```
import plotly.graph_objects as go
import numpy as np

x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
x, y = np.meshgrid(x, y)
z = x**2 * y

fig = go.Figure(data=[go.Surface(z=z, x=x, y=y)])
fig.update_layout(title='3D Plot of the function f(x, y) = x^2y', 
autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))

fig.show()

import plotly.graph_objects as go
import numpy as np

# Create a set of points that lie on the circle
theta = np.linspace(0, 2*np.pi, 100)
x = np.cos(theta)
y = np.sin(theta)

# Create the plot
fig = go.Figure(data=go.Scatter(x=x, y=y, mode='lines'))

# Set the aspect ratio to equal so the circle doesn't look like an 
ellipse
fig.update_yaxes(
    scaleanchor = "x",
    scaleratio = 1,
)

fig.show()

import plotly.graph_objects as go
import numpy as np

# Create grid for the function
x = np.linspace(-1.5, 1.5, 100)
y = np.linspace(-1.5, 1.5, 100)
x, y = np.meshgrid(x, y)
z = x**2 * y

# Create a set of points that lie on the circle
theta = np.linspace(0, 2*np.pi, 100)
x_circle = np.cos(theta)
y_circle = np.sin(theta)
z_circle = x_circle**2 * y_circle
```

```
# Create the surface plot for the function
fig = go.Figure(data=[go.Surface(z=z, x=x, y=y, colorscale='Viridis',
opacity=0.8)])

# Add the circle to the plot
fig.add_trace(go.Scatter3d(x=x_circle, y=y_circle, z=z_circle,
mode='lines'))

fig.show()

import plotly.graph_objects as go
import numpy as np

# Create grid for the function
x = np.linspace(-1.5, 1.5, 100)
y = np.linspace(-1.5, 1.5, 100)
x, y = np.meshgrid(x, y)
z = x**2 * y

# Create a set of points that lie on the circle
theta = np.linspace(0, 2*np.pi, 100)
x_circle = np.cos(theta)
y_circle = np.sin(theta)

# Create the contour plot for the function
fig = go.Figure(data=go.Contour(x=x[0,:], y=y[:,0], z=z))

# Add the circle to the plot
fig.add_trace(go.Scatter(x=x_circle, y=y_circle, mode='lines',
line=dict(color='white')))

# Set aspect ratio
fig.update_layout(
    autosize=False,
    width=500,
    height=500,
    yaxis=dict(scaleanchor="x", scaleratio=1),
)

fig.show()

import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-2, 2, 10)
y = np.linspace(-2, 2, 10)

X, Y = np.meshgrid(x, y)

# Compute the function values
Z = X**2 * Y
```

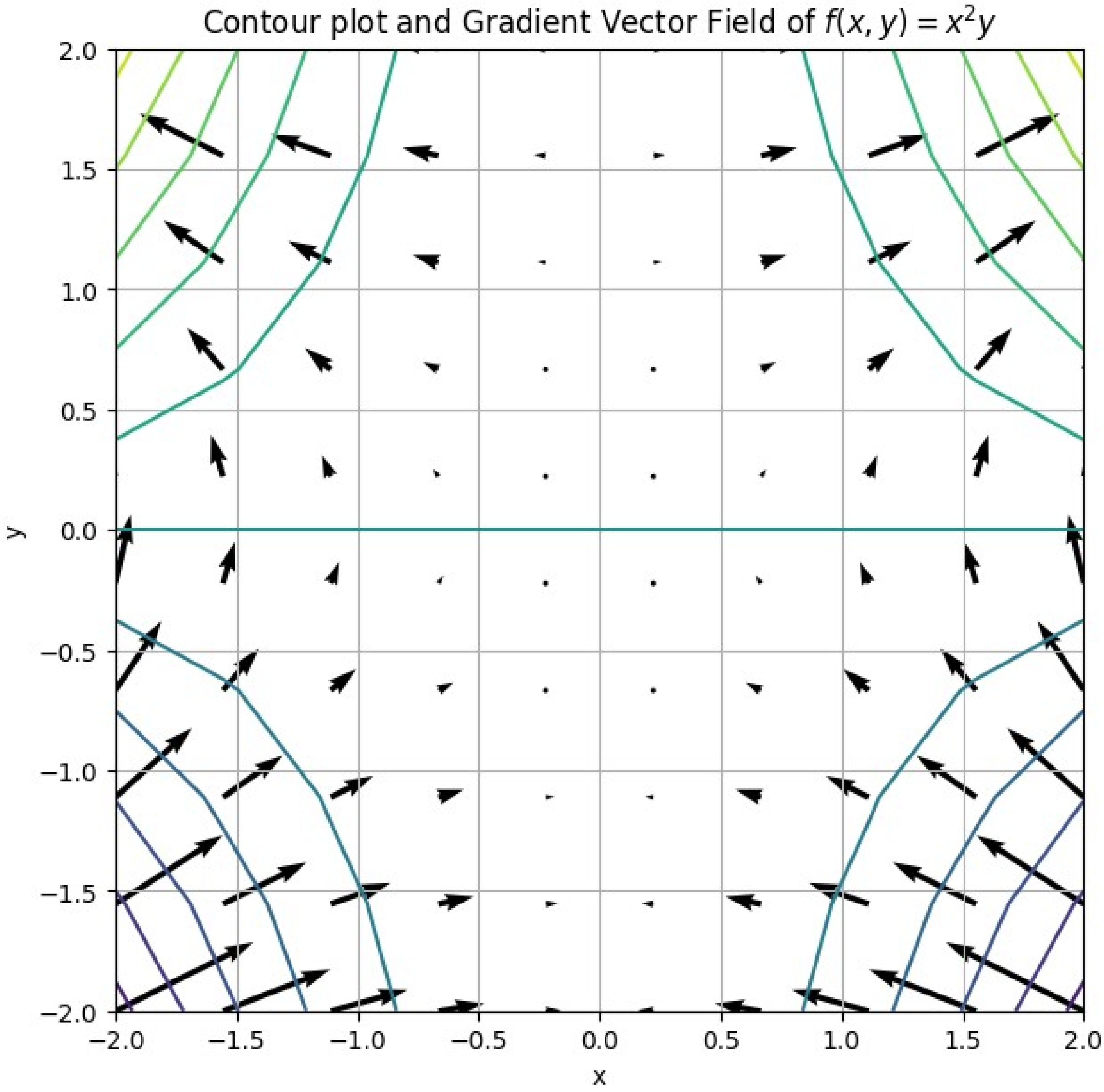
```python
# Compute the gradient
U = 2*X*Y
V = X**2

plt.figure(figsize=(7, 7))

# Draw the contour plot
plt.contour(X, Y, Z, levels=10, cmap='viridis')

# Draw the gradient vector field
plt.quiver(X, Y, U, V)

plt.title('Contour plot and Gradient Vector Field of $f(x, y) =
x^2y$')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```



