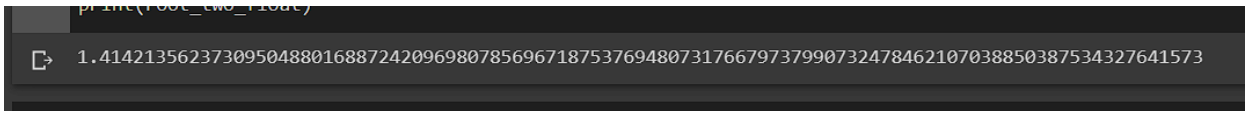
APP WEEK-9 LAB

Q1. Calculate root 2 with 100 decimals

Code:

```
import sympy

# Set the precision to 100 decimal places
sympy_float = lambda x: sympy.Float(x, 100)

# Calculate the square root of 2
root_two = sympy.sqrt(2)

# Evaluate the square root of 2 to 100 decimal places
root_two_float = root_two.evalf(100)

# Print the result
print(root_two_float)
```

SnapShot:

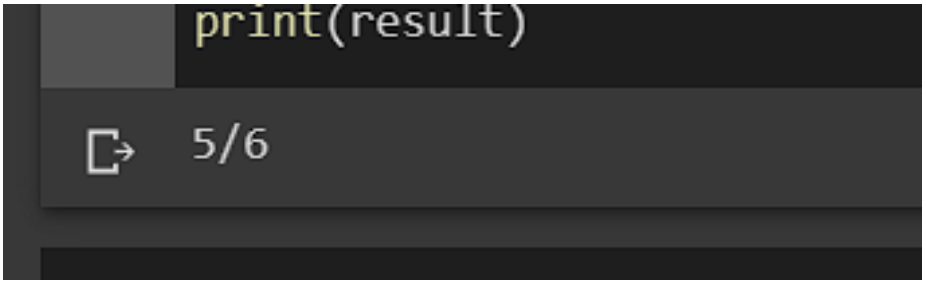


Q2. Calculate 1/2+1/3 in rational arithematic

Code:

```
from fractions import Fraction

# Define the fractions
frac1 = Fraction(1, 2)
frac2 = Fraction(1, 3)

# Add the fractions
result = frac1 + frac2

# Print the result
print(result)
```

SnapShot:

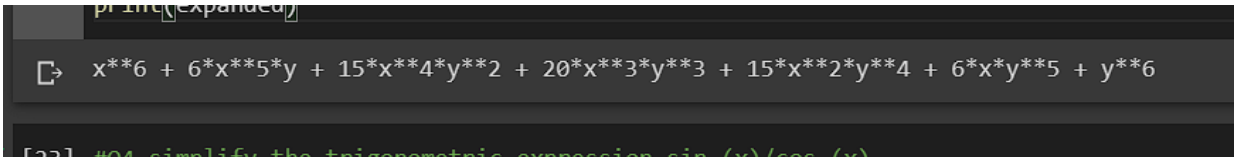


Q3. Calculate the expanded form of (x+y)^6

Code:

```
import sympy

# Define variables
x, y = sympy.symbols('x y')

# Expand the expression
expanded = sympy.expand((x + y)**6)

# Print the result
print(expanded)
```

SnapShot:

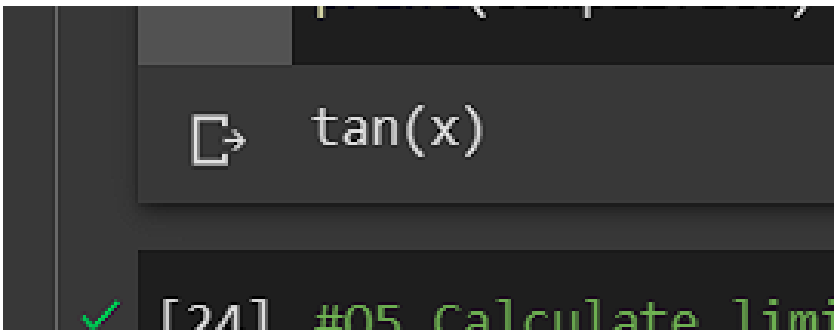


Q4. Simplify the trigonometric expression sin (x)/cos (x)

