Versionamento del codice, Git & GitHub

Ovvero: perché il tuo codice merita meglio di "alieno_VERO_definitivo_v2_QUESTO_SI.py"

💻 III Liceo Scientifico Biella - Scienze Applicate

Python Biella Group

📁 Il problema del "filename hell"

```
alieno.py
alieno_v2.py
alieno_v2_funzionante.py
alieno_v2_funzionante_VERO.py
alieno_v2_funzionante_VERO_definitivo.py
alieno_v2_funzionante_VERO_definitivo_prof_non_guardare.py
```

Domanda esistenziale: Quale versione consegno? 🤔

La tragedia del lavoro di gruppo

Scenario tipico:

- Marco scrive il codice degli alieni
- Sofia aggiunge i punteggi
- Luca modifica... tutto quello che avevano fatto Marco e Sofia
- Risultato: **CAOS ASSOLUTO** 💥

"Ma io avevo già fatto quella parte!" - cit. ogni progetto di gruppo

PBG

 Enter: il Version Control System

Il supereroe che non sapevi di cui avere bisogno

Cos'è un Version Control System?

Un sistema che:

- **Traccia** ogni modifica al codice
- **Ricorda** chi ha fatto cosa e quando
- **Permette** di tornare indietro nel tempo
- **Gestisce** il lavoro in parallelo di più persone

In pratica: La macchina del tempo per il tuo codice! ⏰

PBG

🎮 Git: il boss finale dei VCS

git

- Creato da Linus Torvalds (sì, quello di Linux)
- Lo usano Google, Microsoft, NASA, e... presto voi!
- È come WhatsApp per il codice, ma senza gli status imbarazzanti

PBG

🌟 Git vs il caos

Senza Git	Con Git
50 file con nomi assurdi	1 progetto, infinite versioni
"Chi ha rotto il codice?"	Tracciato ogni singolo byte
Lavoro sovrascritto	Merge intelligente
Backup su chiavetta USB	Cloud infinito

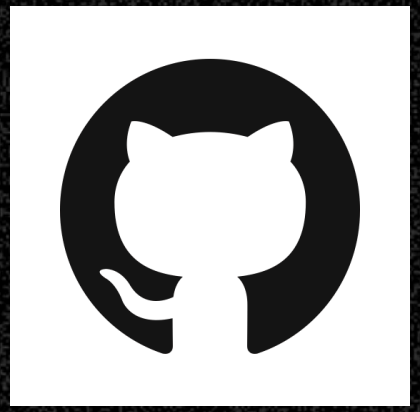
⚔️ SFIDA #1

Trova l'errore nel nome file:

`progetto_pygame_FINALISSIMO_vers3_corretto_ok_davvero.py`

Risposta: tutto. Tutto è l'errore. 😅

GitHub: il social network per nerd



- Git + Hub = dove vive il tuo codice
- Portfolio professionale per sviluppatori
- Dove anche i prof stalkerano i tuoi commit

📝 Creare un account GitHub

Step 1: Vai su github.com

Step 2: Click su "Sign up"

Step 3: Scegli username *SENZA* riferimenti imbarazzanti

- ✅ `mario_rossi_dev`
- ❌ `ilmegagamer2009`
- ❌ `odioinformatica`

Pro tip: Usate l'email della scuola per vantaggi studente!

🎯 Username: scegli con saggezza

Ricordate: questo username potrebbe finire sul vostro CV!

Hall of Shame:

- `xXx_CodeMaster_xXx`
- `programmatore_super_mega`
- `nettonoob123`

Fra 5 anni ringrazierete di aver scelto qualcosa di professionale 💼

⚔️ SFIDA #2

Quale di questi username usereste per candidarvi a Google?

a) `l33t_h4ck3r`

b) `giulia.verdi`

c) `pizza_pasta_mandolino`

Se avete scelto (a) o (c), dobbiamo parlare… 😬

Scaricare un progetto da GitHub

Metodo 1: Il bottone verde (per principianti)

1. Trova il repository
2. Click su "Code" (il bottone verde)
3. "Download ZIP"
4. Estrai e apri in Thonny

È come scaricare musica, ma legale! 🎵

💻 Metodo 2: Git Clone (per pro)

```
git clone https://github.com/username/repository.git
```

In Thonny:

- Strumenti → Apri shell di sistema
- Naviga alla cartella desiderata
- Incolla il comando
- *BOOM!* Progetto scaricato ✨

Il nostro progetto: Colpisci Alieno

Ricordate il gioco della lezione scorsa?

```
import pgzrun

# L'alieno che tutti volevamo colpire durante le verifiche
alieno = Actor('alien')
alieno.pos = 100, 56

def draw():
    screen.fill((0, 0, 255))
    alieno.draw()

pgzrun.go()
```

Ora lo mettiamo su GitHub come dei veri professionisti! 👨‍💻

Creare il tuo primo repository

Step su GitHub:

1. Click sul "+" in alto a destra
2. "New repository"
3. Nome: `colpisci-alieno` (no spazi!)
4. Descrizione: "Il mio primo gioco Python"
5. ✅ "Add a README file"
6. Click "Create repository"

Congratulazioni! Avete appena creato casa per il vostro codice! 🏠

Il README.md: la vetrina del progetto

```
# 🎮 Colpisci Alieno

Gioco sviluppato con PyGameZero durante il corso di IA.

## Come giocare
- Clicca sull'alieno per colpirlo
- Ogni colpo = 1 punto
- Non farti sfuggire l'alieno!

## Requisiti
- Python 3.x
- PyGameZero
```

Il README è come il riassunto del libro: tutti lo leggono per primo!

⚔️ SFIDA #3

Quale README è meglio?

A) "progetto gioco"

B) "# Colpisci Alieno
Gioco Python con PyGameZero. Clicca l'alieno per vincere!"

Se pensate (A), rileggete le slide precedenti 😉

Caricare il codice su GitHub

Metodo GUI (per oggi):

1. Nel tuo repository, click "Add file" → "Upload files"
2. Drag & drop di:
   - `alieno.py` (il tuo codice)
   - cartella `images/` (con alien.png)
   - cartella `sounds/` (se hai suoni)
3. Scrivi un messaggio: "Aggiunto gioco colpisci alieno"
4. Click "Commit changes"

L'arte del commit message

❌ **Messaggi terribili:**

- "fix"
- "roba"
- "asdfghjkl"
- "funziona, non chiedete come"

✅ **Messaggi professionali:**

- "Aggiunta funzione punteggio"
- "Risolto bug movimento alieno"
- "Implementato sistema di livelli"

Scrivete come se il vostro futuro io dovesse capire!

📋 Standard dei progetti GitHub

Ogni progetto dovrebbe avere:

- ✅ `README.md` - la guida del progetto
- ✅ `.gitignore` - file da ignorare (es. `__pycache__`)
- ✅ `LICENSE` - come possono usare il tuo codice
- ✅ Struttura organizzata:

```
colpisci-alieno/
├── README.md
├── alieno.py
├── images/
└── sounds/
```

Il .gitignore: l'amico invisibile

File da NON caricare su GitHub:

```
# Python
__pycache__/
*.pyc
*.pyo

# Thonny
.thonny/

# OS
.DS_Store
Thumbs.db

# File personali
password.txt
```

Git ignora questi file come voi ignorate i messaggi della chat di classe 🙈

🏆 Buone pratiche GitHub

1. **Commit frequenti** - piccoli e significativi
2. **Branch per feature** - (lo vedremo dopo)
3. **Pull prima di push** - sempre aggiornati
4. **Code review** - controllate il codice a vicenda
5. **Issues** - tracciate bug e idee
6. **Documentazione** - README chiaro e aggiornato

👥 Lavorare in gruppo: il workflow

Scenario: Progetto di gruppo su "Colpisci Alieno Pro"

1. **Sofia** aggiunge i livelli
2. **Marco** implementa power-ups
3. **Luca** crea i boss fight
4. Git **merge** tutto insieme magicamente ✨

Niente file sovrascritti, niente drammi, niente lacrime!

🔄 Il ciclo di vita Git

```
Modifica → Stage → Commit → Push
   ↓         ↓        ↓       ↓
Thonny    git add  git commit GitHub
```

Per ora: usiamo l'interfaccia web

Prossimamente: diventerete ninja del terminale 🥷

⚔️ SFIDA FINALE

Missione per casa:

1. Create account GitHub
2. Create repository "colpisci-alieno"
3. Caricate il codice del gioco
4. Scrivete un README decente
5. Inviate il link al prof

Bonus: Chi scrive il commit message più creativo vince... niente, ma ha il nostro rispetto! 🏅

🎓 Recap: cosa abbiamo imparato

- ✅ Git salva dal filename hell
- ✅ GitHub = social per sviluppatori
- ✅ Version control = lavoro di gruppo senza drama
- ✅ README = prima impressione del progetto
- ✅ Commit message = diario del codice
- ✅ .gitignore = privacy per il codice

