# Analysis Knowledge Check and Practical Exercise

## Analysis Knowledge Check (5 points total)

### 1. Linear Regression Formula (1 point):

y = β₀ + β₁x + ε

Where:

- y: Target variable

- x: Input feature

- β₀: Intercept

- β₁: Coefficient (slope)

- ε: Error term

### 2. Definition and Rationale for MSE (1 point):

Mean Squared Error (MSE) is the average of the squared differences between actual and predicted values:

MSE = (1/n) Σ (yᵢ - ŷᵢ)²

We use MSE as a cost function because it:

- Penalizes larger errors more significantly (due to squaring).

- Ensures a smooth, convex optimization surface for gradient-based methods.

### 3. Gradient Descent Update Rule (1 point):

The gradient descent update rule adjusts parameters iteratively as follows:

θⱼ := θⱼ - α ∂J(θ)/∂θⱼ

Where:

- α: Learning rate

- ∂J(θ)/∂θⱼ: Gradient of the cost function with respect to the parameter.

### 4. Derivation of ∂J/∂β₁ in Simple Linear Regression (1 point):

To derive ∂J/∂β₁, we differentiate the MSE cost function with respect to β₁:

J(β₁) = (1/n) Σ (yᵢ - (β₀ + β₁xᵢ))²

The partial derivative is:

∂J/∂β₁ = -(2/n) Σ xᵢ (yᵢ - ŷᵢ)

This shows how the gradient is proportional to the weighted sum of residuals.

### 5. When to Use Multiple Linear Regression (1 point):

We prefer multiple linear regression when the outcome y depends on multiple independent variables (x₁, x₂, ..., xₖ), allowing us to account for more factors and improve prediction accuracy.

## Practical “Learn by Doing” Exercise (5 points)

### Objective:

Build and interpret a simple linear regression model.

### 1. Data Collection (1 point):

Dataset: Hours of study (x) vs. Test scores (y):

x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

y = [50, 55, 60, 65, 70, 75, 80, 85, 90, 95]

### 2. Model Building (2 points):

Using scikit-learn:

import numpy as np  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
  
# Data  
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape(-1, 1)  
y = np.array([50, 55, 60, 65, 70, 75, 80, 85, 90, 95])  
  
# Fit model  
model = LinearRegression()  
model.fit(X, y)  
  
# Parameters  
intercept = model.intercept\_  
coefficient = model.coef\_[0]  
  
print(f"Intercept (β₀): {intercept}")  
print(f"Coefficient (β₁): {coefficient}")

#### Output:

β₀ = 45.0, β₁ = 5.0

### 3. Evaluation (1 point):

Code:

# Predictions and MSE  
predictions = model.predict(X)  
mse = mean\_squared\_error(y, predictions)  
  
print(f"MSE: {mse}")

#### Output:

MSE = 0.0

Comment: Since the data follows a perfect linear relationship, the MSE is 0, indicating an excellent fit.

### 4. Interpretation (1 point):

In plain English, β₁ = 5.0 means that for every additional hour of study, the test score increases by 5 points on average.