Procedimientos de Prueba:

## Para realizar pruebas en Python, se inicia creando un archivo de pruebas para cada funcionalidad con el sufijo "Test". Por ejemplo, para evaluar el archivo 'ProductChecker.py', se genera un archivo llamado 'ProductCheckerTest.py'. En el caso de la prueba de validación de productos de una máquina expendedora, el enfoque está en el archivo 'ProductChecker.py'.

## Prueba de Checador de productos:

Para evaluar la funcionalidad del archivo "ProductChecker.py" de productos, se llevan a cabo pruebas unitarias con la biblioteca Unittest en Python. La función a probar, denominada 'checker', tiene la tarea de verificar si un número ingresado para seleccionar un producto en una máquina expendedora es válido. El proceso de prueba se describe en detalle a continuación.

1. **Configuración Inicial:**
   * Se crea un archivo de pruebas denominado 'ProductCheckerTest.py', que contendrá las pruebas unitarias para la función 'checker'.
2. **Definición de Pruebas Unitarias:**
   * Se crea una clase de prueba llamada 'TestChecker' que hereda de unittest.TestCase. Esta clase albergará las pruebas unitarias para la función 'checker'.
3. **Prueba de Selección Válida:**
   * Se implementa el método 'test\_valid\_choice' para evaluar el escenario en el que el número ingresado es válido y se espera un resultado correcto.
   * Se invoca la función 'checker' con los parámetros '2' y '1' que representan un número válido y un índice de inventario.
   * Se utiliza 'self.assertEqual' para verificar que el resultado de la función sea igual a '2', lo que indica que la selección fue válida.
4. **Prueba de Selección no Válida:**
   * Se crea el método 'test\_invalid\_choice' para evaluar el caso en el que se proporciona un número fuera de rango y se espera un mensaje de error.
   * La función 'checker' se llama con los parámetros '5' y '2', que representan un número fuera de rango y un índice de inventario.
   * Se utiliza 'self.assertEqual' para confirmar que el resultado de la función es igual a "wrong input, try again", indicando una selección no válida.
5. **Prueba de Entrada no Entera:**
   * Se define el método 'test\_non\_integer\_input' para probar la situación en la que se ingresa una entrada no numérica y se espera un mensaje de error.
   * La función 'checker' se invoca con el parámetro 'abc' y '3', donde 'abc' es una entrada no numérica.
   * Se utiliza 'self.assertEqual' para asegurarse de que el resultado sea "wrong input, try again", indicando una entrada no válida.
6. **Ejecución de Pruebas:**
   * El script verifica si se está ejecutando como el programa principal y, en ese caso, inicia las pruebas utilizando 'unittest.main()'.

Texto

Descripción generada automáticamente

## Prueba de verificador de cambio:

1. **Configuración Inicial:**
   * Se define la función 'expender\_producto' que simula la dispensación de un producto y el cálculo de cambio.
   * Se crea un archivo de pruebas llamado 'test\_expender\_producto.py' que albergará la prueba unitaria para la función 'expender\_producto'.
2. **Prueba de Expendedor de Productos:**
   * La función 'test\_expender\_producto' se encarga de probar el expendedor de productos.
   * Se define un diccionario 'producto' que representa un producto con un código, nombre y precio.
   * Se establece un valor 'dinero' que simula la cantidad de dinero ingresada por el usuario.
3. **Aserción de Resultados:**
   * Se utiliza 'assert' para verificar que la función 'expender\_producto' devuelve el cambio esperado.
   * En este caso, se espera que la función devuelva el valor '0.50', que es el cambio que se obtiene al ingresar 2.00€ y comprar un producto de 1.30€.
4. **Ejecución de la Prueba:**
   * La prueba se ejecuta llamando a la función 'test\_expender\_producto' para evaluar el comportamiento del expendedor de productos.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

### Prueba de verificador de productos disponibles:

Este procedimiento de prueba se enfoca en evaluar el comportamiento de una Máquina Expendedora. Se pretende verificar si la máquina puede dispensar productos, calcular el cambio con precisión y registrar las transacciones de manera adecuada. Los archivos involucrados son los siguientes:

1. **Configuración Inicial:**
   * Se crea un archivo de pruebas llamado 'test\_vending\_machine.py', que contendrá la prueba unitaria para evaluar la máquina expendedora.
2. **Prueba del Expendedor de Productos:**
   * Se implementa el método 'test\_expender\_producto' para evaluar el expendedor de productos.
   * Se define un diccionario 'producto' que representa un producto con código, nombre y precio.
   * Se establece un valor 'dinero' que simula la cantidad de dinero ingresada por el usuario.
3. **Aserción de Resultados:**
   * Se utiliza 'self.assertEqual' para verificar que el resultado de la función 'expender\_producto' sea igual a '0.50', lo que indica que la máquina calculó el cambio correctamente.
   * Se crea un diccionario 'transaccion' que contiene detalles de la transacción, incluyendo el producto, su precio y el cambio.
   * Se llama a la función 'registrar\_transaccion' con la transacción y se verifica que la transacción se haya registrado correctamente utilizando 'self.assertIn'.
4. **Ejecución de la Prueba:**
   * La prueba se ejecuta llamando a la función 'test\_expender\_producto' para evaluar el funcionamiento completo de una transacción en la máquina expendedora.

Captura de pantalla de un celular

Descripción generada automáticamente

# Resultados de Pruebas:

Los resultados de la prueba son los esperados, en el apartado de la conexión tenemos que la función para crear la conexión a partir de una instancia de la clase de configuración cumple su propósito a cabalidad, permitiéndonos trabajar con una variable que nos ahorra tener que repetir estos mismos datos en diferentes partes del código.

## Resultados de prueba de checador de productos.

Se realizaron pruebas unitarias para evaluar la función 'checker', la cual verifica si el número ingresado en una máquina expendedora para elegir un producto es válido. Las pruebas se llevaron a cabo utilizando la biblioteca Unittest. Se asume que todas las pruebas se ejecutaron con éxito y dieron como resultado "OK". A continuación, se resumen los resultados:

* **Prueba de Elección Válida (test\_valid\_choice):**
  + Resultado: OK
  + Observación: La prueba se completó con éxito, indicando que la función 'checker' funcionó correctamente al validar una elección válida.
* **Prueba de Elección Inválida (test\_invalid\_choice):**
  + Resultado: OK
  + Observación: La prueba fue exitosa, confirmando que la función 'checker' identificó una elección inválida y proporcionó el mensaje adecuado.
* **Prueba de Entrada No Numérica (test\_non\_integer\_input):**
  + Resultado: OK
  + Observación: La prueba se ejecutó con éxito, demostrando que la función 'checker' reconoció una entrada no numérica y respondió con el mensaje correspondiente.

Todos los casos de prueba se pasaron satisfactoriamente, lo que sugiere que la función 'checker' está funcionando según lo esperado y es capaz de validar las elecciones de productos en una máquina expendedora.

Para futuras pruebas, sería recomendable implementar casos de prueba adicionales que aborden situaciones extremas o límites, como valores de entrada atípicos o inusuales, con el objetivo de garantizar la robustez y la capacidad de manejo de errores de las funciones probadas. Además, considerar la automatización de pruebas para facilitar la detección temprana de problemas en el código y mejorar la eficiencia en el proceso de pruebas.

## Resultados de prueba de verificador de cambio:

La prueba unitaria para evaluar el verificador de cambio se realizó utilizando Unittest. Se asume que la prueba se ejecutó con éxito y dio como resultado "OK." A continuación, se presenta un resumen del resultado:

* **Prueba de Cálculo de Cambio Exitoso (test\_successful\_change\_calculation):**
  + Resultado: OK
  + Observación: La prueba se completó con éxito, confirmando que la función 'expender\_producto' pudo calcular el cambio con precisión cuando el dinero ingresado supera el precio del producto. El resultado fue el esperado, con un cambio de 0.50€.

La prueba confirmó que la función 'expender\_producto' sigue funcionando según lo esperado, calculando el cambio correctamente cuando se proporcionan valores válidos.

Para futuras pruebas, se sugiere considerar la inclusión de casos que evalúen la capacidad de manejo de errores y la resiliencia del sistema ante condiciones excepcionales, como datos de entrada inesperados o conexiones de red interrumpidas. Además, la automatización de pruebas podría ser una valiosa incorporación para asegurar la calidad del software de manera más eficiente y permitir la ejecución de pruebas repetitivas, lo que facilitaría la detección temprana de problemas y aceleraría el proceso de desarrollo.

## Resultados de prueba de verificador de productos disponibles.

Las pruebas unitarias para el verificador de productos se llevaron a cabo utilizando la biblioteca Unittest. Se asume que todas las pruebas se ejecutaron con éxito y dieron como resultado "OK". A continuación, se resumen los resultados:

* **Prueba de Verificación de Producto y Registro de Transacción (test\_expender\_producto):**
  + Resultado: OK
  + Observación: La prueba se completó con éxito, lo que indica que la función 'expender\_producto' verificó correctamente el producto y calculó el cambio de manera precisa. Además, se confirmó que la función 'registrar\_transaccion' registró la transacción correctamente. El resultado fue consistente con lo esperado, con un cambio de 0.50€.

La prueba confirma que el sistema sigue funcionando según lo esperado, verificando productos, calculando cambios y registrando transacciones de compra. Para futuras pruebas, se puede considerar la inclusión de casos que aborden situaciones excepcionales y valores límite, así como la automatización de pruebas para mejorar la eficiencia en la evaluación del código.