P8. Matemáticas Avanzadas en Python:
NumPy y SymPy

Alumna: Daniela Mendez Ramirez

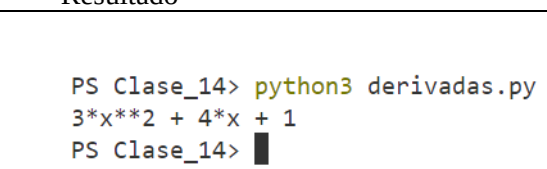
Número de Cuenta: 258331-9

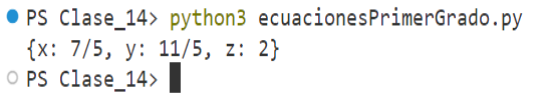
Aplicaciones de Redes

Profesor: Omar Vázquez González

Fecha de Entrega: 16 de febrero de 2025

Ejercicios para entender el funcionamiento de simPy.

Código	Resultado
`import sympy as sp` `# Definir la variable simbólica` `x = sp.Symbol('x')` `# Definir la función a derivar` `f = x**3 + 2*x**2 + x + 5` `# Calcular la derivada` `dfdx = sp.diff(f, x)` `# Mostrar el resultado` `print(dfdx)  # Salida: 3*x**2 + 4*x + 1`	

Código	Resultado
`import sympy as sp` `# Definir las variables simbólicas` `x, y, z = sp.symbols('x y z')` `# Definir las ecuaciones` `eq1 = sp.Eq(2*x + y - z, 3)` `eq2 = sp.Eq(x + 3*y + 2*z, 12)` `eq3 = sp.Eq(3*x - y + 4*z, 10)` `# Resolver el sistema` `solucion = sp.solve((eq1, eq2, eq3),` `(x, y, z))` `# Mostrar el resultado` `print(solucion)  # Devuelve un` `diccionario con los valores de x, y, z`	

Código	Resultado
import sympy as sp # Definir la variable x = sp.Symbol('x') # Definir la ecuación ecuacion = sp.Eq(x**2 - 5*x + 6, 0) # Resolver la ecuación solucion = sp.solve(ecuacion, x) # Mostrar el resultado print("Solución de la ecuación:", solucion)	PS Clase_14> python3 .\EcAlgebraica.py Solución de la ecuación: [2, 3] PS Clase_14>

Código	Resultado
import sympy as sp # Definir la variable x = sp.Symbol('x') # Definir la ecuación exponencial ecuacion = sp.Eq(2**x, 16) # Resolver la ecuación solucion = sp.solve(ecuacion, x) # Mostrar el resultado print("Solución de la ecuación exponencial:", solucion)	PS Clase_14> python3 .\exponencial.py Solución de la ecuación exponencial: [4] PS Clase_14>

Código	Resultado
``python import sympy as sp # Definir la variable y la función x = sp.Symbol('x') f = x**3 + 2*x**2 - 3*x + 5 # Calcular la primera derivada primera_derivada = sp.diff(f, x) # Calcular la segunda derivada segunda_derivada = sp.diff(primera_derivada, x) # Mostrar los resultados print("Primera derivada:", primera_derivada) print("Segunda derivada:", segunda_derivada) ```	``` PS Clase_14> python3 .\exponencial.py Solución de la ecuación exponencial: [4] PS Clase_14> ```

1. Resolver operación.

$$\begin{cases} 2x + 3y + 4z = 20 \\ 3x - 5y - z = -10 \\ -x + 2y - 3z = -6 \end{cases}$$

Código	Resultado
`import sympy as sp` `# Definir las variables simbólicas` `x, y, z = sp.symbols('x y z')` `# Definir las ecuaciones` `eq1 = sp.Eq(2*x + 3*y + 4*z, 20)` `eq2 = sp.Eq(3*x - 5*y - z, -10)` `eq3 = sp.Eq(-x + 2*y - 3*z, -6)` `# Resolver el sistema` `solucion = sp.solve((eq1, eq2, eq3), (x, y, z))` `# Mostrar el resultado` `print(solucion)  # Devuelve un diccionario con los valores de x, y, z`	`PS Clase_14> python3 ecuacionesPrimerGrado.py` `{x: 1, y: 2, z: 3}` `PS Clase_14>`

2. Resolver operación.

$$\begin{aligned} +5x_1 + 4x_2 - 1x_3 + 2x_4 - 9x_5 &= +29 \\ +4x_1 + 3x_2 - 5x_3 - 5x_4 + 4x_5 &= -65 \\ +6x_1 - 6x_2 - 3x_3 - 9x_4 + 4x_5 &= -113 \\ -8x_1 + 8x_2 + 5x_3 - 3x_4 - 6x_5 &= +35 \\ -3x_1 - 7x_2 + 2x_3 - 5x_4 + 4x_5 &= -50 \end{aligned}$$

Código	Resultado
 `` import sympy as sp # Definir las variables x1, x2, x3, x4, x5 = sp.symbols('x1 x2 x3 x4 x5') # Definir las ecuaciones eq1 = sp.Eq(5*x1 + 4*x2 - x3 + 2*x4 - 9*x5, 29) eq2 = sp.Eq(4*x1 + 3*x2 - 5*x3 - 5*x4 + 4*x5, -65) eq3 = sp.Eq(6*x1 - 6*x2 - 3*x3 - 9*x4 + 4*x5, -113) eq4 = sp.Eq(-8*x1 + 8*x2 + 5*x3 - 3*x4 - 6*x5, 35) eq5 = sp.Eq(-3*x1 - 7*x2 + 2*x3 - 5*x4 + 4*x5, -50) # Resolver el sistema de ecuaciones solucion = sp.solve((eq1, eq2, eq3, eq4, eq5), (x1, x2, x3, x4, x5)) # Mostrar la solución print("Solución del sistema:") for var, val in solucion.items(): print(f"{var} = {val}") ```	 ``` PS Clase_14> python3 polinomios.py Solución del sistema: x1 = -2 x2 = 2 x3 = 3 x4 = 8 x5 = -2 PS Clase_14> ```

Conclusión

La librería SymPy es una herramienta poderosa para el cálculo simbólico en Python, que permite manipular expresiones algebraicas, resolver ecuaciones y simplificar expresiones de manera eficiente. Es especialmente útil en áreas como matemáticas, física y química. Una de sus principales ventajas sobre MATLAB es que es completamente gratuita y de código abierto, lo que la hace más accesible. Además, al estar integrada con Python, ofrece la flexibilidad de combinar cálculos simbólicos con otras herramientas y bibliotecas, como NumPy y Pandas, lo que facilita tareas multidisciplinarias. Si bien MATLAB es más eficiente para cálculos numéricos complejos y tiene funciones especializadas en áreas como ingeniería o procesamiento de señales, SymPy es más práctica para trabajos que requieren álgebra simbólica y es ideal para proyectos pequeños o de código abierto.