Universidad de La Sabana

Maestría en Analítica Aplicada

Taller 1: Calculadora Orientada a Objetos en Python

Esteban Bernal - Informatics Engineering - 271938
Juan Montes - Informatics Engineering - 272113
Juan Gomez - Informatics Engineering - 286774

Profesor: Hugo Franco, Ph.D.
Herramientas de Big Data

Agosto de 2025

1. Problema

El objetivo de este taller fue desarrollar una **calculadora orientada a objetos en Python**, capaz de realizar operaciones básicas (suma, resta, multiplicación, división) y extendida con funcionalidades científicas (potencia, raíz cuadrada, seno, coseno y tangente).

El taller se dividió en dos partes: la primera consistió en la construcción de la clase `Calculadora` y su subclase `CalculadoraCientifica`; la segunda en la resolución de retos prácticos planteados en el cuaderno Jupyter adjunto, donde se probaron las funcionalidades implementadas.

Este ejercicio tiene como importancia principal familiarizarse con los conceptos de **programación orientada a objetos (POO)** aplicados en Python, así como con la documentación y presentación de resultados en Reporte Escrito.

2. Método de solución

La solución se abordó mediante la creación de clases en Python, definiendo atributos y métodos que representaran las operaciones matemáticas. Se utilizó herencia para extender la funcionalidad de la calculadora básica hacia una calculadora científica.

2.1. Algoritmo propuesto

A continuación se presenta el pseudocódigo general de la implementación:

Algoritmo 1: Clase Calculadora

```
Clase Calculadora:
    Atributos: operando1, operando2, operacion
    Métodos:
        suma(a,b)
        resta(a,b)
        multiplicacion(a,b)
        division(a,b)
        potencia(base, exp)
        raiz(numero)
```

Algoritmo 2: Clase CalculadoraCientifica (hereda de Calculadora)

```
Clase CalculadoraCientifica:
    Métodos adicionales:
        seno(x)
        coseno(x)
        tangente(x)
```

El código en Python que implementa la solución se encuentra en el archivo `Calculadora.py`, los retos fueron propuestos en el archivo Jupyter `oop_example.ipynb`.

3. Resultados

La ejecución de los retos incluidos en el cuaderno `oop_example.ipynb` permite validar el correcto funcionamiento de las operaciones.

3.1. Ejemplo de uso

Un ejemplo de operación encadenada es el siguiente:

```
result = Calculadora(10).suma(5).multiplicacion(2)
print(result.operando1)
```

La salida es:

```
30
```

3.2. Operaciones científicas

```
result1 = CalculadoraCientifica(90).seno()
print(result1.operando1)
```

Salida:

```
0.8939966636005579
```

3.3. Raíz cuadrada

```
result4 = Calculadora().raiz(9)
print(result4)
```

Salida:

```
3.0
```

4. Discusión

Los resultados obtenidos permiten validar que:

- El uso de programación orientada a objetos facilita la modularidad y reutilización del código.
- La implementación de métodos encadenados (*method chaining*) permite realizar cálculos complejos en una sola línea de código, aumentando la legibilidad.
- El manejo de excepciones (como división por cero o tipos de datos incorrectos) asegura la robustez del programa.
- La extensión hacia operaciones científicas muestra la potencia de la herencia en Python.

En conclusión, el taller cumplió con los objetivos planteados y permitió aplicar los principios de POO en un problema práctico.

Referencias

Documentación oficial de Python: `https://docs.python.org/3/`
Módulo math en Python: `https://docs.python.org/3/library/math.html`