```python
#Q1 MLE Normal Distribution


import numpy as np

# Define a function to calculate the log-likelihood
def log_likelihood(theta_1, theta_2, data):
    n = len(data)
    sum_squared_diff = np.sum((data - theta_1) ** 2)
    return -n/2 * np.log(2 * np.pi * theta_2) - sum_squared_diff / (2 * theta_2)

# Define a function to find MLE for theta_1 (mean)
def mle_theta_1(data):
    return np.mean(data)

# Define a function to find MLE for theta_2 (variance)
def mle_theta_2(theta_1, data):
    n = len(data)
    return np.sum((data - theta_1) ** 2) / n

# Example data
data = np.array([1.5, 2.3, 3.1, 4.2, 5.0])

# Calculate MLE for theta_1 and theta_2
theta_1_mle = mle_theta_1(data)
theta_2_mle = mle_theta_2(theta_1_mle, data)

print("MLE for theta_1 (mean):", theta_1_mle)
print("MLE for theta_2 (variance):", theta_2_mle)
```

[5]

```
MLE for theta_1 (mean): 3.22
MLE for theta_2 (variance): 1.5896000000000001
```

```python
#Q2 MLE Binomial Distribution


from scipy.optimize import minimize
from scipy.special import comb
import numpy as np

# Define the log-likelihood function
def log_likelihood(theta, m, data):
    return -np.sum(np.log(comb(m, data)) + data * np.log(theta) + (m - data) * np.log(1 - theta))

# Example data
data = np.array([4, 5, 6])  # Sample observations
m = 10  # Known positive integer 'm'

# Define the negative log-likelihood function (to minimize)
neg_log_likelihood = lambda theta: -log_likelihood(theta, m, data)

# Initial guess for theta
initial_guess = 0.5

# Minimize the negative log-likelihood to find MLE of theta
result = minimize(neg_log_likelihood, initial_guess, bounds=[(0.01, 0.99)])

# Extract the MLE of theta
theta_mle = result.x[0]

print("MLE for theta:", theta_mle)
```

[7]

```
MLE for theta: 0.5
```