**Algorithm #2 : Clothing Predictor using occurrence based Weightage.**

class SweaterPredictor:

def \_\_init\_\_(self):

self.data = {}

def train\_data(self, temperature, prediction):

if temperature not in self.data:

self.data[temperature] = []

self.data[temperature].append(prediction)

def predict(self, temperature):

if temperature in self.data:

predictions = self.data[temperature]

yes\_count = predictions.count("Yes")

no\_count = predictions.count("No")9

if yes\_count > no\_count:

return "Wear a sweater"

else:

return "No need for a sweater"

else:

closest\_temp = min(self.data.keys(), key=lambda x: abs(x - temperature))

if closest\_temp in self.data:

predictions = self.data[closest\_temp]

yes\_count = predictions.count("Yes")

no\_count = predictions.count("No")

if yes\_count > no\_count:

return "Wear a sweater"

else:

return "No need for a sweater"

predictor = SweaterPredictor()

# Adding temperature data and predictions to the predictor

predictor.train\_data(12, "Yes")

predictor.train\_data(15, "No")

predictor.train\_data(18, "No")

predictor.train\_data(10, "Yes")

predictor.train\_data(18, "Yes")

predictor.train\_data(18, "No")

predictor.train\_data(15, "No")

predictor.train\_data(10, "Yes")

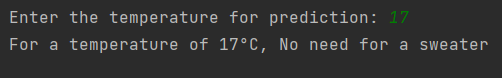
# Get prediction based on the temperature

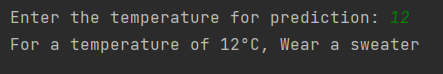
temperature\_to\_predict = int(input("Enter the temperature for prediction: "))

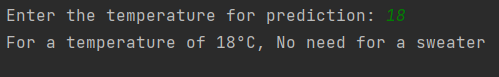
prediction = predictor.predict(temperature\_to\_predict)

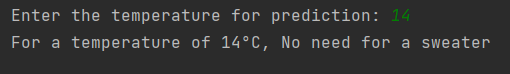
print(f"For a temperature of {temperature\_to\_predict}°C, {prediction}")

**OUTPUT :**

****

****

****

****