🚀 Prossimi passi

Nelle prossime lezioni:

- Git da terminale in Thonny
- Branch e merge
- Pull request
- Gestione conflitti
- Workflow professionali
- GitHub Actions (CI/CD)

Diventerete talmente bravi che vorrete versionare anche i compiti di letteratura! 📚

💡 Pro tips da sviluppatori veri

1. **Commit atomici** - una modifica = un commit
2. **Testare prima di pushare** - funziona sul TUO PC?
3. **Non commitare password** - seriously, DON'T
4. **Backup != Version Control** - sono cose diverse
5. **Leggete la documentazione** - GitHub ha guide ottime

"*La differenza tra un junior e un senior è che il senior fa meno casino con Git*" - Antico proverbio informatico

🎮 Esercizio pratico

Mini-progetto: Migliorate "Colpisci Alieno"

Idee:

- Aggiungete contatore punteggio
- Alieno che si muove random
- Timer di gioco
- Suoni quando colpite
- Livelli di difficoltà

Ogni miglioramento = un commit!

🔗 Risorse utili

- **Git:** git-scm.com
- **GitHub Guide:** guides.github.com
- **Markdown:** markdownguide.org
- **PyGameZero:** pygame-zero.readthedocs.io

Pro tip: GitHub ha student pack con software gratis!
education.github.com

🎯 Quiz veloce!

1. Perché usiamo Git? *Per non impazzire con le versioni*
2. Cos'è un commit? *Uno snapshot del codice*
3. A cosa serve il README? *Spiegare il progetto*
4. Come scarico un repo? *Clone o download ZIP*
5. Perché il .gitignore? *Non caricare file inutili*

Se avete risposto correttamente: pronti per GitHub! 🎉

Se no: rileggete le slide (è permesso!) 📖

😅 Meme corner

Il ciclo di vita dello sviluppatore:

```
git commit -m "funziona"
[2 minuti dopo]
git commit -m "ok ora funziona davvero"
[5 minuti dopo]
git commit -m "non chiedetemi perché ma ora sì"
```

Noi tutti siamo stati lì! 😂

PBG

La filosofia Git

> *"Commita presto, commita spesso, scrivi buoni messaggi"*
>
> *"Un repository senza README è come una pizza senza mozzarella"*
>
> *"Testare è dubitare, committare è credere"*

— *Antica saggezza GitHub*

🎊 Congratulazioni!

Ora sapete:

- ✨ Cos'è il version control
- 🌟 Come usare GitHub
- 🚀 Come organizzare progetti
- 💪 Come lavorare in team
- 🎮 Come essere sviluppatori professionali

Il vostro viaggio nel mondo del version control è appena iniziato!

🎯 Domande?

Ricordate:

Non esistono domande stupide, solo commit senza messaggio!

📧 Homework reminder

Da fare per la prossima lezione:

1. ✅ Account GitHub creato
2. ✅ Repository "colpisci-alieno" online
3. ✅ README.md scritto
4. ✅ Codice del gioco caricato
5. ✅ Link inviato al prof

Bonus challenge: Aggiungete una feature al gioco!

🚀 Buon coding!

"May the commits be with you" ⭐

Prossima lezione: Git in Thonny e comandi avanzati

📎 Cheat sheet finale

```
# Clonare un repository
git clone <url>

# Stato del repository
git status

# Aggiungere file
git add .

# Committare
git commit -m "messaggio"

# Pushare su GitHub
git push
```

Salvate questa slide, vi tornerà utile! 💾

Come trovare l'URL da clonare 🔗

1. Vai sulla pagina del repository
2. Click sul pulsante verde "**Code**"
3. Copia l'URL HTTPS
4. Apri il terminale
5. Naviga dove vuoi salvare il progetto
6. `git clone [URL]`

Pro tip: Il progetto viene scaricato in una cartella con il nome del repo

Esempio pratico 💻

Voglio scaricare un progetto di ML:

```
cd Documenti/Progetti
git clone https://github.com/scikit-learn/scikit-learn.git
cd scikit-learn
```

Boom! Hai appena scaricato scikit-learn completo di tutta la storia di sviluppo! 🎉

Creare un progetto su GitHub 🆕

Metodo 1: Partire da GitHub

1. Click su "+" in alto a destra → "New repository"
2. Scegli un nome (descrittivo!)
3. Descrizione (opzionale ma utile)
4. Public o Private?
5. Aggiungi README? (consigliato: ✅)
6. Aggiungi .gitignore? (per Python: ✅)
7. Click su "Create repository"

Public vs Private 🔓🔒

Public: Visibile a tutti

- ✅ Ottimo per portfolio
- ✅ Altri possono imparare dal tuo codice
- ✅ Puoi ricevere contributi
- ❌ Tutti vedono anche i tuoi errori (ma va bene!)

