Pruebas con Pytest:

Las estrategias, para analizar la eficacia de cada algoritmo, están descritas en el archivo **README.pdf**

Prueba 1: test_Agregar ().

Escenario I: No existe un archivo llamado **“prueba.txt”** dentro de la carpeta.

```
[root@localhost home]# cd jenkinUser
[root@localhost jenkinUser]# ls
cobertura.sh  comandos.py  comandos.pyc  miAmbiente  __pycache__  test_comandos.py  test_comandos.pyc
[root@localhost jenkinUser]#
```

Las 3 pruebas, incluyendo el test_agregar () sale exitoso

```
[root@localhost jenkinUser]# pytest test_comandos.py
============================================= test session starts ==============================================
platform linux2 -- Python 2.6.6, pytest-3.0.3, py-1.4.31, pluggy-0.4.0
rootdir: /home/jenkinUser, inifile:
collected 3 items

test_comandos.py ...

========================================== 3 passed in 0.05 seconds ===========================================
[root@localhost jenkinUser]#
```

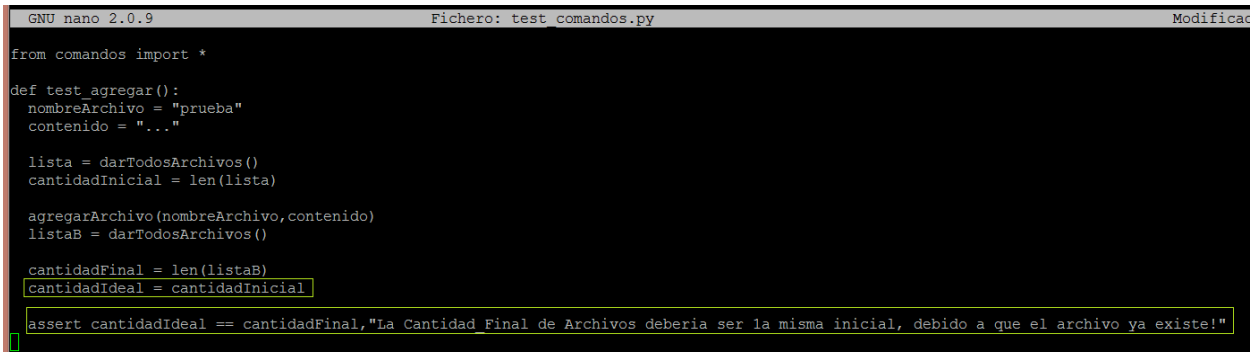
Escenario II: Existe un archivo llamado **“prueba.txt”**, dentro de la carpeta.

```
[root@localhost jenkinUser]# ls
cobertura.sh  comandos.py  comandos.pyc  miAmbiente  prueba.txt  __pycache__  test_comandos.py  test_comandos.pyc
[root@localhost jenkinUser]#
```

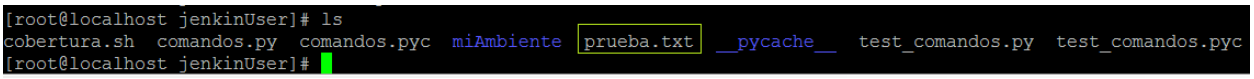
En este caso la prueba falla porque el algoritmo de test_Agregar () solo tiene en cuenta el caso cuando **“prueba.txt”** no existe, para simplificar el código.

```
    def test_agregar():
      nombreArchivo = "prueba"
      contenido = "..."

      lista = darTodosArchivos()
      cantidadInicial = len(lista)

      agregarArchivo(nombreArchivo,contenido)
      listaB = darTodosArchivos()

      cantidadFinal = len(listaB)
      cantidadIdeal = cantidadInicial + 1

>     assert cantidadIdeal == cantidadFinal,"La Cantidad_Final de Archivos deberia ser = La Cantidad_Inicial + 1"
E     AssertionError: La Cantidad_Final de Archivos deberia ser = La Cantidad_Inicial + 1
E     assert 9 == 8

test_comandos.py:16: AssertionError
====================================== 1 failed, 2 passed in 0.06 seconds ======================================
```

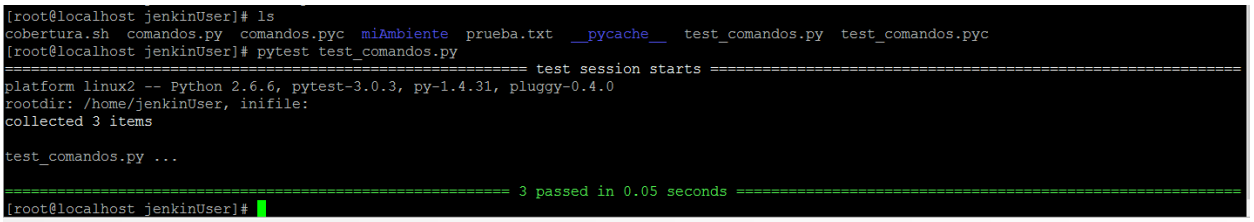
Para tener en cuenta este 2do caso, es necesario modificar el código de la sgte manera:



Aquí la variable, que cuenta la cantidad de Archivos, ya no se incrementa, porque no se creó un archivo nuevo, debido a que ya existía.

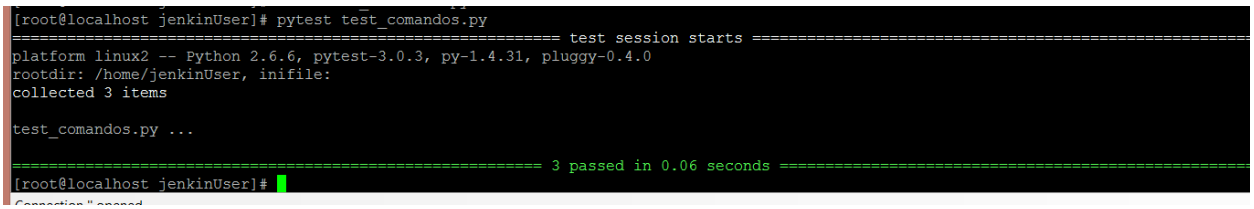


Ahora la prueba si sale exitosa, para el escenario 2



Prueba 2: test_Borrar ()

Escenario I: Existe un archivo llamado **“prueba.txt.”,** dentro de la carpeta.



Tengamos en cuenta que los algoritmos de prueba se ejecutan en el sgte orden:

1. test_agregar ()
2. test_borrar ()
3. test_darTodos ().

Este algoritmo (borrar) pasa exitosamente la prueba, debido a que test_agregar () genera primero el archivo (por el orden de ejecución, descrito arriba) y, cuando el algoritmo test_borrar se va a efectuar, este archivo ya está dentro de la carpeta, dando lugar al escenario I.

Escenario II: No existe un archivo llamado “prueba.txt.” dentro de la carpeta. Se hizo una modificación al test_comandos.py para que el algoritmo test_agregar () no generara el archivo “prueba.txt”.

```
[root@localhost jenkinUser]# ls
cobertura.sh  comandos.py  comandos.pyc  miAmbiente  __pycache__  test_comandos.py  test_comandos.pyc
[root@localhost jenkinUser]# pytest test_comandos.py
============================= test session starts ==============================
platform linux2 -- Python 2.6.6, pytest-3.0.3, py-1.4.31, pluggy-0.4.0
rootdir: /home/jenkinUser, inifile:
collected 2 items

test_comandos.py F.

=================================== FAILURES ===================================
_____________________________ test_borrar ______________________________

    def test_borrar():
      nombreArchivo = "prueba.txt"

      lista = darTodosArchivos()
      cantidadInicial = len(lista)

      borrarArchivo(nombreArchivo)
      listaB = darTodosArchivos()

      cantidadFinal = len(listaB)
      cantidadIdeal = cantidadInicial - 1

>     assert cantidadIdeal == cantidadFinal,"La Cantidad_Final de Archivos deberia ser = La Cantidad_Inicial - 1"
E     AssertionError: La Cantidad_Final de Archivos deberia ser = La Cantidad_Inicial - 1
E     assert 6 == 7

test_comandos.py:30: AssertionError
========================== 1 failed, 1 passed in 0.06 seconds ==========================
[root@localhost jenkinUser]#
```

Para este caso, es normal que la prueba falle, porque la estrategia asume que el archivo ya existe y que fue eliminado con éxito de la carpeta (por eso la variable contadora disminuye en 1). Como simplificación de código de prueba, no se incluye el escenario donde el archivo no existe.

Para que el algoritmo apruebe este escenario, se hace algo análogo al caso del test_agregar (): La variable contadora ya no se disminuye en 1, sino que queda igual, debido a que no se borró nada (y por ende la cantidad de archivos en la carpeta quedó exacta).

De lo anterior resulta:

```
def test_borrar():
  nombreArchivo = "prueba.txt"

  lista = darTodosArchivos()
  cantidadInicial = len(lista)

  borrarArchivo(nombreArchivo)
  listaB = darTodosArchivos()

  cantidadFinal = len(listaB)
  cantidadIdeal = cantidadInicial

  assert cantidadIdeal == cantidadFinal
```

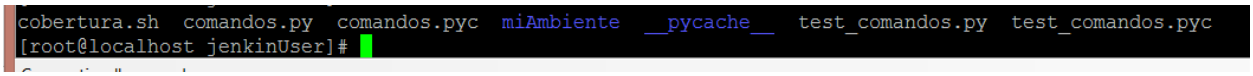
La imagen anterior muestra el algoritmo test_borrar () modificado, para que acepte este 2do escenario.

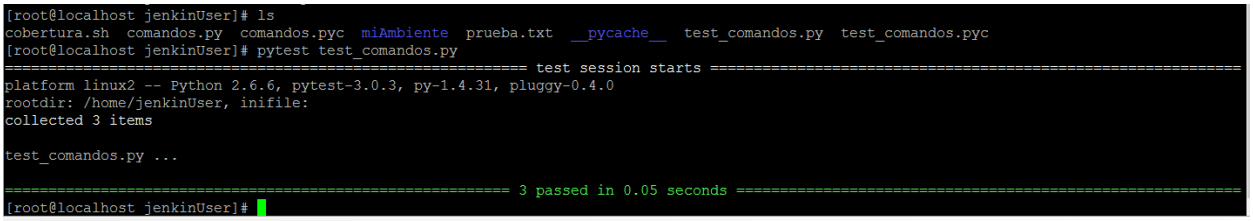
La imagen sgte muestra que la prueba fue exitosa (recordemos que solo se ejecuta el test_borrar () y el test_darTodos () porque el test_agregar () no se ejecutó en este caso, para no crear el archivo **“pruebas.txt”**)

```
============================================================ test session starts ============
platform linux2 -- Python 2.6.6, pytest-3.0.3, py-1.4.31, pluggy-0.4.0
rootdir: /home/jenkinUser, inifile:
collected 2 items

test_comandos.py ..

========================================================= 2 passed in 0.05 seconds ========
[root@localhost jenkinUser]#
```

Prueba 3: test_darTodos()

Escenario I: Ninguno de los sgtes archivos existe, dentro de la carpeta: “primero.txt”, “segundo.txt”, “Tercero.txt”, “Cuarto.txt”.





La prueba es exitosa, sin ningún problema

Escenario II: Alguno de los archivos anteriores existe.

Ocurre lo mismo que los dos algoritmos anteriores (agregar y borrar). Si alguno ya existe dentro de la carpeta, la prueba falla porque no tiene en cuenta el caso donde alguno de los archivos exista.

```
[root@localhost jenkinUser]# pytest test_comandos.py
============================================================ test session starts ============================================================
platform linux2 -- Python 2.6.6, pytest-3.0.3, py-1.4.31, pluggy-0.4.0
rootdir: /home/jenkinUser, inifile:
collected 3 items

test_comandos.py ..F

================================================================= FAILURES ==================================================================
_____________________________________________________________ test_darTodos _____________________________________________________________

    def test_darTodos():

        primerMuestreo = len (darTodosArchivos())

        agregarArchivo("Primero","...");
        agregarArchivo("Segundo","...");

        segundoMuestreo = len(darTodosArchivos())

        borrarArchivo("Primero.txt");
        agregarArchivo("Tercero","...");
        agregarArchivo("Cuarto","...");
        agregarArchivo("Primero","...");

        tercerMuestreo = len(darTodosArchivos());

        borrarArchivo("Primero.txt");
        borrarArchivo("Segundo.txt");
        borrarArchivo("Tercero.txt");
        borrarArchivo("Cuarto.txt");

        cuartoMuestreo = len (darTodosArchivos());

>       assert primerMuestreo == cuartoMuestreo and primerMuestreo + 2 == segundoMuestreo and primerMuestreo + 4 == tercerMuestreo
E       assert (8 == 7)

test_comandos.py:55: AssertionError
========================================================= 1 failed, 2 passed in 0.07 seconds ==========================================
```