

Programming Lab

Parte 1

Introduzione e strumenti

Laura Nenzi

Many slides by Stefano Russo

Benvenuti

Corso: Laboratorio di programmazione I (3 CFU) 

→ Modulo del corso: INTRODUZIONE ALLA PROGRAMMAZIONE E
LABORATORIO (15 CFU) che comprende Modulo prof. Caravagna (9 CFU) e
Laboratorio di programmazione II (3 CFU)

Docente: Laura Nenzi (io )

Sito Web (repository): https://github.com/lauranenzi/ProgrammingLab_I

Ricevimento: libero, scrivetemi a lnenzi@units.it

Tutors: Valentina Blasone e Lucrezia Valeriani

Chi sono

Laura Nenzi

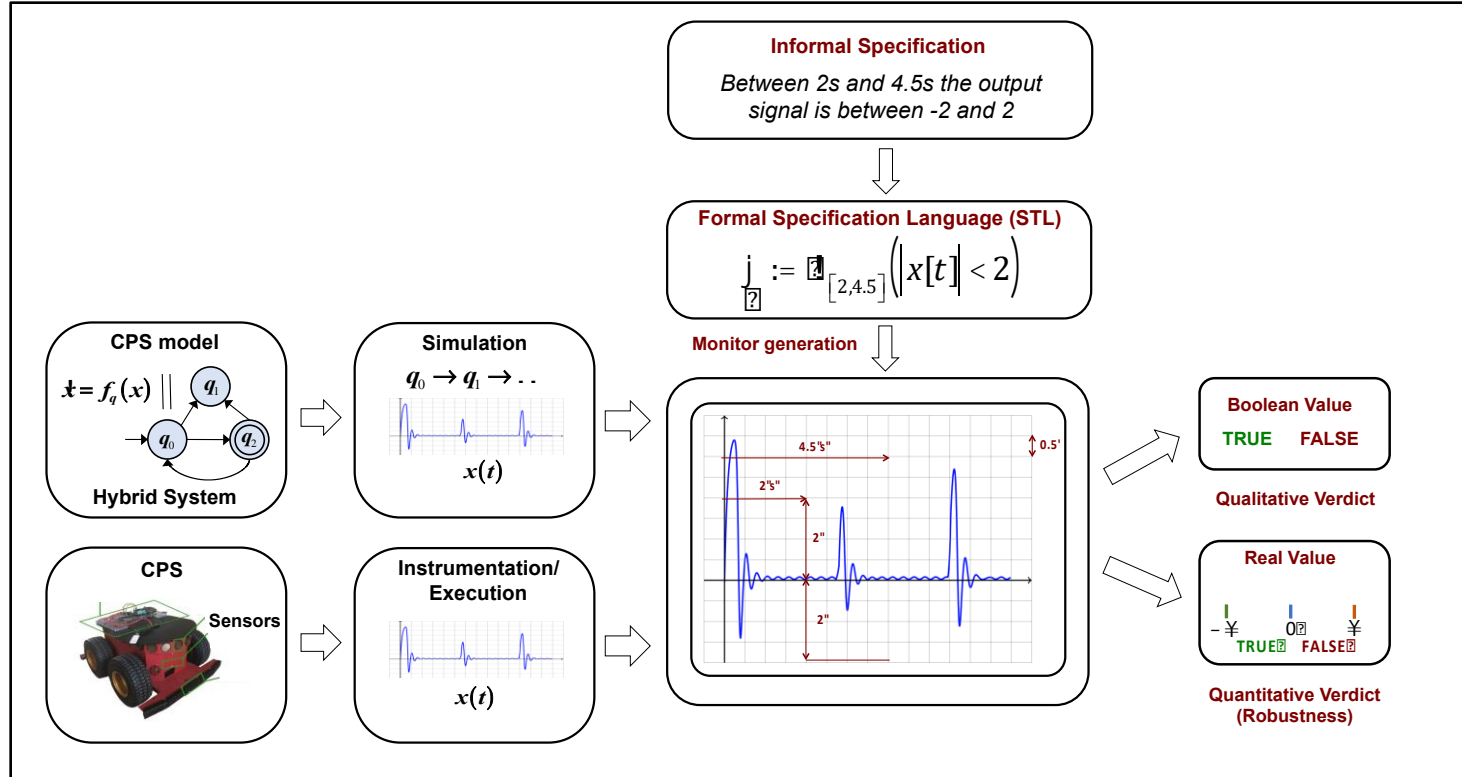
- Professore Associato in Ingegneria Informatica al Dipartimento di Ingegneria ed Architettura
- Laurea in Matematica, Phd in Computer Science
- Ricerca in Metodi Formali e Machine Learning

Ufficio c3 2.55

Mail: lnenzi@units.it



Cosa Faccio



Argomenti del corso

- 1) Intro del corso e strumenti da "laboratorio": la shell, Git, gli IDE, etc.
- 2) Python: tipi dati, costrutti, funzioni, moduli, be pythonic.
- 3) Interagire con i file ed il formato CSV.
- 4) Gli oggetti in Python
- 5) Le eccezioni ed il flusso try-except

Argomenti del corso, lezione per lezione

6) Controllo degli input e sanitizzazione

7) Testing e unit tests

8) Lavorare veramente:

- creiamo un modello
- fittiamo un modello
- valutiamo un modello

Informazioni Utili

- A. Downey J. Elkner C. Meyers, How to Think Like a Computer Scientist, Learning with Python.
 - traduzione italiana (versione pdf) di Andrea Zanella:
https://www.python.it/doc/Howtothink/HowToThink_ITA.pdf
 - traduzione italiana (versione web) di Alessandro Pocaterra:
<https://www.python.it/doc/Howtothink/Howtothink-html-it/index.htm>
- Sito web della comunità italiana di Python: <https://www.python.it/>
- Tutorial ufficiale in inglese (scegliete la versione di Python che avete installato): <https://docs.python.org/3/tutorial/index.html>
- Altro Tutorial di supporto: <https://www.w3schools.com/python/default.asp>

Perchè questo corso (fatto in questo modo)

Concetto: non farvi perdere tempo a far funzionare le cose nei prossimi anni.

Non è un buon uso del vostro tempo

Meglio fare “cose fiche”, no?

..ma serve un po' di sforzo all'inizio (leggi: questo corso 😊)

Perchè questo corso - un'analogia

Mario monta la sua nuova TV alla svelta, tira una teleferica per il cavo dell'alimentazione, la attacca al muro col Patafix, e poi disabilita la chiave del WiFi perchè non riesce a configurarla sulla TV.

Mario passa le prossime serate a litigare con cavi mangiati dal cane, con la TV sbilenca e con il WiFi lento perchè intanto gliel'hanno fregato i vicini.

Perchè questo corso - un'analogia

Bill invece, investe un paio d'ore nel montare la sua nuova TV e passa il cavo per bene, fa un buco col trapano per il supporto, e configura il WiFi correttamente.

Bill passa le prossime serate a godersi la sua serie preferita su Netflix in pace.

ecco, siate come Bill.

Perchè questo corso

“ma io ho fretta”

“ma io voglio solo fare una cosa rapida”

“ma io non sono un ingegnere software!”

“ma io non ho le basi”

“ma io voglio iniziare subito a fare codice”

Perchè questo corso

..cosa vi aspettate che io dica ora?

3..

2..

1..

Perchè questo corso

...che in realtà avete ragione.

Perchè in effetti io ho omesso un dettaglio, torniamo alla slide di prima

Perchè questo corso

Bill invece, investe un paio d'ore nel montare la sua nuova TV e passa il cavo per bene, fa un buco col trapano per il supporto, e configura il WiFi correttamente.

Bill passa le prossime serate a godersi la sua serie preferita su Netflix in pace.

ecco, siate come Bill*.

**** se sapete che poi paga!***

Perchè questo corso

Ecco, lo scopo di questo corso è di mostrarvi che se facciamo le cose per bene nel gestire il codice poi tutto ciò paga. Un po' come montare la TV correttamente. E vi converrà farlo sempre, perchè non perderete tempo su cose stupide.

Perchè di fatto, voi avrete a che fare col codice per tutto il corso di laurea, e probabilmente per buona parte della vostra vita.

Agli statistici in sala: anche voi! Il mestiere sta cambiando tanto, non vorrete essere già vecchi vero?

Organizzazione delle ore

Prima ora: pratica (con me e gli assistenti)

Seconda/Terza ora: Teoria

Oggi, alla fine della lezione dovrete tutti sapere*:

- 1) Usare Visual Studio Code (VSC) per python
- 2) Come si esegue uno script Python dentro a VSC
- 3) Come si fa un commit da VSC su GitHub

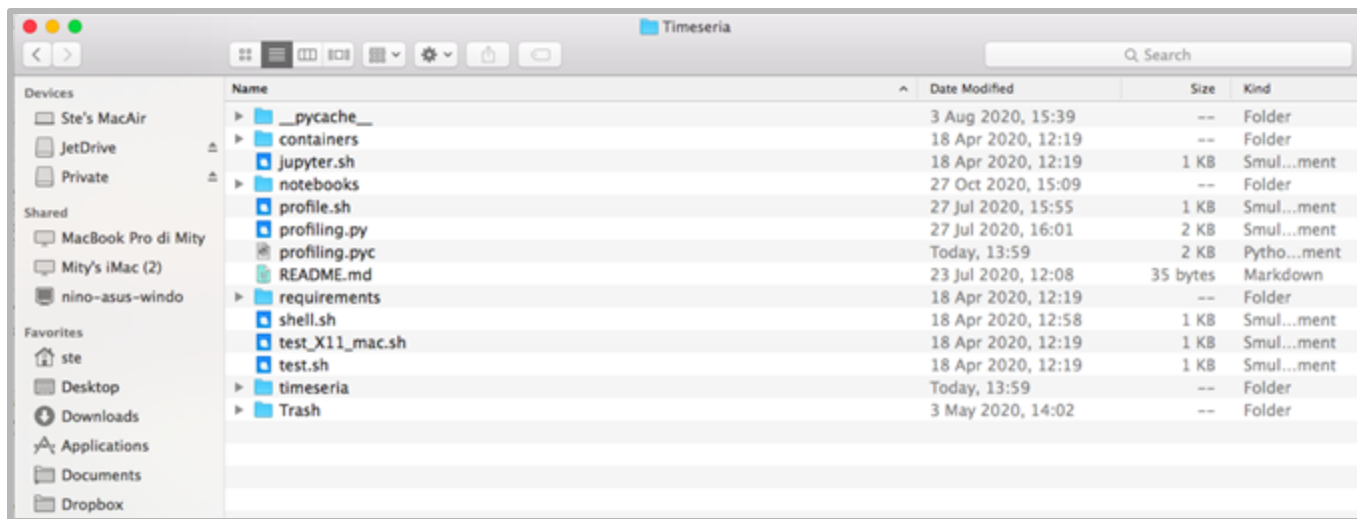
Iniziamo! ...con gli strumenti

“Datemi sei ore per abbattere un’ albero e ne spenderò le prime quattro per affilare l’ascia”

- *Abraham Lincoln*

Strumenti: il File Manager

È quello con cui si vedono le cartelle e i files del computer. Configuratelo per vedere anche i file nascosti e le estensioni!



Strumenti: la Shell (anche Terminale / Console)

È quella cosa con cui si interagisce con il computer in via testuale attraverso una Command-Line Interface (CLI), senza bottoni che automatizzano le cose. È come si fanno le cose sul serio senza usare un ambiente preconfezionato. Su sistemi Unix in genere è “BASH”. Nei sistemi Windows sono il Prompt dei Comandi (CMD) e la PowerShell.

```
stef@stef-MacAir:~$ git status
stef@stef-MacAir:~$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
stef@stef-MacAir:~$ git log
commit fac8594650f2e59d17bd24eaeade026c11612647 (HEAD -> master, origin/master, origin/HEAD)
Author: Stefano Alberto Russo <stefano.russo@gmail.com>
Date:   Fri Nov 6 12:03:07 2020 +0100

    Added ProgrammingLab stuff

commit 31ed8506aa7e2e655560d721247b3e654569d942
Author: Stefano Alberto Russo <stefano.russo@gmail.com>
Date:   Sat Apr 18 18:23:08 2020 +0200

    Added empty index.

commit bd9ee01d8dc6119f8738e7e15e33dd362324fc04
Author: Stefano Alberto Russo <stefano.russo@gmail.com>
Date:   Sat Apr 18 18:20:16 2020 +0200

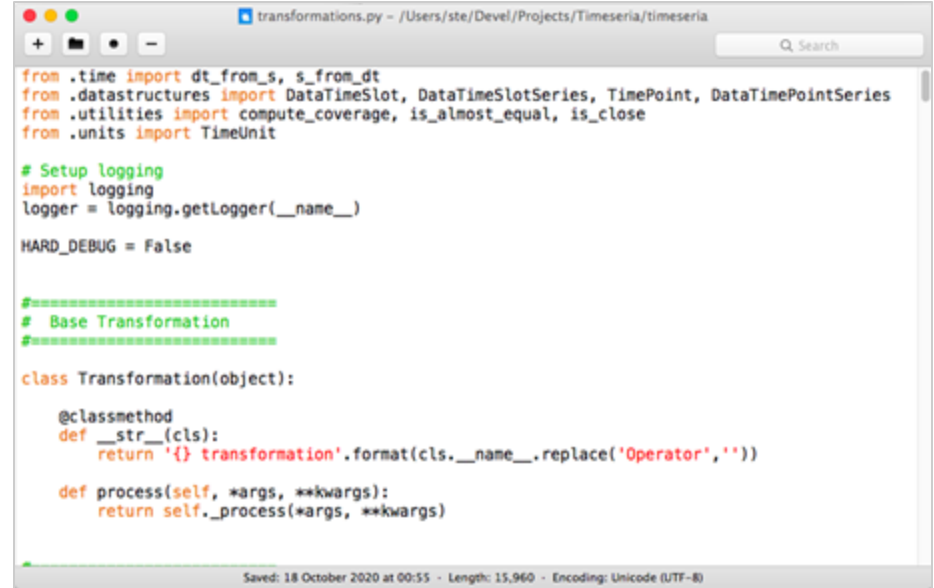
    Added Timeseria doc first stub.

commit 05884aea338a5c6d368afb41cc1459eded514d3
Author: Stefano Alberto Russo <stefano.russo@gmail.com>
Date:   Sat Apr 18 16:04:21 2020 +0200

    Initial commit
stef@stef-MacAir:~$
```

Strumenti: l'Editor del codice

È quella cosa con cui scrivete il codice. A differenza di un programma di videoscrittura (come Microsoft Word), che formatta il testo con font, dimensioni e colori, un editor di testo è progettato principalmente per gestire testo semplice, senza elementi di formattazione avanzati. Gli editor del codice hanno colori specifici per aiutare a programmare.



```
transformations.py - /Users/ste/Devel/Projects/Timeseria/timeseria

from .time import dt_from_s, s_from_dt
from .datastructures import DateTimeSlot, DateTimeSlotSeries, TimePoint, DateTimePointSeries
from .utilities import compute_coverage, is_almost_equal, is_close
from .units import TimeUnit

# Setup logging
import logging
logger = logging.getLogger(__name__)

HARD_DEBUG = False

# =====
# Base Transformation
# =====

class Transformation(object):

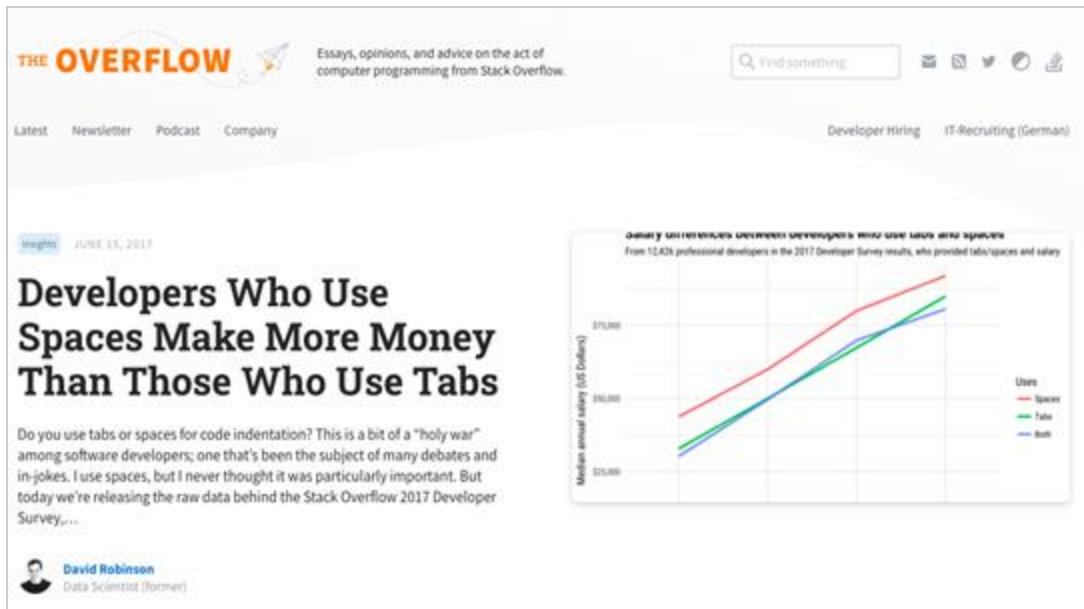
    @classmethod
    def __str__(cls):
        return '{} transformation'.format(cls.__name__.replace('Operator', ''))

    def process(self, *args, **kwargs):
        return self._process(*args, **kwargs)

Saved: 18 October 2020 at 00:55 - Length: 15,960 - Encoding: Unicode (UTF-8)
```

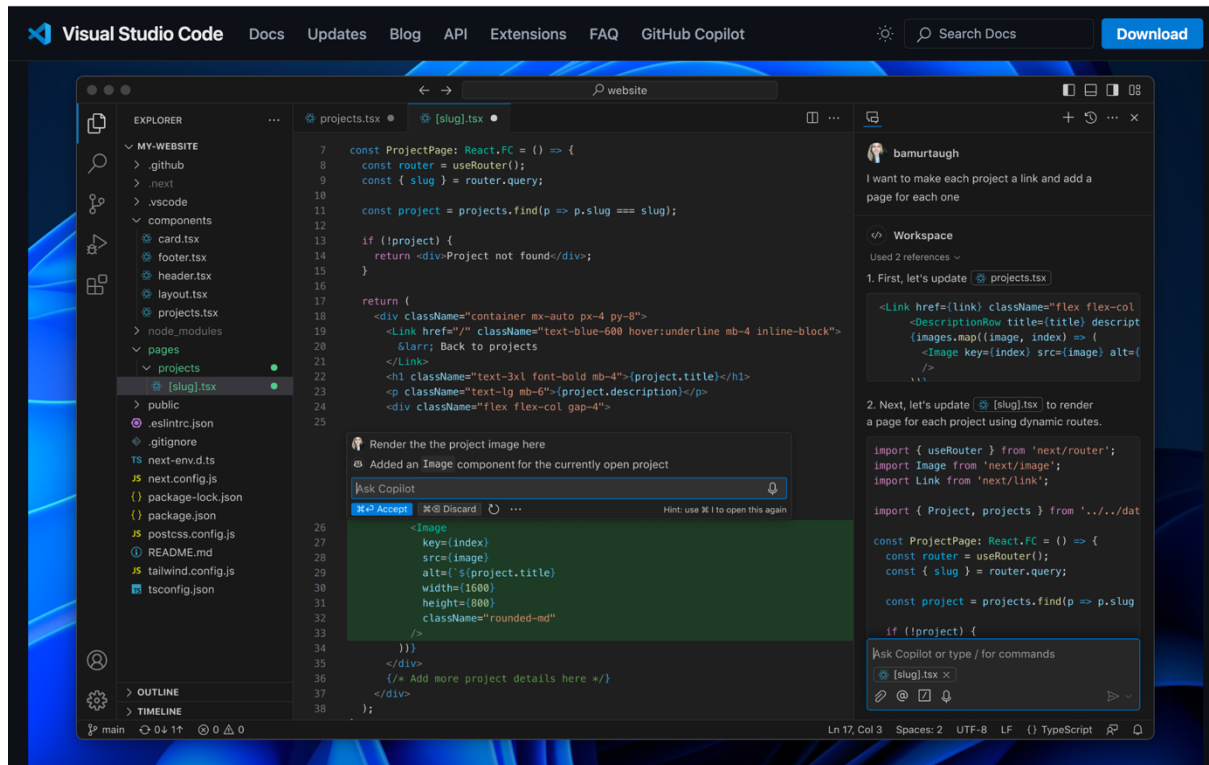
Strumenti: l'Editor del codice

Nota: è tassativo impostare l'editor a usare gli spazi e non i tab. Indentazione a 4 spazi per Python.



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.

