

1) A.

```
[19] ✓ 0s
    # Given data
    x = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
    y = np.array([52, 55, 61, 70, 82])

    print("Study Hours:", x.flatten())
    print("Marks:", y)

▼ Study Hours: [1 2 3 4 5]
  Marks: [52 55 61 70 82]
```

B.

```
[20] ✓ 0s
    # Linear Regression Model
    linear_model = LinearRegression()
    linear_model.fit(x, y)

    # Coefficients
    beta_0 = linear_model.intercept_
    beta_1 = linear_model.coef_[0]

    print(f'Regression Equation:')
    print(f'ŷ = {beta_0:.2f} + {beta_1:.2f}x')

▼ Regression Equation:
  ŷ = 41.50 + 7.50x
```

2) A.

```
[21] ✓ 0s
    # Polynomial Features (degree = 4)
    poly = PolynomialFeatures(degree=4)
    x_poly = poly.fit_transform(x)

    # Polynomial Regression Model
    poly_model = LinearRegression()
    poly_model.fit(x_poly, y)

    # Coefficients
    print("Polynomial Coefficients:")
    for i, coef in enumerate(poly_model.coef_):
        print(f'w{i} = {coef:.4f}')
    print(f'w0 (Intercept) = {poly_model.intercept_:.4f}')

▼ Polynomial Coefficients:
  w0 = 0.0000
  w1 = -1.5000
  w2 = 1.5000
  w3 = 0.0000
  w4 = -0.0000
  w0 (Intercept) = 52.0000
```

b.

```
[22] ✓ 0s
    x_test = np.array([[6]])

    # Prediction using Linear Model
    y_pred_linear = linear_model.predict(x_test)

    # Prediction using Polynomial Model
    x_test_poly = poly.transform(x_test)
    y_pred_poly = poly_model.predict(x_test_poly)

    print(f"Prediction at x = 6 hours:")
    print(f"Linear Model: {y_pred_linear[0]:.2f}")
    print(f"Polynomial Model: {y_pred_poly[0]:.2f}")

▼ Prediction at x = 6 hours:
  Linear Model: 86.50
  Polynomial Model: 97.00
```

3) A.

```
[23] ✓ 0s
    # Predictions on training data
    y_train_pred_linear = linear_model.predict(x)

    # MSE
    mse_linear = mean_squared_error(y, y_train_pred_linear)

    print(f"Training MSE (Linear Model): {mse_linear:.4f}")

▼ Training MSE (Linear Model): 6.3000
```

b.

```
[24] ✓ 0s
    # Predictions on training data
    y_train_pred_poly = poly_model.predict(x_poly)

    # MSE
    mse_poly = mean_squared_error(y, y_train_pred_poly)

    print(f"Training MSE (Polynomial Model): {mse_poly:.4f}")

▼ Training MSE (Polynomial Model): 0.0000
```

4)

Bias–Variance Analysis

- **Higher Bias:**

Model A (Linear Regression)

Reason: It is too simple to capture the non-linear relationship.

- **Higher Variance:**

Model B (Polynomial Regression – Degree 4)

Reason: It closely fits the training data and may overfit, making it sensitive to data changes.