1 Subgrouping data in Pandas with groupby

A very powerful feature in Pandas is to use groupby to create groups of data. Each group may then be further acted on as if it were an independent dataframe. This allows for very sophisticated operations broken down by group.

Here we will create a very simple example to illustrate this.

Let's create our usual dataframe:

```
import pandas as pd
df = pd.DataFrame()

names = ['Gandolf',
         'Gimli',
         'Frodo',
         'Legolas',
         'Bilbo',
         'Sam',
         'Pippin',
         'Boromir',
         'Aragorn',
         'Galadriel',
         'Meriadoc']
types = ['Wizard',
         'Dwarf',
         'Hobbit',
         'Elf',
         'Hobbit',
         'Hobbit',
         'Hobbit',
         'Man',
         'Man',
         'Elf',
         'Hobbit']
magic = [10, 1, 4, 6, 4, 2, 0, 0, 2, 9, 0]
aggression = [7, 10, 2, 5, 1, 6, 3, 8, 7, 2, 4]
stealth = [8, 2, 5, 10, 5, 4 ,5, 3, 9, 10, 6]

df['names'] = names
df['type'] = types
df['magic_power'] = magic
df['aggression'] = aggression
df['stealth'] = stealth
```

And now we will create an object 'groups' that allows us to work on those groups individually. We will group just by one parameter (type), but the groups can combine more parameters (each group produced will have identical values for the parameters used for the grouping).

We create an 'itterable' object (an object that can be stepped through using groupby. We can step through members of that new groupby object. Each member has an index and a datatframe of data.

Lets just print out the index and the data for each group.

```
groups = df.groupby('type') # creates a new object of groups of data
for index, group_df in groups: # each group has an index and a dataframe of data
    print ('group index:', index)
    print ('\nData')
    print (group_df)

OUT:
```

```
group index: Dwarf

Data
   names   type  magic_power  aggression  stealth
1  Gimli  Dwarf            1          10        2
group index: Elf

Data
       names type  magic_power  aggression  stealth
3    Legolas  Elf            6           5       10
9  Galadriel  Elf            9           2       10
group index: Hobbit

Data
       names    type  magic_power  aggression  stealth
2      Frodo  Hobbit            4           2        5
4      Bilbo  Hobbit            4           1        5
5        Sam  Hobbit            2           6        4
6     Pippin  Hobbit            0           3        5
10  Meriadoc  Hobbit            0           4        6
group index: Man

Data
     names type  magic_power  aggression  stealth
7  Boromir  Man            0           8        3
8  Aragorn  Man            2           7        9
group index: Wizard

Data
     names    type  magic_power  aggression  stealth
0  Gandolf  Wizard           10           7        8
```

See what happened? We created six smaller dataframes, each of which was one group of the original data. We can then perform any amount of code in each of those sections. As an example, below is some code for a rather crazy method that extracts the third member of the group, but if there are fewer than three members of the group it will take the highest member that it can (the last).

```
col_names = list(df) # get column names from existing dataframe
groups = df.groupby('type') # creates a new object of groups of data
output_df = pd.DataFrame(columns = col_names) # create empty dataframe to store new data

for index, group_df in groups: # each group has an index and a dataframe of data
    number_of_members = len(group_df)
    get_index = min (3, number_of_members)
    get_index -= 1 # subtract 1 to correct for zero-indexing
    retrieved_data = group_df.iloc[get_index].values
    output_df.loc[index] = retrieved_data

print (output_df)

OUT:
            names    type magic_power aggression stealth
Dwarf       Gimli   Dwarf           1         10       2
Elf     Galadriel     Elf           9          2      10
Hobbit        Sam  Hobbit           2          6       4
Man       Aragorn     Man           2          7       9
Wizard    Gandolf  Wizard          10          7       8
```