Introduzione ad Anaconda e Conda

Rights & Credits

Anaconda

Gli scienziati e, in particolare, quelli che si occupano della gestione di dati hanno spesso bisogno di usare diversi pacchetti di software open source (*open source* significa che il codice sorgente è reperibile gratuitamente, per essere usato o modificato da sviluppatori ove necessario). Tali programmi sono utilizzati per il calcolo scientifico, il calcolo distribuito (cioè quando più computer lavorano assieme per risolvere un problema comune), i *big data*, l'analisi dei dati, la statistica, l'apprendimento automatico, il web e la visualizzazione. I componenti dei pacchetti di software utilizzati per questi scopi hanno componenti scritti in linguaggi generalmente diversi, ma possono essere utilizzati nel framework del linguaggio di programmazione Python.

Anaconda è una *distribuzione software* (cioè, appunto, una collezione di programmi con diversi campi di applicazione distribuiti come un unico set) che installa tutti questi pacchetti e ne garantisce la compatibilità. Centinaia di pacchetti per l'analisi, la modellazione e la visualizzazione sono preinstallati e pronti per l'uso.

Anaconda Navigator è l'interfaccia grafica per l'utente di Anaconda che consente di aggiornare e installare nuovi pacchetti, includendo le migliaia di librerie più comuni e utilizzate in Python e in altri linguaggi di programmazione. Una volta installato, Anaconda Navigator appare come mostrato, per esempio, sotto. Anaconda offre:

- Conda, che è un gestore di pacchetti software da linea di comando;
- Anaconda Navigator, il *navigatore* di Anaconda;
- l'ultima versione di Python supportata dalla distribuzione Anaconda;
- oltre 350 pacchetti software per la scienza dei dati e il machine learning;
- la possibilità di installare facilmente e usare in modo compatibile migliaia di programmi e pacchetti.

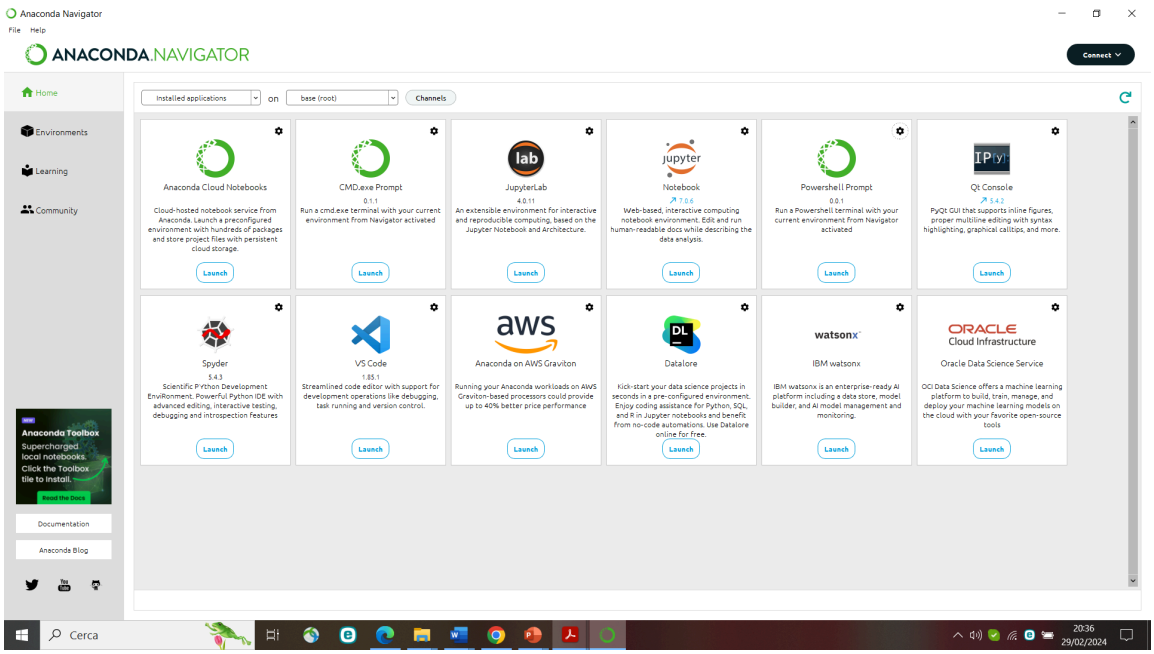
Tra le applicazioni presenti in Anaconda menzioniamo, in quanto le useremo, Jupyter Notebook e JupyterLab. Tali applicazioni sono parte di **Jupyter**, un progetto open-source creato per supportare data science e calcolo scientifico interattivi usando diversi linguaggi di

programmazione. Jupyter fornisce un ambiente basato sul web per lavorare con notebook che supportano codice, dati e altre estensioni.

Jupyter Notebook fornisce un'interfaccia molto semplice per aprire e lavorare su notebook, file di testo e terminale, supportando l'uso di linguaggi di programmazione quali Python, Julia, ecc. Questo documento è scritto su un notebook di Jupyter. Possiamo scrivere questo testo in una *cella* del notebook e poi visualizzarlo opportunamente perché nel menu che appare sopra in questo notebook abbiamo selezionato `Markdown` al posto di `Code` per questa cella. *Markdown*, o meglio *markdown editor* è uno strumento che consente una facile conversione di un semplice testo in formati di tipo HTML, per esempio per creare titoli ed elenchi puntati, e aggiungere immagini (come quella di sotto), video e link.

JupyterLab è un'interfaccia con notevoli proprietà interattive, che consente di usare notebook, consolle, terminali, mappe interattive (si dice *mappa interattiva* un sistema di rappresentazione grafica delle informazioni capace di interagire con le scelte dell'utente o con il verificarsi di altre condizioni esterne), ecc.

Infine, è qui utile sottolineare che Anaconda offre, di default, due utili prompt di comandi, **CMD.exe Prompt** e **Powershell Promt**, che consentono, tra l'altro, di usare il programma di gestione conda (vedi sotto).



Conda

Conda è uno strumento open source per gestire dispense (*repository*) di pacchetti, le loro dipendenze e ambienti per tutti i linguaggi di programmazione: Python, Java, JavaScript, C/C++, FORTRAN, ecc.

Aprendo uno dei prompt di Anaconda, si può controllare che conda sia installato correttamente e attivo attravero l'istruzione

```
conda --version
```

Si può vedere tale informazione anche da questo stesso notebook, ma, per essere riconosciuto, conda deve essere preceduto dal punto esclamativo, come mostrato nella cella seguente.

In [9]:
```
!conda --version
```

```
conda 24.1.2
```

Analogamente, la version di Python usata può essere ottenuta come segue.

In [6]:
```
!python --version
```

```
Python 3.11.7
```

Le linee di comando di sopra sono state lanciate pressando simultaneamente i tasti di shift e invio; alternativamente, si può usare la freccia che appare in cima al notebook. Lo stesso vale per tutte le altre celle del notebook.

Creazione di ambienti, installazione di software e aggiornamenti usando conda

Ambiente virtuale (virtual environment)

Come vedremo, possiamo usare Python nel notebook di Jupyter o da un terminale, incluso uno dei prompt di Anaconda. Tuttavia, è bene definire prima un ambiente virtuale in cui lavorare. Un **ambiente virtuale** e' una struttura autonoma di directory (directory e sue sottocartelle) che contiene l'installazione di una specifica versione di Python e un certo numero di pacchetti e librerie aggiuntivi, opportuni per il tuo progetto, senza influire sull'ambiente di base del sistema. Creare tale ambiente di lavoro è conveniente per vari motivi:

- consente di installare pacchetti aggiuntivi senza permessi di root;
- essendo un ambiente autonomo e separato, il funzionamento dei codici Python (o, in generale, codici che usano un altro linguaggio di programmazione) prodotti non viene compromesso da aggiornamenti o nuove installazioni fatte nel sistema di base complessivo;
- consente di mantenere un ambiente di sviluppo specifico per un codice in cui non vi sono conflitti tra pacchetti aggiuntivi diversi o tra versioni diverso di uno stesso pacchetto, dal momento che l'utente decide quali applicazioni installare nell'ambiente e quando / come aggiornarle.

Creazione/gestione/rimozione di ambienti virtuali con conda

Come buona norma generale, è consigliabile creare ambienti separati per progetti diversi, in modo tale da tenere traccia dei pacchetti e delle loro versioni separatamente. Conda fornisce un modo conveniente per installare e gestire raccolte separate di pacchetti di software. Per esempio, la creazione e l'aggiornamento di un ambiente virtuale possono essere effettuati usando conda, oltre che, come vedremo, usando lo stesso Python.

Una volta lanciato il prompt di comandi di Anaconda, creiamo un ambiente virutale e lo denominiamo, per esempio, ENV1 tramite la riga di comando

```
conda create -n ENV1
```

o, equivalentemente,

```
conda create --name ENV1
```

Con Anaconda in Windows, il nuovo ambiente può anche essere creato direttamente sfruttando l'Anaconda Navigator: si clicca su `Environments` e poi su `Create`.
Il nuovo ambiente virtuale può anche essere creato con una specifica versione di Python. Per esempio, se nell'ambiente che ci accingiamo a creare vogliamo sviluppare ulteriormente un vecchio codice scritto nella versione 3.9 di Python, senza doverlo aggiornare alla più recente versione, possiamo aggiungere (in inglese, *append*) `python=3.9` all'istruzione precedente:

```
conda create --name ENV1 python=3.9
```

Come impostazione predefinita (*by default*), il direttorio che rappresenta il nuovo ambiente viene creato nel sottodirettorio `envs` del direttorio in cui è installato Anaconda (nel mio caso, quest'ultimo è `C:\Users\Agostino\anaconda3` e quindi il direttorio in cui è collocato l'ambiente virtuale è `C:\Users\Agostino\anaconda3\envs`). Si può, tuttavia, specificare un diverso percorso in cui creare l'ambiente virtuale per mezzo della seguente riga di comando:

```
conda create -p /percorso/ENV1
```

dove l'opzione `-p` sta proprio per `*path*'. A questo punto, possiamo attivare l'ambiente creato per mezzo dell'istruzione

```
conda activate ENV1
```

ed eseguire la riga di comando

```
conda list
```

per vedere quali programmi sono già installati in esso. Per installare un qualsiasi pacchetto software con conda, si usa l'istruzione

```
conda install nome_pacchetto
```

Se non sappiamo quali versioni di un programma, o pacchetto di programmi, sia disponibile possiamo cercarlo tramite l'istruzione

```
conda search nome_pacchetto
```

vedere la versione `numero_versione` che ci interessa e installarla con

```
conda install nome_pacchetto=numero_versione
```

Per le attività di questo corso, adesso installiamo alcuni pacchetti software per l'elaborazione scientifica, l'analisi di dati e la loro rappresentazione nel framework del linguaggio Python:

```
conda install jupyter numpy matplotlib pandas scipy
```

Programmi e pacchetti possono anche essere installati usando Anaconda Navigator, dopo averli cercati nella lista di programmi `Not Installed`. Alla fine della propria attività, disattiviamo l'ambiente virtuale con

```
conda deactivate
```

per tornare all'ambiente base (e passare ad un altro ambiente per un altro progetto per esempio ...).
Se, a un certo punto, vogliamo rimuovere un ambiente virtuale creato precedentemente (che, ovviamente, deve essere stato disattivato prima di poterlo rimuovere), di nome `nome_env`, possiamo usare la riga di comando

```
conda remove -n nome_env --all
```

Possiamo vedere l'ambiente in cui ci troviamo e la lista di ambienti esistenti eseguendo la riga di comando

In [7]:
```
!conda env list
```

```
# conda environments:
#
base                  *  C:\Users\Agostino\anaconda3
ENV                      C:\Users\Agostino\anaconda3\envs\ENV
```

oppure

In [8]:
```
!conda info --envs
```

```
# conda environments:
#
base                  *  C:\Users\Agostino\anaconda3
ENV                      C:\Users\Agostino\anaconda3\envs\ENV
```

In ogni caso, il prompt dei comandi è modificato in modo tale da mostrare se ci si trova nell'ambiente di base o in un certo ambiente virtuale.

Aggiornamenti con conda

Si può usare conda per aggiornare lo stesso programma:

```
conda update conda
```

Analogamente, per aggiornare anaconda e anaconda-navigator,

```
conda update anaconda
conda update anaconda-navigator
```

Per aggiornare tutto il software nell'ambiente virtuale in cui si sta lavorando, si usa

```
conda update --all
```

mentre

```
conda update -n OTHER_ENV --all
```

consente di aggiornare tutti i contenuti di un altro ambiente virtuale di nome `OTHER_ENV`.