Private: Solo tu (e chi inviti)

- ✅ Progetti personali
- ✅ Compiti di scuola (finché non li consegni)
- ❌ Non aiuta il portfolio

README.md: La tua vetrina

Il file README è la **homepage** del tuo progetto:

```
# Nome Progetto

## Descrizione
Breve spiegazione di cosa fa

## Come usarlo
Istruzioni per l'uso

## Tecnologie
- Python 3.12
- NumPy, Pandas

## Autore
Il tuo nome
```

SFIDA #5: Crea il README perfetto! 📝

Quale di questi README è meglio?

A) `# progetto` (solo questo)

B) Un saggio di 10 pagine sulla storia dell'informatica

C) Titolo, descrizione chiara, istruzioni, tech stack

D) Solo emoji: 🔥🚀💯

Risposta: C (ma qualche emoji non guasta mai 😉)

Metodo 2: Partire dal tuo PC 💻

Se hai già un progetto in locale:

```
# Nella cartella del tuo progetto
git init
git add .
git commit -m "First commit"

# Collega a GitHub (crea prima il repo vuoto su GitHub)
git remote add origin https://github.com/tuo-user/tuo-repo.git
git push -u origin main
```

I comandi Git essenziali ⌨️

```
git status           # Situazione attuale
git add file.py      # Prepara un file
git add .            # Prepara tutto
git commit -m "msg" # Salva checkpoint
git push             # Invia online
git pull             # Scarica aggiornamenti
git log              # Storia dei commit
```

Regola d'oro: `git status` è il tuo migliore amico!

Fork e Pull Request 🍴

Il vero potere della collaborazione open source!

Scenario: Trovi un progetto figo su GitHub, vuoi migliorarlo ma non sei nel team. Che fai?

Soluzione: Fork + Pull Request!

Cos'è una Fork? 🍴

Una **fork** è la tua copia personale di un repository altrui.

Come fare:

1. Vai sul repository che ti interessa
2. Click su "Fork" in alto a destra
3. GitHub crea una copia nel TUO account
4. Ora puoi modificarla liberamente!

Analogia: È come fotocopiare gli appunti di un compagno per aggiungerci le tue note

Workflow completo con Fork 🔄

1. Fork del progetto originale
   ↓
2. Clone della TUA fork sul tuo PC
   ↓
3. Crei un branch per le modifiche
   ↓
4. Fai le modifiche e commit
   ↓
5. Push sulla TUA fork
   ↓
6. Apri una Pull Request verso l'originale

Esempio pratico di Fork 🛠️

Vuoi aggiungere una funzione al progetto `awesome-ml` di `prof_data`:

```
# 1. Fork su GitHub (click sul bottone)

# 2. Clona LA TUA fork
git clone https://github.com/TUO-USER/awesome-ml.git
cd awesome-ml

# 3. Crea un branch
git checkout -b miglioria-validazione

# 4. Fai modifiche, poi:
git add .
git commit -m "Aggiunta validazione dati"
git push origin miglioria-validazione
```

Cos'è una Pull Request (PR)? 🎁

Una **Pull Request** è una richiesta di integrare le tue modifiche nel progetto originale.

È come dire:

> *"Ehi, ho migliorato il tuo codice! Vuoi dare un'occhiata e magari includerlo?"*

Il proprietario può:

- ✅ Accettarla (merge)
- 💬 Chiedere modifiche
- ❌ Rifiutarla (capita, non piangere)

Come aprire una Pull Request 📤

Dopo aver pushato sul tuo branch:

1. Vai su GitHub (sul TUO fork)
2. Vedrai un banner "Compare & pull request"
3. Click! Si apre una pagina
4. Scrivi un titolo CHIARO
5. Descrivi cosa hai cambiato e perché
6. Click su "Create pull request"

Boom! 🎉 Hai contribuito a un progetto open source!

