In [1]:

```
import pandas as pd

df_tennis = pd.read_csv('PlayTennis.csv')
print("\n Given Play Tennis Data Set:\n\n", df_tennis)
```

```
 Given Play Tennis Data Set:

    PlayTennis   Outlook Temperature Humidity    Wind
0           No     Sunny         Hot     High    Weak
1           No     Sunny         Hot     High  Strong
2          Yes  Overcast         Hot     High    Weak
3          Yes     Rainy        Mild     High    Weak
4          Yes     Rainy        Cool   Normal    Weak
5           No     Rainy        Cool   Normal  Strong
6          Yes  Overcast        Cool   Normal  Strong
7           No     Sunny        Mild     High    Weak
8          Yes     Sunny        Cool   Normal    Weak
9          Yes     Rainy        Mild   Normal    Weak
10         Yes     Sunny        Mild   Normal  Strong
11         Yes  Overcast        Mild     High  Strong
12         Yes  Overcast         Hot   Normal    Weak
13          No     Rainy        Mild     High  Strong
```

In [2]:

```
df_tennis.columns[0]
```

Out[2]:

```
'PlayTennis'
```

In [3]:

```
def entropy(probs):
    import math
    return sum( [-prob*math.log(prob, 2) for prob in probs] )

#Function to calulate the entropy of the given Data Sets/List with respect to target attrib
def entropy_of_list(a_list):
    #print("A-list",a_list)
    from collections import Counter
    cnt = Counter(x for x in a_list)   # Counter calculates the propotion of class
    print("\nClasses:",cnt)
    #print("No and Yes Classes:",a_list.name,cnt)
    num_instances = len(a_list) # = 14
    print("\n Number of Instances of the Current Sub Class is {0}:".format(num_instances ))
    probs = [x / num_instances for x in cnt.values()]  # x means no of YES/NO
    print(probs)
    print("\n Classes:",list(cnt.keys()))#min(cnt),max(cnt))
    print(" \n Probabilities of Class {0} is {1}:".format(min(cnt),min(probs)))
    print(" \n Probabilities of Class {0} is {1}:".format(max(cnt),max(probs)))
    return entropy(probs) # Call Entropy :

# The initial entropy of the YES/NO attribute for our dataset.
print("\n  INPUT DATA SET FOR ENTROPY CALCULATION:\n", df_tennis['PlayTennis'])

total_entropy = entropy_of_list(df_tennis['PlayTennis'])

print("\n Total Entropy of PlayTennis Data Set:",total_entropy)
```

```
  INPUT DATA SET FOR ENTROPY CALCULATION:
 0      No
1       No
2      Yes
3      Yes
4      Yes
5       No
6      Yes
7       No
8      Yes
9      Yes
10     Yes
11     Yes
12     Yes
13      No
Name: PlayTennis, dtype: object

Classes: Counter({'Yes': 9, 'No': 5})

 Number of Instances of the Current Sub Class is 14:
[0.35714285714285715, 0.6428571428571429]

 Classes: ['No', 'Yes']

 Probabilities of Class No is 0.35714285714285715:

 Probabilities of Class Yes is 0.6428571428571429:

 Total Entropy of PlayTennis Data Set: 0.9402859586706309
```

In [4]:

```
def information_gain(df, split_attribute_name, target_attribute_name, trace=0):
    print("Information Gain Calculation of ",split_attribute_name)
    '''
    Takes a DataFrame of attributes, and quantifies the entropy of a target
    attribute after performing a split along the values of another attribute.
    '''
    # Split Data by Possible Vals of Attribute:
    df_split = df.groupby(split_attribute_name)
    print("split:",type(df_split))
    for name,group in df_split:
        print("Name:\n",name)
        print("Group:\n",group)

    # Calculate Entropy for Target Attribute, as well as
    # Proportion of Obs in Each Data-Split

    nobs = len(df.index)
    print("NOBS",nobs)
    #define the aggregation based on target attribute name df_agg_ent=df_ent_prob
    df_ent_prob = df_split.agg({target_attribute_name : [entropy_of_list, lambda x: len(x)/
    #print(target_attribute_name)
    #print("df is",df_agg_ent)
    #print(" Entropy List ",entropy_of_list)
    print(df_ent_prob.columns)
    print("the entropy and the probability value for each attribute is",df_ent_prob)#[targe
    df_ent_prob.columns = ['Entropy', 'PropObservations']
    #if trace: # helps understand what fxn is doing:
     #   print(df_agg_ent)

    # Calculate Information Gain:
    new_entropy = sum( df_ent_prob['Entropy'] * df_ent_prob['PropObservations'] )
    overall_entropy = entropy_of_list(df[target_attribute_name])
    return overall_entropy - new_entropy


print('Info-gain for Outlook is :'+str( information_gain(df_tennis, 'Outlook', 'PlayTennis'
print('\n Info-gain for Humidity is: ' + str( information_gain(df_tennis, 'Humidity', 'Play
print('\n Info-gain for Wind is:' + str( information_gain(df_tennis, 'Wind', 'PlayTennis'))
print('\n Info-gain for Temperature is:' + str( information_gain(df_tennis, 'Temperature','
```

```
3         Yes    Rainy        Mild     High    Weak
4         Yes    Rainy        Cool   Normal    Weak
5          No    Rainy        Cool   Normal  Strong
9         Yes    Rainy        Mild   Normal    Weak
13         No    Rainy        Mild     High  Strong
Name:
 Sunny
Group:
    PlayTennis Outlook Temperature Humidity    Wind
0          No   Sunny         Hot     High    Weak
1          No   Sunny         Hot     High  Strong
7          No   Sunny        Mild     High    Weak
8         Yes   Sunny        Cool   Normal    Weak
10        Yes   Sunny        Mild   Normal  Strong
NOBS 14

Classes: Counter({'Yes': 4})

 Number of Instances of the Current Sub Class is 4:
```

