```
class NaiveBayes:
   def __init__(self, f, r):
      self.features = f
      self.response = r

   def predict(self,custcase):
      anskeys = list(self.response.keys())
      ansvalues = dict.fromkeys(anskeys,0)
      #print(custcase)
      for key in anskeys :
         ansvalues[key] = self.response[key]

         for ikey, ival in custcase.items() :
            ansvalues[key] = ansvalues[key] * self.features[ikey][ival][key]

      print(ansvalues)

      #calculating MAP
      maxkey=""
      maxans=-1
      for ikey, ival in ansvalues.items():
         if ival > maxans :
            maxans= ival
            maxkey = ikey
      return maxkey


#precalculated values from worksheet - "naive bayes classifier working"
response = {"Wait":0.4, "Leave":0.6}

features = {
   "Reservation":
               {
                  "Yes" : {"Wait":0.5, "Leave":0.666667},
                   "No" : {"Wait":0.5, "Leave":0.333333}
```

```
                } ,
   "Time>30":
                {
                    "Yes" : {"Wait":0.25, "Leave":0.83333},
                     "No" : {"Wait":0.75, "Leave":0.16667}
                }
}



nb = NaiveBayes(features, response)

#print("Probability :", nb.features["Reservation"]["Yes"]["Wait"])
#print("Probability :", nb.features["Time>30"]["No"]["Leave"])

resstatus = input("Manager asks Customer, have you reserved table?(Yes/No):")
timestatus = input("Customer asks Manager, Will it take more than 30
mins?(Yes/No):")

custcase = {"Reservation":resstatus, "Time>30":timestatus}

print("Manager predicts that Customer will :" , nb.predict(custcase) )
```