3. Write a program to demostrate the working of the decision tree based ID3 algorithm. Use an appropriate dataset for building the decision tree and apply this knowleolge to classify a new sample.

```
import pandas as pd
from pandas import DataFrame
df_tennis = pd.read_csv('C:/Users/user/Desktop/
                    MLLab/Tennis_data.csv')
 attribute_names = list(df_tennis.columns)
 attribute-names.remove('Play Tennis')
 print(attribute-names)

def entropy_of_list(lst):
    from collections import counter
     count = counter(x for x in lst)
     num_instances = len(lst)*L
     probs = [x]/num_instances for x in
                    count.values()]
     return entropy(probs)

def entropy(probs):
    import math
    return sum([-prob * math.log(prob,2) for
                    prob in probs])
total_entropy = entropy_of_list(df_tennis('PlayTennis')]
```

Teacher's Signature :

```
def information_gain (df, split_attribute_name,
                      target_attribute_name, trace=0)
   df_split = df. groupby (split_attribute_name)
    nobs = len(df. index)* 1
    df_agg_ent = df_split.agg ({target_attribute_
                   name :[entropy_of_list, lambda x: len(x)/nobs]})
   df_agg_ent.columns = ['Entropy', 'propobservations']
   new_entropy = sum [df, agg_ent ['Entropy']* df_agg_ent
                    ['propobservations']])
   old_entropy = entropy_of_list (df [target_attribute_name])
  print (split attribute_name, 'IG: ', old_entropy-
                              new entropy)
   return   old entropy-new entropy

def id3 (df, target_attribute_name, attribute_names,
                          default_class=None):
   from collections import Counter
   count = Counter( x for X in df [target_attribute_name]
   if len (count) == 1:
        return next (iter (count))
    elif df.empty or (not attribute_names):
        return default_class
    else:
        default_class = max(count.keys())
        gain = [
          information gain (df, attr, target_attribute_name)
            for attr in attribute_names]
    index_of_max = gain. index [max(gain)]
```

```
best_attr = attribute_names [index_of_max ]

tree = { best_attr: { } }

remaining_attribute_names = [i for i in attribute_names
                        if i != best_attr]

for attr_values, data_subset in df.groupby (best_attr):
        subtree = id3 (data_subset, target_attribute_name,
                        remaining_attribute_names, default_class)
    test [best_attr] [attr_val] = subtree
  return tree

from pprint import pprint
    tree = id3 (df_tennis, 'PlayTennis', attribute_names)
print ("\n The Resultant Decision Tree is: \n")
pprint (tree)
```

Teacher's Signature :

output :

```
['outlook', 'Temperature', 'Humidity', 'wind']

outlook IG : 0.2467498197744391
Temperature IG : 0.029222565658954647
Humidity IG : 0.15183550136234136
Wind IG : 0.04812703040826927

Temperature IG : 0.01997309402197489
Humidity IG : 0.01997309402197489
Wind IG : 0.9709505944546686

Temperature IG : 0.5709505944546686
Humidity IG : 0.9709505944546686
Wind IG : 0.01997309402197489

The Resultant Decision Tree is :

{'outlook' : {'overcast' : 'Yes',
             'Rain' : {'wind' : {'strong' : 'NO', 'weak' : 'Yes'}},
            'Sunny' : {'Humidity' : {'High' : 'NO', 'Normal' : 'Yes'}}}}
```