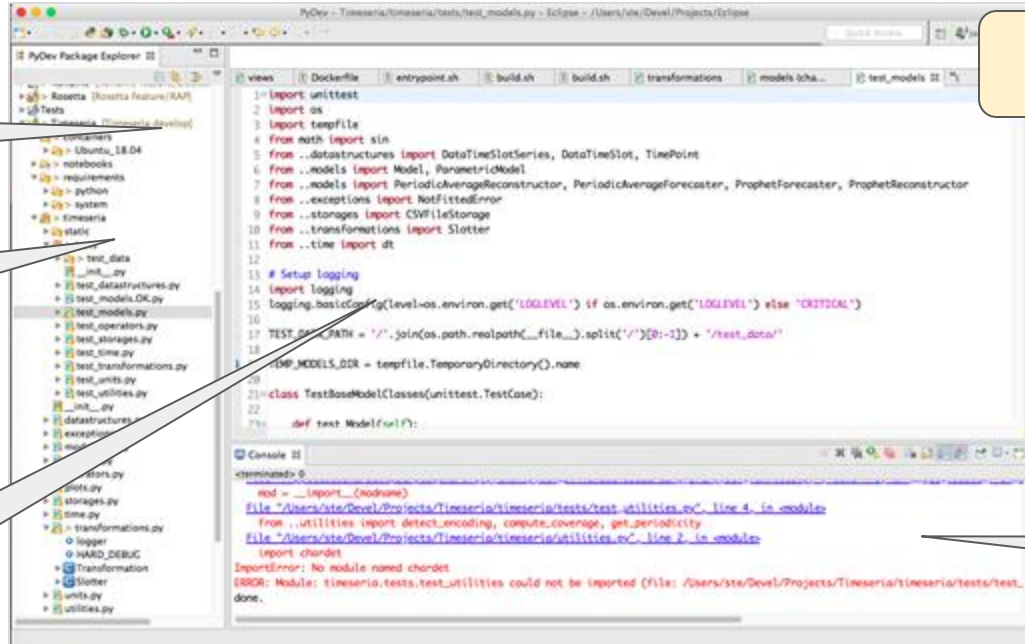
Sistema di
versionamento

File
Manager

Editor del
codice

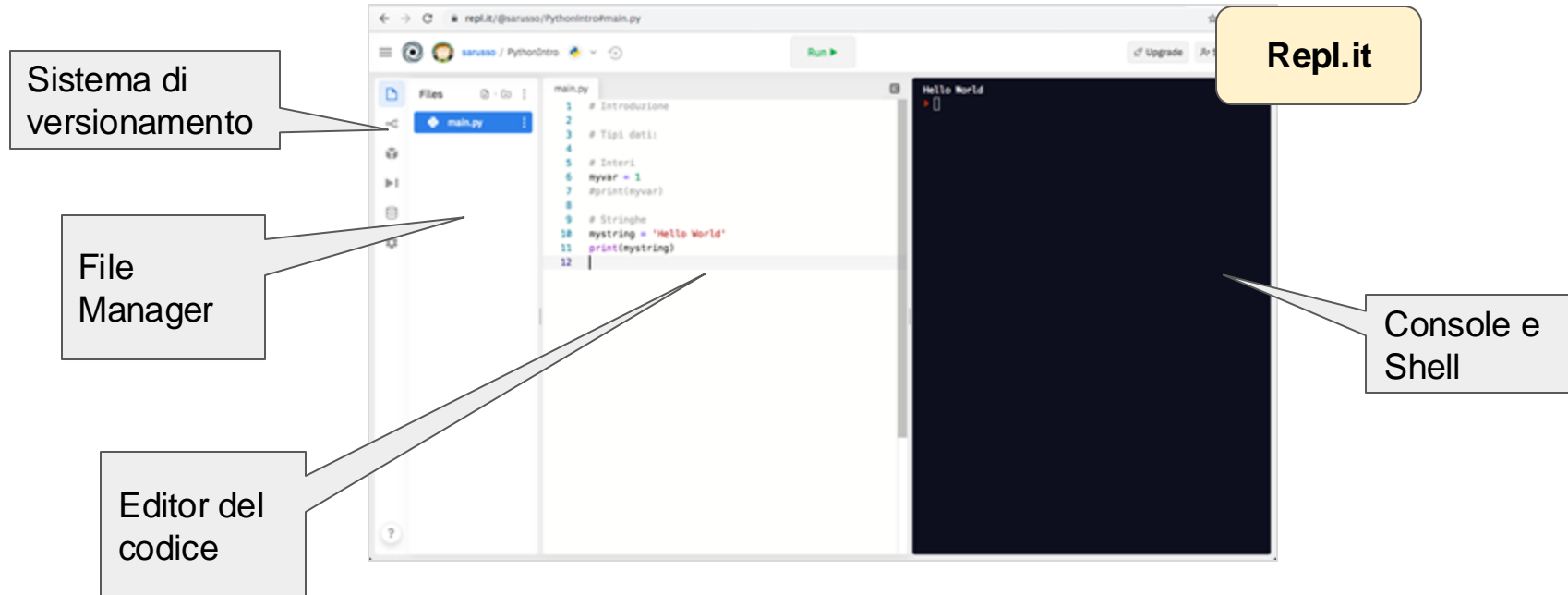
Eclipse

Shell



Strumenti: l'IDE (Integrated Development Environment)

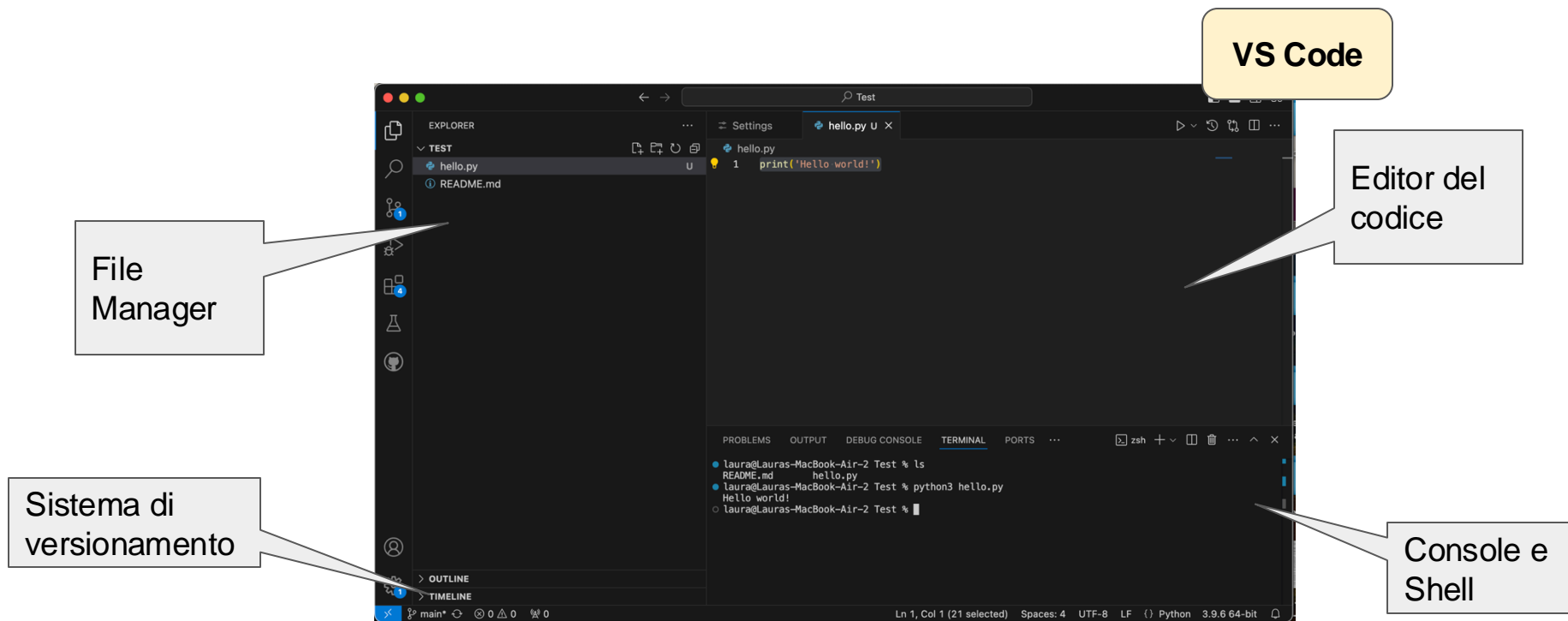
- Il tutto preconfigurato per uno o più linguaggi di programmazione specifici (ad esempio, PyCharm per Python, IntelliJ IDEA per Java).



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 displays 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 guides and categories, with 'PYTHON' and 'Quick Start' highlighted. The main content area features the article title '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, titled 'IN THIS ARTICLE', lists links for creating a project, UI tour, code actions, commands, debugging, and next steps, as well as links to subscribe, ask questions, follow on GitHub, request features, report issues, and watch videos. A footer note mentions leveraging the 'Python profile template'.

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 · ★★★★★
IntelliSense (Pylance), Linting, Debugging (Python Debugger),...
[Disable] [Uninstall] [Switch to Pre-Release Version]

This extension is enabled globally.

DETAILS · FEATURE CONTRIBUTIONS · CHANGELOG · EXTENSION PACK

Python extension for Visual Studio Code

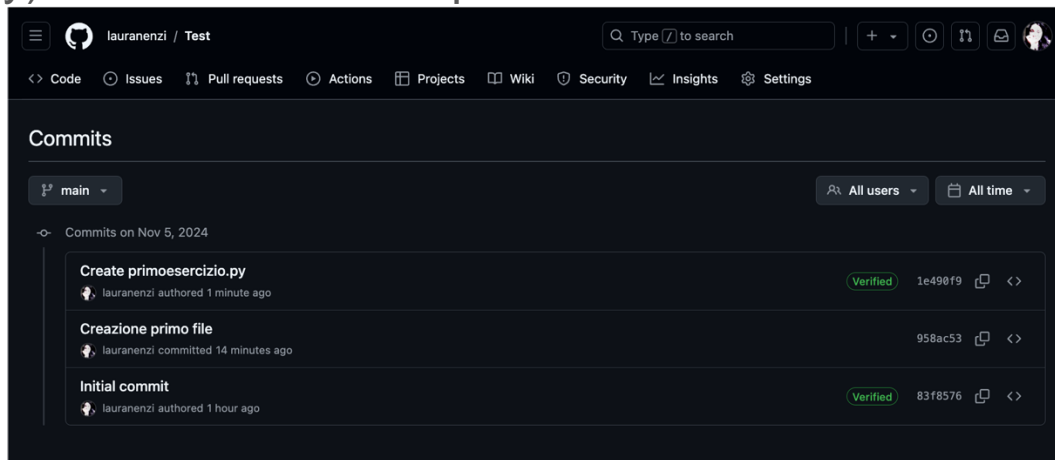
A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the Python language)

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.

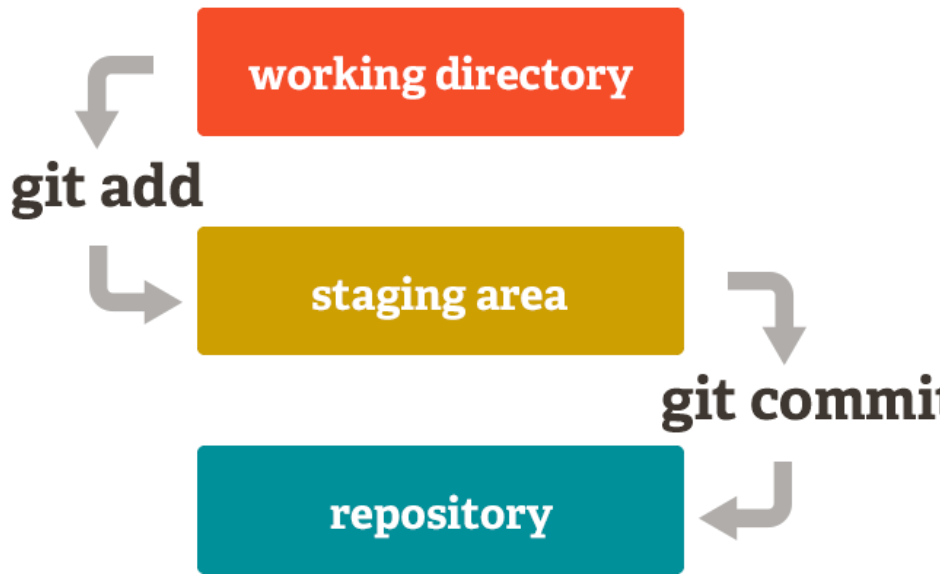


Tutorial di Michele Rispoli (tutor dell'anno scorso):

https://github.com/drpOpZ/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.



Python

Sarà il linguaggio di riferimento del laboratorio

→ *Versione 3, in particolare ≥ 3.6*

Vi verrà spiegato anche al corso programmazione, focalizzandosi sulla Teoria, in particolare quella relativa ai linguaggi ad oggetti.



Cos'è Python?

«Python è un linguaggio di **programmazione** ad **alto livello**, **orientato agli oggetti**, adatto, tra gli altri usi, a sviluppare applicazioni distribuite, scripting, computazione numerica e system testing» (wikipedia)

«È un linguaggio **multi-paradigma** che ha tra i principali obiettivi: dinamicità, semplicità e flessibilità. Supporta il paradigma object oriented, la **programmazione strutturata** e molte caratteristiche di **programmazione funzionale** ...» (wikipedia)

«Python è un linguaggio di programmazione **interpretato**, interattivo, orientato agli oggetti. Include moduli, eccezioni, tipizzazione dinamica...» (python.it)

Cos'è Python?

Linguaggio di programmazione ad alto livello

Multi Paradigma

- Strutturata
- A oggetti
- Funzionale

Interpretato



Compilato

Tipizzazione
Dinamica



Tipizzazione
Statica

Linguaggio di programmazione

È un linguaggio formale (che ha delle regole precise e non è ambiguo) adatto a descrivere algoritmi (procedure che prendono dati in input e producono dati in output)

Questo permette al PC di interpretare un programma (insieme di frasi appartenenti al linguaggio) mediante l'analisi dei simboli e della struttura.

Basso livello

Alto Livello

```
00100111101111011111111111100000
1010111110111111100000000000010100
101011111010010000000000000100000
101011111010010100000000000100100
10101111101000000000000000011000
10101111101000000000000000011100
10001111101011100000000000011100
10001111101110000000000000011000
00000001110011100000000000011001
0010010111001000000000000000001
00101001000000010000000001100101
10101111101010000000000000011100
0000000000000000000111100000010010
00000011000011111100100000100001
0001010000100000111111111110111
10101111101110010000000000011000
00111100000001000001000000000000
10001111101001010000000000011000
0000110000010000000000001101100
00100100100001000000010000110000
10001111101111110000000000010100
00100111101111010000000000100000
0000001111100000000000000001000
000000000000000000001000000100001
```

Codice Macchina a 32 bit

```
.text
.align 2
.globl main

main:
    subu    $sp, $sp, 32
    sw      $ra, 20($sp)
    sd      $a0, 32($sp)
    sw      $0, 24($sp)
    sw      $0, 28($sp)

loop:
    lw      $t6, 28($sp)
    mul     $t7, $t6, $t6
    lw      $t8, 24($sp)
    addu    $t9, $t8, $t7
    sw      $t9, 24($sp)
    addu    $t0, $t6, 1
    sw      $t0, 28($sp)
    ble     $t0, 100, loop
    la      $a0, str
    lw      $a1, 24($sp)
    jal     printf
    move    $v0, $0
    lw      $ra, 20($sp)
    addu    $sp, $sp, 32
    jr      $ra

.data
.align 0
str:
.asciiz "The sum from 0 .. 100 is %d\n"
```

Codice Assembly

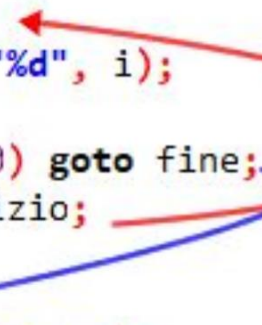
```
a = 5
b = 10
c = a+b
print("la somma è "+str(c))
```

Python

Paradigmi di programmazione

PROGRAMMAZIONE NON STRUTTURATA

```
1  #include <stdio.h>
2  int main() {
3      int i=0;
4
5      inizio:
6      printf("%d", i);
7      i++;
8      if (i>10) goto fine;
9      goto inizio;
10
11     fine:
12     printf("fine");
13 }
```



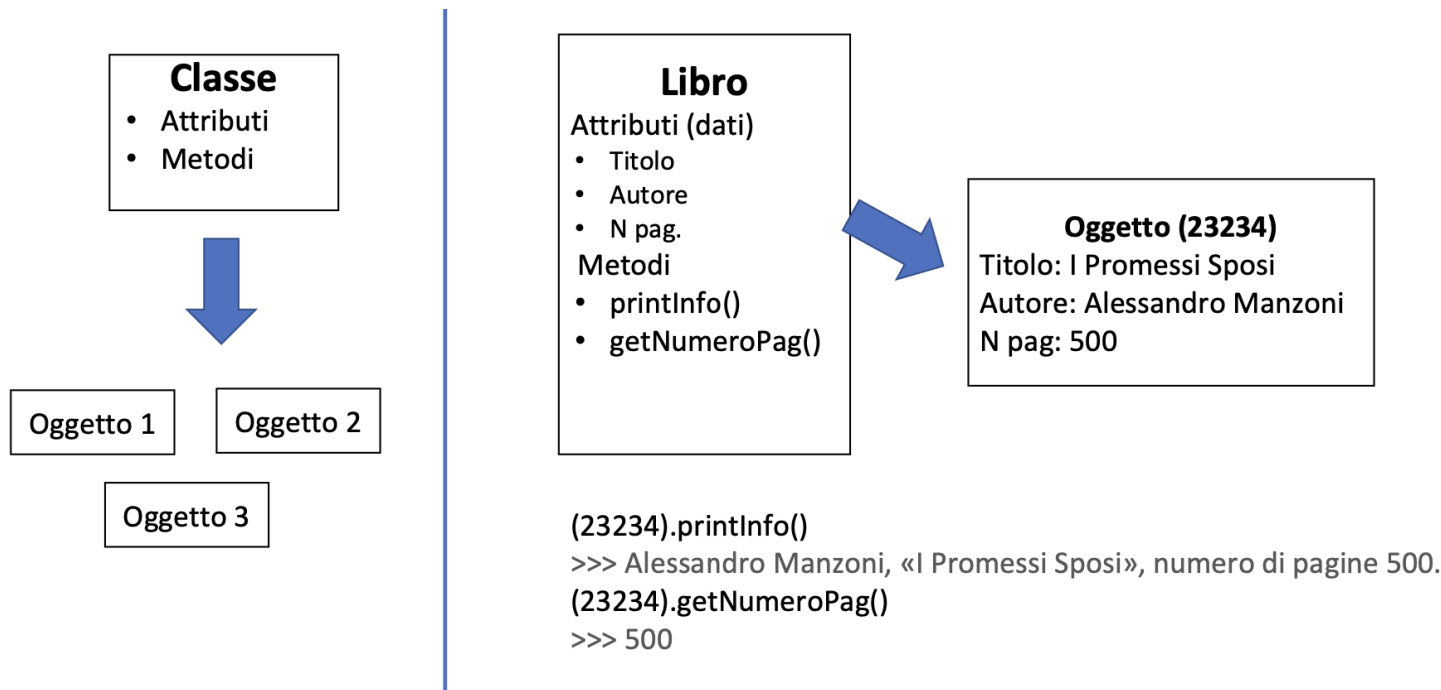
PROGRAMMAZIONE STRUTTURATA

```
1  #include <stdio.h>
2  int main() {
3      int i=0;
4
5      while(i<=10) {
6          printf("%d", i);
7          i++;
8      }
9
10     printf("fine");
11
12 }
13
```

WWW.OKPEDIA.IT

«Spaghetti Programming» o «goto programming»

Linguaggio orientato ad oggetti



Linguaggio funzionale

Paradigma di programmazione, dove il flusso di esecuzione assomiglia a una serie di valutazioni di funzioni matematiche



- Output (Y) è determinato esclusivamente dall' input (X)

Interpretato



Compilato

Python: un linguaggio interpretato

Python è un linguaggio interpretato, non deve essere tradotto in linguaggio macchina come per il c (ovvero, compilato), ma viene eseguito “come sta”

Essendo un linguaggio interpretato, ha anche un interprete interattivo, che potete (e dovrete) usare ogni qualvolta vogliate testare delle cose in rapidità

```
ste@Stes-MacAir:ProgrammingLab (master) $ python
Python 2.7.10 (default, Oct 23 2015, 19:19:21)
[GCC 4.2.1 Compatible Apple LLVM 7.0.0 (clang-700.0.59.5)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+2
4
>>> for item in [1,2,3]: print item
...
1
2
3
>>> █
```

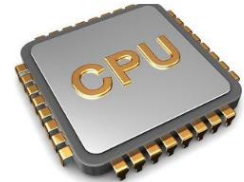
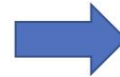

Linguaggio Compilato

```
#include <stdio.h>

int main(void){
    printf("hello, world\n");
}
```



Compilatore

[illegible]

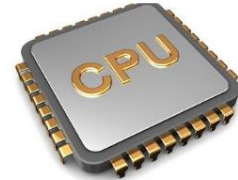
Linguaggio Interpretato

```
>>> print("Hello world!")
```

Interprete

```
>>> print("Hello world!")  
Hello world!
```

```
00100111101111011111111111110000  
10101111101111111000000000010100  
101011111010010000000000010000  
101011111010010100000000010010  
101011111010000000000000011000  
101011111010000000000000011100  
100011111010111000000000011000  
1000111110111000000000011000  
0000001110011100000000011001  
001001011100100000000000000001  
001010010000001000000001100101  
101011111010100000000000011000  
0000000000000001111000010010  
00000011000111111001000010001  
000101000010000111111111110111  
101011111011100100000000011000  
0011110000001000010000000000  
100011111010010100000000011000  
0000110000100000000001101100  
0010010010001000000100011000  
1000111101111100000000010100  
0010011101111010000000010000  
0000001111100000000000010000  
00000000000000001000010001
```



Benevolent dictator for «life»

Guido Van Rossum

- Olandese
- Crea Python nel 1989



Perché creare Python?

Per avere un linguaggio

- semplice, intuitivo e potente
- open source, aperto allo sviluppo condiviso
- facilmente comprensibile, come l'inglese parlato
- ottimo per risolvere problemi quotidiani (degli sviluppatori)

Voleva essere un linguaggio di rottura con il passato,
lievemente provocatorio

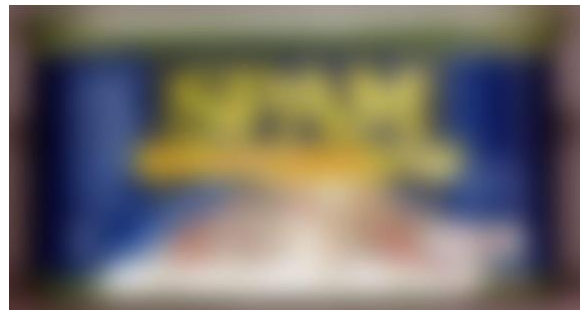


Perché chiamarlo Python?

