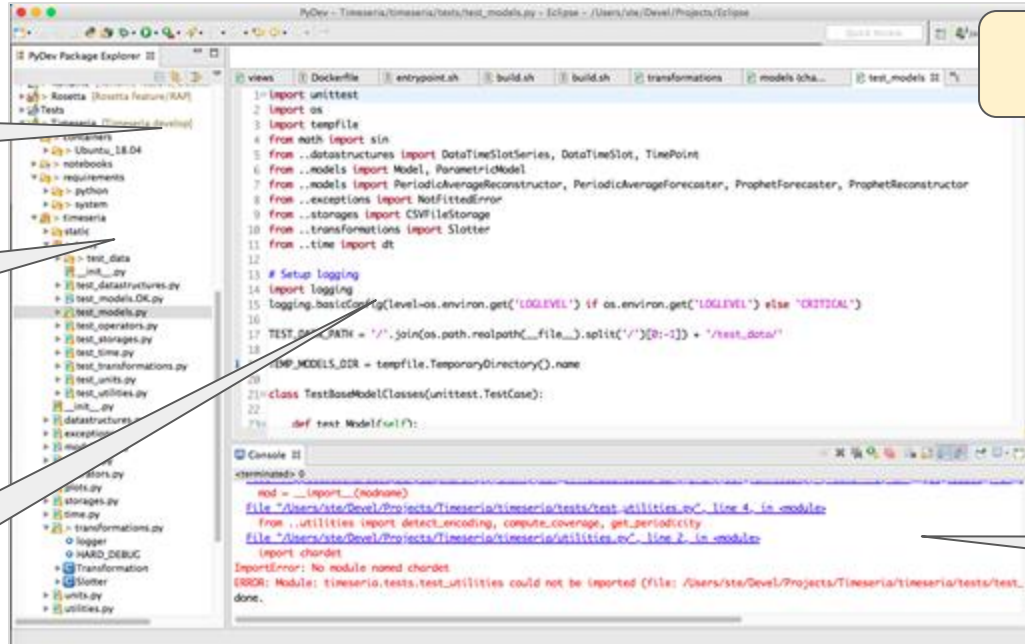
Sistema di  
versionamento

File  
Manager

Editor del  
codice

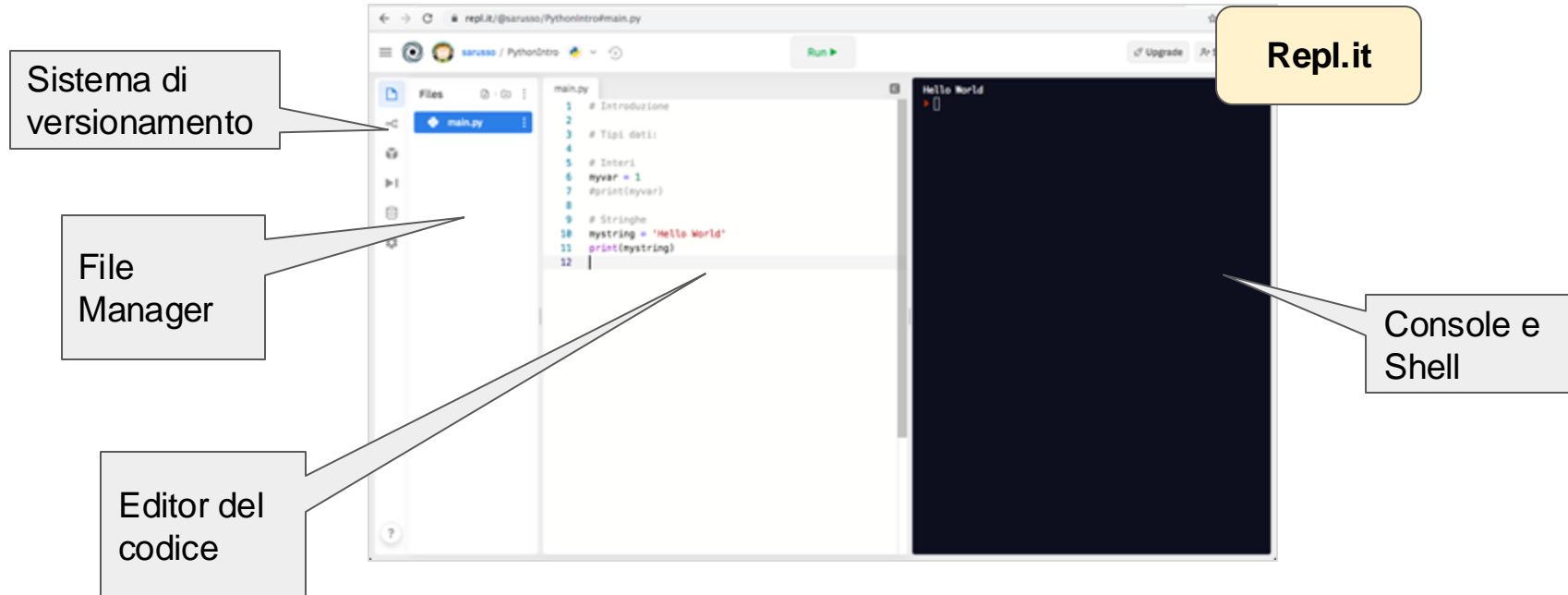
Eclipse

Shell



# Strumenti: l'IDE (Integrated Development Environment)

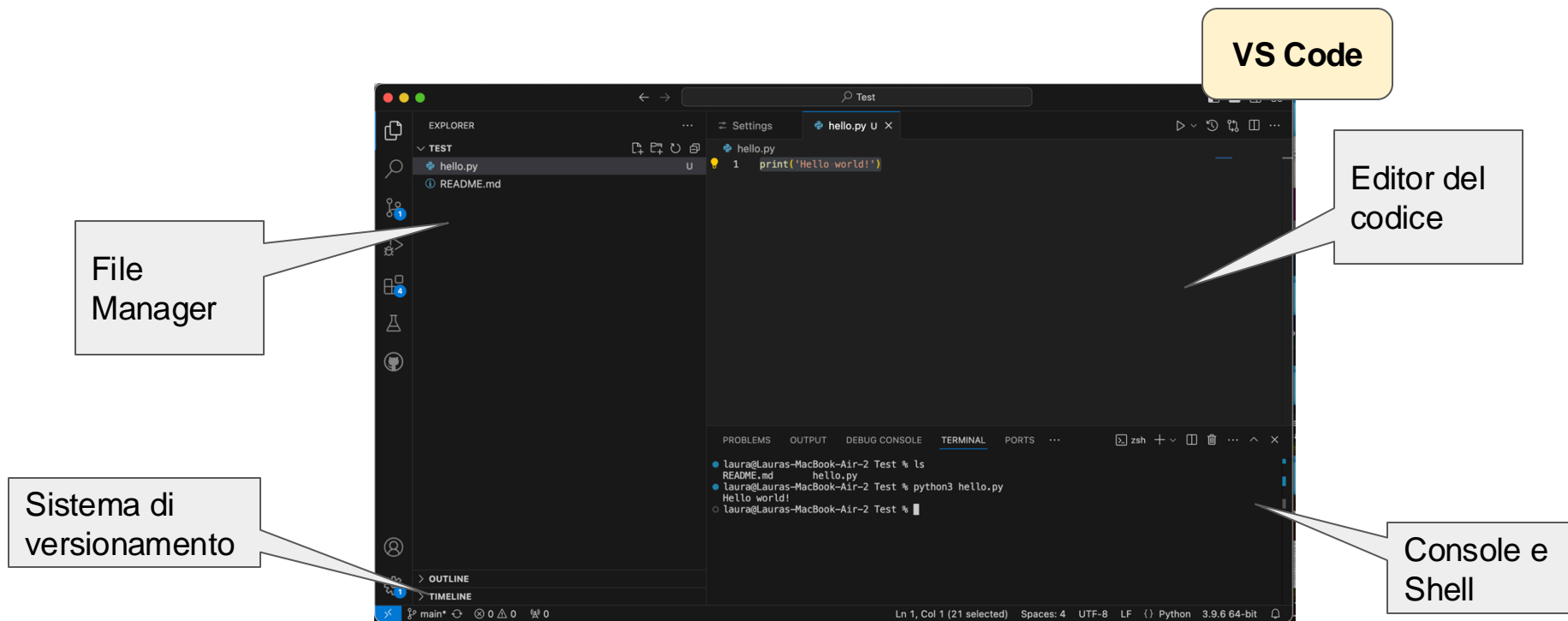
- Il tutto preconfigurato per uno o più linguaggi di programmazione specifici (ad esempio, PyCharm per Python, IntelliJ IDEA per Java).



# VS Code non è un IDE ma...

- VS Code è molto flessibile grazie al suo sistema di **estensioni**.
- Questo significa che può essere configurato per comportarsi quasi come un IDE completo aggiungendo estensioni, e.g. per il debugging, il controllo della versione, l'integrazione di database e altro.
- Gli **IDE** di solito offrono tutte queste funzionalità già pronte, senza necessità di aggiungere estensioni.
- Quindi, pur non essendo un IDE completo, VS Code può essere configurato per offrire un'esperienza simile

# VS Code usato come IDE



# VS Code per Python

The screenshot shows the Visual Studio Code website with a dark theme. The top navigation bar includes links for Docs, Updates, Blog, API, Extensions, FAQ, and GitHub Copilot, along with a search bar and a Download button. A banner for Version 1.95 is visible. The left sidebar lists various sections like Overview, Setup, Get Started, User Guide, Source Control, Terminal, GitHub Copilot, Languages, Node.js / JavaScript, TypeScript, and Python. The main content area features the article 'Quick Start Guide for Python in VS Code' with an 'Edit' button. The article text explains that the Python extension makes VS Code an excellent Python editor and provides instructions on how to install it. A screenshot of the VS Code interface shows the Python extension installed and its details page. The right sidebar contains a section 'IN THIS ARTICLE' with links to various guides and resources.

**Visual Studio Code** Docs Updates Blog API Extensions FAQ GitHub Copilot Search Docs Download

Version 1.95 is now available! Read about the new features and fixes from October.

## Quick Start Guide for Python in VS Code Edit

The Python extension makes Visual Studio Code an excellent Python editor, works on any operating system, and is usable with a variety of Python interpreters.

Get started by installing:

- [VS Code](#)
- [A Python Interpreter](#) (any [actively supported Python version](#))
- [Python extension](#) from the VS Code Marketplace

**IN THIS ARTICLE**

- How to create and open a Python project or file
- UI tour
- Code Actions
- Python commands
- Run, debug, and test
- Next steps

[Subscribe](#)

[Ask questions](#)

[Follow @code](#)

[Request features](#)

[Report issues](#)

[Watch videos](#)

**Python** v2023.22.1  
Microsoft [microsoft.com](#) 108,969,315 5 stars

IntelliSense (Pylance), Linting, Debugging (Python Debugger),...

[Disable](#) [Uninstall](#) [Switch to Pre-Release Version](#)

This extension is enabled globally.

**Python extension for Visual Studio Code**

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the Python language).

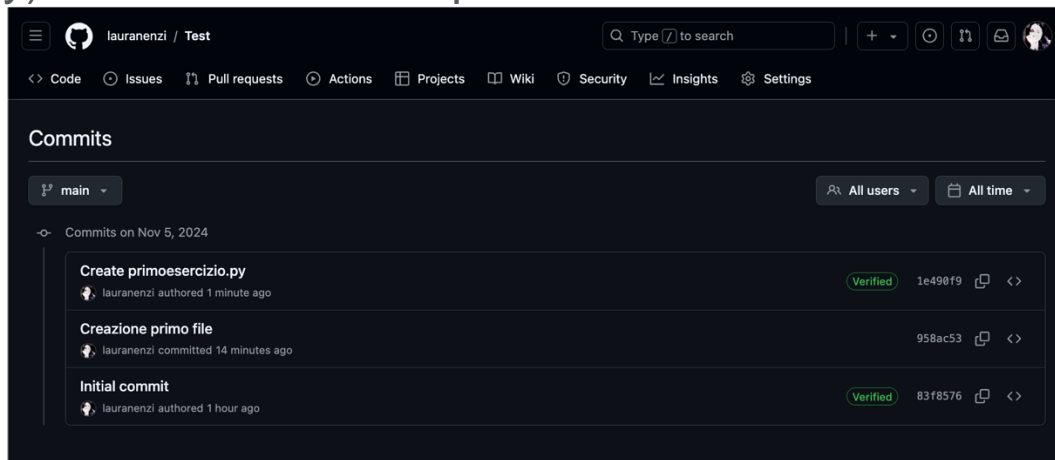
**Categories**

- Programming Languages
- Linters
- Debuggers
- Formatters
- Data Science

To further customize VS Code for Python, you can leverage the [Python profile template](#), automatically

# Strumenti: il sistema di versionamento (Git)

È quella cosa dove viene tenuta traccia di tutte le modifiche che avete fatto nel codice. Usate SEMPRE un sistema di versionamento, mai che vada Dropbox (che ha la history). Git è la soluzione più indicata.

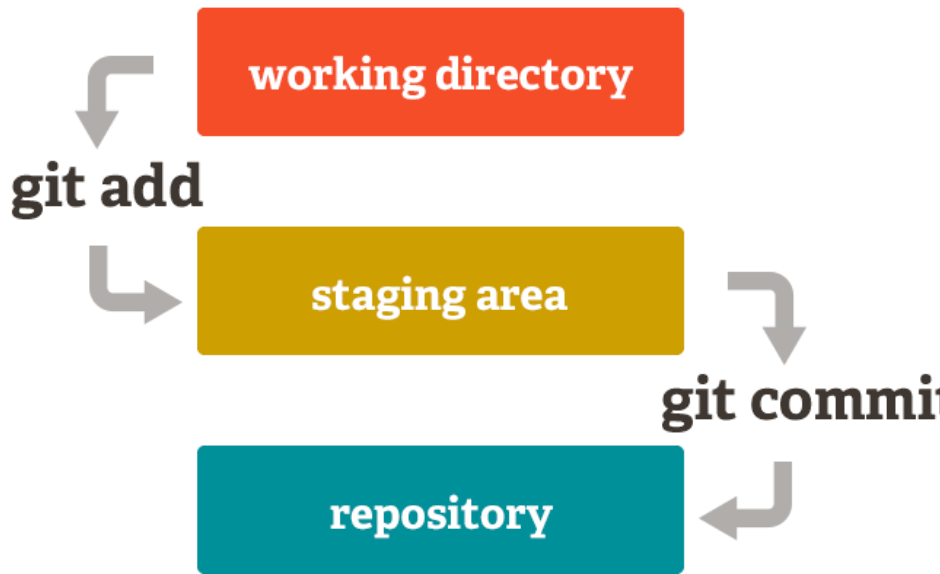


Tutorial di Michele Rispoli (tutor dell'anno scorso):

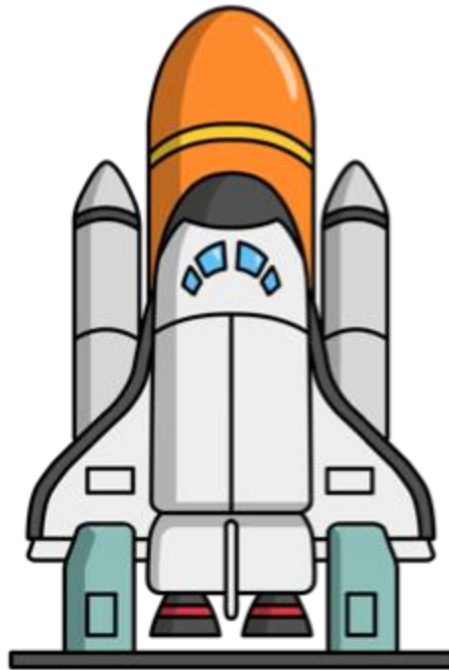
[https://github.com/drOpZ/proglab2021-tutors/blob/master/git\\_quickstart.md](https://github.com/drOpZ/proglab2021-tutors/blob/master/git_quickstart.md)

# Staging e Commit in Git

- **Staging** è l'area temporanea dove metti le modifiche che vuoi salvare. Ti permette di selezionare esattamente quali modifiche includere nel prossimo commit.
- **Commit** è il salvataggio permanente delle modifiche nel repository. Ogni commit è un punto di salvataggio che crea una cronologia del progetto.







Cominciamo

# Setup dell'ambiente

- 1) Scaricate VS Code (dovreste averlo già)
- 2) Scaricate l'estensione Python ed essere sicuri di avere python installato nel computer



# Setup dell'ambiente

3) Registratevi su GitHub se non lo siete già

4) Createvi un repository su GitHub chiamato "ProgrammingLab"


## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).


Required fields are marked with an asterisk (\*).

Owner \*

Repository name \*

 lauranenzi


/ ProgrammingLab

 ProgrammingLab is available.


Great repository names are short and memorable. Need inspiration? How about [solid-succotash](#) ?

**Description** (optional)

Repo for the programming Lab course

☒  **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:**

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs](#).

**Add .gitignore**

.gitignore template: **None**


Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

**Choose a license**

License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

This will set `main` as the default branch. Change the default name in your [settings](#).

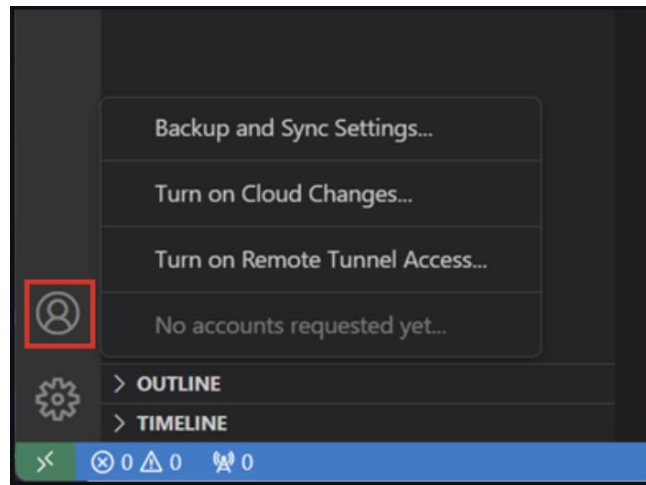
 You are creating a public repository in your personal account.

Create repository

# Setup dell'ambiente

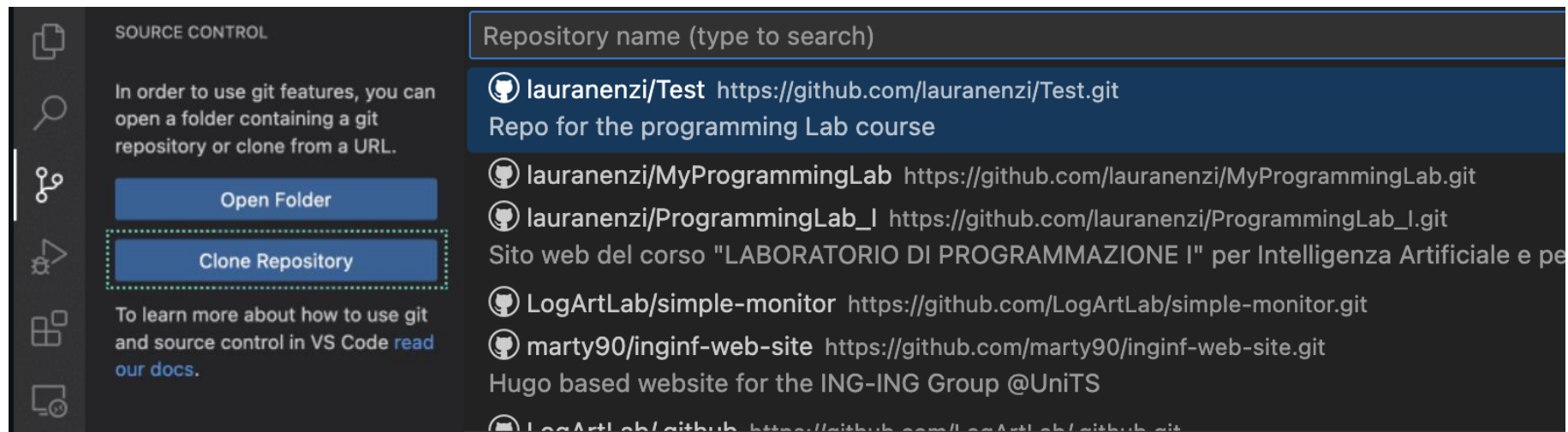
5) Accedete a VS Code col vostro account GitHub nel menu Account in basso a destra della barra Attività. Se Git è mancante, vengono mostrate le istruzioni su come installarlo. Assicuratevi di riavviare VS Code in seguito.

Sito git <https://git-scm.com/downloads>



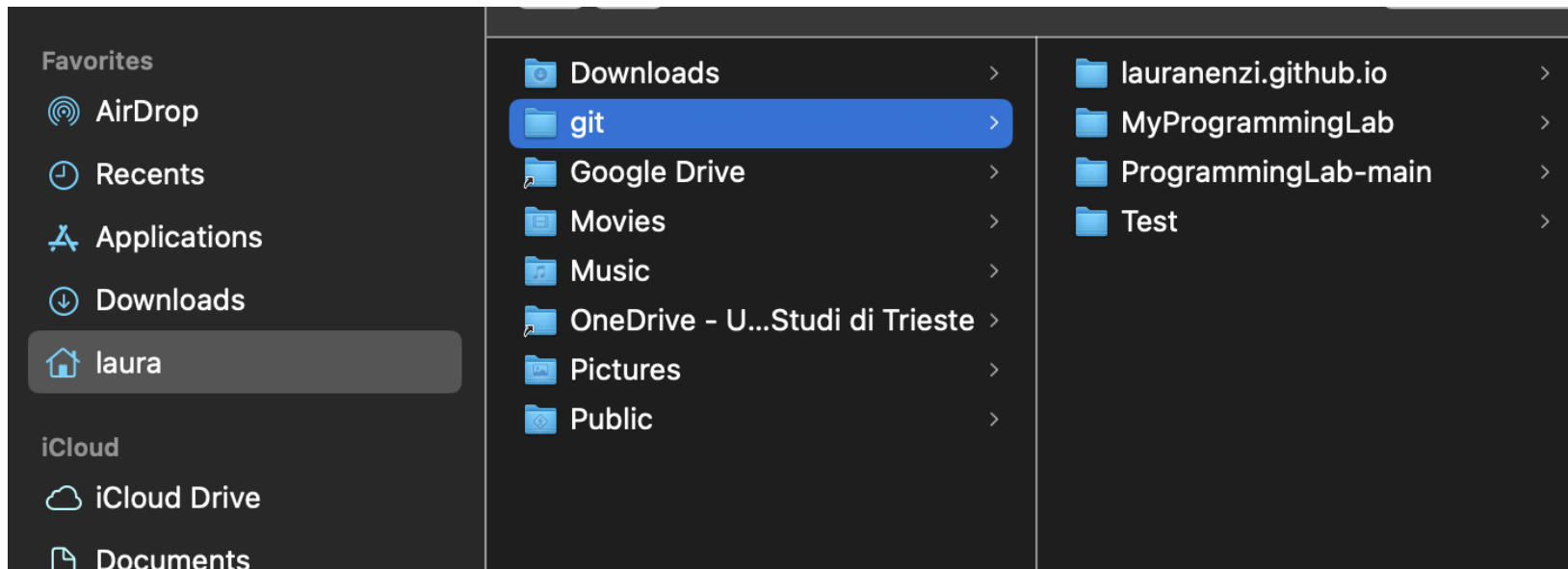
# Setup dell'ambiente

6) Scrivete il comando "Git: Clone" nel Command Palette ( $\uparrow$   $\text{Ctrl}+\text{P}$ ) o selezionate il Clone Repository nel Source Control. Scegliete quindi il repository "ProgrammingLab" che dovrebbe comparire tra i vostri repository



# Setup dell'ambiente

7) Salvate il repository localmente sul computer. Vi consiglio di creare una cartella git dove salvate tutti i repository localmente.



# Setup dell'ambiente

## 8) Installate un interprete python.

- 8a) Vedete dal sito di VS code come fare per i diversi sistemi operativi
- 8b) Oppure (Consigliato per AIDA)

- Installate miniconda: <https://docs.conda.io/en/latest/miniconda.html>


- Verificate che sia installato. Da terminale: `$ conda --version`

- Potreste dover inizializzare Cond, specialmente su macOS e Linux: `$ conda init`

- Crea un "virtual enviroment" `$ conda create -n nome_ambiente python=3.12`

# Setup dell'ambiente

9a) Si può creare l'ambiente anche dalla linea di comando di VScode ma ve lo sconsiglio.

Open the Command Palette () , start typing the **Python: Create Environment** command to search, and then select the command.

The command presents a list of environment types, Venv or Conda. For this example, select **Venv**.

Select an environment type

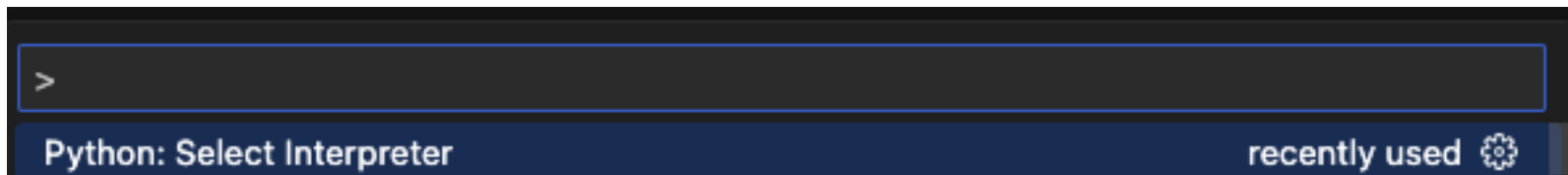
**Venv** Creates a `.venv` virtual environment in the current workspace

**Conda** Creates a `.conda` Conda environment in the current workspace



# Setup dell'ambiente

9b) Selezionate un “virtual environment”, i.e. un ambiente virtuale. Una volta attivato quell'ambiente, tutti i pacchetti che installi successivamente sono isolati dagli altri ambienti, incluso l'ambiente globale dell'interprete, riducendo molte complicazioni che possono sorgere da conflitti tra versioni dei pacchetti.



# Primi comandi

Creiamo adesso uno script "hello.py" con dentro il contenuto:

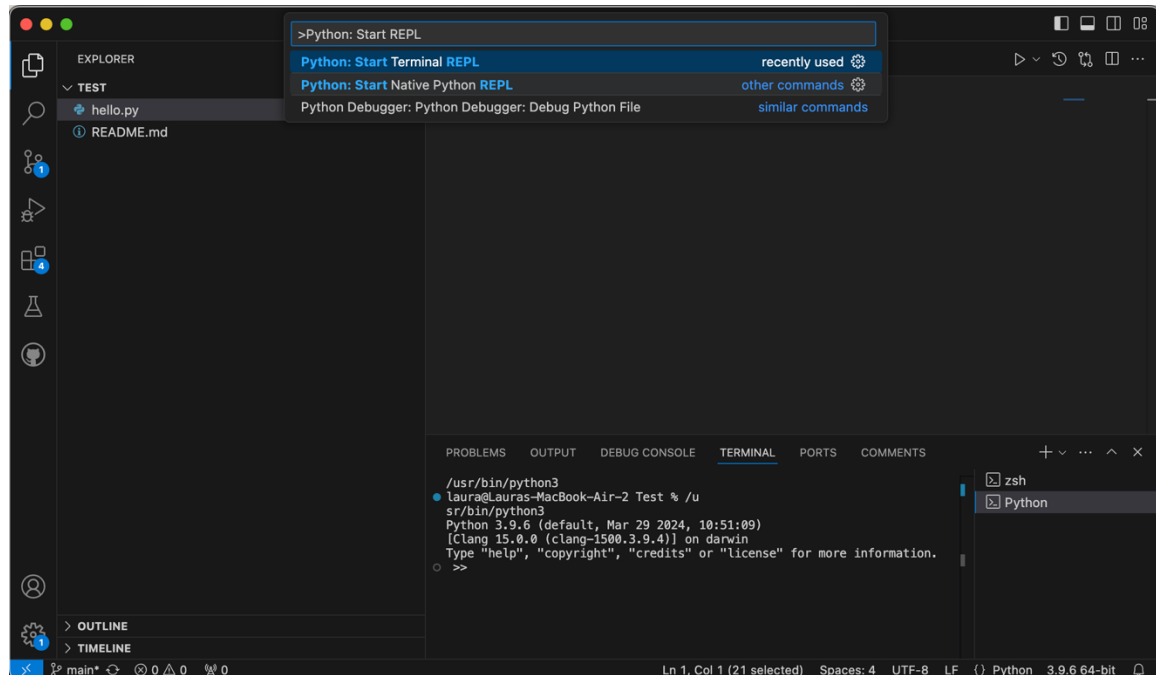
```
print('Hello world!')
```

Poi, eseguiamo lo script dalla shell

```
$ python hello.py  
Hello world!
```

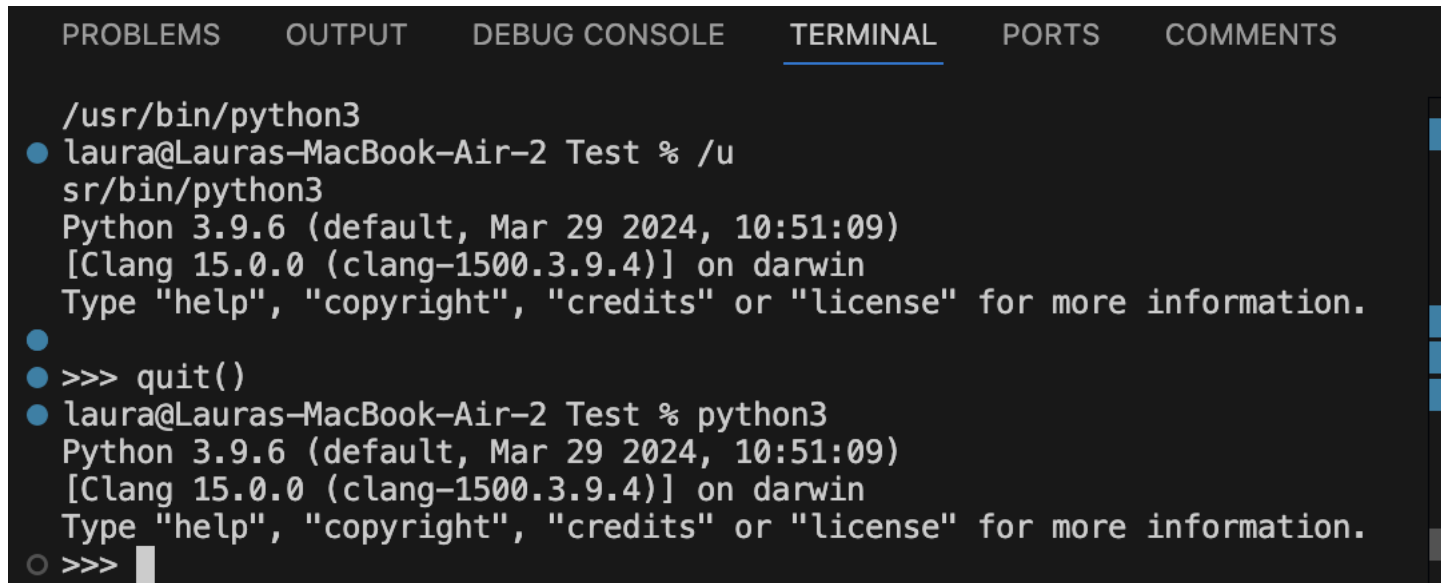
# Console

- Dalla Command Palette ( $\uparrow \text{CMD} P$ ), seleziona il comando Python: Start REPL per aprire un terminale REPL per l'interprete Python attualmente selezionato. In REPL, puoi quindi immettere ed eseguire le righe di codice una alla volta.



# Console

- `quit()` per uscire. Stesso comando scrivendo “python3” direttamente nella shell

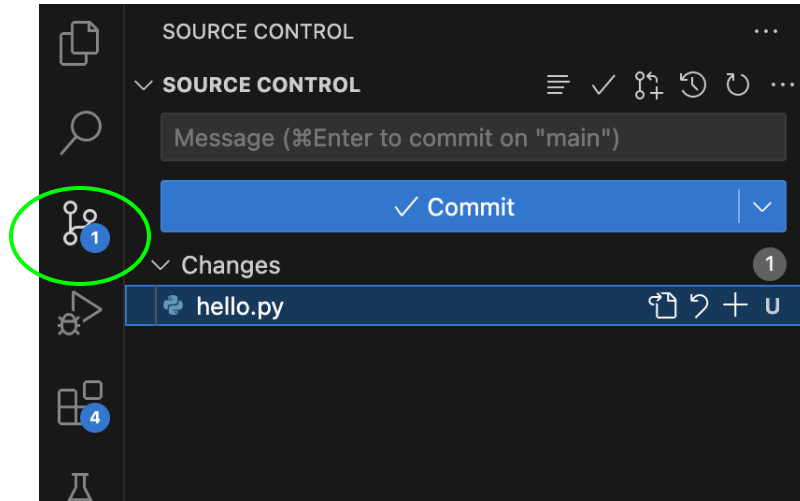


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

/usr/bin/python3
● laura@Lauras-MacBook-Air-2 Test % /usr/bin/python3
Python 3.9.6 (default, Mar 29 2024, 10:51:09)
[Clang 15.0.0 (clang-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
●
● >>> quit()
● laura@Lauras-MacBook-Air-2 Test % python3
Python 3.9.6 (default, Mar 29 2024, 10:51:09)
[Clang 15.0.0 (clang-1500.3.9.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
○ >>> 
```

# Source Control

- Puoi accedere alla “Source Control view” dalla Barra dell'attività che elenca tutti i file modificati nel tuo spazio di lavoro.



# Staging

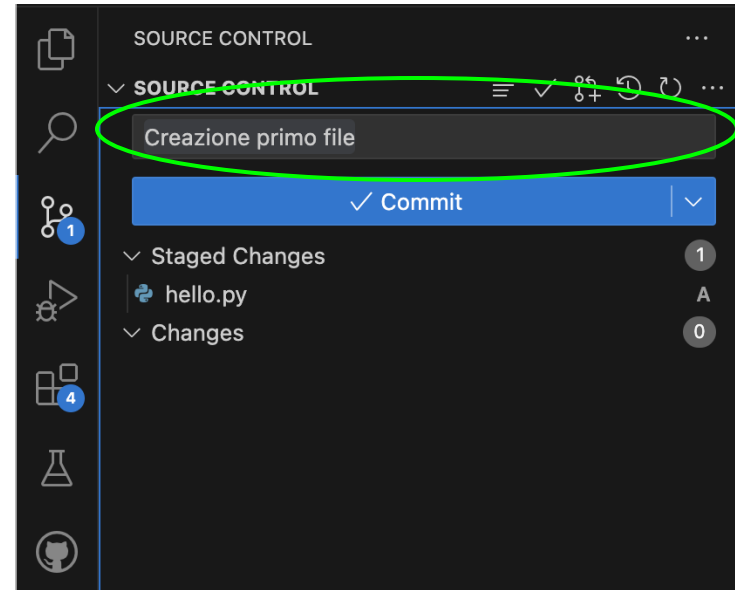
- Quando si seleziona un file, l'editor mostra una vista diff che evidenzia le modifiche del file rispetto al file precedentemente modificato.



- Per mettere in staging un file, seleziona l'icona + (più) accanto al file nella vista Controllo sorgente. Questo aggiunge il file alla sezione Modifiche in staging, indicando che verrà incluso nel prossimo commit.

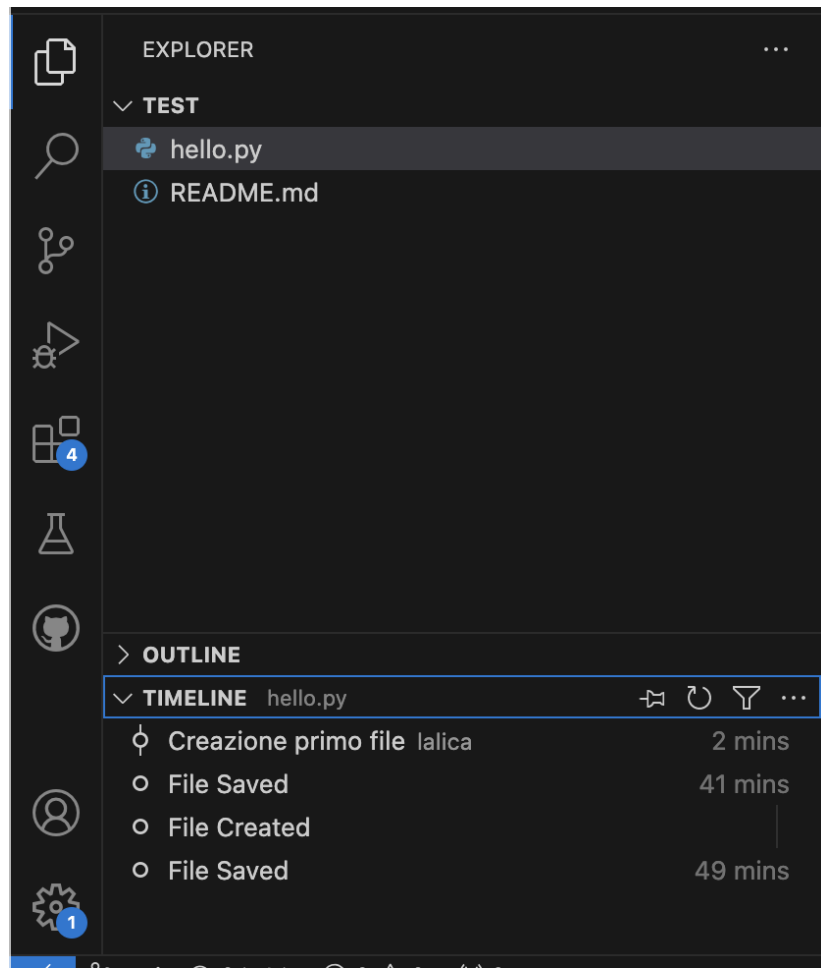
# Commit

- Per eseguire il commit delle modifiche in stage, digita un messaggio di commit nella casella di testo superiore, quindi seleziona il pulsante Commit. Questo salva le modifiche nel repository Git locale, consentendoti di ripristinare le versioni precedenti del codice se necessario.
- Suggerimento: eseguite il commit delle modifiche in anticipo e spesso. Ciò rende più facile tornare alle versioni precedenti del tuo codice, se necessario.



# Timeline

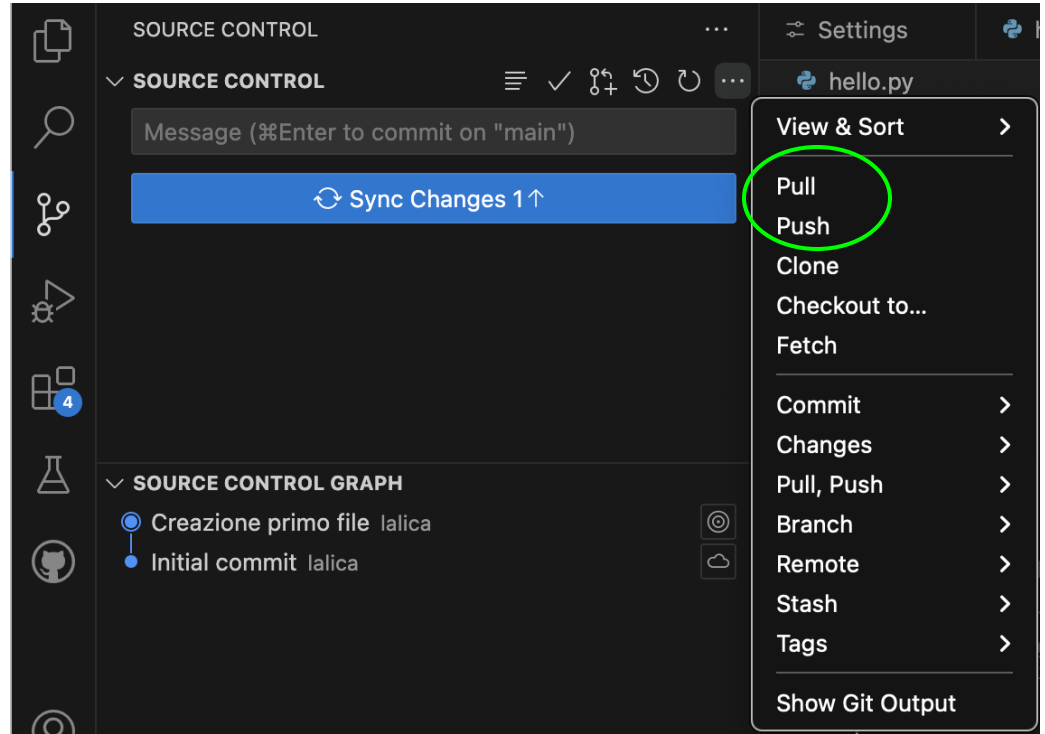
- È possibile navigare e rivedere tutte le modifiche dei file locali nella vista “Timeline” disponibile nella parte inferiore della vista Esplora.



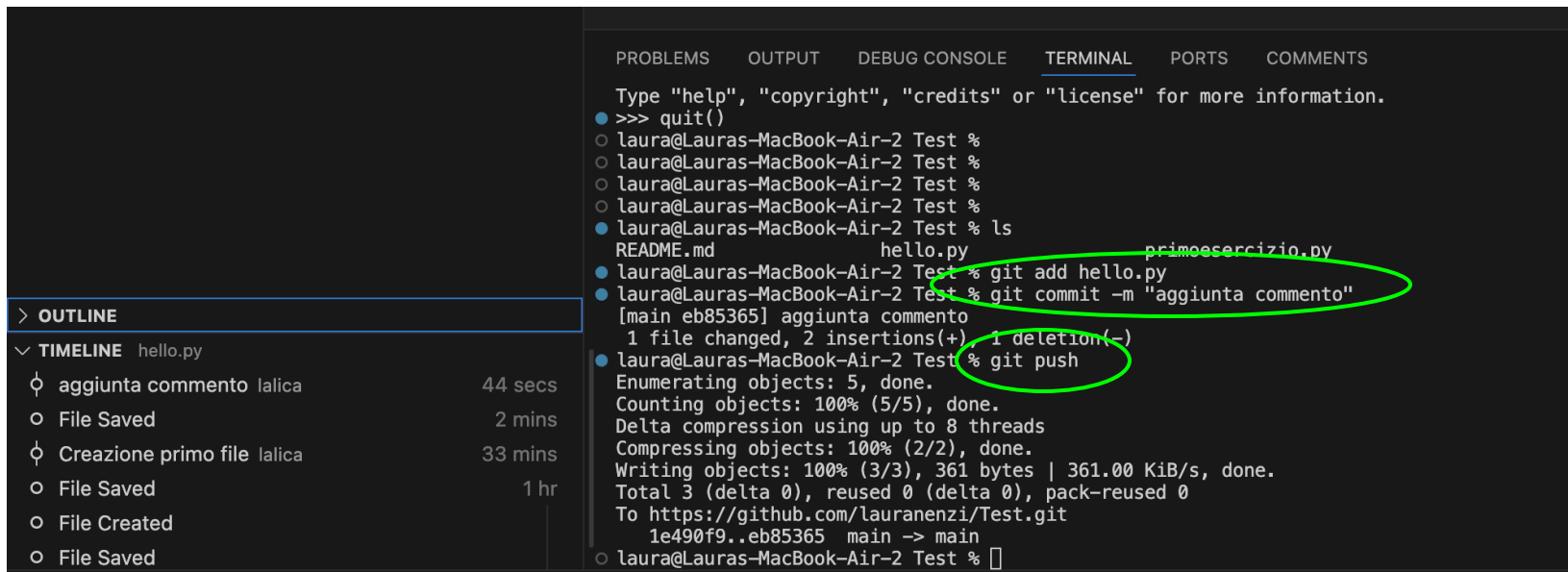


# Push and Pull

- Push carica le tue modifiche sul remoto.
- Pull scarica le modifiche dal remoto.
- È possibile accedere ai comandi Push e Pull dal menu Source Control.



# Git da terminale



The image shows a screenshot of an IDE interface. On the left, there is an 'OUTLINE' panel with a 'TIMELINE' section for the file 'hello.py'. The timeline shows a sequence of actions: 'aggiunta commento' (44 secs), 'File Saved' (2 mins), 'Creazione primo file' (33 mins), 'File Saved' (1 hr), 'File Created', and 'File Saved'. The main terminal window on the right displays the output of Git commands. The terminal tabs at the top are 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (selected), 'PORTS', and 'COMMENTS'. The terminal output shows the following sequence of commands and their outputs:

```
Type "help", "copyright", "credits" or "license" for more information.
• >>> quit()
○ laura@Lauras-MacBook-Air-2 Test %
○ laura@Lauras-MacBook-Air-2 Test %
○ laura@Lauras-MacBook-Air-2 Test %
○ laura@Lauras-MacBook-Air-2 Test %
• laura@Lauras-MacBook-Air-2 Test % ls
  README.md      hello.py      primoesercizio.py
• laura@Lauras-MacBook-Air-2 Test % git add hello.py
• laura@Lauras-MacBook-Air-2 Test % git commit -m "aggiunta commento"
[main eb85365] aggiunta commento
  1 file changed, 2 insertions(+), 1 deletion(-)
• laura@Lauras-MacBook-Air-2 Test % git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 361 bytes | 361.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/lauranenzi/Test.git
  1e490f9..eb85365  main -> main
○ laura@Lauras-MacBook-Air-2 Test %
```

Red circles highlight the following parts of the terminal output:

- The file `primoesercizio.py` in the `ls` command output.
- The command `git commit -m "aggiunta commento"`.
- The command `git push`.

# Cronologia su Github

The screenshot shows the GitHub interface for the repository 'lauranenzi / Test'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A search bar is located on the right. The main content area is titled 'Commits' and shows a list of commits for the 'main' branch. The commits are filtered by 'All users' and 'All time'. The commit history is grouped by date, showing commits from November 6, 2024, and November 5, 2024. The commits are listed in reverse chronological order, with the most recent commit at the top.


Commits on Nov 6, 2024

- aggiunta commento**  
lauranenzi committed 2 minutes ago  
eb85365

Commits on Nov 5, 2024

- Create primoesercizio.py**  
lauranenzi authored 22 minutes ago  
1e490f9
- Creazione primo file**  
lauranenzi committed 35 minutes ago  
958ac53
- Initial commit**  
lauranenzi authored 1 hour ago  
83f8576


# Estensione Jupyter notebook



## Jupyter



Microsoft [microsoft.com](https://microsoft.com) | 88,996,208 | ★★★★★ (330)



Jupyter notebook support, interactive programming and computing that supports Intellisense, debugging and...

[Disable](#) [Uninstall](#) [Switch to Pre-Release Version](#) ☒ Auto Update 

[DETAILS](#) [FEATURES](#) [CHANGELOG](#)

### Extension Pack (4)

**Jupyter Keymap**  
Jupyter keymaps for notebooks  
Microsoft 

**Jupyter Notebook Renderers** 1ms  
Renderers for Jupyter Notebooks (with pl...  
Microsoft 

### Jupyter Extension for Visual Studio Code

A [Visual Studio Code](#) extension that provides basic notebook support for [language kernels](#) that are supported in [Jupyter Notebooks](#) today, and allows any Python environment to be used as a Jupyter kernel. This is **NOT a Jupyter kernel**--you must have Python environment in which you've installed the [Jupyter package](#), though many language kernels will work with no modification. To enable advanced features, modifications may be needed in the VS Code language extensions.

### Installation

Identifier	ms-toolsai.jupyter
Version	2025.1.0
Last Updated	2025-02-11, 09:25:26
Size	16.55MB

### Marketplace

Published	2020-11-11, 20:14:18
Last Released	2025-03-12, 11:06:11

### Categories

[Extension Packs](#)[Data Science](#)