

Configurazione PyCharm e Anaconda. Introduzione al controllo versione Git

Introduzione all'Ingegneria Finanziaria
Seminario Professionalizzante

Vincenzo Eugenio Corallo, Ph. D

Sapienza Università di Roma

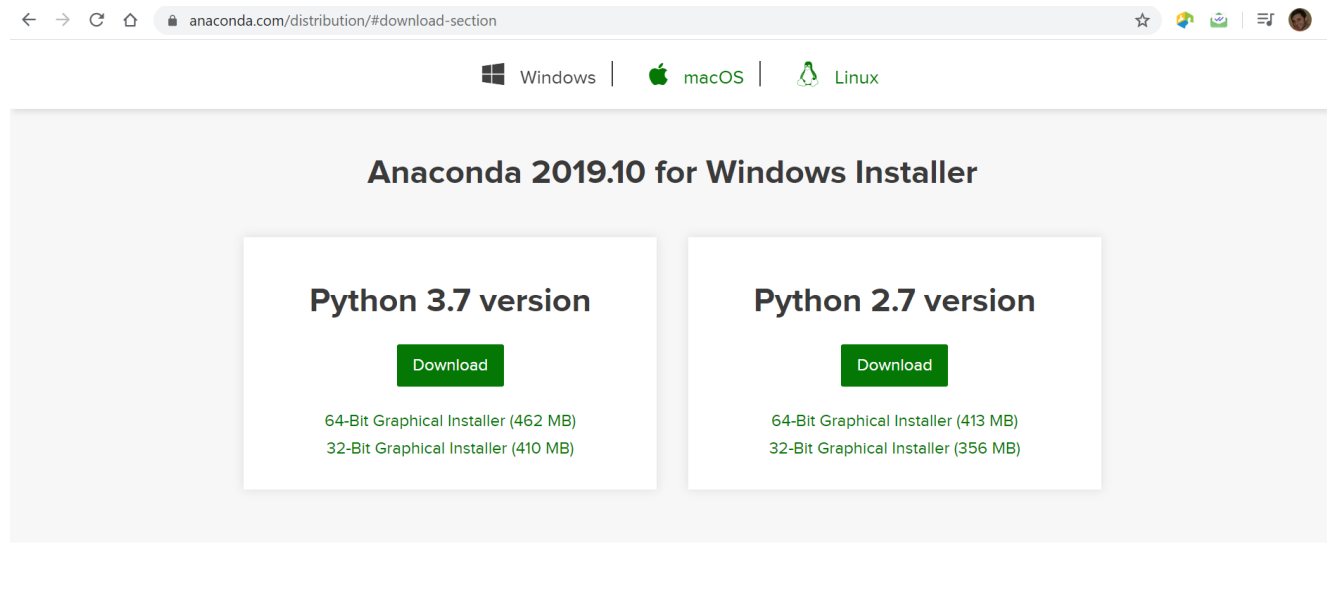
vincenzo.corallo@uniroma1.it

Cos'è Anaconda?

- Anaconda è probabilmente la più usata distribuzione *open-source* di Python;
- Anaconda include numerosi moduli gratuiti sviluppati per l'algebra lineare (ad es. **NumPy**), calcolo scientifico (ad es. **SciPy**), il plotting (ad es. **PyPlotLib**) data science (ad es. **Pandas**) e il machine learning (ad. es **SciKitLearn**);
- Anaconda include il **Jupyter Notebook**, usatissimo ambiente integrato *browser-based* per lo sviluppo di codice in svariati linguaggi di programmazione.