```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-2, 2, 10)
y = np.linspace(-2, 2, 10)

X, Y = np.meshgrid(x, y)

# Compute the function values
Z = X**2 + Y**2

# Compute the gradient
U = 2*X
```

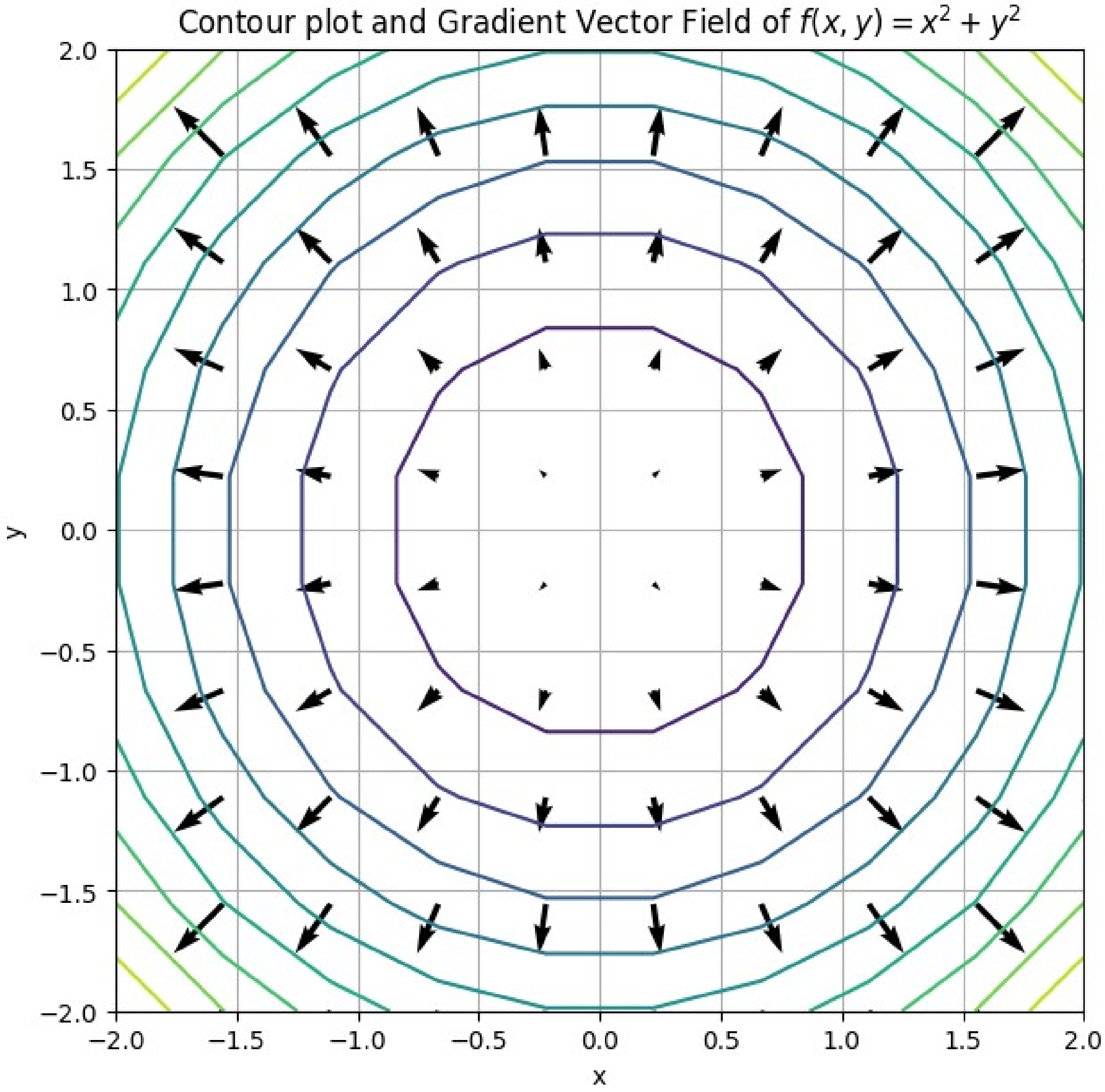
```python
V = 2*Y

plt.figure(figsize=(7, 7))

# Draw the contour plot
plt.contour(X, Y, Z, levels=10, cmap='viridis')

# Draw the gradient vector field
plt.quiver(X, Y, U, V)

plt.title('Contour plot and Gradient Vector Field of $f(x, y) = x^2 + y^2$')
plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.show()
```


Contour plot and Gradient Vector Field of $f(x, y) = x^2 + y^2$
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x