Α) **ΚΩΔΙΚΑΣ:**

```
import os


def read_data(filename):
  if not os.path.exists(filename):
    raise FileNotFoundError(f"Το αρχείο {filename} δεν βρέθηκε.")


  X = []
  y = []
  with open(filename, 'r') as file:
    next(file)
    for line in file:
      parts = line.strip().split(',')
      if len(parts) == 3:
        x1, x2, label = float(parts[0]), float(parts[1]), float(parts[2])
        y.append(int(label))
        X.append([x1, x2])
  return X, y


def step_function(x):
  return 1 if x >= 0 else -1


def train_perceptron(X, y, learning_rate, epochs):
  num_features = len(X[0])
  weights = [0.0] * num_features
  bias = 0.0


  for _ in range(epochs):
    for i in range(len(X)):
      linear_output = sum(weights[j] * X[i][j] for j in range(num_features)) + bias
      prediction = step_function(linear_output)
```

```
        if prediction != y[i]:
            for j in range(num_features):
                weights[j] += learning_rate * y[i] * X[i][j]
            bias += learning_rate * y[i]

    return weights, bias

def main():
    training_file = 'training_data.csv'
    learning_rate = 0.1
    epochs = 100

    X_train, y_train = read_data(training_file)
    weights, bias = train_perceptron(X_train, y_train, learning_rate, epochs)
    print(f"Τελικά βάρη: {weights}")
    print(f"Bias: {bias}")

if __name__ == "__main__":
    main()
```

ΠΕΡΙΓΡΑΦΗ:

Ο παραπάνω κώδικας υλοποιεί ένα απλό perceptron για την ταξινόμηση δεδομένων σε δύο κατηγορίες {-1, 1}. Πρώτα, φορτώνει δεδομένα από το αρχείο training_data.csv, όπου κάθε γραμμή περιέχει δύο χαρακτηριστικά X1,X2 και μία ετικέτα Label. Στη συνέχεια, εκπαιδεύει το perceptron χρησιμοποιώντας τη μέθοδο της εποπτευόμενης μάθησης. Για κάθε δείγμα, υπολογίζει τη γραμμική έξοδο και εφαρμόζει τη βήμα ενεργοποίησης (step function) για να προβλέψει την κατηγορία. Εάν η πρόβλεψη είναι λανθασμένη, ενημερώνει τα βάρη και το bias με βάση έναν προκαθορισμένο ρυθμό μάθησης (learning rate). Η εκπαίδευση επαναλαμβάνεται για συγκεκριμένο αριθμό εποχών (epochs), και στο τέλος επιστρέφει τα τελικά βάρη και το bias.

ΑΠΑΝΤΗΣΗ:

Τελικά βάρη: [0.25227230205190393, 0.29233324611090 45]

Bias: -0.1 .

Β) **ΚΩΔΙΚΑΣ:**

```
def read_data(filename):
```

```
    X = []
    y = []
    with open(filename, 'r') as file:
        for line in file:
            parts = line.strip().split(',')
            if len(parts) == 3:
                x1, x2, label = float(parts[0]), float(parts[1]), float(parts[2])
                label = 1 if label > 0 else -1
                X.append([x1, x2])
                y.append(label)
    return X, y

def step_function(x):
    return 1 if x >= 0 else -1

def test_perceptron(X, y, weights, bias):
    correct_predictions = 0
    for i in range(len(X)):
        linear_output = sum(weights[j] * X[i][j] for j in range(len(weights))) + bias
        prediction = step_function(linear_output)
        if prediction == y[i]:
            correct_predictions += 1
    return correct_predictions / len(X)

def main():
    testing_file = 'test_data.csv'
    weights = [0.25227230205190393, 0.29233332461109045]
    bias = -0.1
    X_test, y_test = read_data(testing_file)
    accuracy = test_perceptron(X_test, y_test, weights, bias)
    print(f"Ακρίβεια: {accuracy * 100:.2f}%")
```

```
if __name__ == "__main__":
    main()
```

ΠΕΡΙΓΡΑΦΗ:

Ο παραπάνω κώδικας υπολογίζει την ακρίβεια ενός ήδη εκπαιδευμένου perceptron στα δεδομένα αξιολόγησης που βρίσκονται στο αρχείο test_data.csv. Αρχικά, φορτώνει τα δεδομένα με τη συνάρτηση read_data, η οποία διαβάζει τις τιμές εισόδου (X1, X2) και τις ετικέτες (-1 ή 1) από το αρχείο. Στη συνέχεια, χρησιμοποιεί τις τιμές των βαρών και του bias που προέκυψαν από την εκπαίδευση του perceptron, για να προβλέψει τις ετικέτες των δεδομένων αξιολόγησης με τη βοήθεια της συναρτησιακής εξόδου και της συνάρτησης ενεργοποίησης step_function. Τέλος, συγκρίνει τις προβλέψεις με τις πραγματικές ετικέτες για να υπολογίσει τη συνολική ακρίβεια, η οποία εκφράζεται ως ποσοστό σωστών προβλέψεων επί του συνολικού αριθμού δειγμάτων.

ΑΠΑΝΤΗΣΗ:

```
Φόρτωση δεδομένων αξιολόγησης...

Υπολογισμός ακρίβειας...

Ακρίβεια: 96.00% .
```