Control de Versiones & Desarrollo basado en tests

Introducción

En este boletín vamos a practicar con algunos conceptos relativos a la creación y gestión de un repositorio git. El proyecto a alojar en dicho repositorio permitirá desarrollar una serie de pruebas unitarias mediante `pytest`.

Objetivos

- Creación de un repositorio git de forma local y en GitHub mediante línea de comandos.
- Gestión de los ficheros alojados en dicho repositorio mediante comandos git.
- Creación y gestión de diferentes `branches` en el repositorio.
- Creación y ejecución de pruebas unitarias mediante `pytest`.

Pasos guiados

1. Juego del tres en raya

Una vez que en la sesión anterior hemos dejado listo el entorno de desarrollo, con un repositorio git en marcha, vamos ahora a dicho repositorio programando el juego del 3 en raya. Para ello, primeros vamos a crear una nueva rama en nuestro repositorio llamada `tresenraya` con el comando `git branch tresenraya`

Podemos establecer ahora dicha rama como activa con `git checkout tresenraya`. Si ahora ejecutamos `git branch` vemos que efectivamente ahora tenemos dos ramas, `main` y `tresenraya`, siendo esta última en la que nos encontramos actualmente.

```
C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git>git branch
  main
* tresenraya
```

vamos en primer lugar a crear un fichero llamado `juego_3_en_raya.py` en nuestro directorio local `src`. Primero, implementaremos la función para mostrar el tablero por pantalla con una dimensión n y asumiendo que los jugadores han realizado movimientos (`movimientos_jugadores`).

```
fichas= ['o','x']

def mostrar_tablero(n, movimientos_jugadores):

    for i in range(n):
        for j in range(n):
            casilla_vacia = True
            for k in range(len(movimientos_jugadores)):
                movimientos_jugador= movimientos_jugadores[k]

                if i in movimientos_jugador:
                    if j in movimientos_jugador[i]:
                        print(fichas[k],end='')
                        casilla_vacia= False
            if casilla_vacia:
                print('_ ',end='')
        print('\n')(t, len(t))

if __name__ == "__main__":
    #Pedimos el tamaño del tablero en que se va a realizar el juego
    n=int(input('Introduce el tamaño del tablero cuadrado:'))

    casillas_libres = n*n
    jugador_activo = 0

    movimientos_jugador_1 = {}
    movimientos_jugador_2 = {}
    movimientos_jugadores = [movimientos_jugador_1, movimientos_jugador_2]

    mostrar_tablero(n,movimientos_jugadores)
```

Podemos testear que dicho código genera correctamente el tablero con las dimensiones adecuadas con el siguiente código

```
import pytest

@pytest.fixture
def tablero_dimension():
    return 3
```

```python
def test_mostrar_tablero(tablero_dimension, movimientos_ambos_jugadores, capsys):
    mostrar_tablero(tablero_dimension, movimientos_ambos_jugadores)
    captured = capsys.readouterr()
    lineas = captured.out.strip().split("\n")
    lineas= [l for l in lineas if l]
    assert len(lineas) == tablero_dimension
    for linea in lineas:
        assert len(linea.replace(' ', '')) == tablero_dimension
```

El código de arriba hace uso de una funcionalidad muy interesante de la librería `pytest` como son las `fixtures`. En `pytest`, una `fixture` es una función que proporciona datos, configuraciones o recursos que pueden ser reutilizados en múltiples tests. Se definen con el decorador @pytest.fixture y se inyectan automáticamente en los tests que las necesiten pasando su nombre como argumento.

Ahora debemos forzar a `pytest para` que evalúe nuestro fichero `juego_3_en_raya.py`. Para ello, en la línea de comandos escribimos

```
pytest juego_3_en_raya.py -v
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> pytest .\juego_3_en_raya.py -v
============================= test session starts =============================
platform win32 -- Python 3.11.4, pytest-7.4.3, pluggy-1.3.0 -- C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\Scripts\python.ex
cachedir: .pytest_cache
rootdir: C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src
collected 1 item

juego_3_en_raya.py::test_generar_tablero PASSED                          [100%]
```

La siguiente fase va a consistir en desarrollar un método que permita determinar si el movimiento de un jugador (colocar una ficha en la casilla con coordenadas (x,y)) es válido o no.

```python
def movimiento_valido(n, x, y, movimientos_otro_jugador):
    if x > n or y > n:
        return False

    if x in movimientos_otro_jugador:
        movimientos_en_columna= movimientos_otro_jugador[x]
        if y in movimientos_en_columna:
            return False

    return True
```

A partir de este método podemos diseñar varios test unitarios para comprobar su correcta funcionalidad junto con varias fixtures:

```python
@pytest.fixture
def movimientos_vacios():
    return {}, {}

@pytest.fixture
```

```python
def movimientos_vacios():
    return {}, {}

@pytest.fixture
def movimientos_ocupados():
    return {2: [3]}

@pytest.fixture
def movimientos_fuera_tablero(tablero_dimension):
    return tablero_dimension + 1, tablero_dimension + 1


def test_movimiento_columna_fuera_tablero(tablero_dimension, movimientos_vacios):
    movimientos_otro_jugador, _ = movimientos_vacios
    x = 1
    y = tablero_dimension + 1
    assert not movimiento_valido(tablero_dimension, x, y, movimientos_otro_jugador)


def   test_movimiento_fila_y_columna_fuera_tablero(tablero_dimension,   movimientos_vacios,
movimientos_fuera_tablero):
    movimientos_otro_jugador, _ = movimientos_vacios
    x, y = movimientos_fuera_tablero
    assert not movimiento_valido(tablero_dimension, x, y, movimientos_otro_jugador)


def test_movimiento_incorrecto(tablero_dimension, movimientos_ocupados):
    x = 2
    y = 3
    assert not movimiento_valido(tablero_dimension, x, y, movimientos_ocupados)
```

De nuevo, ejecutando

```
pytest juego_3_en_raya.py -v
```

podemos re-evaluar todos los tests

Sin embargo, vemos que con la instrucción anterior se está volviendo a re-evaluar el test `test_mostrar_tablero`, lo cual es algo redundante. En este caso podemos limitar la ejecución sólo de aquellas pruebas que contengan la cadena `movimiento` en su título. Para ello ejecutamos el comando

```
pytest juego_3_en_raya.py -v –k movimiento
```

```
========================================================== 5 passed in 0.02s ===========================================================
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> pytest .\juego_3_en_raya.py -v -k movimiento
========================================================== test session starts ==========================================================
platform win32 -- Python 3.11.4, pytest-7.4.3, pluggy-1.3.0 -- C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\Scripts\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src
collected 5 items / 1 deselected / 4 selected

juego_3_en_raya.py::test_movimiento_fila_fuera_tablero PASSED                                                                  [ 25%]
juego_3_en_raya.py::test_movimiento_columna_fuera_tablero PASSED                                                               [ 50%]
juego_3_en_raya.py::test_movimiento_fila_y_columna_fuera_tablero PASSED                                                        [ 75%]
juego_3_en_raya.py::test_movimiento_incorrecto PASSED                                                                          [100%]
```

En este punto ya podríamos subir pensar en subir la versión actual del juego a la rama `tresenraya` de nuestro repositorio git. Para ello, ejecutamos los siguientes comandos git en la línea de comandos:

```
git add juego_3_en_raya.py
git commit –m 'tablero y comprobacion de movimientos completada'
```

Si comprobamos nuestro repositorio en github veremos que la rama `tresenraya` no ha sido subida a github, para ello deberemos ejecutar el comando

```
git push –u origin tresenraya
```

```
PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git push -u origin tresenraya
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 946 bytes | 946.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'tresenraya' on GitHub by visiting:
remote:      https://github.com/fterroso/pcd_boletin1/pull/new/tresenraya
remote:
To https://github.com/fterroso/pcd_boletin1.git
 * [new branch]      tresenraya -> tresenraya
branch 'tresenraya' set up to track 'origin/tresenraya'.
```

Vamos ahora a implementar el método que permita determinar si un nuevo movimiento de un jugador le permite ganar el juego

```python
def jugada_ganadora(movimientos_jugador):
    #Comprobamos si hay 3 fichas en una fila
    for fila in movimientos_jugador:
        movimientos_columna = movimientos_jugador[fila]
        if len(movimientos_columna)==3:
            return True
    return False
```

Sobre este método vamos a definir dos test unitarios diferentes

```python
@pytest.fixture
def movimientos_no_ganador():
    return {2: [2, 3]}

@pytest.fixture
def movimientos_ganador():
    return {2: [1, 2, 3]}

def test_no_ganador(movimientos_no_ganador):
    assert not jugada_ganadora(movimientos_no_ganador)
```

```
def test_ganador(movimientos_ganador):
    assert jugada_ganadora(movimientos_ganador)
```

Que podemos ejecutar con el comando

```
pytest .juego_3_en_raya.py -v -k ganador
```

viendo que los dos test han sido superados satisfactoriamente.

