[1]:
```python
import math  # math 모듈을 임포트합니다. 이 모듈은 수학적 연산을 수행하는데 필요한 여러 함수를 제공합니다.

# 측정값들을 담은 리스트를 생성합니다. 이 예제에서는 총 10개의 측정값이 있습니다.
measurements = [2.1, 2.2, 2.0, 2.3, 2.1, 2.2, 2.1, 2.4, 2.3, 2.2]

# 가능도(likelihood)를 계산하는 함수를 정의합니다. 이 함수는 데이터, 평균(mu), 그리고 표준편차(sigma)를 인자로 받습니다.
def calculate_likelihood(data, mu, sigma):
    likelihood = 1.0  # 가능도를 초기화합니다. 가능도는 모든 데이터 포인트에 대한 확률밀도함수의 곱이므로, 초기값은 곱셈의 항등원인 1로 설정합니다.
    for measurement in data:  # 데이터의 각 측정값에 대해 반복합니다.
        exponent = -(measurement-mu)**2/(2*sigma**2)  # 정규 분포의 지수 부분을 계산합니다. 이는 측정값에서 평균을 뺀 값의 제곱을 2*표준편차의 제곱으로 나눈 것임
        likelihood *= (1/(sigma*math.sqrt(2*math.pi)))*math.exp(exponent)  # 정규 분포의 확률 밀도 함수를 계산하고, 이를 현재까지의 가능도에 곱합니다.
    return likelihood  # 계산된 가능도를 반환합니다. 이 값은 주어진 데이터가 특정 평균과 표준편차를 가진 정규 분포에서 나왔을 가능성을 나타냅니다.
```

[6]:
```python
### 1. 가능도 계산 ###
print("### 1. Calculate the likelihood ###")

# 모수의 평균을 2.2라고 가정한 값
mean = 2.2

# 표준편차 역시 가정한 값
std_dev = 0.1

# 가능도 계산 함수 호출
# 'measurements'는 이미 정의된 데이터
likelihood = calculate_likelihood(measurements, mean, std_dev)

# 만일 mean이 2.2이고 std_dev이 0.1일 때의 가능도 출력
print("Likelihood:", likelihood)
```

```
### 1. Calculate the likelihood ###
Likelihood: 1535.2762294855227
```

[7]:
```python
### 2. 평균 값을 변경하여 가능도 계산 ###
print("### 2. Calculate the likelihoods by changing mean values ###")

# 직관을 위해 여러 가정 값을 설정
mean_list = [2.0, 2.1, 2.2, 2.3, 2.4]

# 표준편차는 고정된 값
std_dev = 0.1

# 각 평균 값에 대한 가능도 계산
for mean in mean_list:
    likelihood = calculate_likelihood(measurements, mean, std_dev)
    # 각각의 평균과 표준편차에 대한 가능도 출력
    print(f"Mean: {mean}, Likelihood: {likelihood}")
```

```
### 2. Calculate the likelihoods by changing mean values ###
Mean: 2.0, Likelihood: 2.3382225877603254e-05
Mean: 2.1, Likelihood: 28.119565013714418
Mean: 2.2, Likelihood: 1535.2762294855227
Mean: 2.3, Likelihood: 3.8055692956214573
Mean: 2.4, Likelihood: 4.282604055890112e-07
```

[8]:
```python
### 3. 평균과 표준편차를 모두 변경하여 가능도 계산 ###
print("### 3. Calculate the likelihoods by changing both mean and std values ###")

# 여러 평균 값 설정
mean_list = [2.0, 2.1, 2.2, 2.3, 2.4]

# 여러 표준편차 값 설정
std_dev_list = [0.1, 0.2, 0.3, 0.4, 0.5]

# 평균과 표준편차를 모두 변경하여 가능도 계산
for mean in mean_list:
    for std_dev in std_dev_list:
        likelihood = calculate_likelihood(measurements, mean, std_dev)
        # 각각의 평균과 표준편차에 대한 가능도 출력
        print(f"Mean: {mean}, std: {std_dev}, Likelihood: {likelihood}")
```

```
### 3. Calculate the likelihoods by changing both mean and std values ###
Mean: 2.0, std: 0.1, Likelihood: 2.3382225877603254e-05
Mean: 2.0, std: 0.2, Likelihood: 2.181458723602369
Mean: 2.0, std: 0.3, Likelihood: 1.1366899530712442
Mean: 2.0, std: 0.4, Likelihood: 0.210614040322775
Mean: 2.0, std: 0.5, Likelihood: 0.039245694837152616
Mean: 2.1, std: 0.1, Likelihood: 28.119565013714418
Mean: 2.1, std: 0.2, Likelihood: 72.23999156132467
Mean: 2.1, std: 0.3, Likelihood: 5.385316291542496
Mean: 2.1, std: 0.4, Likelihood: 0.5052368178928953
Mean: 2.1, std: 0.5, Likelihood: 0.06870635870641578
Mean: 2.2, std: 0.1, Likelihood: 1535.2762294855227
Mean: 2.2, std: 0.2, Likelihood: 196.36865634918325
Mean: 2.2, std: 0.3, Likelihood: 8.39906583033428
Mean: 2.2, std: 0.4, Likelihood: 0.6487369156209131
Mean: 2.2, std: 0.5, Likelihood: 0.08062765884824176
Mean: 2.3, std: 0.1, Likelihood: 3.8055692956214573
Mean: 2.3, std: 0.2, Likelihood: 43.81576973932554
Mean: 2.3, std: 0.3, Likelihood: 4.312224181175323
Mean: 2.3, std: 0.4, Likelihood: 0.445869926862178
Mean: 2.3, std: 0.5, Likelihood: 0.06342396282259623
Mean: 2.4, std: 0.1, Likelihood: 4.282604055890112e-07
Mean: 2.4, std: 0.2, Likelihood: 0.8025138161774141
Mean: 2.4, std: 0.3, Likelihood: 0.72882330563465
Mean: 2.4, std: 0.4, Likelihood: 0.16402637952920995
Mean: 2.4, std: 0.5, Likelihood: 0.033442975099142944
```

[5]:
```python
import numpy as np
### 4. Maximum likelihood estimation ###
print("### 4. Maximum likelihood estimation ###")

max_likelihood = -99
for mean in np.arange(2, 3, 0.001):
    for std in np.arange(0.1, 0.5, 0.001):
        likelihood = calculate_likelihood(measurements, mean, std)
        if max_likelihood < likelihood:
            max_likelihood = likelihood
            best_mean = mean
            best_std_dev = std

print("Maximum Likelihood Estimation:")
print(f"Best Mean: {best_mean}")
print(f"Best Standard Deviation: {best_std_dev}")
print(f"Max Likelihood: {max_likelihood}")
```

```
### 4. Maximum likelihood estimation ###
Maximum Likelihood Estimation:
Best Mean: 2.189999999999979
Best Standard Deviation: 0.11400000000000002
Max Likelihood: 1925.8398074486447
```

[5]:

[5]:

[5]:

[ ]: