```python
#diagonizable property
import numpy as np

# Define the matrix
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])

# Find the eigenvalues and eigenvectors
eigvals, eigvecs = np.linalg.eig(A)

# Check if the matrix is diagonalizable
if np.linalg.matrix_rank(eigvecs) == A.shape[0]:
    print("The matrix is diagonalizable")
else:
    print("The matrix is not diagonalizable")

# Print the eigenvalues and eigenvectors
print("Eigenvalues:", eigvals)
print("Eigenvectors:", eigvecs)


#Cayley-Hamilton theorem

import numpy as np

# Define the matrix
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
```

```
# Find the eigenvalues and eigenvectors
eigvals, eigvecs = np.linalg.eig(A)

# Compute the characteristic polynomial
poly = np.poly1d([1, -np.sum(eigvals),
np.linalg.det(A)])
print("Characteristic polynomial:", poly)

# Evaluate the polynomial at A
result = poly(A)
print("Result of polynomial evaluated at A:\n",
result)
```

```
#gradient of a scalar field
import numpy as np

# Define the scalar field
def f(x, y, z):
    return np.sin(x) * np.cos(y) * np.exp(z)

# Define the coordinates
x = np.linspace(0, 2*np.pi, 10)
y = np.linspace(0, np.pi, 10)
z = np.linspace(0, 1, 10)
```

```
# Compute the gradient
grad = np.gradient(f(x, y, z))

# Print the gradient
print("Gradient of f:")
print("df/dx =", grad[0])
print("df/dy =", grad[1])
print("df/dz =", grad[2])
```