Anatomia di una buona PR 📋

```
# Titolo
Aggiungi validazione input nel modulo dataset

# Descrizione
- Aggiunta funzione validate_input()
- Controlla tipo e range dei dati
- Aggiunge test unitari
- Risolve issue #42


## Come testare
1. Esegui test dataset.py
2. Prova con dati non validi
```

Pro tip: Sii specifico! Il maintainer ringrazierà.

SFIDA #6: PR Roulette! 🎰

Quale PR ha più probabilità di essere accettata?

A) "fixed stuff" (nessuna descrizione)

B) "Corretto bug validazione + aggiornato README con esempi (issue #23)"

C) "GUARDAMI HO RISCRITTO TUTTO!!1!"

D) "prova" (commit fatto per sbaglio)

Risposta: B (chiara, descrittiva, professionale)

Branch: lavorare in parallelo 🌳

I **branch** permettono di sviluppare feature diverse contemporaneamente:

Ogni branch è indipendente finché non lo unisci (merge)!

Comandi Branch essenziali 🌿

```
# Crea e passa a nuovo branch
git checkout -b nome-feature

# Lista branch
git branch

# Passa a branch esistente
git checkout main

# Merge (da dentro main)
git merge nome-feature

# Elimina branch (dopo merge)
git branch -d nome-feature
```

SFIDA FINALE: Debug Story! 🐛

È venerdì sera. Devi consegnare lunedì. Il codice è rotto. Cosa fai?

A) Panico totale

B) `git log` → trovi l'ultimo commit funzionante → `git checkout [hash]`

C) Ricominci da zero

D) Fingi malattia lunedì

Risposta: B (Git ti salva SEMPRE)

Situazioni comuni e soluzioni SOS

"Ho fatto commit di cose sbagliate!"

```
git reset HEAD~1  # Annulla ultimo commit (mantieni modifiche)
```

"Ho modifiche che non voglio commitare"

```
git stash         # Metti da parte
git stash pop     # Recupera dopo
```

"Il mio collega ha pushato, io ho modifiche locali"

```
git pull --rebase # Sincronizza intelligentemente
```

Best practices da ricordare 🏆

- ✅ Commit spesso, messaggi chiari
- ✅ Usa branch per ogni feature
- ✅ Pull prima di pushare (in team)
- ✅ Non commitare password o file enormi
- ✅ README sempre aggiornato
- ✅ .gitignore configurato (file da ignorare)

- ❌ Mai `git push --force` su `main` condiviso
- ❌ Commit con msg tipo "asdf" o "test"

.gitignore: cosa NON versionare 🚫

File da NON includere nel repository:

```
# Python
__pycache__/
*.pyc
venv/

# Dati sensibili
.env
config.ini
passwords.txt

# File grandi
*.csv  # (meglio usare Git LFS)
dataset/
```

GitHub offre template .gitignore per ogni linguaggio!

Comandi Git Cheat Sheet 📝

```
git init                    # Inizializza repo
git clone [url]             # Scarica repo
git status                  # Situazione
git add [file]              # Stage file
git commit -m "msg"         # Commit
git push                    # Carica online
git pull                    # Scarica aggiornamenti
git branch                  # Lista branch
git checkout -b [nome]      # Nuovo branch
git merge [branch]          # Unisci branch
```

Pro tip: Stampa questo slide e tienilo vicino! 📌

Risorse utili 🔗

Documentazione:

- git-scm.com/doc - Documentazione ufficiale
- docs.github.com - Guide GitHub

Tutorial interattivi:

- learngitbranching.js.org - Visualizza Git!
- GitHub Skills - Tutorial pratici

Cheat Sheet:

- education.github.com/git-cheat-sheet-education.pdf

GitHub per studenti 🎓

GitHub Student Developer Pack:

- Account Pro gratis
- Tanti tool gratuiti
- Hosting gratuito
- E molto altro!

Come ottenerlo:

1. Vai su education.github.com
2. Verifica con email scolastica
3. Profit! 🎉

Progetti pratici per esercitarsi 💪

Livello Base:

1. Crea repo per esercizi di Python
2. Committa ogni esercizio
3. Condividi con compagni

Livello Medio:

1. Fork un progetto open source piccolo
2. Correggi un typo nel README
3. Apri la tua prima PR!

Livello Avanzato:

1. Progetto di gruppo con branch
2. Code review reciproche

L'importanza della Community 🌍

GitHub non è solo codice, è **social networking**:

- ⭐ Star ai progetti che ti piacciono
- 👁️ Watch per seguire aggiornamenti
- 🐛 Apri issue per bug o richieste
- 💬 Partecipa alle discussioni
- 🤝 Contribuisci a progetti open source

Il tuo profilo GitHub è il tuo portfolio!

