```
1.import numpy as np

# Generate sample data
np.random.seed(0)
num_samples = 1000

# Square footage (in square feet)
square_footage = np.random.randint(1000, 5000,
num_samples)

# Number of bedrooms
num_bedrooms = np.random.randint(1, 6, num_samples)

# Number of bathrooms
num_bathrooms = np.random.randint(1, 4, num_samples)

# Generating prices (in dollars)
# Price = 100 * square_footage + 5000 * num_bedrooms +
7000 * num_bathrooms + random noise
prices = 100 * square_footage + 5000 * num_bedrooms +
7000 * num_bathrooms + np.random.normal(0, 20000,
num_samples)

# Reshape features into a matrix
X = np.column_stack((square_footage, num_bedrooms,
num_bathrooms))

# Reshape target variable into a column vector
y = prices.reshape(-1, 1)

# Add a column of ones for the intercept term
X = np.column_stack((np.ones(num_samples), X))

# Compute the coefficients using the normal equation
(closed-form solution)
```

```
coefficients = np.linalg.inv(X.T @ X) @ X.T @ y

# Print the coefficients
print("Intercept:", coefficients[0])
print("Coefficient for square footage:", coefficients[1])
print("Coefficient for number of bedrooms:", coefficients[2])
print("Coefficient for number of bathrooms:",
coefficients[3])
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:/Users/ASUS/AppData/Local/Programs/Python/Python311/task 1.py ===
Intercept: [-465.2404927]
Coefficient for square footage: [100.49147184]
Coefficient for number of bedrooms: [4672.7807812]
Coefficient for number of bathrooms: [6882.61113518]
```

```
2.import numpy as np

# Generate a custom dataset representing purchase
history of customers
# Each row represents a customer, and each column
represents a feature (e.g., total spent, num purchases, avg
purchase amount)
# Here, we generate 100 customers with 3 features each
num_customers = 100
purchase_history = np.random.randint(1, 100,
size=(num_customers, 3))

# Define the number of clusters (K)
k = 5

# Randomly initialize centroids
centroids =
purchase_history[np.random.choice(purchase_history.sha
pe[0], k, replace=False)]

# Define a function to assign each data point to the
closest centroid
def assign_clusters(data, centroids):
    distances = np.sqrt(((data - centroids[:,
np.newaxis])**2).sum(axis=2))
```

```
    return np.argmin(distances, axis=0)

# Define a function to update centroids based on the mean
of data points assigned to each cluster
def update_centroids(data, clusters, k):
    new_centroids = np.zeros_like(centroids)
    for i in range(k):
        new_centroids[i] = np.mean(data[clusters == i], axis=0)
    return new_centroids

# Perform K-means clustering
max_iter = 100
for i in range(max_iter):
    clusters = assign_clusters(purchase_history, centroids)
    new_centroids = update_centroids(purchase_history,
clusters, k)
    if np.all(centroids == new_centroids):
        print("Converged after", i+1, "iterations.")
        break
    centroids = new_centroids
```

IDLE Shell 3.11.0

File Edit Shell Debug Options Window Help

```
>>>
== RESTART: C:/Users/ASUS/AppData/Local/Programs/Python/Python311/task new.py ==
Converged after 14 iterations.
Customer 1 is in cluster 4
Customer 2 is in cluster 3
Customer 3 is in cluster 2
Customer 4 is in cluster 5
Customer 5 is in cluster 2
Customer 6 is in cluster 5
Customer 7 is in cluster 5
Customer 8 is in cluster 5
Customer 9 is in cluster 4
Customer 10 is in cluster 5
Customer 11 is in cluster 5
Customer 12 is in cluster 1
Customer 13 is in cluster 2
Customer 14 is in cluster 2
Customer 15 is in cluster 5
Customer 16 is in cluster 2
Customer 17 is in cluster 1
Customer 18 is in cluster 5
Customer 19 is in cluster 3
Customer 20 is in cluster 3
```