1 Reshaping Pandas data with stack, unstack, pivot and melt

Sometimes data is best shaped where the data is in the form of a wide table where the description is in a column header, and sometimes it is best shaped as as having the data descriptor as a variable within a tall table.

To begin with you may find it a little confusing what happens to the index field as we switch between different formats. But hang in there and you'll get the hang of it!

Lets look at some examples, beginning as usual with creating a 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
```

When we look at this table, the data descriptors are columns, and the data table is 'wide'.

```
print (df)

OUT:

        names    type  magic_power  aggression  stealth
0     Gandolf  Wizard           10           7        8
1       Gimli   Dwarf            1          10        2
2       Frodo  Hobbit            4           2        5
3     Legolas     Elf            6           5       10
4       Bilbo  Hobbit            4           1        5
5         Sam  Hobbit            2           6        4
```

```
6      Pippin  Hobbit            0           3        5
7     Boromir     Man            0           8        3
8     Aragorn     Man            2           7        9
9   Galadriel     Elf            9           2       10
10   Meriadoc  Hobbit            0           4        6
```

1.1 Stack and unstack

We can convert between the two formats of data with *stack* and *unstack*. To convert from a wide table to a tall and skinny, use *stack*. Notice this creates a more complex index which has two levels the first level is person id, and the second level is the data header. This is called a multi-index.

```
df_stacked = df.stack()
print(df_stacked.head(20)) # pront forst 20 rows

OUT:

0  names          Gandolf
   type            Wizard
   magic_power         10
   aggression           7
   stealth              8
1  names            Gimli
   type             Dwarf
   magic_power          1
   aggression          10
   stealth              2
2  names            Frodo
   type            Hobbit
   magic_power          4
   aggression           2
   stealth              5
3  names          Legolas
   type               Elf
   magic_power          6
   aggression           5
   stealth             10
dtype: object
```

We can convert back to wide table with *unstack*. This recreates a single index for each line of data.

```
df_unstacked = df_stacked.unstack()
print (df_unstacked)

OUT:

        names    type magic_power aggression stealth
0     Gandolf  Wizard          10          7       8
1       Gimli   Dwarf           1         10       2
2       Frodo  Hobbit           4          2       5
3     Legolas     Elf           6          5      10
4       Bilbo  Hobbit           4          1       5
5         Sam  Hobbit           2          6       4
6      Pippin  Hobbit           0          3       5
7     Boromir     Man           0          8       3
8     Aragorn     Man           2          7       9
9   Galadriel     Elf           9          2      10
10   Meriadoc  Hobbit           0          4       6
```

Returning to our stacked data, we can convert our multi-index to two separate fields by resetting the index. By default this method names the separated index field 'level_0' and 'level_1' (multi-level indexes may have further levels as well), and the data field '0'. Let's rename them as well (comment out that row with a # to see what it would look like without renaming them). You can see the effect below:

```
reindexed_stacked_df = df_stacked.reset_index()
reindexed_stacked_df.rename(
    columns={'level_0': 'ID', 'level_1': 'variable', 0:'value'},inplace=True)

print (reindexed_stacked_df.head(20)) # print first 20 rows
```

OUT:

```
    ID     variable    value
0    0        names  Gandolf
1    0         type   Wizard
2    0  magic_power       10
3    0   aggression        7
4    0      stealth        8
5    1        names    Gimli
6    1         type    Dwarf
7    1  magic_power        1
8    1   aggression       10
9    1      stealth        2
10   2        names    Frodo
11   2         type   Hobbit
12   2  magic_power        4
13   2   aggression        2
14   2      stealth        5
15   3        names  Legolas
16   3         type      Elf
17   3  magic_power        6
18   3   aggression        5
19   3      stealth       10
```

We can return to a multi-index, if we want to, by setting the index to the two fields to be combined. Whether a multi-index is preferred or not will depend on what you wish to do wit the dataframe, so it useful to know how to convert back and forth between multi-index and single-index.

```
reindexed_stacked_df.set_index(['ID', 'variable'], inplace=True)
print (reindexed_stacked_df.head(20))
```

OUT:

```
                 value
ID variable
0  names        Gandolf
   type          Wizard
   magic_power       10
   aggression         7
   stealth            8
1  names          Gimli
   type           Dwarf
   magic_power        1
   aggression        10
   stealth            2
2  names          Frodo
   type          Hobbit
   magic_power        4
```

```
   aggression         2
   stealth            5
3  names        Legolas
   type             Elf
   magic_power        6
   aggression         5
   stealth           10
```

1.2 Melt and pivot

melt and *pivot* are like *stack* and *unstack*, but offer some other options.

melt de-pivots data (into a tall skinny table)

pivot will re-pivot data into a wide table.

Let's return to our original dataframe created (which we called 'df') and create a tall skinny table of selected fields using *melt*. We will separate out one or more of the fields, such as 'names' as an ID field, as below:

```
unpivoted = df.melt(id_vars=['names'], value_vars=['type','magic_power'])
print (unpivoted)

OUT:
```

```
        names     variable   value
0     Gandolf         type  Wizard
1       Gimli         type   Dwarf
2       Frodo         type  Hobbit
3     Legolas         type     Elf
4       Bilbo         type  Hobbit
5         Sam         type  Hobbit
6      Pippin         type  Hobbit
7     Boromir         type     Man
8     Aragorn         type     Man
9   Galadriel         type     Elf
10   Meriadoc         type  Hobbit
11    Gandolf  magic_power      10
12      Gimli  magic_power       1
13      Frodo  magic_power       4
14    Legolas  magic_power       6
15      Bilbo  magic_power       4
16        Sam  magic_power       2
17     Pippin  magic_power       0
18    Boromir  magic_power       0
19    Aragorn  magic_power       2
20  Galadriel  magic_power       9
21   Meriadoc  magic_power       0
```

And we can use the *pivot* method to re-pivot the data, defining which field identifies the data to be grouped together, which column contains the new column headers, and which field contains the data.

```
pivoted = unpivoted.pivot(index='names', columns='variable', values='value')
print (pivoted_2)

OUT:
```

```
variable  magic_power    type
names
Aragorn             2     Man
```

```
Bilbo              4  Hobbit
Boromir            0     Man
Frodo              4  Hobbit
Galadriel          9     Elf
Gandolf           10  Wizard
Gimli              1   Dwarf
Legolas            6     Elf
Meriadoc           0  Hobbit
Pippin             0  Hobbit
Sam                2  Hobbit
```