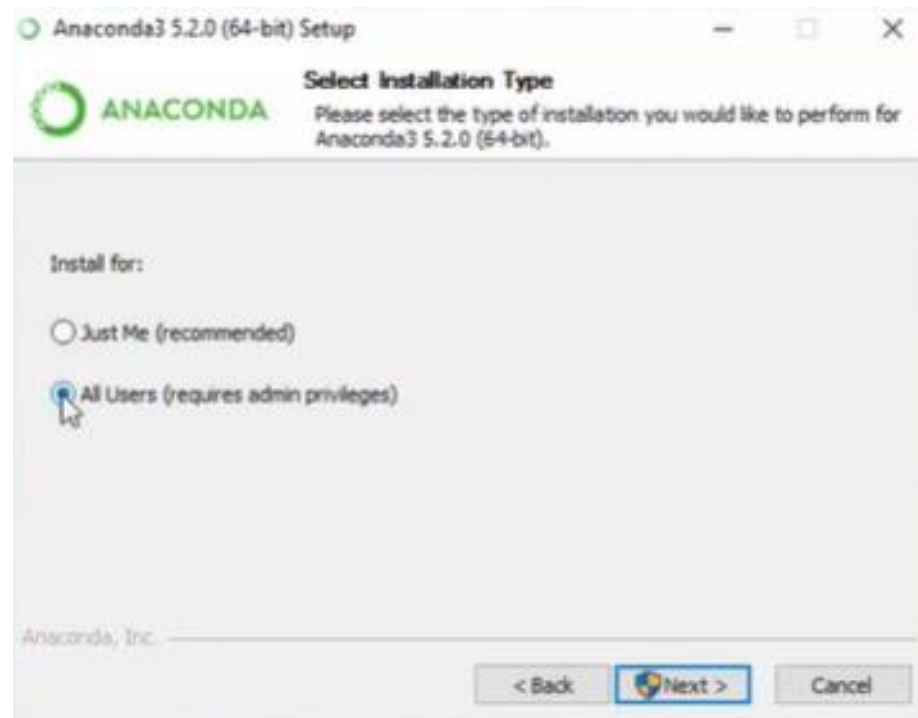
Installare Anaconda - 1

- Recarsi all' URL <https://www.anaconda.com/distribution/#download-section>;
- Scaricare il «**64-bit Graphical Installer**» (dopo aver verificato dai setting di Windows che la propria macchina ha un'architettura a 64 bit) di **Python 3.7** (versione delle API del linguaggio più recente).



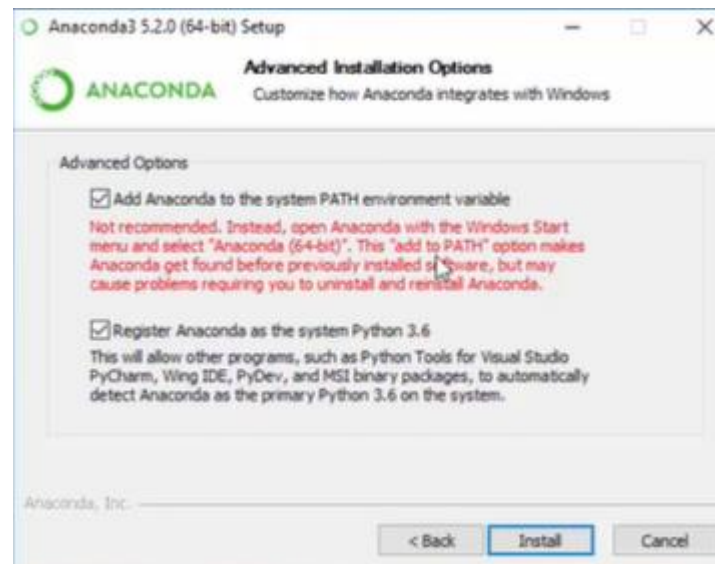
Installare Anaconda - 2

- Eseguire il wizard di installazione.



Installare Anaconda - 3

- Aggiungere la variabile di ambiente Anaconda al *path* di sistema di modo che anche il prompt nativo di Windows riconosca il comandi «conda»;
- Settare Anaconda come l'interprete Python di sistema.



Anaconda prompt

Alcuni comandi utili:

>> **conda info** per vedere le informazioni sulla versione in uso di Anaconda

>> **jupyter notebook** per avviare l'ambiente sul proprio default browser

>> **pip install QuantLib-Python** esempio di installazione di un modulo aggiuntivo.

A screenshot of an Anaconda Prompt window titled "Anaconda Prompt - conda info". The window has a black background with white text. The prompt is "(base) C:\Users\learn>". The first command entered is "python --version", which outputs "Python 3.6.5 :: Anaconda, Inc.". The second command entered is "conda info", and the cursor is positioned at the end of this command, ready for execution.

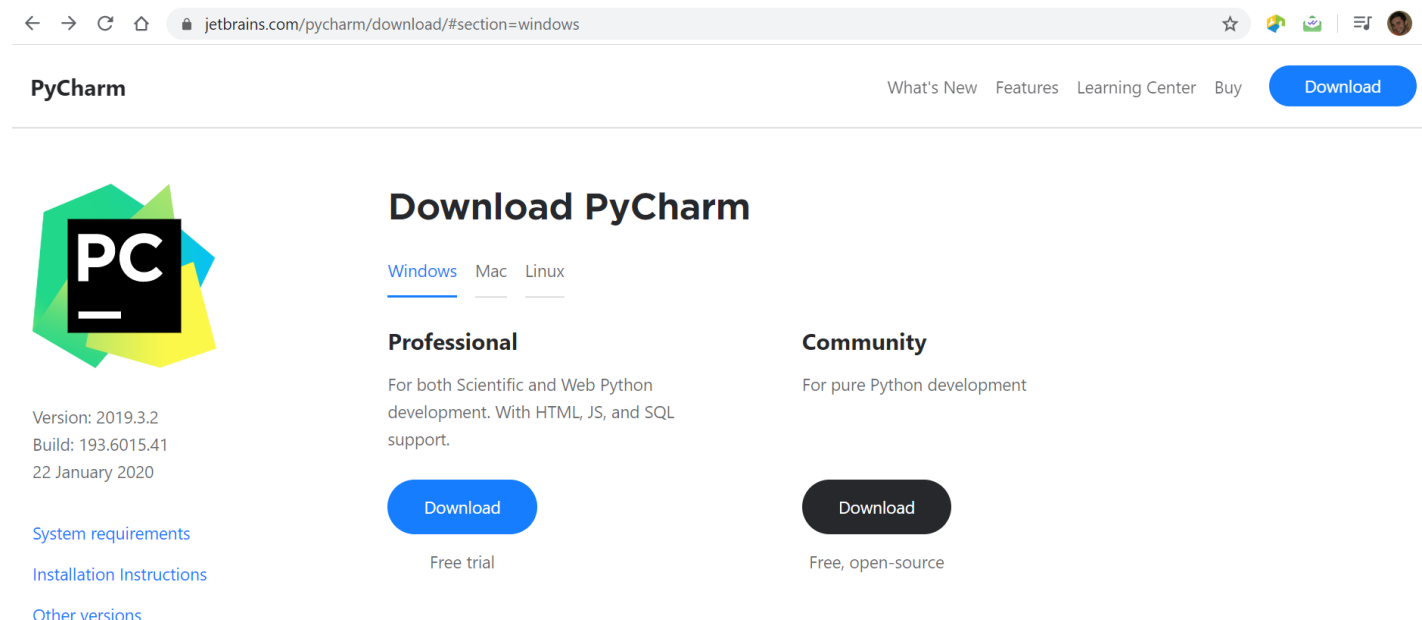
```
Anaconda Prompt - conda info
(base) C:\Users\learn>python --version
Python 3.6.5 :: Anaconda, Inc.
(base) C:\Users\learn>conda info
```

