```
# Q1: Linear Regression - Predict salary for 3.5 years of experience
import numpy as np
from sklearn.linear_model import LinearRegression

X_lr = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)
y_lr = np.array([35, 40, 45, 50, 55, 60])
model_lr = LinearRegression()
model_lr.fit(X_lr, y_lr)
predicted_salary = model_lr.predict(np.array([[3.5]]))[0]
print(f"Q1 Linear Regression predicted salary for 3.5 years: {predicted_salary}k")
```

```
# Q2: Logistic Regression - Predict probability of passing exam after 3.5 hours studied
import numpy as np
from sklearn.linear_model import LogisticRegression

X_logr = np.array([1, 2, 3, 4, 5, 6]).reshape(-1, 1)
y_logr = np.array([0, 0, 0, 1, 1, 1])
model_logr = LogisticRegression()
model_logr.fit(X_logr, y_logr)
predicted_prob = model_logr.predict_proba(np.array([[3.5]]))[0][1]
print(f"Q2 Logistic Regression predicted pass probability for 3.5 hours: {predicted_prob:.2f}")
```

```
# Q3: Naive Bayes - Predict if tennis game will be played given Sunny and Cool
from sklearn.naive_bayes import GaussianNB
import numpy as np

# Encode features manually as numbers
weather_map = {'Sunny': 0, 'Overcast': 1, 'Rainy': 2}
temperature_map = {'Hot': 0, 'Mild': 1, 'Cool': 2}

# Dataset
X_nb = np.array([
    [weather_map['Sunny'], temperature_map['Hot']],
    [weather_map['Sunny'], temperature_map['Hot']],
    [weather_map['Overcast'], temperature_map['Hot']],
    [weather_map['Rainy'], temperature_map['Mild']],
    [weather_map['Rainy'], temperature_map['Cool']],
    [weather_map['Rainy'], temperature_map['Mild']],
    [weather_map['Overcast'], temperature_map['Cool']],
    [weather_map['Sunny'], temperature_map['Mild']],
    [weather_map['Sunny'], temperature_map['Cool']],
    [weather_map['Rainy'], temperature_map['Mild']]
])

y_nb = np.array([0, 0, 1, 1, 1, 0, 1, 0, 1, 1])  # Play Tennis (0=No, 1=Yes)

model_nb = GaussianNB()
model_nb.fit(X_nb, y_nb)
```

```
# Predict for Sunny and Cool
test_sample = np.array([[weather_map['Sunny'], temperature_map['Cool']]])
pred_nb = model_nb.predict(test_sample)[0]
pred_nb_label = 'Yes' if pred_nb == 1 else 'No'
print(f"Q3 Simple Naive Bayes prediction (Sunny, Cool): {pred_nb_label}")
```

```
# Q4: k-NN - Predict flower class for petal length 4.6
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

X_knn = np.array([1.4, 1.3, 4.7, 4.5, 5.1, 5.9]).reshape(-1, 1)
y_knn = ['Setosa', 'Setosa', 'Versicolor', 'Versicolor', 'Virginica', 'Virginica']

model_knn = KNeighborsClassifier(n_neighbors=3)
model_knn.fit(X_knn, y_knn)

pred_knn = model_knn.predict(np.array([[4.6]]))[0]
print(f"Q4 k-NN predicted class for petal length 4.6: {pred_knn}")
```

```
# Q5: Apriori Algorithm (manual) - Frequent itemsets and association rules
transactions = [
    ['Bread', 'Milk'],
    ['Bread', 'Diaper', 'Juice', 'Eggs'],
    ['Milk', 'Diaper', 'Juice', 'Cola'],
    ['Bread', 'Milk', 'Diaper', 'Juice'],
    ['Bread', 'Milk', 'Diaper', 'Cola']
]

min_support = 0.6  # 60%
min_support_count = int(min_support * len(transactions))

item_counts = {}

for transaction in transactions:
    for item in transaction:
        if item in item_counts:
            item_counts[item] += 1
        else:
            item_counts[item] = 1

print("Q5 Frequent single items (min support 60%):")
for item, count in item_counts.items():
    if count >= min_support_count:
        print(f"{item} appears {count} times")
```

```
# Q6: K-Means Clustering - Cluster example data points into 2 clusters
import numpy as np
from sklearn.cluster import KMeans

data_points = np.array([[1, 2], [2, 1], [4, 3], [2, 5], [8, 5], [8, 6], [6, 9]])
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(data_points)

print("Q6 K-Means cluster centroids:")
print(kmeans.cluster_centers_)
print("Q6 Cluster labels for data points:")
print(kmeans.labels_)
```