**Versions i proves unitàries en Git**

Objectiu

Aprendre a implementar proves unitàries en python per validar funcionalitats d'un projecte i gestionar diferents versions del codi utilitzant Git, etiquetes (tags) i releases.

En Git, els tags i les releases s’utilitzen per marcar punts importants en la història del projecte, com versions estables o llançaments significatius.

***Tags***

*Un tag (etiqueta) és un punt específic en la història del codi que normalment es fa servir per marcar una versió. Els tags són com "marcadors" que s’associen a un commit concret i es mantenen fixes en el temps. Hi ha dos tipus de tags en Git:*

*- Annotated tags: Inclouen metadades addicionals (autor, data i un missatge).*

*- Lightweight tags: Són simplement un nom assignat a un commit, sense informació addicional.*

*Exemple de com crear un tag:*

*git tag -a v1.0 -m "Primera versió estable"*

*git push origin v1.0*

***Releases***

*Una release és una versió oficial publicada del projecte, normalment associada a un tag. A GitHub, les releases són més completes, ja que permeten:*

*- Afegir notes de llançament que descriguin canvis, novetats i millores.*

*- Adjuntar binaris o altres fitxers per a la distribució de programari.*

*Les releases són útils per distribuir versions de codi preparades per a l’ús en producció o per a que altres desenvolupadors i usuaris finals puguin accedir-hi fàcilment.*

*En resum, els tags marquen versions específiques al codi, mentre que les releases proporcionen una manera estructurada de documentar i publicar aquestes versions.*

**1. Configurar el Repositori de Git**

- Crea un nou repositori remot a GitHub anomenat `gestio-versions`.

- Clona el repositori al teu sistema local:

**2. Crea el fitxer principal del projecte, `calculadora.py`:**

Haureu de completar el codi.

# calculadora.py

def sumar(a, b):

return ...

def restar(a, b):

...

- Afegeix aquest fitxer al repositori, fes un commit i puja’l al repositori remot:

**3. Crear Proves Unitàries**

- Crea un fitxer de proves, `test\_calculadora.py`, on implementarem les proves unitàries de les funcions de `calculadora.py`.

- Implementa les proves utilitzant `unittest` o `pytest`. Aquí tens un exemple amb `pytest`:

# test\_calculadora.py

from calculadora import sumar, restar

def test\_sumar():

assert sumar(2, 3) == 5

assert sumar(-1, 1) == 0

assert sumar(0, 0) == 0

def test\_restar():

assert restar(3, 2) == 1

assert restar(0, 0) == 0

assert restar(5, 10) == -5

- Executa les proves per assegurar-te que funcionen correctament:

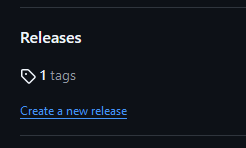
**4. Assignar una Etiqueta (Tag) a la Primera Versió**

- Un cop les proves han passat correctament, assigna una etiqueta (tag) a aquesta primera versió del projecte i puja-la a github.

- A GitHub, crea una release per la versió `v1.0`:

- Ves a la pàgina del repositori a GitHub, a la secció de Releases.

- Selecciona el tag `v1.0` i crea una nova release amb una descripció.



**5. Afegir Noves Funcionalitats i Proves**

- Crea una nova branca que es digui **mes-operacions** per implementar noves operacions, com `multiplicar()` i `dividir()`:

- Afegeix aquestes funcions a `calculadora.py`:

def multiplicar(a, b):

...

def dividir(a, b):

if b != 0:

...

else:

...

- Afegeix també proves per aquestes noves funcions en `test\_calculadora.py`:

def test\_multiplicar():

...

def test\_dividir():

...

- Executa de nou les proves per assegurar-te que les noves funcions funcionen correctament:

**6. Fusionar i Crear una Nova Versió**

-Fusiona la branca ` mes-operacions ` amb `main`:

- Assigna una nova etiqueta a aquesta versió (v2.0) i puja-la a GitHub.

- Crea una release a GitHub per la versió `v2.0`, amb una descripció de les noves funcionalitats.

**7. Treballar amb Múltiples Versions**

- Per accedir a una versió específica del codi, utilitza l'ordre `git checkout` seguit del tag de la versió:

**git checkout v1.0**

Així podràs veure les funcions implementades en aquella versió concreta i les proves associades.