Guido V.R prende ispirazione da un gruppo comico inglese i Monty Python

«Il programma televisivo fu una rivoluzione — i Monty Python hanno rappresentato quello che i Beatles sono stati per la musica: un punto di partenza e di non ritorno. Il loro umorismo — anarchico, a tratti demenziale, sempre intellettuale, mai volgare, corrosivo, capace di toccare le vette rarefatte dell'assurdo — sfidò tutte le convenzioni comiche dell'epoca, sia in termini di stile che di contenuti»

Corriere della Sera, 22/10/20



Perché chiamarlo Python?

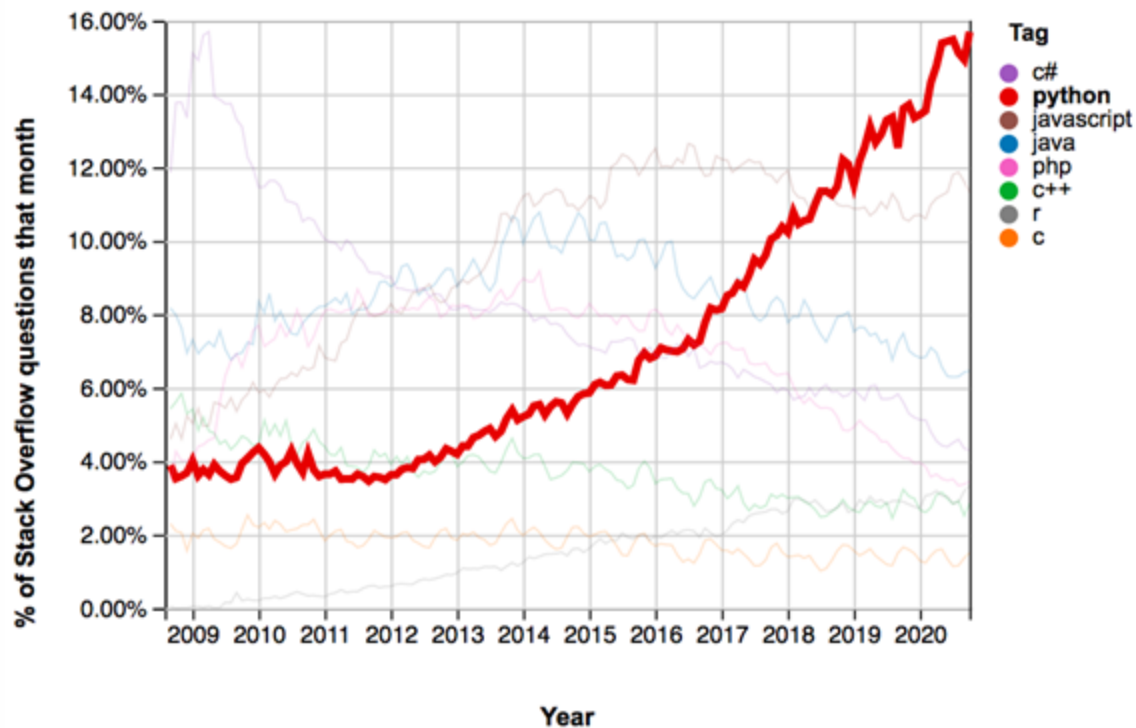
Guido V.R prende ispirazione da un gruppo comico inglese i Monty Python

«Il programma televisivo fu una rivoluzione — i Monty Python hanno rappresentato quello che i Beatles sono stati per la musica: un punto di partenza e di non ritorno. Il loro umorismo — anarchico, a tratti demenziale, sempre intellettuale, mai volgare, corrosivo, capace di toccare le vette rarefatte dell'assurdo — sfidò tutte le convenzioni comiche dell'epoca, sia in termini di stile che di contenuti» Corriere della Sera, 22/10/20



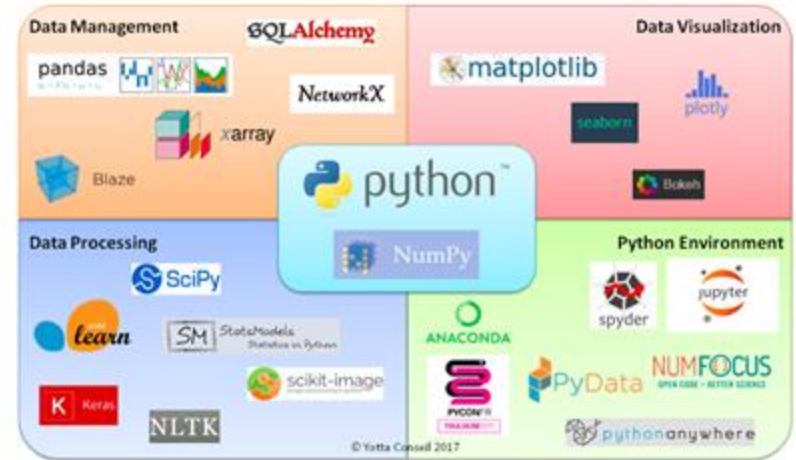
<https://www.youtube.com/watch?v=yckNt0MhTkk>

Perchè Python?



Perchè Python?

- E' un linguaggio semplice, intuitivo e potente
- Facilmente comprensibile, quasi pseudo-codice
- E' il linguaggio della Data Science
- Ha un ecosistema di software per il calcolo scientifico / statistico invidiabile



Pseudo Codice (parentesi)

Lo pseudo-codice sarà il vostro migliore amico, ancora prima di Python.

Fare pseudo-codice vuol dire scrivere, in linguaggio naturale (Italiano/Inglese), quello che dovrebbe fare il programma, con un minimo di sintassi.

Non ci si focalizza sui dettagli nello pseudo-codice!

Ovvero, non ci si focalizza sul **come**, ma sul **cosa** fare.

Pseudo Codice (parentesi)

Esempio: trova i numeri in una lista minori di 5 e stampali

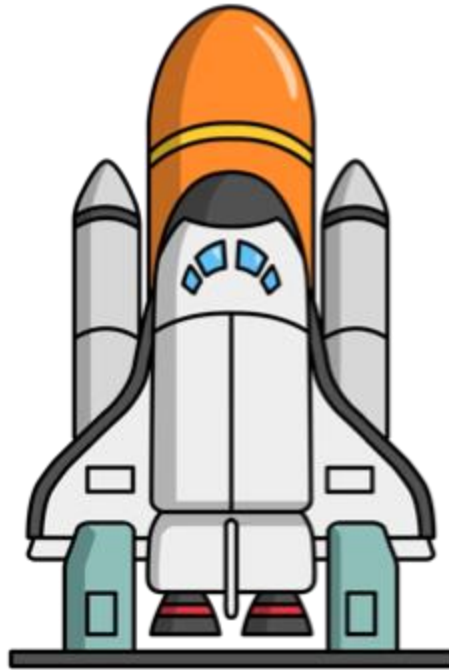
```
data una lista di numeri;  
  
per ogni elemento della lista:  
    se l'elemento è minore di 5:  
        stampa l'elemento
```

Python: un codice minimale

Esempio: trova i numeri in una lista minori di 5 e stampali

```
number_list = [13,12,34,4,51,8,27,18]

for item in number_list:
    if item < 5:
        print(item)
```



Cominciamo

Setup dell'ambiente

- 1) Scaricate VS Code (dovreste averlo già)
- 2) Scaricate l'estensione Python



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](#).

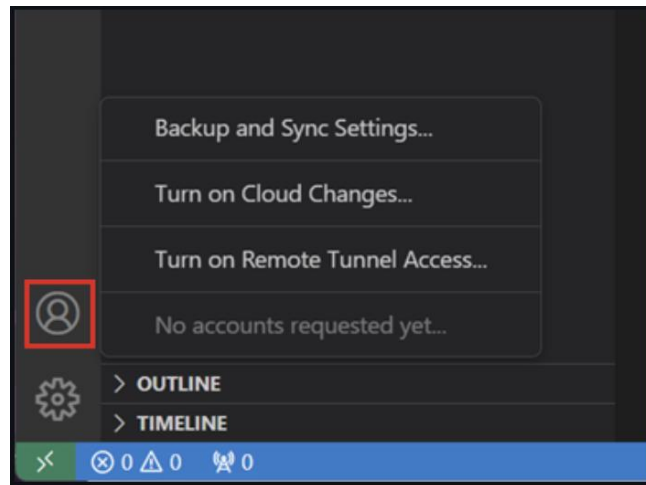
 You are creating a public repository in your personal account.

Create repository

Setup dell'ambiente

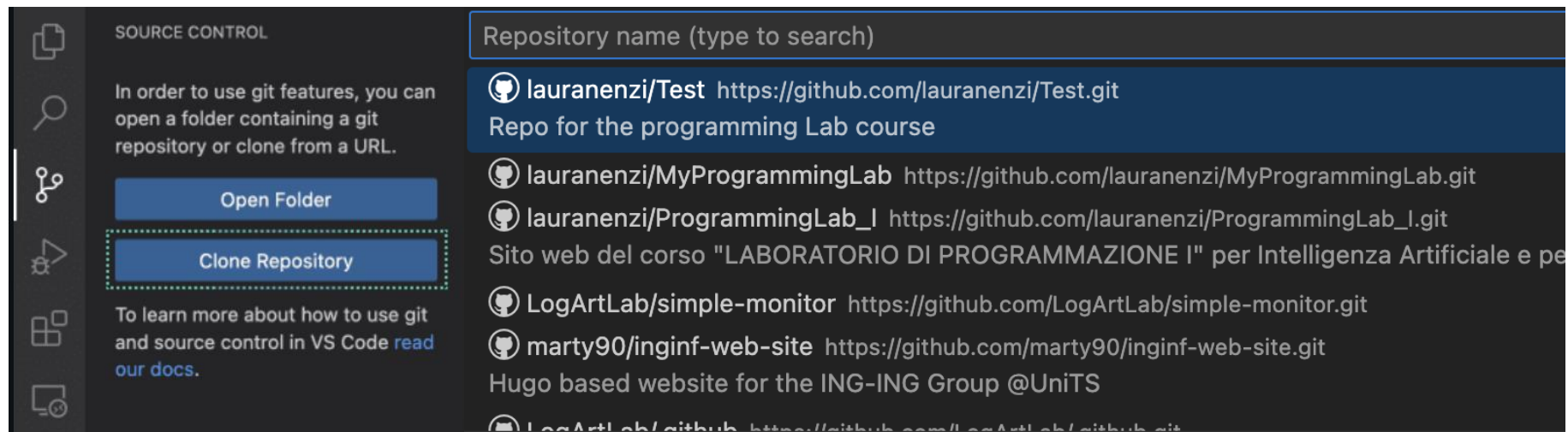
5) Accedete a VS Code col vostro account GitHub nel menu Account in basso a destra della barra Attività. Se Git è mancante, vengono mostrate le istruzioni su come installarlo. Assicuratevi di riavviare VS Code in seguito.

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



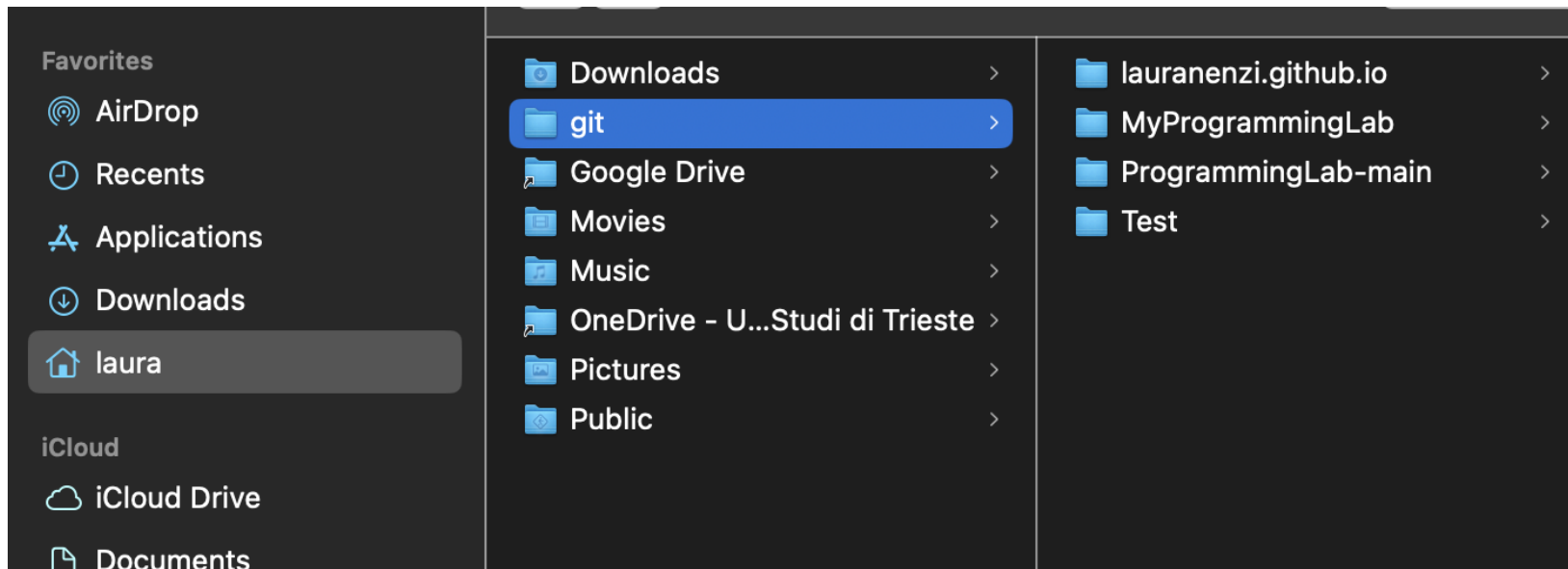
Setup dell'ambiente

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) Create 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.

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**.

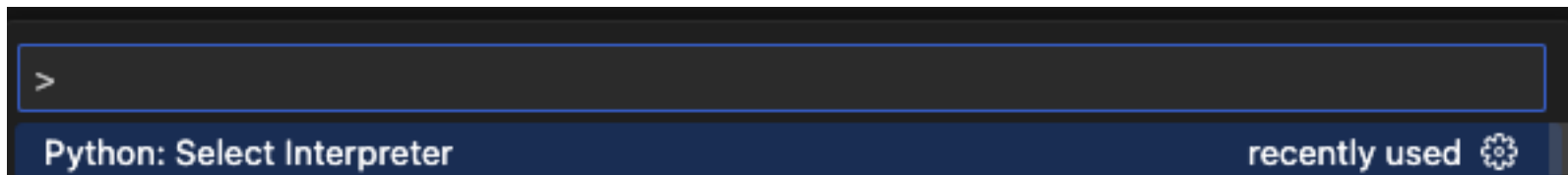
Select an environment type

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

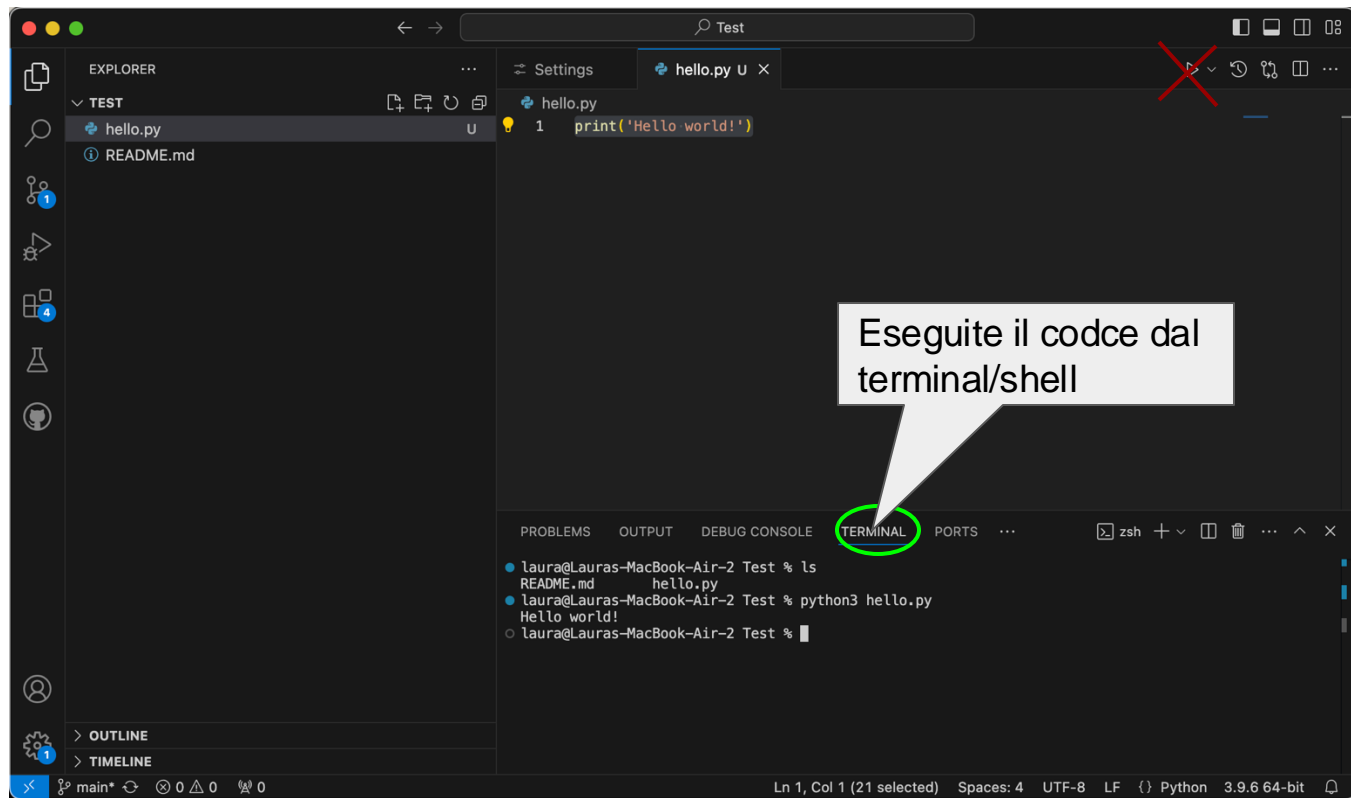
Creiamo adesso uno script "hello.py" con dentro il contenuto:

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

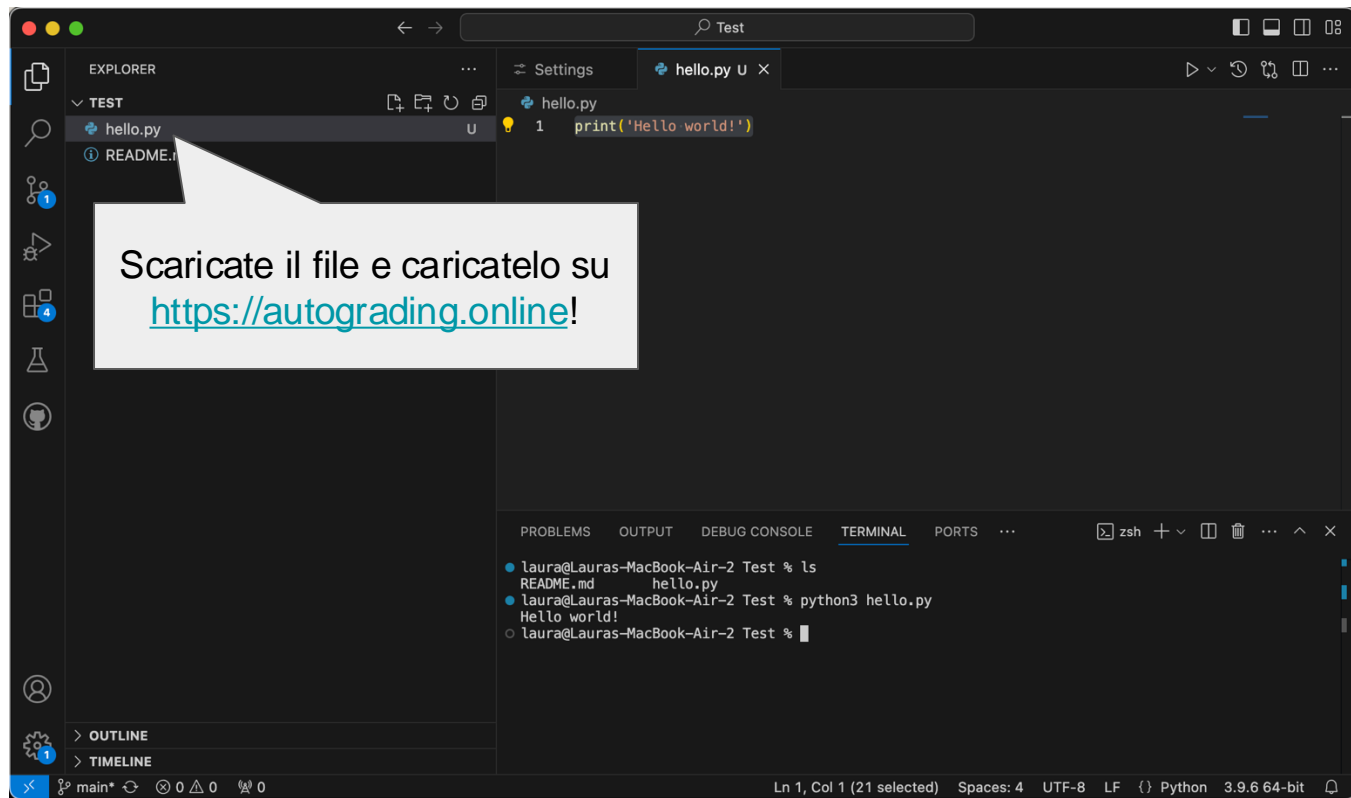
Poi, eseguiamo lo script dalla shell

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

Primi comandi

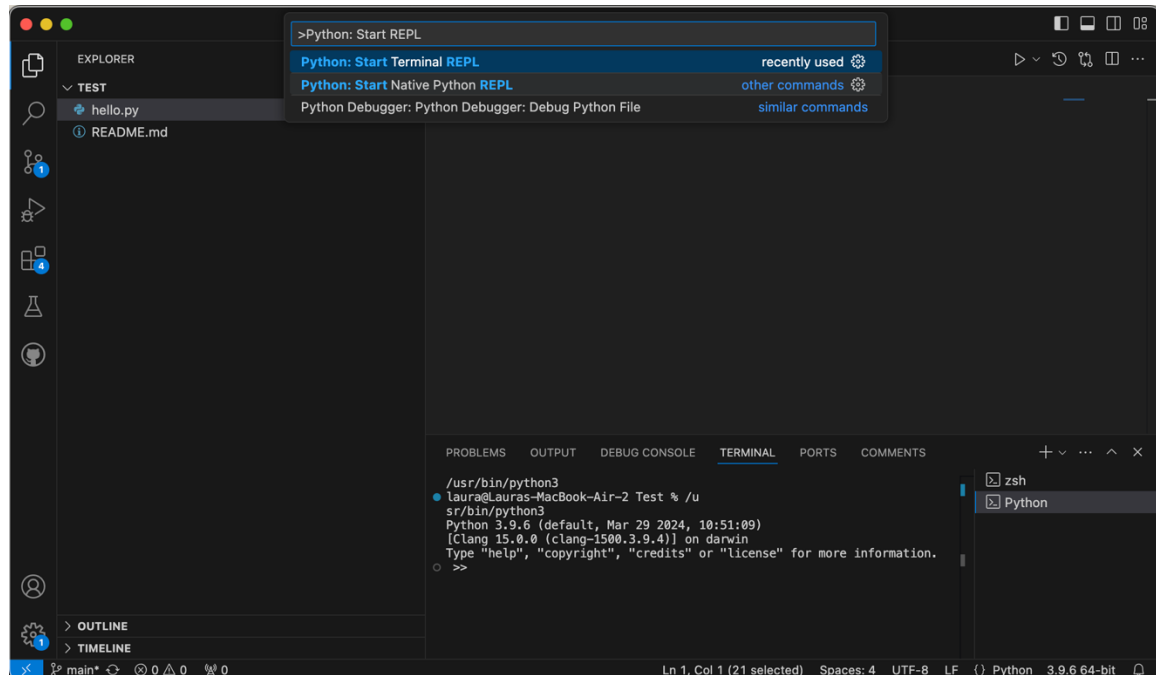


Primi comandi



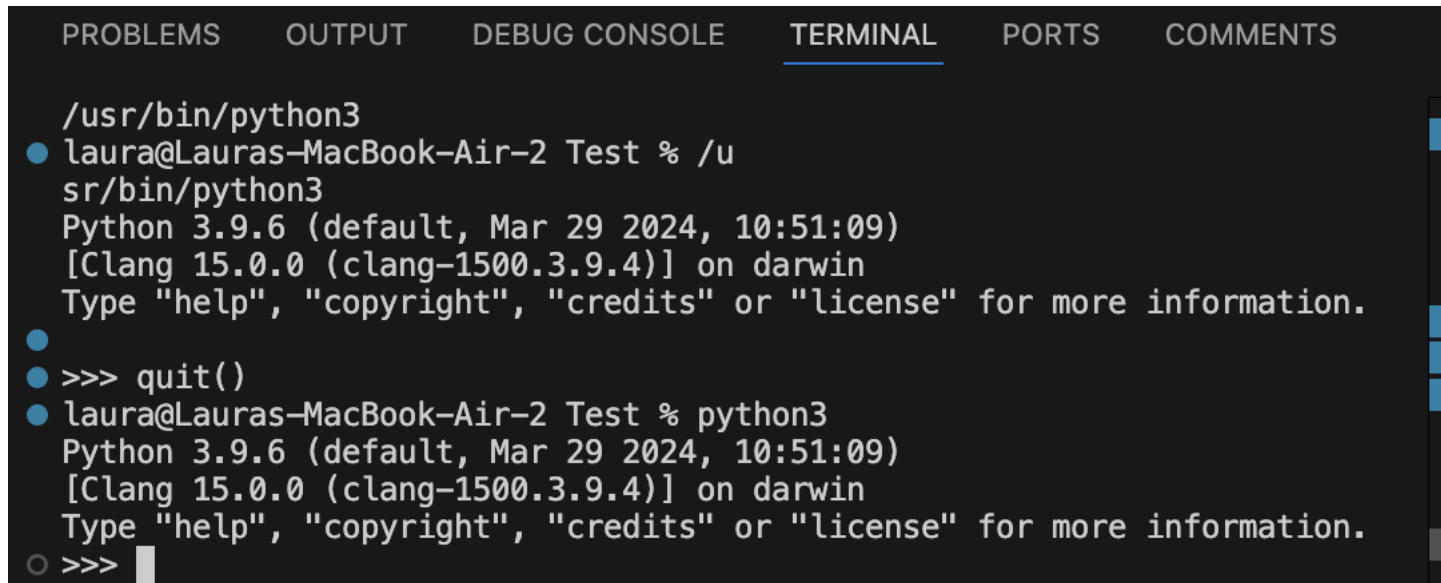
Console

- 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

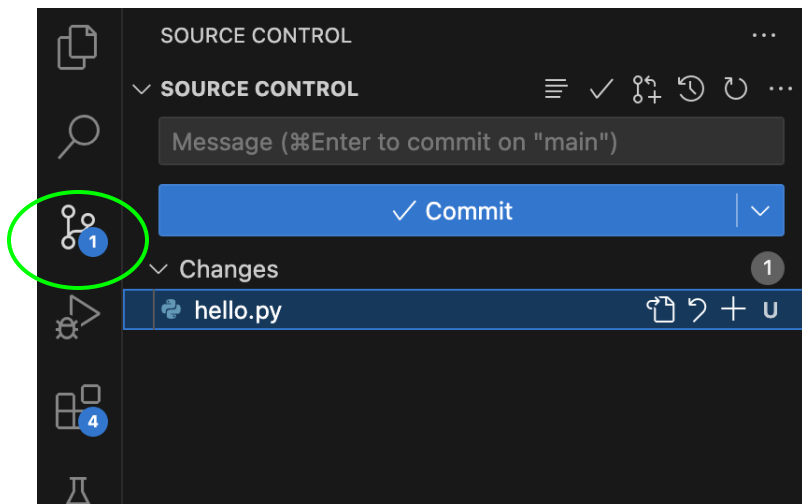


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

/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

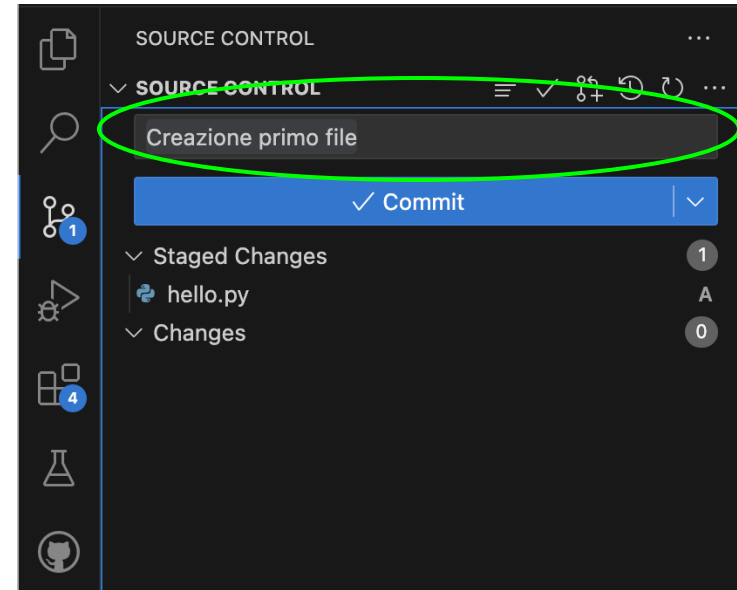
- Quando si seleziona un file, l'editor mostra una vista diff che evidenzia le modifiche del file rispetto al file precedentemente modificato.



- 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.

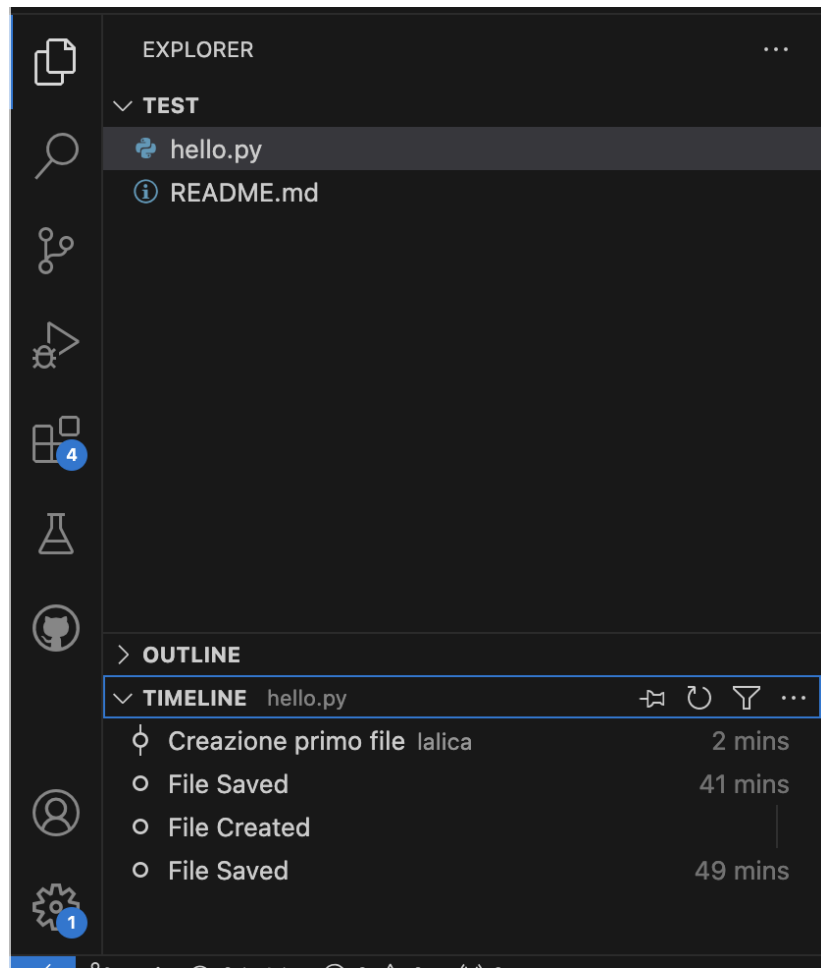
Commit

- 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.