Cos'è PyCharm?

- PyCharm è un **IDE** (*Integrated Development Environment*) pensato per lo sviluppo di analitiche in Python sviluppato da JetBrains;
- PyCharm è dotato di una comoda **Python console** integrata che permette di testare i propri script;
- È possibile affidarsi al **debugging tool** incluso in PyCharm per verificare la presenza di bug e imperfezioni nel codice;
- PyCharm è multiplatforma, infatti è disponibile per Linux, Windows e MacOS;
- La **Community Edition** è gratuita.

Installare PyCharm - 1

- Recarsi all' URL <https://www.jetbrains.com/pycharm/download/#section=windows> e scaricare la Community Edition (gratuita).



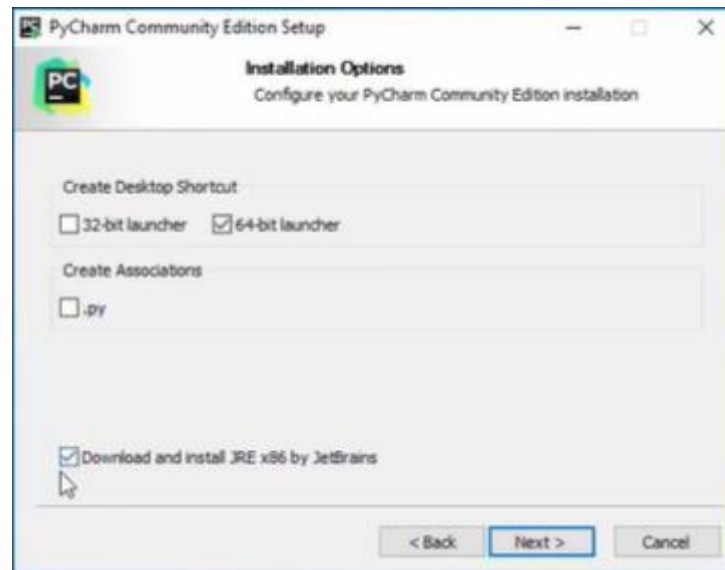
Installare PyCharm - 2

- Eseguire il wizard di installazione.



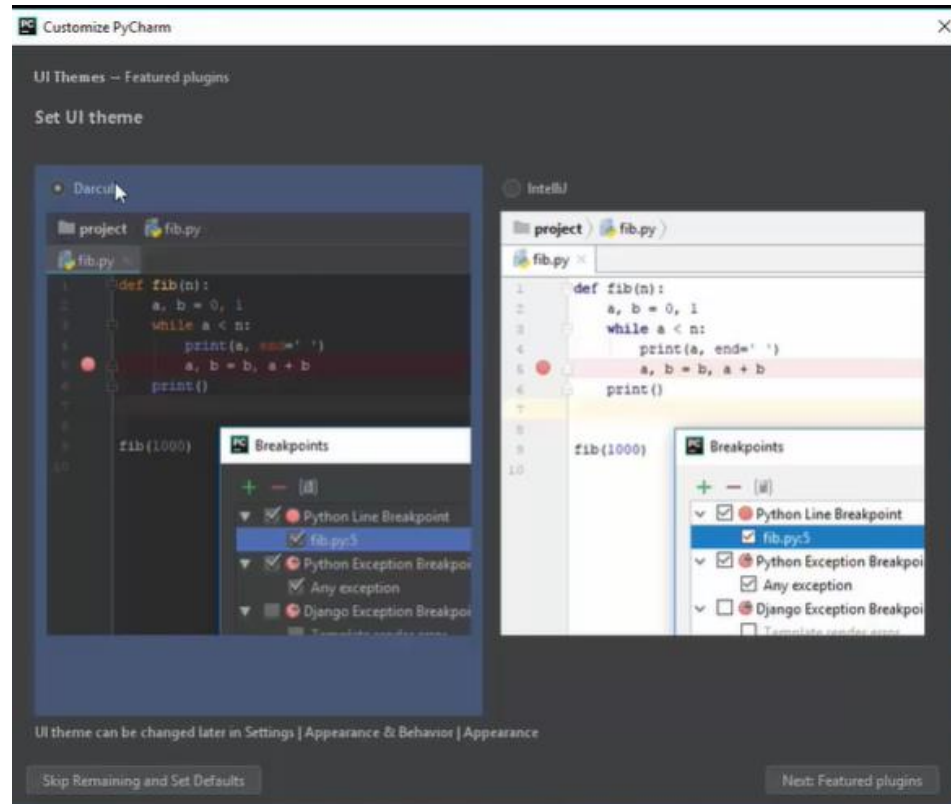
Installare PyCharm - 3

- Flaggare il box «64-bit launcher» se si è verificato dai setting di Windows che la propria macchina ha un'architettura a 64 bit;
- Associare i file con **estensione .py** a PyCharm.

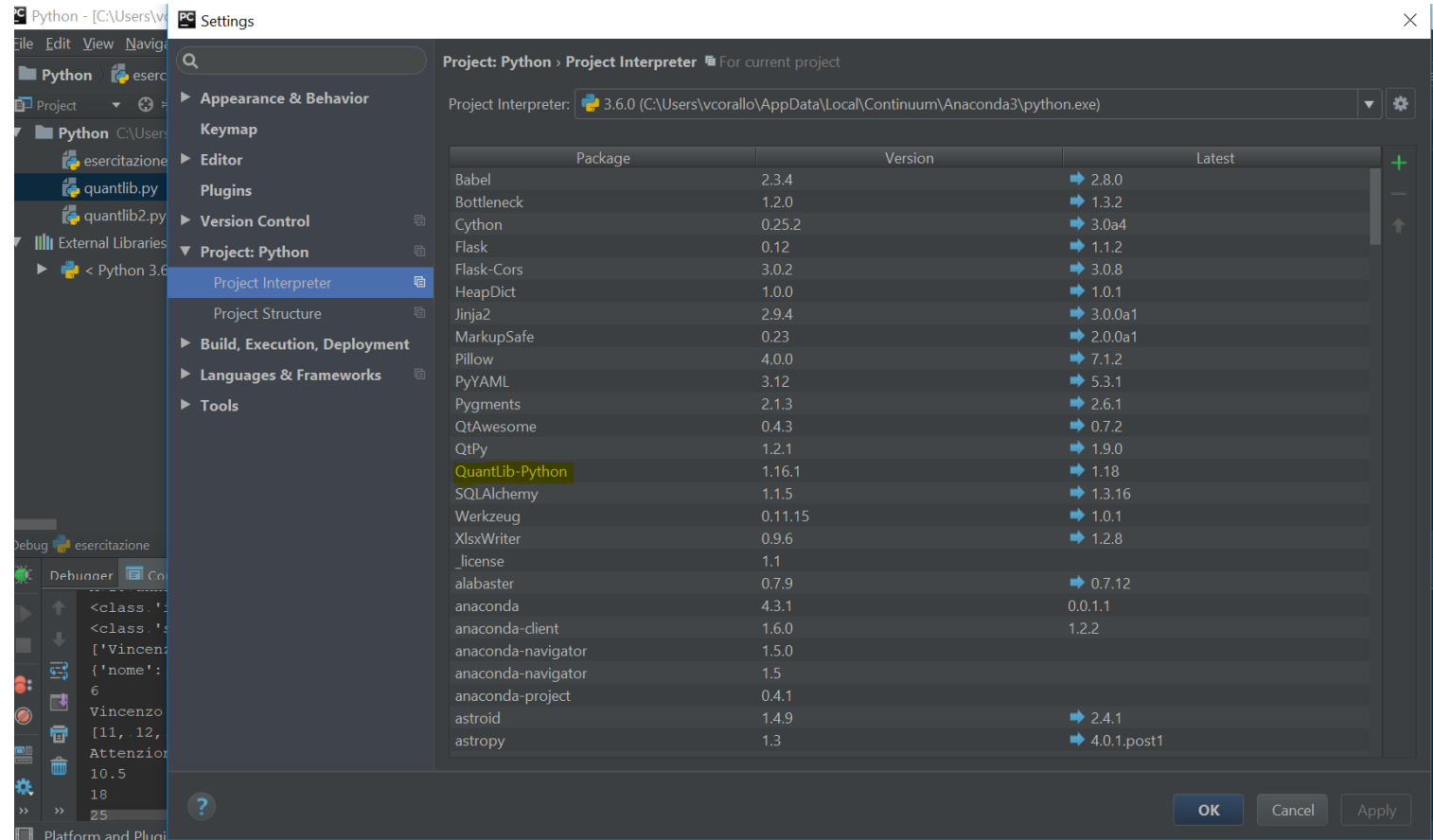


Installare PyCharm - 4

- Completare l'installazione selezionando il tema e qualche altra configurazione.



Configurare Anaconda come interprete del progetto in PyCharm



Caratteristiche e vantaggi del linguaggio Python

- Python è un linguaggio *open source*;
- Per tale motivo, Python ha un fortissimo supporto della community di developers che alimentano migliaia di moduli e librerie gratuite;
- Python è un linguaggio *interpretato* ovvero è possibile eseguire un'operazione da linea di comando senza compilare un eseguibile
- Python è un linguaggio di *alto livello* ovvero tanti aspetti finemente tecnici, quali ad esempio la gestione della memoria, sono oscurati all'utente;
- Ciò che ha favorito la sua enorme diffusione è la sua **semplicità**: la sintassi di Python è molto intuitiva e potrebbe essere usata come pseudo-codice;
- Gli *scope* all'interno programmi vengono definiti per *indentazione*;
- Consente di sviluppare codice sia con un **paradigma di programmazione funzionale** che con un **paradigma di programmazione orientato agli oggetti**.

Cos'è un repository?

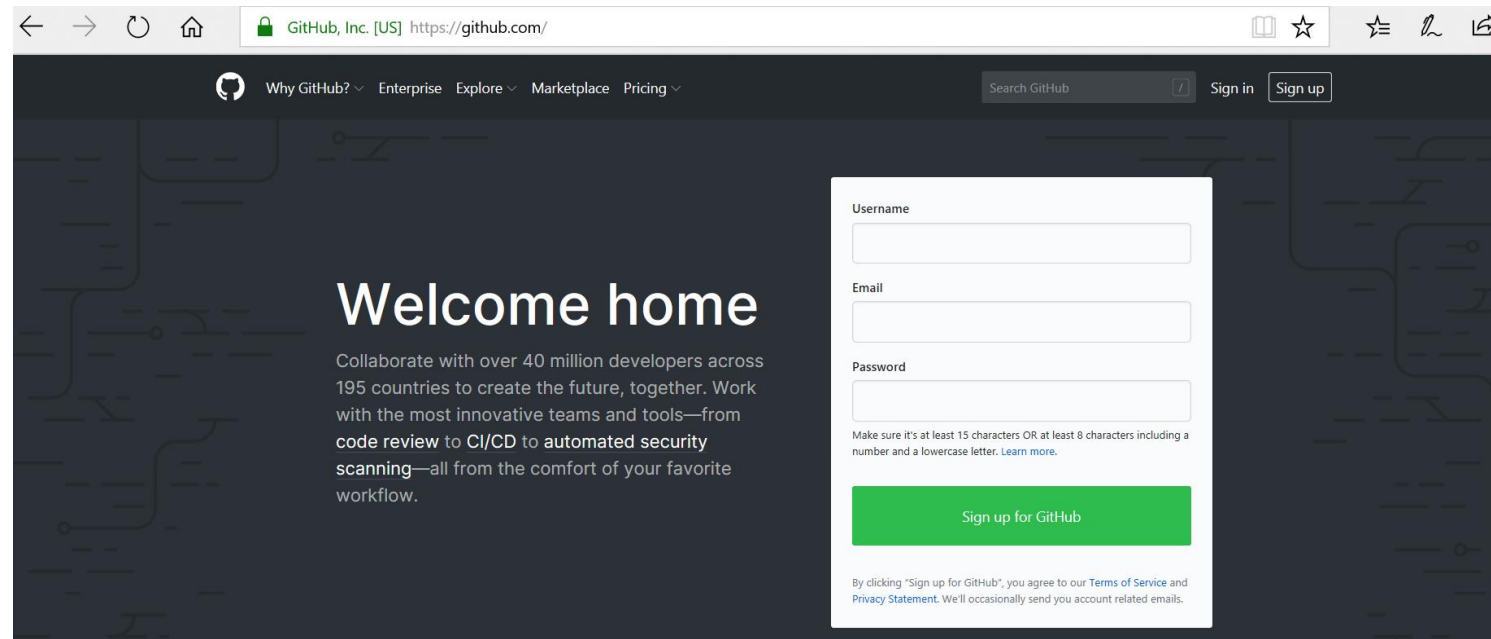
- Un **repository** (in italiano andrebbe tradotto con il sostantivo «ripostiglio») è l'archivio nel quale i file di un progetto di sviluppo codice sono memorizzati, spesso su server in remoto;
- Un **check-out** effettua una copia dei file di progetto dal repository (può essere visto come l'operazione inversa dell'importazione) alla propria directory locale. Il check-out è il prelievo del file nonché lo stato corrispondente;
- Un **check-in** si effettua quando si copiano le modifiche eseguite sui file in locale nel server remoto del repository (il software di controllo versione controlla quali file sono stati modificati dall'ultima sincronizzazione). Il check-in è l'inserimento del file nonché lo stato corrispondente;
- Un **conflitto** si presenta quando diversi soggetti apportano modifiche in contemporanea allo stesso file non avendo visibilità l'uno le modifiche che sta apportando l'altro e che potrebbero sovrapporsi. Non essendo il software abbastanza intelligente da decidere quale tra le modifiche è quella «corretta», si richiede ad un utente di risolvere il conflitto.

Cos'è GitHub?

- Il sito **GitHub** è utilizzabile dagli sviluppatori, che caricano il codice sorgente dei loro programmi e lo rendono scaricabile dagli utenti. Questi ultimi possono interagire con lo sviluppatore tramite un sistema di *issue tracking*, *pull request* e commenti che permette di migliorare il codice del repository risolvendo bug o aggiungendo funzionalità. Inoltre Github elabora dettagliate pagine che riassumono come gli sviluppatori lavorano sulle varie diramazioni, dette **branches**, dei **repository** versionato con Git;
- Moltissime aziende che offrono servizi a livello internazionale usano GitHub, le principali sono Google, Apple, Microsoft, NASA, Facebook, Twitter, JetBrains, e GitHub stesso.

