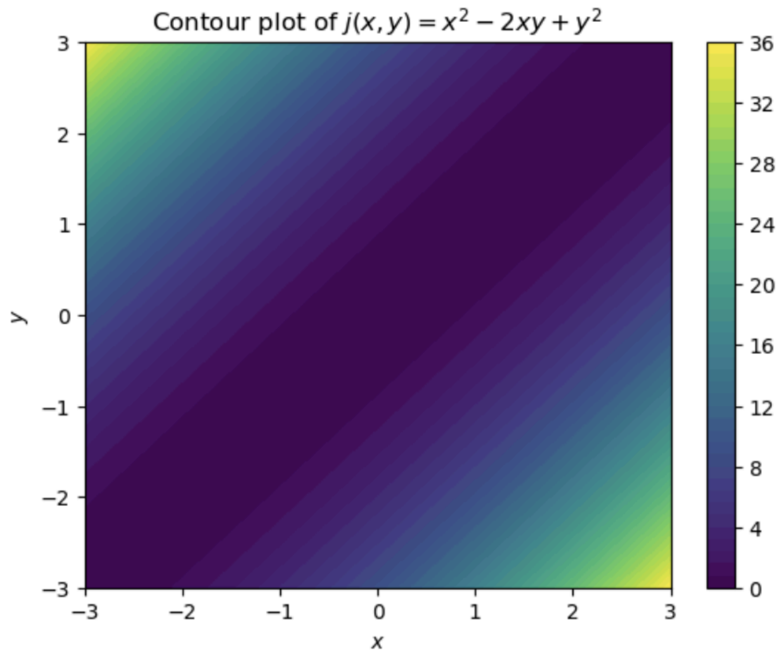
```python
#4a
import sympy as sp
x, y = sp.symbols('x y') # define x and y in sympy
# Define a function
f = sp.exp(x)*sp.sin(y)+y**3
df_dx=sp.diff(f,x) # find partial derivative wrt x
df_dy=sp.diff(f,y) # find partial derivative wrt y
print("df/dx=", df_dx)
print("df/dy=", df_dy)
```

```
df/dx= exp(x)*sin(y)
df/dy= 3*y**2 + exp(x)*cos(y)
```

```python
#4c
x, y = sp.symbols('x y') # define x and y in sympy
# Define a function
f = sp.log(x**2+y**2)
secondderivx=sp.diff(sp.diff(f,x),x) # find second partial derivative
secondderivy=sp.diff(sp.diff(f,y),y) # find second partial derivative
mixedderiv=sp.diff(f,x,y) # find second partial derivative
print("2nd derivative x=", secondderivx)
print("2nd derivative y=", secondderivy)
print("Mixed derivative=", mixedderiv)
# comment on symmetry: as per Clairout's Theorem, we know that our mixed derivatives
# fxy(x,y) and fyx(x,y) are symmetrical and will be the same as each other
```

```
2nd derivative x= -4*x**2/(x**2 + y**2)**2 + 2/(x**2 + y**2)
2nd derivative y= -4*y**2/(x**2 + y**2)**2 + 2/(x**2 + y**2)
Mixed derivative= -4*x*y/(x**2 + y**2)**2
```

```python
#4d based on greg given code
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
x, y = sp.symbols('x y')
j = x**2 - 2*x*y + y**2
j_func = sp.lambdify((x, y), j, 'numpy')
x_vals = np.linspace(-3, 3, 400)
y_vals = np.linspace(-3, 3, 400)
X, Y = np.meshgrid(x_vals, y_vals)
Z = j_func(X, Y)
plt.contourf(X, Y, Z, levels=50, cmap='viridis')
plt.colorbar()
plt.title('Contour plot of $j(x, y) = x^2 - 2xy + y^2$')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.show()
```



]:

```
#4b
x, y = sp.symbols('x y') # define x and y in sympy
# Define a function
f = ((x**2)*y+(x*(y**2)))
df_dx=sp.diff(f,x)
df_dy=sp.diff(f,y)
gradientf=df_dx+df_dy
x=1
y=-1
def gradientxatpoint(x,y): # defining gradient x at point
    return 2*x*y+(y**2)

def gradientyatpoint(x,y):  # defining gradient y at point
    return x**2+2*x*y
print("Gradient at (1,-1) is:", (gradientxatpoint(x,y),gradientyatpoint(x,y)))
def magnitude(gradientxatpoint,gradientyatpoint): # define magnitude
    return sp.sqrt((gradientxatpoint)**2+(gradientyatpoint)**2)
print("Magnitude at (1,-1) is:", (magnitude(x,y)))
```

```
Gradient at (1,-1) is: (-1, -1)
Magnitude at (1,-1) is: sqrt(2)
```