Instituto Politécnico Nacional

Escuela Superior de Computo

Unidad de académica:

Paradigmas de programación

Práctica de laboratorio 3:

“Implementación de la programación funcional”

Equipo:

- Elizalde Hernández Alan
- Reyes Ruíz Yoselyn Estefany
- Solares Velasco Arturo Misael
- Solís Lugo Mayra

Toral Hernández Leonardo Javier

Grupo: 3CV1

Profesor: García Floriano Andrés

Fecha:

19 marzo 2024

Reporte de Práctica: Implementación de una Calculadora con Programación Funcional

Introducción

En el marco del aprendizaje de la programación funcional, se ha desarrollado una calculadora en Python que ofrece una variedad de operaciones matemáticas. Este proyecto se ha centrado en aplicar los principios de la programación funcional, priorizando la creación de funciones puras para cada operación, evitando la dependencia excesiva de la función principal.

Objetivo

El objetivo principal de esta práctica fue desarrollar una calculadora funcional en Python, utilizando funciones puras para cada operación matemática. Se propuso la implementación de al menos 10 funciones matemáticas, así como la creación de un menú de opciones para seleccionar la operación deseada.

Implementación

Para lograr el objetivo propuesto, se ha seguido el siguiente enfoque:

Definición de Funciones: Se han creado funciones para cada operación matemática requerida, siguiendo los principios de las funciones puras. Estas funciones realizan cálculos sin modificar ningún estado fuera de su alcance, lo que garantiza una mayor modularidad y reutilización del código.

A continuación, se muestra un ejemplo de las funciones que implementa la calculadora.

```python
# Definir las funciones para las operaciones aritméticas
def suma(a, b):
    return a + b
def resta(a, b):
    return a - b
def multiplicacion(a, b):
    return a * b
def division(a, b):
    if b == 0:
        return "Error: No se puede dividir por cero."
    else:
        return a / b
```

Ilustración 1 Funciones para las operaciones aritméticas.

```python
# Funciones para otras operaciones
def raiz_cuadrada(a):
    if a < 0:
        return "Error: No se puede calcular la raíz cuadrada de un número negativo."
    else:
        return math.sqrt(a)
def raiz_n(a, n):
    if a < 0 and n % 2 == 0:
        return "Error: No se puede calcular la raíz par de un número negativo."
    else:
        return a ** (1/n)
def exponencial(a, b):
    return a ** b
def seno(a):
    return math.sin(a)
def coseno(a):
    return math.cos(a)
def tangente(a):
    return math.tan(a)
def arcoseno(a):
    return math.asin(a)
def arcocoseno(a):
    return math.acos(a)
def arcotangente(a):
    return math.atan(a)
```

Ilustración 2 Funciones para otras operaciones.

Menú de Opciones: Se ha implementado un menú de opciones que permite al usuario seleccionar la operación matemática que desea realizar. Esto se ha logrado utilizando un bucle while que solicita al usuario que ingrese el nombre de la operación deseada.

Validación de Entrada: Se ha incorporado una validación de entrada para manejar posibles errores durante la interacción con el usuario. Esto incluye la comprobación de la validez de la operación ingresada, así como la detección de errores al ingresar números.

Función Principal: La función principal (calculadora) actúa como punto de entrada del programa. Se encarga de llamar a las funciones correspondientes según la operación seleccionada por el usuario y mostrar el resultado.

A continuación, se muestra el código de la función principal que se ejecutará al final y que llevará a cabo todas las otras funciones que se declararon anteriormente.

Ilustración 3 Implementación de la función 'calculadora' (principal)

Ilustración 4 Punto de entrada del programa con la función principal.

Conclusión

La implementación de una calculadora funcional en Python ha permitido aplicar los principios de la programación funcional de manera efectiva. El uso de funciones puras ha mejorado la legibilidad, modularidad y mantenibilidad del código, mientras que el menú de opciones ofrece una experiencia de usuario intuitiva. Este proyecto ha demostrado cómo la programación funcional puede utilizarse para crear programas eficientes, estructurados, y ordenados.