Creare un account su GitHub

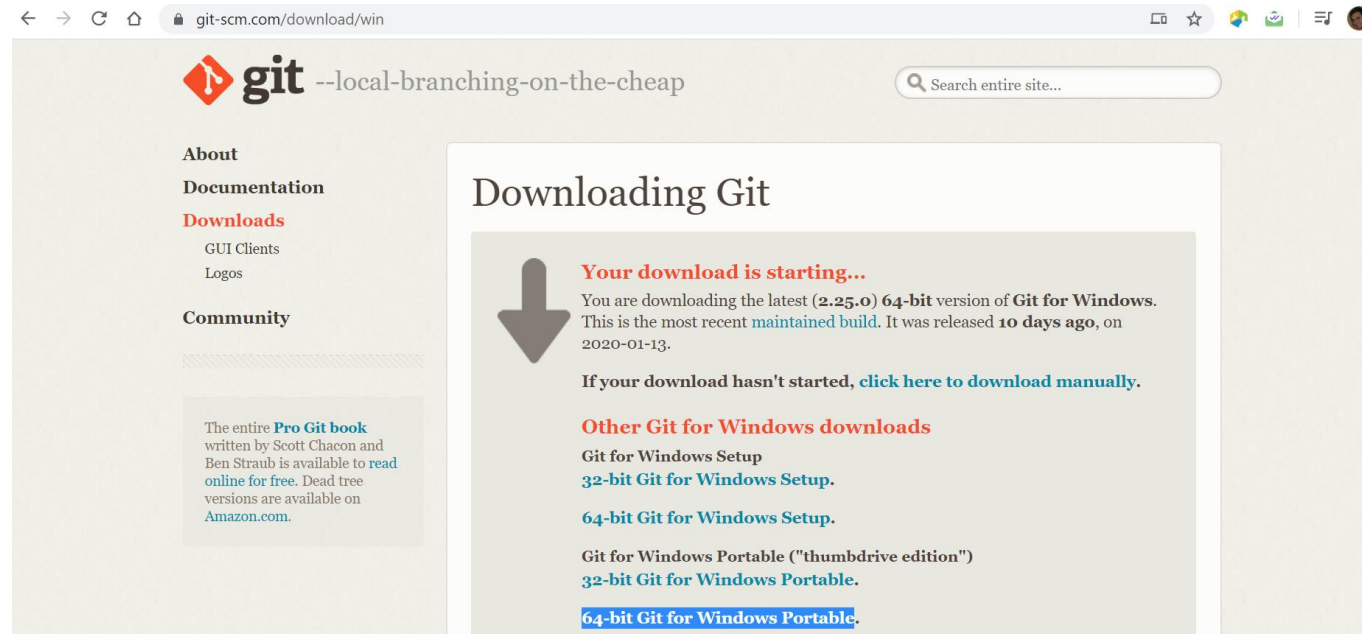
- Recarsi all' URL <https://github.com/> e creare il proprio account seguendo le istruzioni a video.




The screenshot shows the GitHub homepage in a web browser. The browser's address bar displays "GitHub, Inc. [US] https://github.com/". The page features a dark background with a faint, stylized tree diagram. On the left, the text "Welcome home" is prominently displayed, followed by a paragraph: "Collaborate with over 40 million developers across 195 countries to create the future, together. Work with the most innovative teams and tools—from code review to CI/CD to automated security scanning—all from the comfort of your favorite workflow." On the right side, there is a white sign-up form with the following fields: "Username", "Email", and "Password". Below the "Password" field, a note states: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)". A large green button labeled "Sign up for GitHub" is positioned below the form. At the bottom of the form, a small disclaimer reads: "By clicking 'Sign up for GitHub', you agree to our [Terms of Service](#) and [Privacy Statement](#). We'll occasionally send you account related emails."

Installare la versione «portable» di Git e Git-Bash

- Recarsi all' URL <https://git-scm.com/download/win> e scaricare, dopo aver verificato dai setting di Windows che la propria macchina ha un'architettura, la «64-bit Git for Windows Portable»



Git-Bash

 MINGW64:/C:/Users/vcorallo/Desktop/Università di Catania/Slides-Ingegneria-Finanziaria-Unict-2020

```
vcorallo@ITPC007781 MINGW64 /  
$ cd /C/  
  
vcorallo@ITPC007781 MINGW64 /C  
$ cd Users/vcorallo/Desktop/Università\ di\ Catania/Slides-Ingegneria-Finanziaria-Unict-2020/  
  
vcorallo@ITPC007781 MINGW64 /C/Users/vcorallo/Desktop/Università di Catania/Slides-Ingegneria-Finanziaria-Unict-2020 (master)  
$ 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 checkout -- <file>..." to discard changes in working directory)  
  
        modified:   Richiami di Finanza Matematica.pptx  
  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
  
        Introduzione al CCR e agli XVA.pptx  
        Programma Ingegneria Finanziaria 2020 v2.pdf  
        Python, Git, Anaconda.pptx  
        Python/  
        ~$Python, Git, Anaconda.pptx  
  
no changes added to commit (use "git add" and/or "git commit -a")  
  
vcorallo@ITPC007781 MINGW64 /C/Users/vcorallo/Desktop/Università di Catania/Slides-Ingegneria-Finanziaria-Unict-2020 (master)  
$ |
```

Terminale di Git-Bash: comandi di base (UNIX)

- **Change Directory:** con il seguente comando ci si dirigerà nella folder prescelta. Con il tasto Tab si attiva l'auto compilazione (affichè ciò funzioni occorre che la directory annidata sia univoca)

\$ cd C/Users/vcorallo

Rispettivamente, con il comandi:

\$ cd .. si risale di un livello;

\$ cd ../.. si risale di due livelli e così via.

