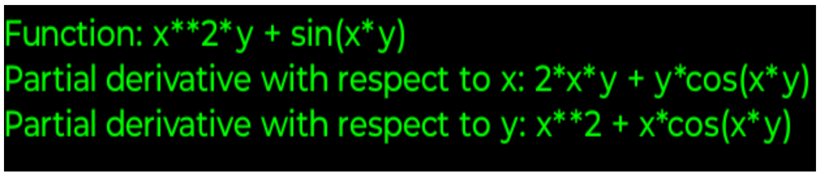
MML - Practical - QB - Solutions - By - Th3_

Practical 1: Compute Partial Derivative

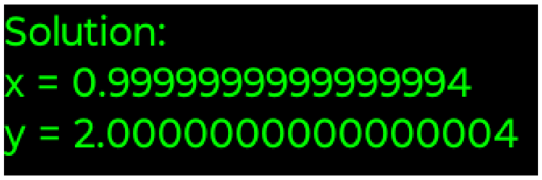
```
import sympy as sp

x, y = sp.symbols('x y')

f = x**2 * y + sp.sin( x * y)

partial_x = sp.diff(f, x)
partial_y = sp.diff(f, y)

print("Function:", f)
print("Partial derivative with respect to x:", partial_x)
print("Partial derivative with respect to y:", partial_y)
```



```
Function: x**2*y + sin(x*y)
Partial derivative with respect to x: 2*x*y + y*cos(x*y)
Partial derivative with respect to y: x**2 + x*cos(x*y)
```

Practical 2: Solve System of Linear Equations

```
import numpy as np

A = np.array([[2, 3], [3, 4]])
B = np.array([8, 11])

solution = np.linalg.solve(A, B)

print("Solution:")
print(f"x = {solution[0]}")
print(f"y = {solution[1]}")
```

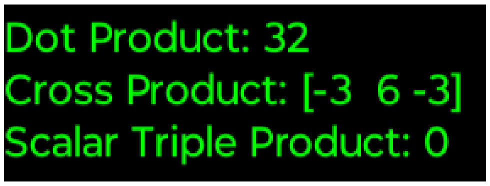


```
Solution:
x = 0.9999999999999994
y = 2.0000000000000004
```

MML - Practical - QB - Solutions - By - Th3_

Practical 3: Vector Operations (Dot Product, Cross Product, Scalar Triple Product)

```
import numpy as np

def dot_product(v1, v2):
    return np.dot(v1, v2)

def cross_product(v1, v2):
    return np.cross(v1, v2)

def scalar_triple_product(v1, v2, v3):
    return np.dot(v1, np.cross(v2, v3))

v1 = np.array([1, 2, 3])
v2 = np.array([4, 5, 6])
v3 = np.array([7, 8, 9])

dot_prod = dot_product(v1, v2)
cross_prod = cross_product(v1, v2)
scalar_triple_prod = scalar_triple_product(v1, v2, v3)

print("Dot Product:", dot_prod)
print("Cross Product:", cross_prod)
print("Scalar Triple Product:", scalar_triple_prod)
```

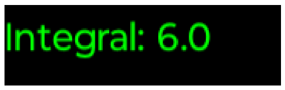


```
Dot Product: 32
Cross Product: [-3  6 -3]
Scalar Triple Product: 0
```

MML - Practical - QB - Solutions - By - Th3_

Practical 4: Numerical Integration Using Simpson's One-Third Rule

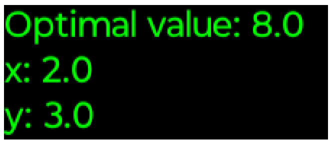
```
def simpsons_one_third(x, y):
    n = len(x) - 1
    if n % 2 != 0:
        raise ValueError("Number of intervals must be even.")

    h = (x[-1] - x[0]) / n
    integral = y[0] + y[-1]

    for i in range(1, n, 2):
        integral += 4 * y[i]

    for i in range(2, n, 2):
        integral += 2 * y[i]

    integral *= h / 3
    return integral

x = [0, 1, 2, 3, 4]
y = [1, 2, 0, 2, 1]

integral = simpsons_one_third(x, y)
print("Integral:", integral)
```



```
Integral: 6.0
```

Practical 5: Simplex Method for 2 Equations in 2 Variables

```
from scipy.optimize import linprog

c = [-1, -2]
A = [[1, 1], [-1, 2]]
b = [5, 4]

result = linprog(c, A_ub=A, b_ub=b, method='simplex')

if result.success:
    print("Optimal value:", -result.fun)
    print("x:", result.x[0])
    print("y:", result.x[1])
else:
    print("No solution found.")
```



```
Optimal value: 8.0
x: 2.0
y: 3.0
```