

**Fundamentos de Python**

Unidad 4. PROGRAMACION ORIENTADA A OBJETOS

PROJECTOS

1. Calculadora básica con POO

ESTUDIANTE

Armando Gabriel Jacinto López

INSTITUCION

INFOTEC

FECHA

21/09/2025

**1.- Introducción**

El presente informe técnico detalla un programa en Python diseñado para funcionar como una calculadora interactiva. Este sistema no solo realiza las cuatro operaciones aritméticas fundamentales (suma, resta, multiplicación y división), sino que también gestiona la interacción con el usuario de manera robusta y amigable. El código está estructurado en un enfoque de Programación Orientada a Objetos (POO), lo que permite una organización lógica del código, y separa las funcionalidades en módulos distintos: una clase principal para las operaciones de la calculadora y funciones para el control del flujo del programa y la validación de las entradas.

El programa cumple con todos los requisitos establecidos, incluyendo la gestión de un historial de operaciones, el manejo de errores comunes como la división por cero y la entrada de datos no válidos, y un bucle continuo que permite múltiples operaciones en una sola sesión.

**2.- Desarrollo**

El programa está compuesto por una clase principal, Calculator, y dos funciones, expressions y main, que se encargan de la lógica principal, la validación y el control del flujo.

**2.1.- La Clase Calculator**

Esta clase es el núcleo del programa. Su objetivo es encapsular toda la lógica relacionada con las operaciones matemáticas y el registro del historial.

* **Método Constructor (\_\_init\_\_)**: Este método se llama cuando se crea un nuevo objeto de la clase. Inicializa tres atributos "protegidos" con un guion bajo: \_records, una lista vacía que almacenará el historial de operaciones, y \_num1 y \_num2, que se inicializan a 0 para guardar los números que se van a operar.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 1 Clase Calculator

* **Getters y Setters (@property)**: El código utiliza decoradores (@property) para crear getters y setters. Estos métodos permiten acceder y modificar los atributos de la clase de manera controlada. Por ejemplo, el setter para num1 y num2 incluye una validación que asegura que el valor que se le asigna es de tipo int o float. Si no es así, lanza un error (raise ValueError), lo cual es una excelente práctica para la gestión de errores.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 2 Getters, y Setters

* Texto

  El contenido generado por IA puede ser incorrecto.**Método de Registro (\_register)**: Este método privado, indicado por el guion bajo, es el responsable de guardar las operaciones en el historial. Recibe el tipo de operación (+, -, \*, /) y el resultado, y añade un diccionario a la lista \_records con la operación y el resultado.

Ilustración 3 Método para registrar, y mostrar el historial

* **Método de Historial (look\_register)**: Este método público recorre la lista \_records y muestra cada operación guardada en un formato legible para el usuario. Si la lista está vacía, notifica al usuario que no hay operaciones registradas.
* **Métodos de Operación (sum, sub, mul, div)**: Cada uno de estos métodos ejecuta la operación aritmética correspondiente. Después de calcular el resultado, llaman al método privado \_register para guardar la operación en el historial antes de devolver el valor.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 4 Métodos para realizar las operaciones aritméticas

**2.2.- La Función expressions**

Esta función es crucial para la validación de las entradas del usuario. Su principal objetivo es asegurar que los datos ingresados son correctos antes de que la calculadora intente realizar cualquier operación. Esta función fue modificada para hacer más eficiente su validación.

* **Bloque try...except**: La función utiliza un bloque try...except ValueError para manejar errores de conversión. El código dentro del bloque try intenta convertir las entradas del usuario a float. Si el usuario ingresa texto que no se puede convertir a un número, el bloque except se activa, imprime un mensaje de error y la función retorna None.
* **Validación de la Operación y División por Cero**: Una vez que se ha verificado que los números son válidos, la función comprueba que el operador ingresado sea uno de los cuatro permitidos. También incluye una condición para verificar si la operación es una división y si el segundo número es 0, evitando así un error de división por cero y mostrando un mensaje de error claro al usuario.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 5 Operación para verificar las entradas

**2.3.- La Función main y el Bucle del Programa**

La función main es el punto de entrada del programa. Su propósito es manejar la interacción con el usuario y orquestar el flujo de la aplicación.

* **Instancia de la Clase**: La línea calc = Calculator() crea un objeto de la clase Calculator, lo cual nos da acceso a todos sus métodos y atributos.
* **Bucle while True**: Este bucle asegura que el programa se ejecute continuamente. Solo se detendrá cuando el usuario ingrese el comando 'exit'.
* **Manejo de Comandos**: El programa lee la entrada del usuario y utiliza una serie de sentencias if/elif para decidir qué acción tomar:
  + Si la entrada es 'exit', el bucle se detiene.
  + Si la entrada es 'records', se llama al método calc.look\_register() para mostrar el historial.
  + Texto

    El contenido generado por IA puede ser incorrecto.Si la entrada es 'operation', se piden los dos números y el operador. Se llama a la función expressions para validar las entradas, y si la validación es exitosa, se utiliza otra estructura if/elif para llamar al método de operación correspondiente de la instancia calc.

Ilustración 6 Función main(), ciclo while, y operaciones

* Texto

  El contenido generado por IA puede ser incorrecto.**Mensajes de Error**: La función también maneja entradas no válidas por parte del usuario, como comandos que no son 'exit', 'records' u 'operation', mostrando un mensaje de error y reiniciando el bucle.

Ilustración 7 Validación de entradas, y ejecución de la operación seleccionada



Ilustración 8 Ejecución de la función principal

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 9 Ejecución del programa

**3.- Conclusiones**

El programa de la calculadora es un ejemplo muy completo de la eficiencia de la programación orientada a objetos. La clase Calculator encapsula la lógica de desarrollo, hace que el código sea modular, fácil de leer y mantener. La separación de responsabilidades entre la clase Calculator, la función de validación expressions y la función de control main es un patrón de diseño sólido.

Las funcionalidades de manejo de errores, como la validación de tipos de datos y la prevención de la división por cero, demuestran un enfoque en la fiabilidad. Además, la inclusión de un historial de operaciones añade una funcionalidad práctica que mejora la experiencia del usuario. En resumen, este programa es una solución completa y funcional que cumple con todos los requisitos y sirve como una excelente demostración de las mejores prácticas de codificación en Python.