Error comuni e come risolverli 🔧

"fatal: not a git repository"

→ Non sei in una cartella Git, fai `git init`

"Your branch is ahead of 'origin/main'"

→ Hai commit locali, fai `git push`

"Merge conflict"

→ Git non sa come unire modifiche, devi scegliere manualmente

"Permission denied"

→ Problemi di autenticazione, verifica credenziali

Git GUI: Alternative grafiche 🖱️

Se il terminale ti spaventa (all'inizio è normale):

GitHub Desktop - Facile e ufficiale

GitKraken - Potente e bello graficamente

VS Code - Integrato nell'editor

Ma ricorda: Imparare i comandi base è FONDAMENTALE!

Le GUI sono comode, ma sapere cosa fanno è essenziale.

SFIDA MEGA FINALE! 🏆

Quiz rapido - vero o falso?

1. Git salva solo le differenze tra commit → **VERO**
2. Posso cancellare un commit dopo il push → **COMPLICATO**
3. Fork e branch sono la stessa cosa → **FALSO**
4. GitHub è l'unico servizio di hosting Git → **FALSO**
5. Un buon programmatore non fa mai errori → **FALSISSIMO**

Se hai risposto tutto giusto: congratulazioni! 🎓

Esercizio per la prossima volta 📚

Homework pratico:

1. Crea un account GitHub
2. Crea un repository "esercizi-python"
3. Aggiungi un file README.md
4. Carica un tuo esercizio di Python
5. Fai almeno 3 commit con messaggi chiari
6. (Bonus) Trova un progetto interessante e fai una fork

Prossima lezione: Rivediamo insieme i vostri repository! 🔍

Conclusioni: Perché Git cambierà la tua vita 🚀

- ✨ **Mai più perderai codice**
- ✨ **Lavoro di gruppo sarà un piacere** (quasi)
- ✨ **Portfolio professionale pronto**
- ✨ **Parteciperai alla community open source**
- ✨ **Skill richiesta da TUTTE le aziende tech**

In breve: Git non è "una cosa in più da imparare", è **LA** cosa da imparare.

Pensiero finale 💭

> *"Git è difficile solo finché non lo usi. Poi diventa difficile vivere senza."*
>
> *— Ogni programmatore, ever*

Remember:

- Tutti hanno rotto qualcosa con Git
- Tutti hanno perso ore su un merge conflict
- Tutti hanno fatto `git push --force` e si sono pentiti
- **E tutti sono ancora qui a raccontarlo** 😄

Domande? 🙋

Ricorda: L'unica domanda stupida è quella non fatta!

(Ok, forse "Posso usare Git per versionare le mie foto delle vacanze?" è un po' stupida... ma tecnicamente potresti!)

Grazie per l'attenzione! 🎉

Ora andate e committate con fierezza!

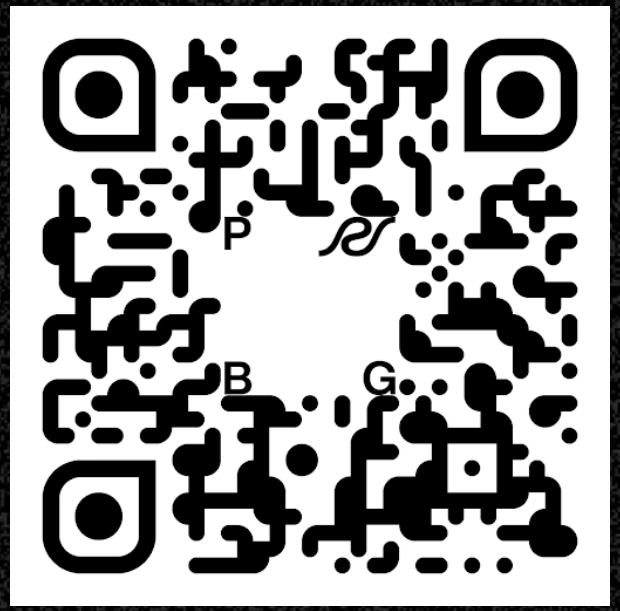
```
git commit -m "Ho imparato Git! 🚀"
git push origin main
```

Contatti e risorse:

- Repository delle slides: [github.com/...]
- Materiale extra: [...]

"May the Git be with you!" ⭐

Grazie per l'attenzione...



> *"C'è sempre qualcosa da imparare per migliorarci e crescere...**insieme!**"*