Code:

```
import sympy

# Define variable
x = sympy.symbols('x')

# Simplify the expression
simplified = sympy.tan(x)

# Print the result
print(simplified)
```

SnapShot:

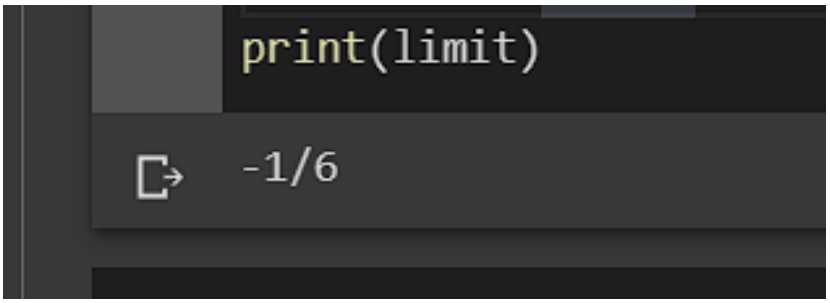


Q5. Calculate limit x tends to 0 ((sin(x)-x)/x^3)

Code:

```
import sympy

# Define the variable and expression
x = sympy.symbols('x')
expr = (sympy.sin(x) - x) / x**3

# Calculate the limit as x approaches 0
limit = sympy.limit(expr, x, 0)

# Print the result
print(limit)
```

SnapShot:

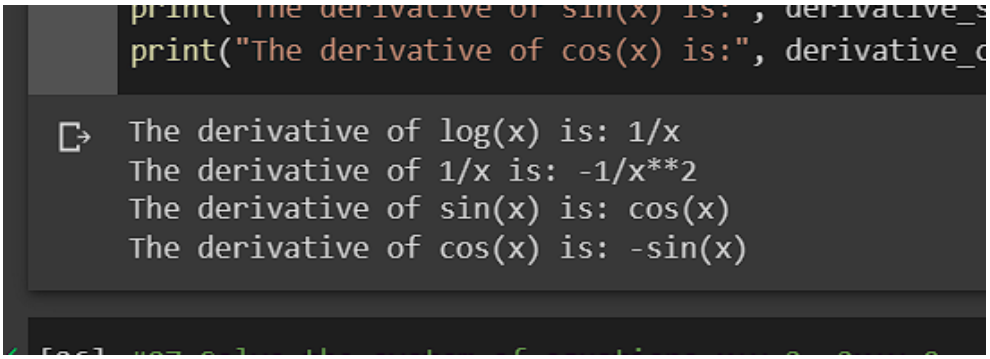


Q6. Calculate the derivative of log(x), 1/x, sin(x), cos(x) for x

Code:

```
import sympy

# Define variable
x = sympy.symbols('x')

# Calculate the derivatives
derivative_log = sympy.diff(sympy.log(x), x)
derivative_inv = sympy.diff(1/x, x)
derivative_sin = sympy.diff(sympy.sin(x), x)
derivative_cos = sympy.diff(sympy.cos(x), x)

# Print the results
print("The derivative of log(x) is:", derivative_log)
print("The derivative of 1/x is:", derivative_inv)
print("The derivative of sin(x) is:", derivative_sin)
print("The derivative of cos(x) is:", derivative_cos)
```

SnapShot:

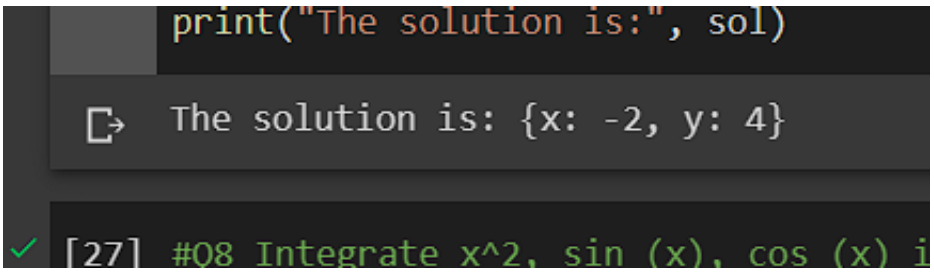


Q7. Solve the system of equations x+y=2, 2x+y=0

Code:

```
import sympy

# Define the variables
x, y = sympy.symbols('x y')

# Define the equations
eq1 = sympy.Eq(x + y, 2)
eq2 = sympy.Eq(2*x + y, 0)

# Solve the system of equations
sol = sympy.solve((eq1, eq2), (x, y))

# Print the solution
print("The solution is:", sol)
```