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> pytest .\juego_3_en_raya.py -v -k ganador
============================================= test session starts =============================================
platform win32 -- Python 3.11.4, pytest-7.4.3, pluggy-1.3.0 -- C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\Scripts\pyt
hon.exe
cachedir: .pytest_cache
rootdir: C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src
collected 7 items / 5 deselected / 2 selected

juego_3_en_raya.py::test_no_ganador PASSED                                                           [ 50%]
juego_3_en_raya.py::test_ganador PASSED                                                              [100%]

========================================== 2 passed, 5 deselected in 0.03s ==========================================
```

Vamos a hacer un nuevo `commit` con los cambios realizados, pero esta vez vamos a usar un tag (`v0.5`) para etiquetar dicho `commit` y subirlo al repositorio remoto.

```
git add juego_3_en_raya.py

git commit –m 'añadida comprobación de jugada ganadora'

git tag v0.5

git push origin tresenraya:tresenraya
```

En este punto vamos a añadir la lógica del juego que permita hacerlo interactivo. Para ello, implementamos el bucle que permite a cada uno de los dos jugadores ir añadiendo movimientos:

```
if __name__ == "__main__":
    #Pedimos el tamaño del tablero en que se va a realizar el juego
    n=int(input('Introduce el tamaño del tablero cuadrado:'))

    casillas_libres = n*n
    jugador_activo = 0

    movimientos_jugador_1 = {}
    movimientos_jugador_2 = {}
    movimientos_jugadores = [movimientos_jugador_1, movimientos_jugador_2]

    mostrar_tablero(n,movimientos_jugadores)


    while casillas_libres > 0:

        casilla_jugador = input(f"JUGADOR {jugador_activo+1}: Introduce movimiento (x,y):
")
```

```
        casilla_jugador= casilla_jugador.strip()
        x= int(casilla_jugador.split(',')[0])-1
        y= int(casilla_jugador.split(',')[1])-1

        print(casilla_jugador,x,y)

        movimientos_jugador_activo= movimientos_jugadores[jugador_activo]
        movimientos_otro_jugador = movimientos_jugadores[(jugador_activo+1)%2]
        if movimiento_valido(x,y, movimientos_otro_jugador):
            mov_col= movimientos_jugador_activo.get(x,[])
            mov_col.append(y)
            movimientos_jugador_activo[x]= mov_col

            clear = lambda: os.system('cls')
            clear()
            mostrar_tablero(n, movimientos_jugadores)
            if jugada_ganadora(movimientos_jugador_activo):
                print(F"ENHORABUENA EL JUGADOR {jugador_activo+1} HA GANADO")
                break
        else:
            frequency = 2000 # Set Frequency To 2500 Hertz
            duration = 1000  # Set Duration To 1000 ms == 1 second
            print('\a')
            winsound.Beep(frequency, duration)
            print("Movimiento invalido. Turno para el siguiente jugador")


        casillas_libres= casillas_libres -1
        jugador_activo = (jugador_activo+1) % 2
```

Ahora podemos confirmar los cambios, crear un nuevo tag `v1.0` sobre dicho `commit` y subirlo todo al servidor remoto

```
git add juego_3_en_raya.py

git commit -m 'añadida primera version de la lógica del juego'

git tag v1.0

git push origin tresenraya:tresenraya
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git add juego_3_en_raya.py
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git commit -m 'añadida primera version de la lógica
 del juego'
[tresenraya 1d240da] añadida primera version de la lógica del juego
 1 file changed, 62 insertions(+), 3 deletions(-)
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git tag v1.0
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git push origin tresenraya:tresenraya
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 1.15 KiB | 1.15 MiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/fterroso/pcd_boletin1.git
   735a16c..1d240da  tresenraya -> tresenraya
```

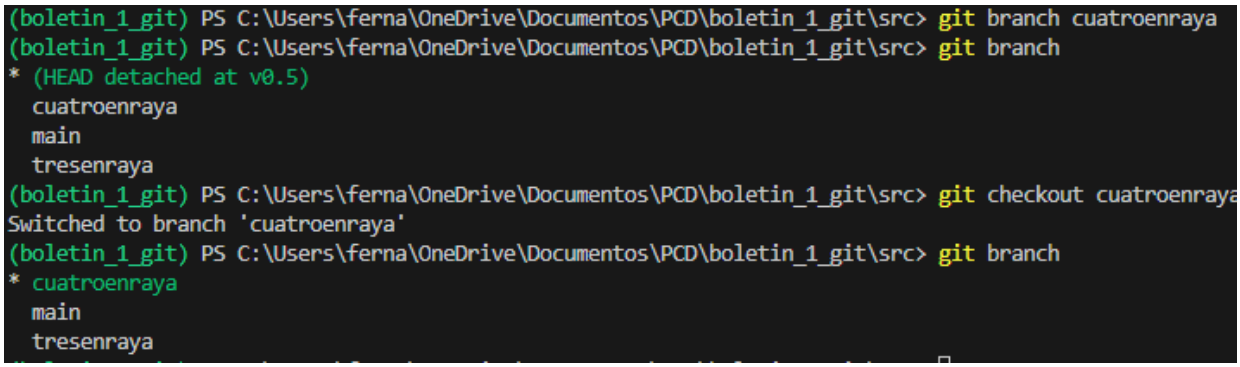
2. *Trabajando con ramas y tags*

En este punto tenemos en nuestro repositorio dos ramas diferentes: `master` y `tresenraya`. Además, esta última contiene dos de sus commits etiquetados (`v0.5` y `v1.0`). Estos tags nos permiten movernos fácilmente entre las diferentes versiones del juego. Así ejecutando...

```
git checkout v0.5
```

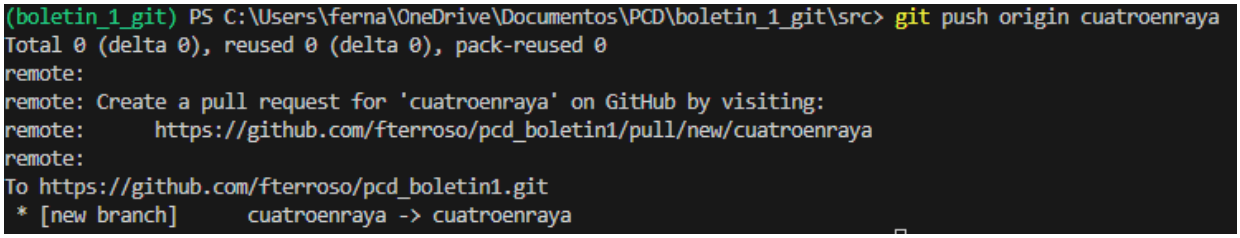
...restauraremos el proyecto tal y como estaba al hacer dicho *commit* y podríamos crear una nueva rama `4enraya` a partir de dicho punto:

```
git branch cuatroenraya
git checkout cuatroenraya
```



Posteriormente, podemos subir dicha rama al servidor remoto con

```
git push origin cuatroenraya
```



En este punto podríamos empezar a trabajar en un nuevo desarrollo para el juego del 4 en raya a partir del estado actual del proyecto.

Nosotros, sin embargo, vamos a volver al HEAD de la rama principal pues lo que vamoa a hacer es unirla con la de 3enraya con

```
git checkout main
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git branch
  cuatroenraya
* main
  tresenraya
```

Ahora podemos fusionar ambas ramas en una única `main` con

```
git merge tresenraya
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git merge tresenraya
Updating a764b2d..1d240da
Fast-forward
 src/juego_3_en_raya.py | 153 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 file changed, 153 insertions(+)
 create mode 100644 src/juego_3_en_raya.py
```

Subimos los cambios al servidor remoto con

```
git push origin main
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git push origin
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/fterroso/pcd_boletin1.git
   a764b2d..1d240da  main -> main
```

Ahora borramos la rama `tresenraya` pues ya no nos hace falta

```
git branch –d tresenraya
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git branch -d tresenraya
Deleted branch tresenraya (was 1d240da).
```

Finalmente borramos también dicha rama en nuestro servidor local con

```
git push origin –-delete tresenraya
```

```
(boletin_1_git) PS C:\Users\ferna\OneDrive\Documentos\PCD\boletin_1_git\src> git push origin --delete tresenraya
To https://github.com/fterroso/pcd_boletin1.git
 - [deleted]         tresenraya
```

Ejercicios opcionales

1. Como habrás comprobado, la función que comprueba si un jugador ha ganado una partida al 3 en raya, `jugada_ganadora`, está incompleta. Termina dicha función y sube los cambios a la rama main de tu repositorio.
2. Implementa la versión del 4 en raya a partir del *commit* inicial de la rama git `cuatroenraya`. Ejecuta los comandos necesarios para situarte en dicha rama.