In [5]:

```
def id3(df, target_attribute_name, attribute_names, default_class=None):

    ## Tally target attribute:
    from collections import Counter
    cnt = Counter(x for x in df[target_attribute_name])# class of YES /NO

    ## First check: Is this split of the dataset homogeneous?
    if len(cnt) == 1:
        return next(iter(cnt))  # next input data set, or raises StopIteration when EOF is

    ## Second check: Is this split of the dataset empty?
    # if yes, return a default value
    elif df.empty or (not attribute_names):
        return default_class  # Return None for Empty Data Set

    ## Otherwise: This dataset is ready to be devied up!
    else:
        # Get Default Value for next recursive call of this function:
        default_class = max(cnt.keys()) #No of YES and NO Class
        # Compute the Information Gain of the attributes:
        gainz = [information_gain(df, attr, target_attribute_name) for attr in attribute_na
        index_of_max = gainz.index(max(gainz)) # Index of Best Attribute
        # Choose Best Attribute to split on:
        best_attr = attribute_names[index_of_max]

        # Create an empty tree, to be populated in a moment
        tree = {best_attr:{}} # Iniiate the tree with best attribute as a node
        remaining_attribute_names = [i for i in attribute_names if i != best_attr]

        # Split dataset
        # On each split, recursively call this algorithm.
        # populate the empty tree with subtrees, which
        # are the result of the recursive call
        for attr_val, data_subset in df.groupby(best_attr):
            subtree = id3(data_subset,
                          target_attribute_name,
                          remaining_attribute_names,
                          default_class)
            tree[best_attr][attr_val] = subtree
        return tree
```

In [6]:

```
# Get Predictor Names (all but 'class')
attribute_names = list(df_tennis.columns)
print("List of Attributes:", attribute_names)
attribute_names.remove('PlayTennis') #Remove the class attribute
print("Predicting Attributes:", attribute_names)
```

```
List of Attributes: ['PlayTennis', 'Outlook', 'Temperature', 'Humidity', 'Wi
nd']
Predicting Attributes: ['Outlook', 'Temperature', 'Humidity', 'Wind']
```

In [7]:

```
from pprint import pprint
tree = id3(df_tennis,'PlayTennis',attribute_names)
print("\n\nThe Resultant Decision Tree is :\n")
#print(tree)
pprint(tree)
attribute = next(iter(tree))
print("Best Attribute :\n",attribute)
print("Tree Keys:\n",tree[attribute].keys())
```

```
Information Gain Calculation of  Outlook
split: <class 'pandas.core.groupby.generic.DataFrameGroupBy'>
Name:
 Overcast
Group:
    PlayTennis   Outlook Temperature Humidity    Wind
2          Yes  Overcast         Hot     High    Weak
6          Yes  Overcast        Cool   Normal  Strong
11         Yes  Overcast        Mild     High  Strong
12         Yes  Overcast         Hot   Normal    Weak
Name:
 Rainy
Group:
    PlayTennis Outlook Temperature Humidity    Wind
3          Yes   Rainy        Mild     High    Weak
4          Yes   Rainy        Cool   Normal    Weak
5           No   Rainy        Cool   Normal  Strong
9          Yes   Rainy        Mild   Normal    Weak
13          No   Rainy        Mild     High  Strong
```

In [ ]: