Experimental techniques for GW detection

Introduzione alla programmazione

corso per il XXXVIII ciclo di dottorato in fisica

14 ore

Mateusz Bawaj mateusz.bawaj@unipg.it

Piano della lezione

1) python
2) ambienti virtuali
3) ambienti di sviluppo
4) librerie generiche e particolari GW
5) versioning

Programmazione

nozioni base per il campo di onde gravitazionali

useremo *python* come linguaggio di programmazione

per programmare in *python* istalleremo *conda* (o *anaconda*) e *jupyter notebook*

python è un linguaggio di programmazione e scripting di alto livello presente in maggior parte delle piattaforme (hardware e sistemi operativi)

si trova nella versione più datata 2.7 che è ancora in uso (rilasciata nel 2010) e nella versione più recente 3.11

Tutorial di base (consigliato) (preparato da W3)

Tutorial python

interprete di python viene chiamato con

```
$ python
```

← ed è molto scomodo!

scriviamo *“hellow world”* (nell’interprete scomodo)

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

Tutorial python

il primo ‘programma’ ← script = comandi python in un file di testo con estensione *.py

indentazione funzionale:

```
print("eseguito sempre all'inizio")
if 1 == 1:
    print('eseguito in mezzo solo se la condizione è vera')
    print('eseguito in mezzo solo se la condizione è vera')
print('eseguito sempre alla fine')
```

il file dovrebbe contenere la shebang all’inizio

il file dovrebbe essere eseguibile:

```
$ chmod u+x my_file.py
```

shebang consigliato:

```
#!/usr/bin/env python3
```

Tutorial python

`help()` – in qualsiasi ambiente di sviluppo è sempre sotto mano

`exit()` – per uscire dall'interprete

che cos'è duck typing?

le librerie python frequentemente sono ottimizzate e scritte in linguaggi di programmazione di più basso livello come C/C++.

un libreria per questo task è Simplified Wrapper and Interface Generator (SWIG)

Ambienti virtuali

per la numerosità delle versioni di *python* e delle sue librerie conviene usare gli ambienti virtuali per poter installare le versioni necessarie ed isolarle fra quelle installate per altre applicazioni.

per creare un ambiente nuovo:

```
$ conda create --name corsonde python=3.7
```

per attivare un ambiente:

```
$ conda activate corsonde
```

← all'inizio del prompt appare il nome dell'ambiente

per disattivare un ambiente:

```
$ conda deactivate
```

Ambienti virtuali

per installare le librerie dentro un ambiente virtuale:

```
$ conda install -c conda-forge some-package
```

se l'ambiente base attivato in automatico da *conda* non è richiesto:

```
$ conda config --set auto_activate_base false
```

per aggiungere canale conda-forge ai preferiti (e non dover più specificarlo attraverso `-c conda-forge` durante installazione)

```
$ conda config --add channels conda-forge
```

i file temp di conda occupano molto spazio sul disco. Per liberalo:

```
$ conda clean --all
```

← pulire il cache aiuta anche nel caso dei problemi

Ambienti virtuali

esempio di creazione di un ambiente virtuale per lo scopo del corso:

```
$ conda create --name corsonde python=3.7 sympy einsteinpy numpy scipy nb_conda_kernels matplotlib ipython jupyter
```

dove con ‘`python=3.7`’ inizia la lista dei pacchetti da installare dentro l’ambiente.

N.B. durante il lavoro verranno scelti con la priorità tutti i pacchetti installati dentro l’ambiente virtuale attivo (indicato dal nome in parentesi davanti al prompt). Altri pacchetti non installati dentro l’ambiente verranno cercati nel sistema operativo. Nel caso di apparente incompatibilità installa tutti pacchetti di cui hai bisogno dentro l’ambiente virtuale.

Ambiente di sviluppo

useremo l'ambiente interattivo *jupyter notebook* basato sul motore web per scrivere e condividere con facilità il codice *python*

per installare *jupyter* nell'ambiente: `corsonde`

```
$ conda activate corsonde
```

```
$ conda install nb_conda_kernels
```
← aggiunge tutti gli ambienti virtuali alla lista di jupyter

per attivare jupyter notebook:

```
$ jupyter notebook
```

scrittura del codice ed esecuzione del programma avviene su una pagina speciale nel browser!

Documentazione Jupyter

N.B. nel caso di problemi con jupyter disattiva ad-blocker nel tuo browser!

Ambiente di sviluppo

esistono altri ambienti di sviluppo:

Spyder – più simile a Matlab

Visual Studio – classico ambiente per gli sviluppatori software

PyCharm – buona alternativa al Visual Studio

Eclipse + PyDev – un’altra alternativa

non dimentichiamo gli ambienti in riga di comando:

GNU Emacs

Vi / Vim

libreria efficiente per le operazioni sulle matrici che contiene anche una lunga lista delle funzioni matematiche.

libreria base per diverse applicazioni del calcolo numerico.

classico modo per importarla:

```
$ import numpy as np
```

N.B. *NumPy* fornisce un potentissimo meccanismo chiamato *slice notation* che permette facilmente e in modo efficace a scegliere elementi o sotto-matrici

```
$ array[start:end:step]
```

← sceglie gli elementi da *start* a *stop* con il passo *step*
← indici negativi sono permessi!

N.B. operazioni *element-wise* sono le operazioni che operano sugli elementi delle matrici indipendentemente.

SciPy è una libreria con le funzioni matematiche piu avanzate in rispetto a *NumPy* (FFT, integrazione numerica, interpolazione, algebra lineare e altre).

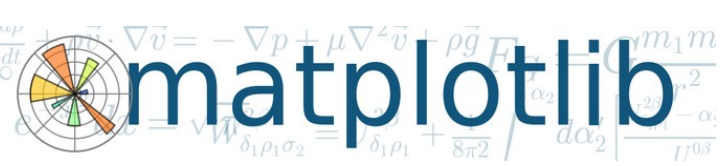
è un corrispondente di *MATLAB*, *Octave* o *Scilab* per python.

contiene una lunga lista di costanti fisiche:

```
$ from scipy import constants
```

nei nostri programmi useremo la trasformata Fourier. Per importare i moduli necessari:

```
$ from scipy.fft import fft, fftfreq
```



matplotlib è una libreria per creare i grafici. Data la sua semplicità è molto utile durante sviluppo del codice per monitorare i dati.

```
$ import matplotlib.pyplot as plt
```

```
$ plt.plot(xdata, ydata)
```
← crea un grafico di punti XY collegati con le rette

per usare in *jupyter notebook* inserire dopo import:

```
%matplotlib inline
```

per cambiare le dimensioni del grafico incluso nel *jupyter notebook*

```
mpl.rcParams['figure.figsize'] = (10,10)
```

```
plt.rcParams['figure.dpi'] = 150
```

le opzioni avanzate permettono a creare i grafici complessi 2D e 3D

SymPy

SymPy è una libreria per algebra computazionale in attivo sviluppo. Mira di diventare un completo sistema CAS (computer algebra system).

```
$ conda install sympy
```

operazioni sulle forme simboliche di equazioni e funzionalità:

- semplificazione
- differenziazione
- soluzione di equazioni lineari
- integrazione indefinita e definita
- …

depends on: mpmath
supports Python 3.5, 3.6, 3.7

EinsteinPy è una libreria python che fornisce gli algoritmi numerici per modellare gli effetti relativistici usando le soluzioni analitiche:

- spazio-tempo di Schwarzshild
- metrica di Kerr
- metrica di Kerr-Newman

per installare EinsteinPy non attivando precedentemente l'ambiente virtuale:

```
$ conda install -n corsonde -c conda-forge einsteinpy
```

contiene dodici moduli per le sue principali funzionalità:

- visualizzazione della ipersuperficie per lo spaziotempo di Schwarzschild (hypersurface)
- simulazione dell’ombra di un buco nero di Schwarzschild (rays)
- visualizzazione delle metriche, corpi e geodiche (metric, bodies, geodesic, coordinates)
- calcolo simbolico

insieme dei pacchetti python per l’astronomia.

funzioni principali sono:

- fornitura delle strutture di dati
- conversione delle unita di misura
- lista delle costanti fisiche
- accesso ai formati di dati caratteristici per l’astronomia (FITS, VO, HDF5)
- computazione di conversioni e trasformazioni cosmologiche

astropy has the following strict requirements:
- Python 3.7 or later
- Numpy >=1.17 or later
- PyERFA >=1.7.3 or later

installazione in conda:

```
$ conda install astropy
```

Virtual Astronomical Observatory

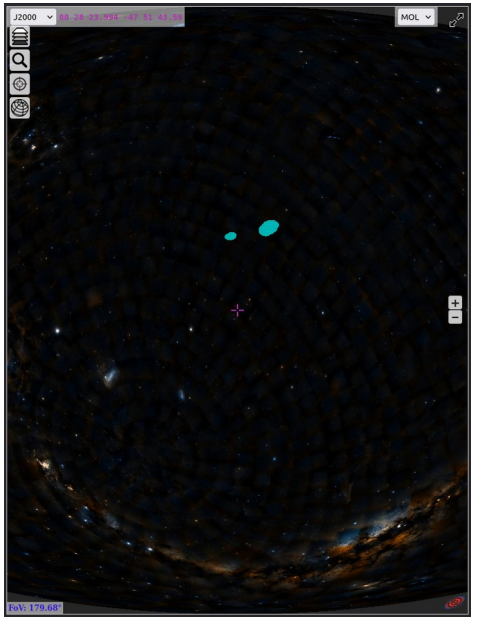
EuroVO

International Virtual Observatory Alliance

offre Aladin lite per facilitare le indagini astronomiche.

installazione in conda:

```
$ conda install -c bmatthieu3 ipyaladin==0.2.1
```



GWpy docs

pacchetto per lo studio sui dati provenienti dagli osservatori delle OG. Contiene funzioni per gli studi su astrofisica e su esperimento di rivelazione.

installazione in conda:

```
$ conda install gwpy
```

l'idea e funzionamento del pacchetto è spiegato nello script *jupyter*.

il pacchetto si trova nella versione di conda preparata dalla comunità di OG: https://docs.ligo.org/lscsoft/conda/

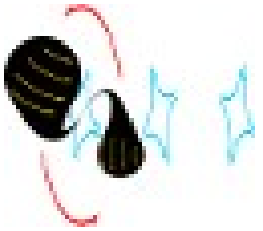
pacchetto software per lo studio delle sorgenti di onde gravitazionali attraverso i dati raccolti negli osservatori interferometrici Virgo e LIGO.

installazione in conda:

```
$ conda install -c conda-forge pycbc
```

funzioni di PyCBC sono usate nei jupyter notebook in corso delle lezioni.

LALSuite



LALSuite is comprised of the following components:

- LAL: Core gravitational wave analysis routines
- LALFrame: LAL wrapping of the LIGO/Virgo Frame library
- LALMetaIO: LAL wrapping of the MetaIO LIGO_LW XML library
- LALSimulation: LAL routines for gravitational waveform and noise generation
- LALBurst: LAL routines for burst gravitational wave data analysis
- LALInspiral: LAL routines for inspiral and ringdown CBC gravitational wave data analysis
- LALPulsar: LAL routines for pulsar and continuous wave gravitational wave data analysis
- LALInference: LAL routines for Bayesian inference data analysis
- LALApps: Collection of gravitational wave data analysis codes and pipelines utilising the LAL libraries

Ambiente virtuale per il corso

per referenza trovate il file di configurazione dell'ambiente virtuale in cui sono stati verificati gli esempi forniti come Jupyter notebook.

Nome del file: `corsonde_env.yml`

ambiente virtuale con una lista predefinita di pacchetti si crea:

```
$ conda env create -f corsonde_env.yml
```

Non solo per i sviluppatori di software!!!

è uno dei sistemi di versioning attualmente più diffusi.

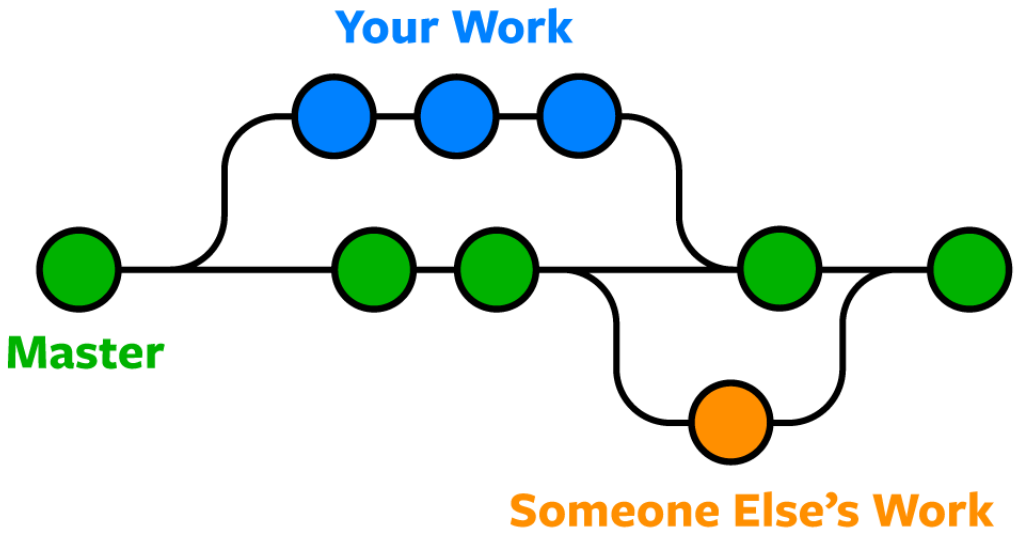
aiuta ad organizzare i documenti (file) in forma testuale.

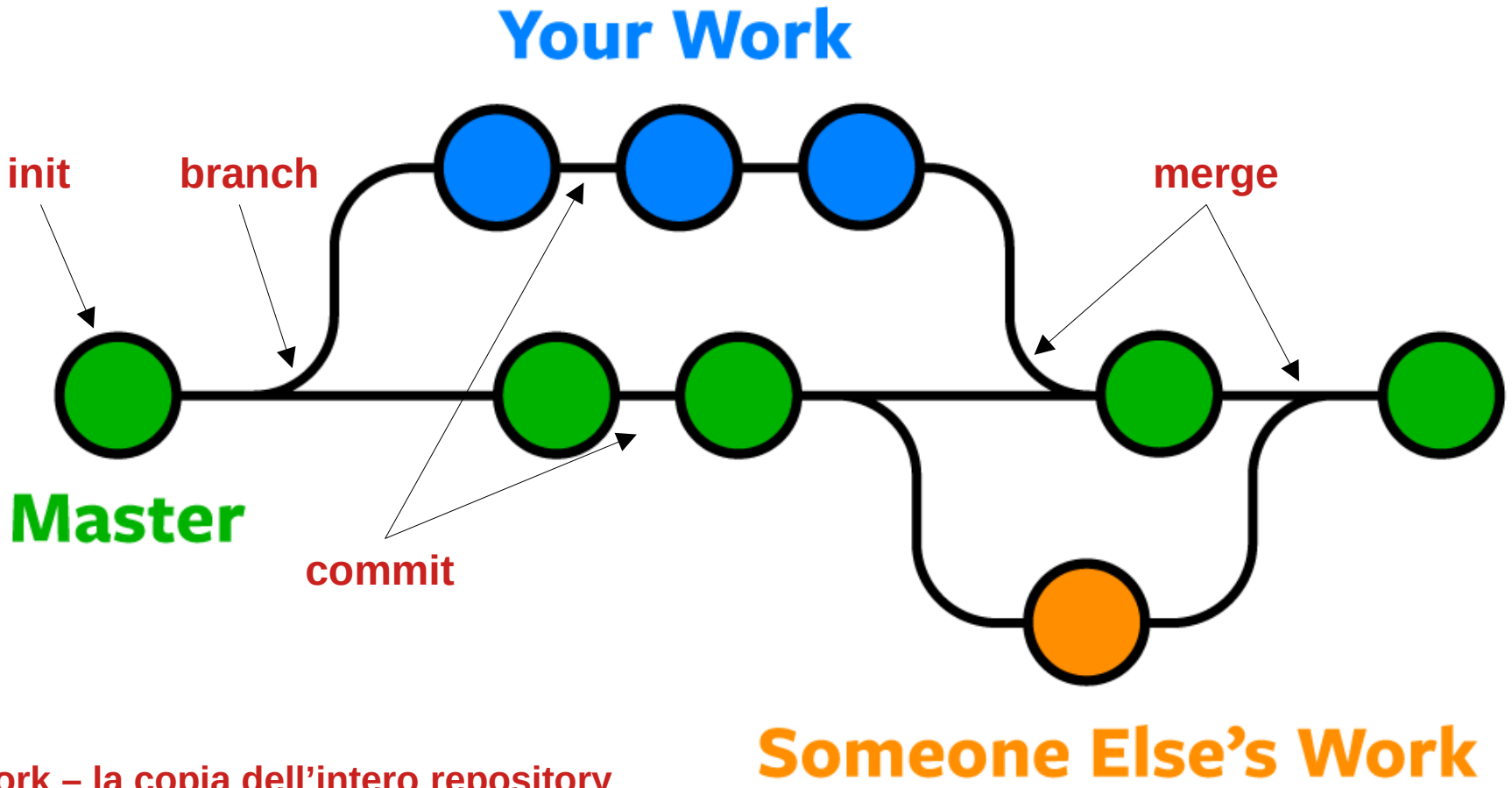
supporta il repositorio remoto e il lavoro in contemporanea.

tutti repositori sono equivalenti (a differenza di SVN dove il repositorio remoto è principale).

funzione *Staging Area* permette di selezionare i cambiamenti da includere nel commit.

è gratuito e open-source.

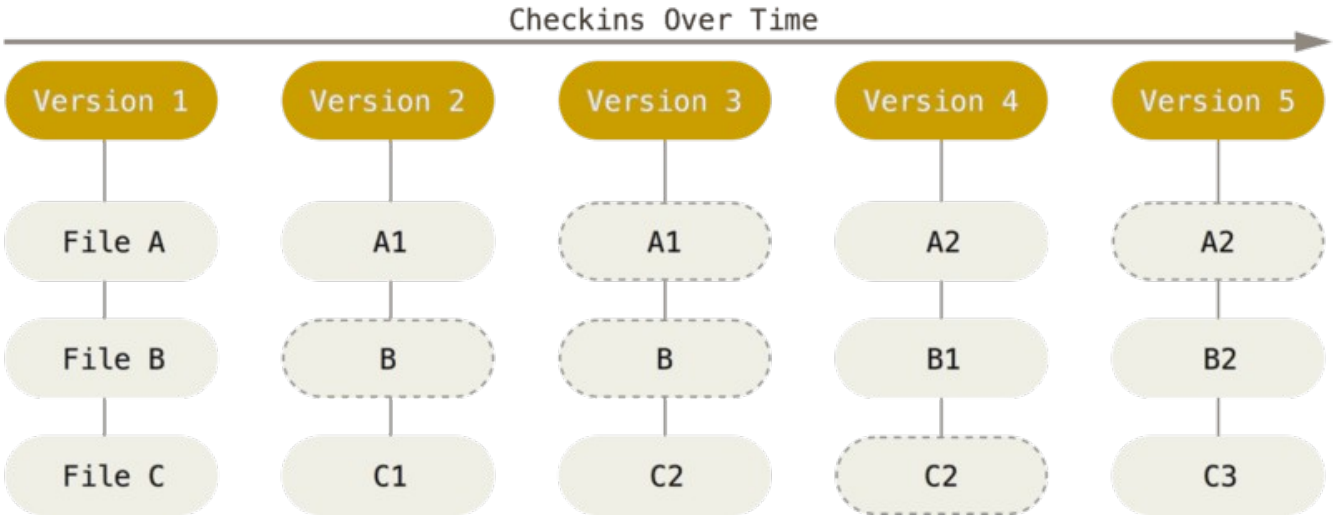




branch parallele

fork – la copia dell'intero repository
clone – la copia locale del repository remoto
pull – download delle modifiche dal repositorio remoto
push – invio delle modifiche locali nel repositorio remoto

“Git considera i propri dati come una sequenza di istantanee (snapshot) di un mini filesystem. Con Git, ogni volta che registri (commit), o salvi lo stato del tuo progetto, fondamentalmente lui fa un’immagine di tutti i file in quel momento, salvando un riferimento allo snapshot. Per essere efficiente, se alcuni file non sono cambiati, Git non li risalva, ma crea semplicemente un collegamento al file precedente già salvato. Git considera i propri dati più come un flusso di istantanee.”



Creare repository

Crea un nuovo repository o clonane uno esistente da un URL

```
$ git init [project-name]
```

Crea un nuovo repository locale con il nome specificato

```
$ git clone [url]
```

Scarica un progetto esistente e il suo storico di cambiamenti

nel repositorio git non possiamo usare più i comandi del sistema operativo (rm, mv)

Refactoring dei nomi di file

Ricerca e rimuovi file dallo storico

```
$ git rm [file]
```

Rimuovi un file dalla directory e prepara l'eliminazione definitiva

```
$ git rm --cached [file]
```

Elimina il file dallo storico di versione ma mantieni il file locale

```
$ git mv [file-original] [file-renamed]
```

Modifica il nome del file in preparazione a una commit

Effettuare modifiche

Rivedi i cambiamenti al codice e prepara una commit

```
$ git status
```

Elenca tutti i file nuovi o modificati

```
$ git diff
```

Mostra le differenze non ancora nell’area di staging

```
$ git add [file]
```

Crea uno snapshot del file in preparazione al versioning

```
$ git reset [file]
```

Rimuovi un file dall’area di staging, ma mantieni le modifiche

```
$ git commit -m "[descriptive message]"
```

Salva gli snapshot dei file in maniera permanente nello storico

Modifiche di gruppo

Aggrega una serie di commit all’interno di un branch

```
$ git branch
```

Elenca tutti i branch nel repository corrente

```
$ git branch [branch-name]
```

Crea un nuovo branch

```
$ git switch -c [branch-name]
```

Passa al branch specificato e aggiorna la directory corrente

```
$ git merge [branch-name]
```

Unisci lo storico del branch specificato con quello corrente

```
$ git branch -d [branch-name]
```

Elimina il branch specificato

Rivedere lo storico

Esplora l’evoluzione dei file del progetto

```
$ git log
```

Elenca lo storico di versione per il branch corrente

```
$ git log --follow [file]
```

Elenca lo storico di versione per il file specificato, incluse rinominazioni

```
$ git diff [first-branch]...[second-branch]
```

Mostra la differenza tra due branch

```
$ git show [commit]
```

Mostra i metadati e i cambiamenti della commit specificata

Annullare commit

Elimina errori e altera lo storico dei cambiamenti

```
$ git reset [commit]
```

Annulla tutte le commit effettuate dopo [commit], preservando i cambiamenti locali

```
$ git reset --hard [commit]
```

Elimina tutto lo storico e i cambiamenti fino alla commit specificata

Sincronizzare i cambiamenti

Collegati a un URL remoto e ottieni lo storico dei cambiamenti

```
$ git fetch [remote]
```

Scarica lo storico dei cambiamenti dal repository remoto

```
$ git merge [remote]/[branch]
```

Unisci il branch remoto con quello locale

```
$ git push [remote] [branch]
```

Carica tutti i cambiamenti dal branch locale su remote repo

```
$ git pull
```

Scarica lo storico e unisci i cambiamenti

Copia il repositorio con il file yml e crea l’ambiente virtuale.

https://github.com/matib12/corso_gw

Referenze

Software generico per il calcolo scientifico

1) https://www.python.org/
2) https://docs.conda.io/ & https://anaconda.org/
3) https://jupyter.org/
4) https://numpy.org/
5) https://www.scipy.org/
6) https://matplotlib.org/
7) https://git-scm.com/doc

Software specifico per il campo delle OG

8) https://www.astropy.org/
9) https://einsteinpy.org/
10) https://gwpy.github.io/
11) https://pycbc.org/
12) https://git.ligo.org/lscsoft/lalsuite/
13) https://aladin.cds.unistra.fr/AladinLite/

Repositori

14) https://www.gw-openscience.org/
15) https://git.ligo.org/explore/projects/starred

Dove cercare aiuto tecnico

16) https://stackoverflow.com/