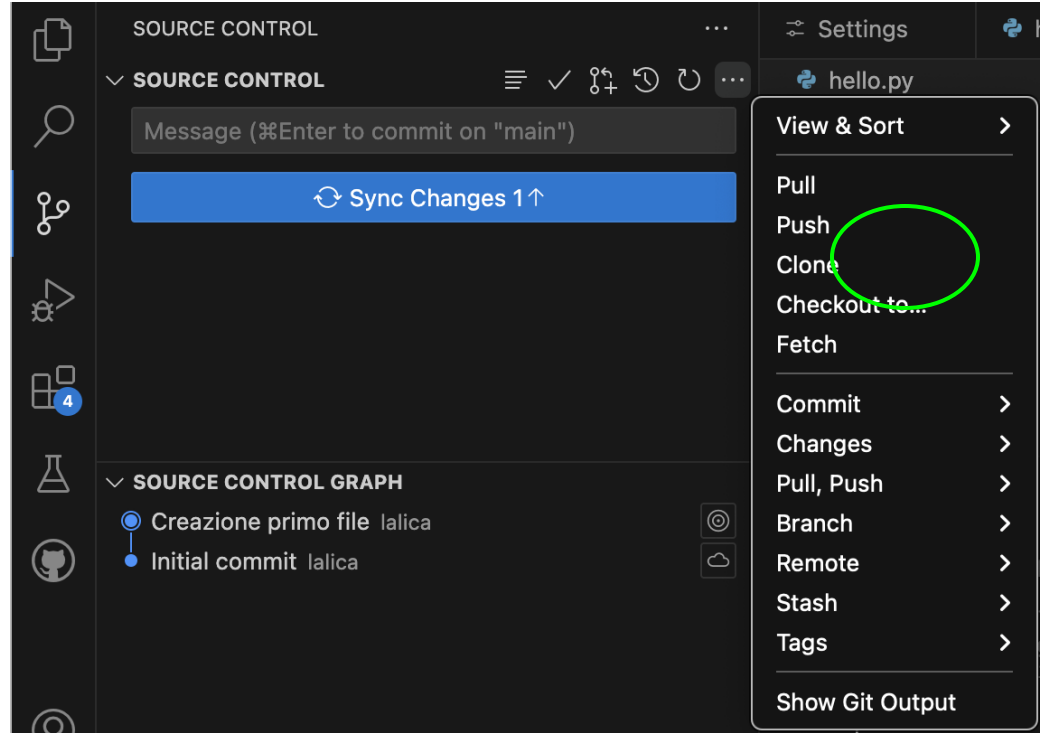
Timeline

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

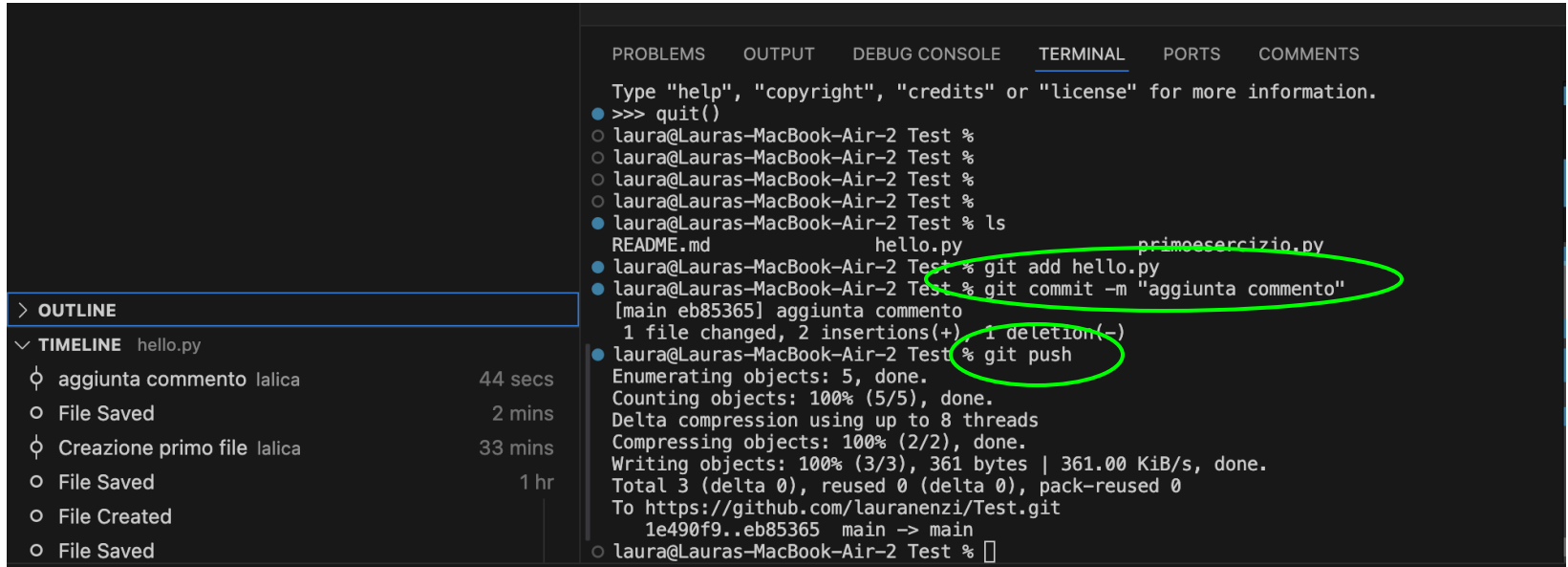


Push and Pull

- 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

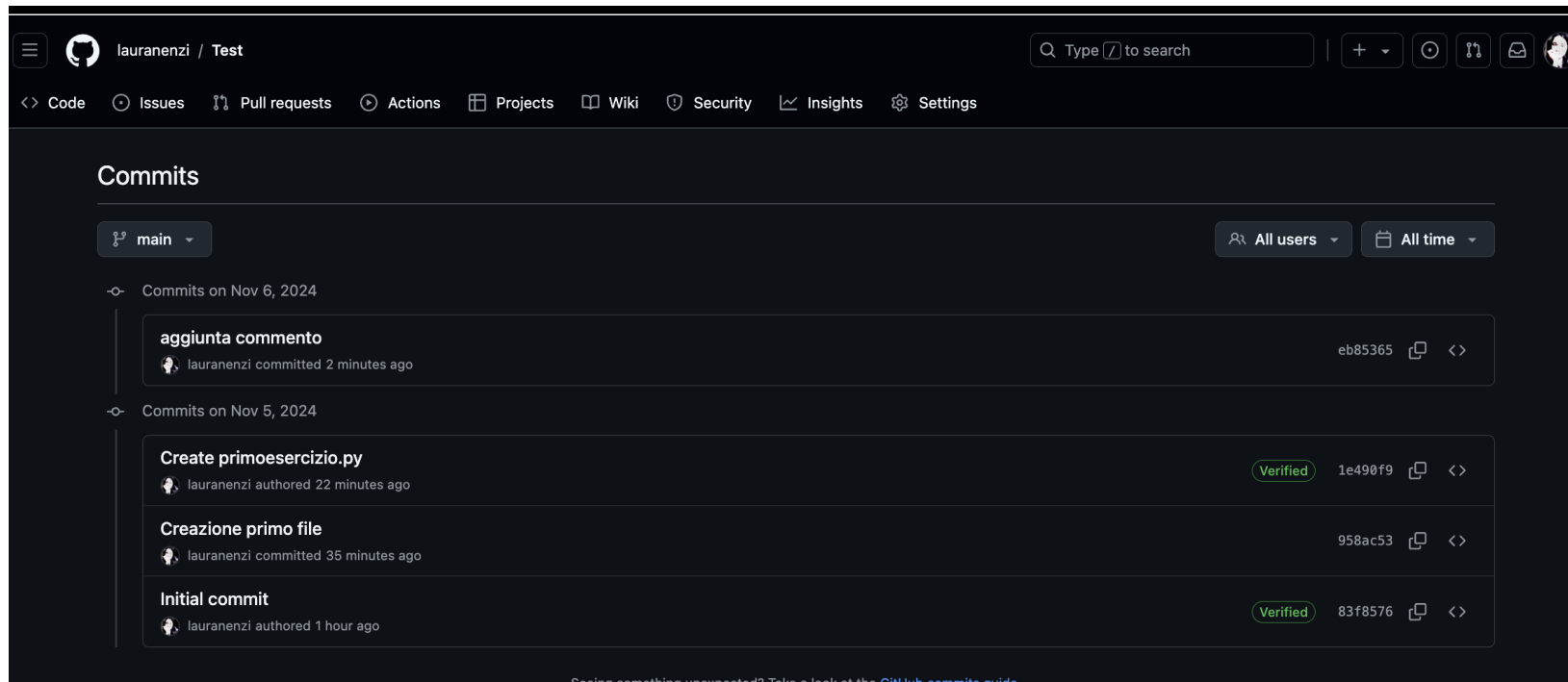


The image shows a screenshot of an IDE interface. On the left, there is an 'OUTLINE' panel with a 'TIMELINE' section for the file 'hello.py'. The timeline shows a sequence of actions: 'aggiunta commento' (44 secs), 'File Saved' (2 mins), 'Creazione primo file' (33 mins), 'File Saved' (1 hr), 'File Created', 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 commands and their results:

```
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 file names 'hello.py' and 'primoesercizio.py' in the 'ls' command output, and the 'git add hello.py' and 'git commit -m "aggiunta commento"' commands in the terminal history.

Cronologia su Github



The screenshot displays the GitHub interface for the repository 'lauranenzi / Test'. The top navigation bar includes a search bar with the text 'Type / to search' and a series of icons for repository management. Below this, a secondary navigation bar lists various repository features: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The main content area is titled 'Commits' and shows a list of recent commits. The first commit, 'aggiunta commento', is dated 'Nov 6, 2024' and was committed 2 minutes ago. The second commit, 'Create primoesercizio.py', is dated 'Nov 5, 2024' and was authored 22 minutes ago. The third commit, 'Creazione primo file', was committed 35 minutes ago. The fourth commit, 'Initial commit', was authored 1 hour ago. Each commit entry includes the commit message, the author's name and profile picture, the commit hash, and icons for viewing the commit details and the associated code changes. A 'Verified' badge is present next to the commit hashes for the second, third, and fourth commits. At the bottom of the page, there is a small text link: 'Explore something unexpected? Take a look at the [GitHub commits guide](#)'.

lauranenzi / Test

Type / to search

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Commits

main

All users All time

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
Verified 1e490f9
- Creazione primo file**
lauranenzi committed 35 minutes ago
958ac53
- Initial commit**
lauranenzi authored 1 hour ago
Verified 83f8576

Explore something unexpected? Take a look at the [GitHub commits guide](#)