# Equipo 5: Prueba Unitaria en Unittest

## Descripción:

**Lenguaje a emplear:**

* Python

**Paqueterías o librerías a emplear:**

* Unittest

# Código a probar:

En esta sección se realizarán distintas pruebas a una sección de un código en un proyecto determinado de Python utilizando unittest.

En el siguiente código se tendrá una función que su función principal es realizar la suma del 1 hasta un numero ingresado.

Esta función recibirá como parámetro un numero donde dicho parámetro por defecto será ‘None’ o nulo, después se verificará si este número es ‘None’ o nulo, si lo es devolverá la suma de los números del numero 1 al numero 100, pero si no es ‘None’ o nulo, se devolverá la suma de los números del numero 1 al numero ingresado.

Texto

Descripción generada automáticamente

# Entradas y salidas:

Antes de realizar las pruebas es importante realizar la importación de la librería unittest así como el código al que vamos a probar.

Texto

Descripción generada automáticamente

**Prueba 1:** En esta prueba se realizarán diferentes pruebas donde el objetivo es verificar el resultado cuando no se le pase ningún parámetro a la función o el parámetro es nulo. Primero compararemos el resultado si no le pasamos ningún parámetro a la función y lo compararemos con el valor esperado que es la suma del numero 1 al número 100 (5050).

Después realizaremos una segunda prueba donde compararemos la función si le pasamos como parámetro ‘None’ o nulo y lo compararemos con el valor esperado que es la suma del numero 1 al número 100 (5050).

La salida esperada de estas pruebas es un aprobado o un ‘OK’, esto nos indica que se realizaron las pruebas con éxito y que todas las pruebas dieron un resultado esperado.

Texto

Descripción generada automáticamente Fondo negro con letras blancas

Descripción generada automáticamente con confianza media

**Prueba 2:** En estas prueba se realizaran diferentes pruebas donde el objetivo es probar a la funcion con distintos parametros de todo tipo. Primeramente se ingresara la sumatoria de los numeros del 1 al 10 como parametro y lo compararemos con el valor esperado (55).

Despues ingresaremos una lista de numeros [1, 2, 3] como parametro y lo compararemos con su valor esperado (6).

Despues se ingresaran una tupla de numeros (1,2,3) como parametro y se compararan con su valor esperado (6).

Finalmente se ingresara una lista vacia [] como parametro y se comparara con su valor esperado (0), se espera que sea 0 ya que una lista vacia no es un valor nulo.

La salida esperada de estas pruebas es un aprobado o un ‘OK’, esto nos indica que se realizaron las pruebas con éxito y que todas las pruebas dieron un resultado esperado. Texto

Descripción generada automáticamenteForma, Rectángulo

Descripción generada automáticamente

# Discusión Equipo 5, pruebas unitarias en Unittest:

Experiencias Obtenidas:

Durante la ejecución de las pruebas en el código en Python utilizando la biblioteca Unittest, se llevaron a cabo diversas experiencias para evaluar el comportamiento de la función sum\_numbers. Se observaron casos en los que no se proporcionaron parámetros a la función o se le pasó None, lo que generó un resultado esperado de 5050. Además, se sometió la función a pruebas con una variedad de tipos de parámetros, como rangos, listas y tuplas, y se compararon con los valores esperados correspondientes. Estas experiencias permitieron comprender cómo la función reacciona a diferentes entradas y si produce los resultados deseados en cada caso.

Resultados Obtenidos:

Los resultados de las pruebas en el código fueron en general satisfactorios. En el caso de la primera prueba, cuando no se proporcionaron parámetros o se pasó None, la función sum\_numbers produjo el resultado esperado de 5050, lo que indica que la función maneja correctamente estas situaciones. En la segunda prueba, se probaron distintos tipos de parámetros, como rangos, listas y tuplas, y se compararon con los valores esperados, lo que resultó en una ejecución exitosa de las pruebas, confirmando que la función es capaz de manejar múltiples tipos de entradas y generar resultados coherentes en todos los casos. Estos resultados son esenciales para verificar la fiabilidad de la función y su capacidad para adaptarse a diferentes situaciones de entrada.

# Equipo 6: Prueba Unitaria en Pytest

## Descripción:

**Lenguaje a emplear:**

* Python

**Paqueterías o librerías a emplear:**

* Django
* Pytest

# Entradas y salidas:

**Prueba 1:** En estas pruebas se enfocaran principalmente al uso de la librería pytest. En esta prueba se evaluaría si un numero es igual a otro, en este caso se evaluara si el primer numero (numero 1) es igual al segundo numero (numero 2). La salida esperada es que el test sea aprobado ya que los dos números son iguales.

**Texto

Descripción generada automáticamente con confianza media**

**Texto

Descripción generada automáticamente**

**Prueba 2:** En esta prueba se evaluara la misma prueba vista en la Prueba 1 donde se compararan dos números (El numero 1 y otra vez el numero 1), a su vez tendremos otra segunda función donde se compararan otros dos números (El numero 1 y el numero 2). En este caso la salida esperada debería de ser que las pruebas fallaron con una prueba pasada y otra fallida, ya que aunque la primera función sea correcta la segunda función es falsa.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**Prueba 3:** En esta prueba ahora se invertieron las funciones, ahora se correra la primera función donde se compararan dos números iguales (el numero 1 y el numero 2) y otra función donde se compararan otros dos números iguales (el numero 1 y el numero 1). La salida esperada de esta función va a ser fallida, sin embargo a diferencia de la prueba anterior, este código se parará al fallar en la primera función y no probara la siguiente función.

Texto

Descripción generada automáticamente

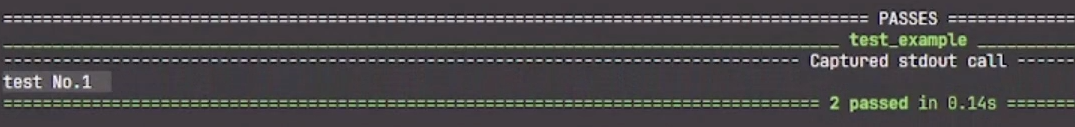
Captura de pantalla de un celular con letras

Descripción generada automáticamente

**Prueba 4:** En esta prueba ahora tendremos dos funciones idénticas donde se compararán un numero con el mismo numero (el numero 1 y el numero 1) pero en esta ocasión se tendrá en la primera función que antes de realizar la comparación se imprimirá una cadena de texto “test No.1”. La salida esperada es que la prueba se pase con una impresión en la consola de “test No.1”.

Texto

Descripción generada automáticamente



**Prueba 5:** En esta prueba se utilizará las mismas funciones que la prueba anterior. Ahora añadiremos @pytest.mark.skip lo que indicara que se saltara la primera función. La salida esperada es 1 función pasada y 1 función salteada pero como se salteo la primera función, no tendremos impreso en la consola la cadena “test No.1” a diferencia de la prueba anterior.

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Prueba 6:** En esta prueba cambiaremos la línea @pytest.mark.skip por @pytest.mark.xfail, esto es para indicar a una función que es posible que pueda fallar. La salida esperada es una prueba pasada y una prueba ‘xpassed’.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente con confianza media

**Prueba 7:** En esta prueba cambiaremos la línea @pytest.mark.xfail por @pytest.mark.slow, esto es para marcar las pruebas que son lentas o afectan al tiempo de ejecución total del conjunto de pruebas. La salida esperada es una prueba pasada y una prueba deseleccionada.

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

# Discusión Equipo 6, pruebas unitarias en Pytest:

Experiencias Obtenidas:

Durante la ejecución de las pruebas en Python con el uso de Django y Pytest, se observaron distintas situaciones que proporcionaron información valiosa. Se identificaron casos en los que las pruebas de igualdad de números resultaron en aprobación y, en contraste, se evidenciaron situaciones en las que algunas pruebas fallaron, lo que permitió detectar problemas en el código. Además, se exploraron las características de los marcadores Pytest, como @pytest.mark.skip, @pytest.mark.xfail, y @pytest.mark.slow, lo que enriqueció la comprensión de cómo gestionar y categorizar pruebas en un entorno de desarrollo de Python.

Resultados Obtenidos:

Los resultados de las pruebas realizadas en el código revelaron un comportamiento diverso. Se obtuvo una validación exitosa en los escenarios de igualdad de números, confirmando la integridad de las funciones relacionadas. Sin embargo, se detectaron situaciones en las que las pruebas fallaron, lo que señala la presencia de errores o incoherencias en el código. La utilización de los marcadores Pytest, como @pytest.mark.skip, @pytest.mark.xfail y @pytest.mark.slow, permitió gestionar las pruebas de manera efectiva, logrando resultados acordes con los objetivos de cada marcador. Estos resultados son esenciales para evaluar la calidad y confiabilidad del código bajo prueba y para orientar futuros ajustes y mejoras.

# Propuesta Python, Checador de productos:

La función a probar en Python será la del ‘checker’ la cual su función es la de checar o verificar si el numero a ingresar en una maquina expendedora para elegir un producto sea correcta o no. La función es el siguiente:  
Texto

Descripción generada automáticamente

El propósito de esta función es verificar si ingresamos un numero correcto de los números que existen en el inventario de la maquina expendedora, si ingresamos un numero incorrecto nos dirá la maquina expendedora que el numero es incorrecto y lo tendremos que volver a intentar, este ciclo continuará hasta que ingresemos un numero correcto.

Para probar la función anterior realizaremos pruebas unitarias con Unittest de la siguiente manera:

Texto

Descripción generada automáticamente

El objetivo de realizar estas pruebas unitarias es garantizar que la función checker se comporte correctamente y devuelva resultados esperados en diferentes situaciones. Las pruebas unitarias te ayudarán a detectar errores y asegurar la calidad de la función y así evitar errores en la compra de productos.

En estas pruebas unitarias, se evalúan varios aspectos de la función checker. La función test\_valid\_choice prueba si la función devuelve el número seleccionado cuando la entrada es válida. La función test\_invalid\_choice verifica si la función devuelve el mensaje "wrong input, try again" cuando se proporciona una entrada fuera de rango. Finalmente, la función test\_non\_integer\_input verifica si la función devuelve el mensaje "wrong input. Try again" cuando se proporciona una entrada no numérica.

Los criterios de aceptación de cada prueba son los siguientes:

* **test\_valid\_choice:**
  + **Entrada**: Se proporciona una opción válida, por ejemplo, "2".
  + **Resultado Esperado:** La función debería devolver el número seleccionado, que en este caso es 2.
  + **Criterio de Éxito:** La función devuelve el número seleccionado correctamente.
* **test\_invalid\_choice:**
  + **Entrada**: Se proporciona una opción fuera de rango, por ejemplo, "5".
  + **Resultado Esperado:** La función debería devolver el mensaje "wrong input, try again".
  + **Criterio de Éxito**: La función devuelve el mensaje de error correspondiente cuando se proporciona una opción fuera de rango.
* **test\_non\_integer\_input:**
  + **Entrada**: Se proporciona una opción no numérica, por ejemplo, "abc".
  + **Resultado Esperado:** La función debería devolver el mensaje "wrong input. Try again".
  + **Criterio de Éxito:** La función devuelve el mensaje de error correspondiente cuando se proporciona una entrada no numérica.