- **Directory listing:** per visualizzare i file e le folder presenti nella directory corrente occorre digitare il comando:

\$ ls

Sono disponibili Git manager dotati di interfaccia grafica (eg. **Git Extensions**) e «indicatori grafici» sullo stato dei file nel progetto (eg. **TortoiseGit**)

Funzionamento di un sistema di controllo versione

Un sistema di controllo versione (VCS) conserva le modifiche effettuate sui file tracciati in:

- *locale*: i file tracciati risiedono sulla propria macchina;
- *globale*: i file tracciati risiedono su un server remoto;
- *distribuito*: i file tracciati risiedono sulle macchine di ciascun utente.

Benefici del versionamento

Git è un VCS decentralizzato, che tiene traccia delle modifiche ai file tramite particolari «snapshot» chiamati ***commit***, che includono le modifiche eseguite in locale rispetto ai commit precedenti.

- nei progetti di sviluppo su cui lavorano più developers ogni modifica viene salvata previo un meccanismo di controllo di integrità e relativo feedback (***code review***);
- Git è utile nel caso di modifica o cancellazione involontaria di informazioni contenute nei file tracciati

Inizializzare un repository locale

La directory in cui file siano tracciati da Git sono storati si configura come versione locale del repository.

Per inizializzare un repository locale occorre:

- attraverso Git-bash, dirigersi nella directory radice del progetto che si vuole tracciare;
- digitare il comando (senza mai includere il simbolo dollaro, che semplicemente compare sempre all'inizio della riga di comando):

\$ git init

Configurare il proprio account di GitHub su GitBash

Al fine di configurare il proprio account GitHub in Git-Bash occorre digitare, per esempio:

```
$ git config --global user.name «Vincenzo Corallo»
```

→ Imposta il nome utente che sarà associato alle modifiche

```
$ git config -- global user.email vincenzo.corallo@gmail.com
```

→ Imposta anche una mail da associare al nome utente (*n.b.* verrà richiesto l'inserimento della password scelta)

Clonare un repository remoto in locale

Al fine di clonare un repository remoto in locale, occorre digitare, per esempio:

```
$ git clone https://github.com/lballabio/QuantLib.git
```

Per verificare lo stato di sincronizzazione tra il repository locale e la versione del repository in remoto occorre digitare:

```
$ git status
```

Per osservare la storia dei commit con i relativi autori, data del commit, identificativo univoco (detto *hash*) del commit e il titolo della stessa occorre digitare:

```
$ git log
```


Le branch - 1

Le branch locali (le equivalenti remote prendono il nome di *sandbox*) sono delle diramazioni di quella principale detta «**master**», che vengono create per eseguire degli sviluppi specifici.



Per creare una branch occorre digitare:

\$ git checkout -b <nomeBranch>

Le branch - 2

Per navigare verso un'altra branch occorre digitare:

\$ git checkout <nomeBranch>

Per visualizzare tutte le branch presenti nel repository locale occorre digitare:

\$ git branch

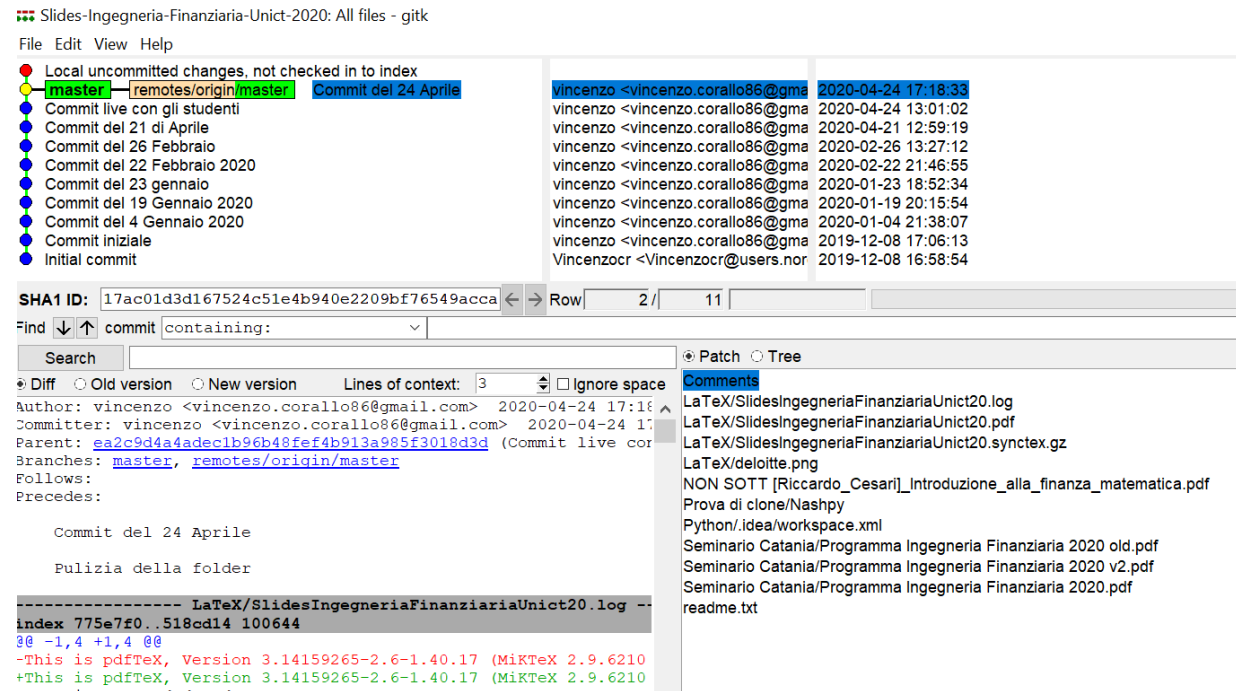
Per eliminare una branch, terminati gli sviluppi, occorre digitare:

\$ git branch -D <nomeBranch>

Comando gitk

Per visualizzare un'interfaccia grafica dello stato del repository digitare il comando:

\$ gitk



The screenshot shows the gitk graphical user interface. At the top, it displays the repository name 'Slides-Ingegneria-Finanziaria-Unict-2020: All files - gitk'. Below this is a menu bar with 'File', 'Edit', 'View', and 'Help'. The main area is divided into several panes. On the left, there's a list of commits with colored circles indicating their status (e.g., red for local uncommitted changes, yellow for master, blue for other branches). The middle pane shows the commit history with details like 'SHA1 ID', 'Author', 'Committer', 'Parent', and 'Branches'. The right pane shows the commit message and the list of files changed in that commit. At the bottom, there's a diff view showing the changes between the current commit and its parent, with a list of files and their corresponding changes.

```
gitk Slides-Ingegneria-Finanziaria-Unict-2020: All files - gitk
File Edit View Help
Local uncommitted changes, not checked in to index
master remotes/origin/master Commit del 24 Aprile
Commit live con gli studenti
Commit del 21 di Aprile
Commit del 26 Febbraio
Commit del 22 Febbraio 2020
Commit del 23 gennaio
Commit del 19 Gennaio 2020
Commit del 4 Gennaio 2020
Commit iniziale
Initial commit

SHA1 ID: 17ac01d3d167524c51e4b940e2209bf76549acca Row 2 / 11
Find commit containing:
Search
Diff Old version New version Lines of context: 3 Ignore space
Author: vncenzo <vincenzo.corallo86@gmail.com> 2020-04-24 17:16
Committer: vncenzo <vincenzo.corallo86@gmail.com> 2020-04-24 17:16
Parent: ea2c9d4a4adec1b96b48fef4b913a985f3018d3d (Commit live con gli studenti)
Branches: master, remotes/origin/master
Follows:
Precedes:

Commit del 24 Aprile
Pulizia della folder

----- LaTeX/SlidesIngegneriaFinanziariaUnict20.log -----
index 775e7f0..518cd14 100644
@@ -1,4 +1,4 @@
-This is pdfTeX, Version 3.14159265-2.6-1.40.17 (MiKTeX 2.9.6210
+This is pdfTeX, Version 3.14159265-2.6-1.40.17 (MiKTeX 2.9.6210
-----
```

Stage e commit

Una volta effettuate le modifiche necessarie in locale, è possibile aggiungere i file modificati alla *stage area*, che contiene il contenuto del prossimo commit, digitando il comando:

\$ git add <nomeFile> → aggiungere un file specifico alla stage area

\$ git add . → aggiungere alla stage area tutti i file modificati alla stage area

I file modificati, che prima di essere aggiunti alla stage area erano evidenziati in **rosso**, dopo l'aggiunta verranno visualizzati in **verde** digitando il comando `$git status`

Per inserire il contenuto della stage area nel commit (che come accennavo rappresenta uno snapshot dello stato dei file tracciati al netto delle modifiche) occorre digitare il comando:

\$ git commit

Pull e Push

Avendo associato dei repository remoti a un progetto, possiamo caricare o scaricare modifiche da e verso di essi, digitando i comandi:

\$ git pull origin master → scaricare dal repository remoto i commit che non abbiamo in locale. Questo è sconsigliabile se siamo consapevoli vi sono delle modifiche in locale. In tal caso è più *safe* digitare in sequenza:

\$ git fetch origin master → per scaricare i commit mancanti in locale

\$ git rebase -i origin master → per posizionare «on top» ai commit in remoto i nostri commit locali

\$ git push origin master → caricare le modifiche al progetto sul repository remoto

Risoluzione dei conflitti

Nel caso in cui due commit contengano modifiche allo stesso file, bisogna scegliere manualmente la modifica da tenere prima di poter caricare o scaricare entrambi i commit nello stesso repository