Question 1: What is Simple Linear Regression?
Answer: simple linear regression attempts to determine the strength and characteristics of relationship between one independent and another dependent variable.

Question 2: What are the key assumptions of Simple Linear Regression?
Answer: **Linearity**:
There is a linear relationship between the independent variable (X) and the dependent variable (Y).

- **Independence of Errors**:
  The residuals (errors) are independent of each other.

- **Homoscedasticity**:
  The variance of residuals is constant across all levels of X.

- **Normality of Errors**:
  The residuals are normally distributed.

- **No Measurement Error in X**:
  The independent variable is measured accurately (no error in X).

Question 3: What is heteroscedasticity, and why is it important to address in regression models?
Answer:**Heteroscedasticity** refers to a condition in a regression model where the **variance of the residuals (errors) is not constant** across all levels of the independent variable.

In simple terms, the spread of errors changes as the value of X changes — for example, errors might increase or decrease as X increases.

**Violates Regression Assumptions**:

- It breaks the assumption of **homoscedasticity** (constant variance), which is key for valid regression analysis.

**Inaccurate Predictions**:

- Leads to **biased or inefficient estimates** of standard errors, making confidence intervals and hypothesis tests unreliable.

**Invalid Significance Tests**:

- p-values and t-statistics may be incorrect, possibly leading to wrong conclusions about the importance of predictors.

**Reduced Model Efficiency**:

- Ordinary Least Squares (OLS) is no longer the Best Linear Unbiased Estimator (BLUE) under heteroscedasticity.

Question 4: What is Multiple Linear Regression?
Answer: **Multiple Linear Regression (MLR)** is a statistical technique used to model the relationship between **one dependent variable (Y)** and **two or more independent variables ($X_1$, $X_2$, ..., $X$□)**.

Question 5: What is polynomial regression, and how does it differ from linear regression?
Answer:**Polynomial Regression** is a type of regression that models the relationship between the **independent variable (X)** and the **dependent variable (Y)** as an **nth-degree polynomial**.
**Linear Regression** fits a straight line.

**Polynomial Regression** fits a **curved line** to better model non-linear patterns.

Question 6: Implement a Python program to fit a Simple Linear Regression model to the following sample data: ● X = [1, 2, 3, 4, 5] ● Y = [2.1, 4.3, 6.1, 7.9, 10.2] Plot the regression line over the data points.
Answer:import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

```
# Sample Data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)  # Reshape for sklearn
Y = np.array([2.1, 4.3, 6.1, 7.9, 10.2])

# Create and train the model
model = LinearRegression()
model.fit(X, Y)

# Predict values
```

```
Y_pred = model.predict(X)

# Plot original data points
plt.scatter(X, Y, color='blue', label='Data Points')

# Plot regression line
plt.plot(X, Y_pred, color='red', label='Regression Line')

# Labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Linear Regression')
plt.legend()
plt.grid(True)
plt.show()
```

Question 7: Fit a Multiple Linear Regression model on this sample data: ● Area = [1200, 1500, 1800, 2000] ● Rooms = [2, 3, 3, 4] ● Price = [250000, 300000, 320000, 370000]
Answer:

```
import numpy as np
from sklearn.linear_model import LinearRegression

# Input features: Area and Rooms
X = np.array([
    [1200, 2],
    [1500, 3],
    [1800, 3],
    [2000, 4]
])

# Target variable: Price
Y = np.array([250000, 300000, 320000, 370000])

# Create and train the model
model = LinearRegression()
model.fit(X, Y)

# Print coefficients and intercept
print("Intercept (β₀):", model.intercept_)
print("Coefficients (β₁ for Area, β₂ for Rooms):", model.coef_)

# Predict for the same data
Y_pred = model.predict(X)
print("\nPredicted Prices:", Y_pred)
```

Question 8: Implement polynomial regression on the following data: ● X = [1, 2, 3, 4, 5] ● Y = [2.2, 4.8, 7.5, 11.2, 14.7] Fit a 2nd-degree polynomial and plot the resulting curve.
Answer:import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

```
# Sample Data
X = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)
Y = np.array([2.2, 4.8, 7.5, 11.2, 14.7])

# Transform input features to include polynomial terms (degree = 2)
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Create and train the model
model = LinearRegression()
model.fit(X_poly, Y)

# Predict using the model
X_range = np.linspace(1, 5, 100).reshape(-1, 1)
X_range_poly = poly.transform(X_range)
Y_pred = model.predict(X_range_poly)

# Plot original data
plt.scatter(X, Y, color='blue', label='Data Points')

# Plot polynomial regression curve
plt.plot(X_range, Y_pred, color='red', label='2nd Degree Polynomial Curve')

# Labels and title
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Polynomial Regression (Degree 2)')
plt.legend()
plt.grid(True)
plt.show()
```

Question 9: Create a residuals plot for a regression model trained on this data: ● X = [10, 20, 30, 40, 50] ● Y = [15, 35, 40, 50, 65] Assess heteroscedasticity by examining the spread of residuals.
Answer:import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

```
# Sample Data
X = np.array([10, 20, 30, 40, 50]).reshape(-1, 1)
Y = np.array([15, 35, 40, 50, 65])

# Train the regression model
model = LinearRegression()
model.fit(X, Y)

# Predict values
Y_pred = model.predict(X)

# Calculate residuals
residuals = Y - Y_pred

# Plot residuals
plt.scatter(X, residuals, color='purple', marker='o')
plt.axhline(y=0, color='black', linestyle='--')

# Labels and title
plt.xlabel('X')
plt.ylabel('Residuals (Y - Y_pred)')
plt.title('Residuals Plot')
plt.grid(True)
plt.show()
```

Question 10: Imagine you are a data scientist working for a real estate company. You need to predict house prices using features like area, number of rooms, and location. However, you detect heteroscedasticity and multicollinearity in your regression model. Explain the steps you would take to address these issues and ensure a robust model.
Answer:As a data scientist working for a real estate company, predicting house prices using features like area, number of rooms, and location, it's crucial to ensure the regression model is statistically sound. If **heteroscedasticity** is detected, which means the variance of errors is not constant across all values of the predictors, it can lead to inefficient estimates and unreliable hypothesis testing. To address this, I would apply a **log transformation** or **Box-Cox transformation** on the target variable (house price) or certain input features like area. This often stabilizes the variance. If the problem persists, I might use **Weighted Least Squares (WLS)** or **robust regression** techniques that adjust for non-constant error variance.

On the other hand, **multicollinearity** arises when independent variables are highly correlated with each other, making it hard to determine the individual effect of each feature. To handle this, I would calculate the **Variance Inflation Factor (VIF)** to detect which variables are causing multicollinearity. Based on the results, I could **remove one of the correlated variables**, combine them into a new meaningful feature (e.g., area per room), or use **dimensionality reduction techniques** like **Principal Component Analysis (PCA)**.

Additionally, applying **regularization methods** such as **Ridge or Lasso Regression** can help manage multicollinearity by penalizing large or unnecessary coefficients.

Throughout the process, I would ensure proper **feature scaling**, validate results using **cross-validation**, and monitor the **residual plots** to confirm improvements. These steps would help in building a **robust, interpretable, and accurate model** for predicting house prices, ensuring the insights are reliable for business decisions.