SnapShot:

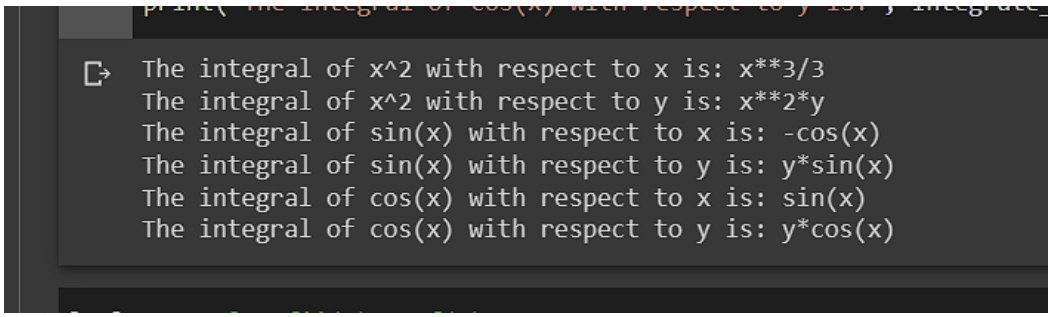


Q8. Integrate x^2, sin (x), cos (x) interms of x and y

Code:

```
import sympy

# Define variables
x, y = sympy.symbols('x y')

# Integrate x^2 with respect to x
integrate_x2_x = sympy.integrate(x**2, x)
print("The integral of x^2 with respect to x is:", integrate_x2_x)

# Integrate x^2 with respect to y
integrate_x2_y = sympy.integrate(x**2, y)
print("The integral of x^2 with respect to y is:", integrate_x2_y)

# Integrate sin(x) with respect to x
integrate_sin_x = sympy.integrate(sympy.sin(x), x)
print("The integral of sin(x) with respect to x is:", integrate_sin_x)

# Integrate sin(x) with respect to y
integrate_sin_y = sympy.integrate(sympy.sin(x), y)
print("The integral of sin(x) with respect to y is:", integrate_sin_y)

# Integrate cos(x) with respect to x
integrate_cos_x = sympy.integrate(sympy.cos(x), x)
print("The integral of cos(x) with respect to x is:", integrate_cos_x)

# Integrate cos(x) with respect to y
integrate_cos_y = sympy.integrate(sympy.cos(x), y)
print("The integral of cos(x) with respect to y is:", integrate_cos_y)
```

SnapShot:

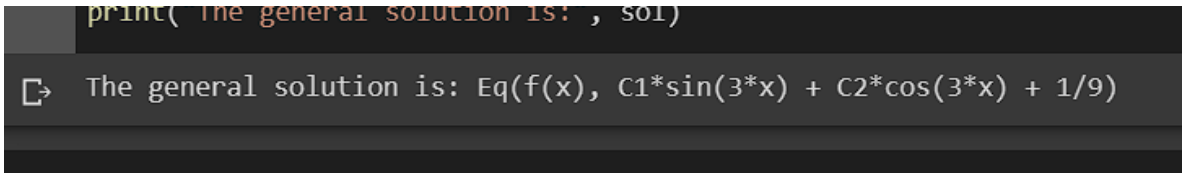


Q9. Solve f''(x) + 9f(x)=1

Code:

```
import sympy

# Define the function and the variable
f = sympy.Function('f')
x = sympy.Symbol('x')

# Define the differential equation
deq = sympy.diff(f(x), x, 2) + 9*f(x) - 1

# Solve the differential equation
sol = sympy.dsolve(deq, f(x))

# Print the solution
print("The general solution is:", sol)
```

SnapShot:



Q10. Using matrices solve the linear equations: 3x+7y=12z and 4x-2y=5z

Code:

```
import numpy as np

# Define the coefficient matrix and the right-hand side vector
A = np.array([[3, 7], [4, -2]])
b = np.array([[12], [5]])

# Solve the system of equations using matrix inversion
x = np.linalg.inv(A) @ b

# Print the solution
print("The solution is:", x)
```

SnapShot: