

Data manipulation: Data wrangling, aggregation, and group operations.

**AAA-Python Edition** 



## Plan

- 1- Hierarchical indexing
- 2- Combining and merging Data Sets
- 3- Reshaping and pivoting
- 4- Group by Mechanics
- 5- Data aggregation
- 6- Other aggregation operations



## Data Wrangling and hierarchical indexing

**Data wrangling** is the process of **cleaning** and **unifying messy** and complex data sets for easy access and analysis. (from:

https://www.datawatch.com/what-is-data-wrangling/

Indices of level 1

**Hierarchical indexing**: is the use of **multiple indexes** at different levels

```
ser1
  1 # creating a Series with a hirearchical index
                                                                      i1
    hind = [list("AAABBBCCCD"),["i1","i2","i3"]*3 +["i1"]]
                                                                      i2
  3 ser1 = S(range(10), index= hind)
                 i1, i2, i3 will be
   hind
                  In the level A
                                                                      i3
 ['A', 'A', 'A', 'B', 'B', 'B', 'C', 'C', 'C', 'D'],
 ['i1', 'i2', 'i3', 'i1', 'i2', 'i3', 'i1', 'i2', 'i3', 'i1']]
                                                                      i2
                     level 1
                                           level 2
ser1.index
                                                                   dtype: int64
MultiIndex(levels=\{['A', 'B', 'C', 'D'],
          labels=[{0, 0, 0, 1, 1, 1, 2, 2, 2, 3}, [0, 1, 2, 0, 1, 2, 0, 1, 2, 0]])
```

Indices of level 2





## Reordering and sorting

Reordering enables interchanging the index levels using the swaplevel method

 Sorting enables sorting the data by sorting one level values, using the sort index method.

```
# naming the levels
ser1.index.names=["the_level0","the_level1"]
# rearanging the levels
ser1.swaplevel("the_level0","the_level1")
```

the lev	vel0	the_level1	
Α		i1	0
В		i1	3
С		i1	6
D		i1	9
Α		i2	1
В		i2	4
C		i2	7
Α		i3	2
В		i3	5
C		i3	\ 8∕
dtype:	int64	1	

The hierarchy of the indexes changed

The hierarchy of the Indexes didn't change

The order of the data (so the indexes too) changed: **the\_level1** index was **sorted** 

i1 B 3
i2 B 4
i3 B 5
i1 C 6
i2 C 7
i3 C 8
i1 D 9
dtype: int64

The order of the data (so

the level1

i2

the level0

The order of the data (so the indexes too) remains the same

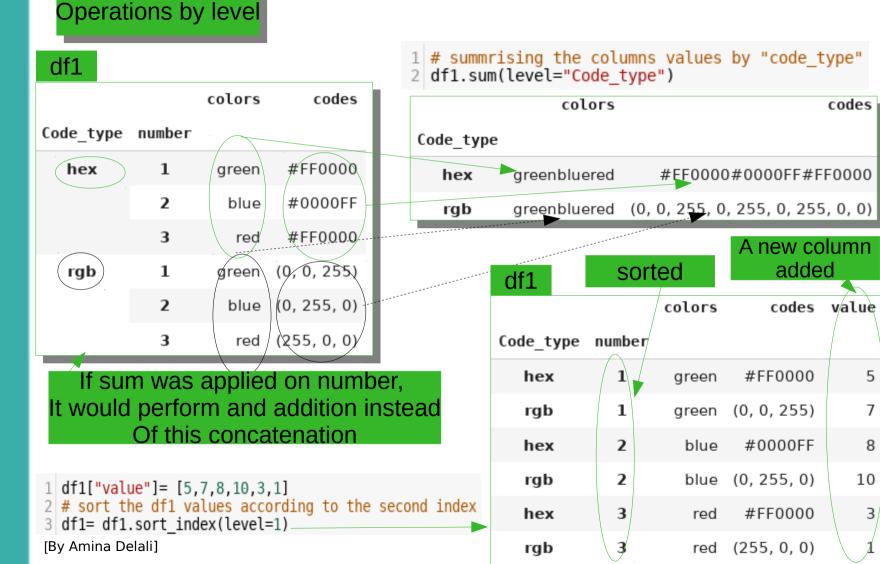
[By Amina Delali]

# sorting the values following the second level : level=1
ser1.sort index(level=1)

0



## Hierarchical indexing



8



## 1- Hierarchical indexing

## indexing

1 # creating a new DataFrame using df1 columns
2 df2 =df1.set index(["colors","codes"])

### df2 value colors codes #FF0000 green (0, 0, 255)blue #0000FF 8 (0, 255, 0)10 #FF0000 3 red (255, 0, 0)

The previous df1 columns are now indexes

The **indexes** are converted into **columns** 

0	green	#FF0000	5
1	green	(0, 0, 255)	10
2	blue	#0000FF	7
3	blue	(0, 255, 0)	3
4	red	#FF0000	8
5	red	(255, 0, 0)	1

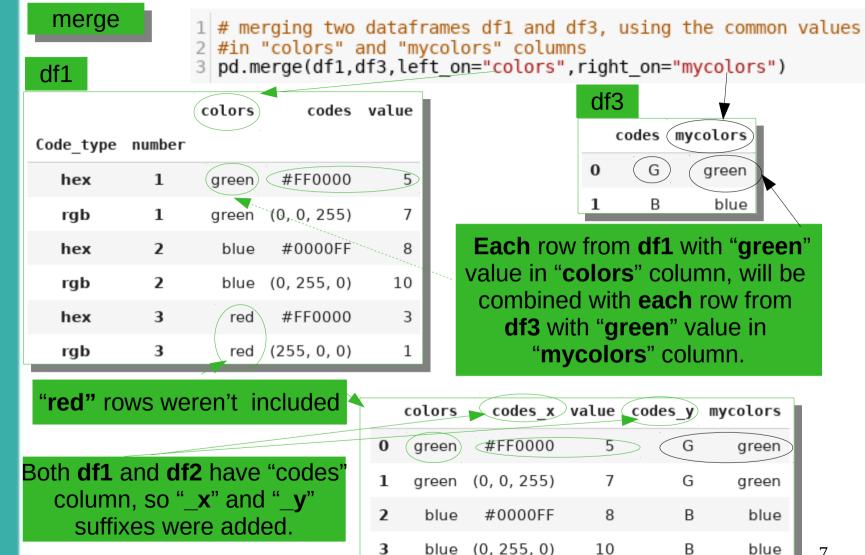
codes

value

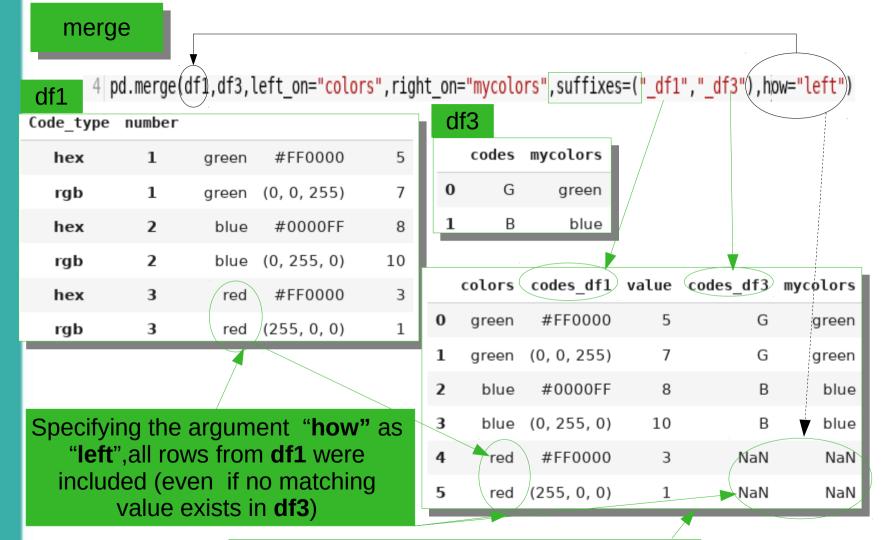
colors

- 1 # the indexes are converted into columns
- 2 df2.reset\_index()



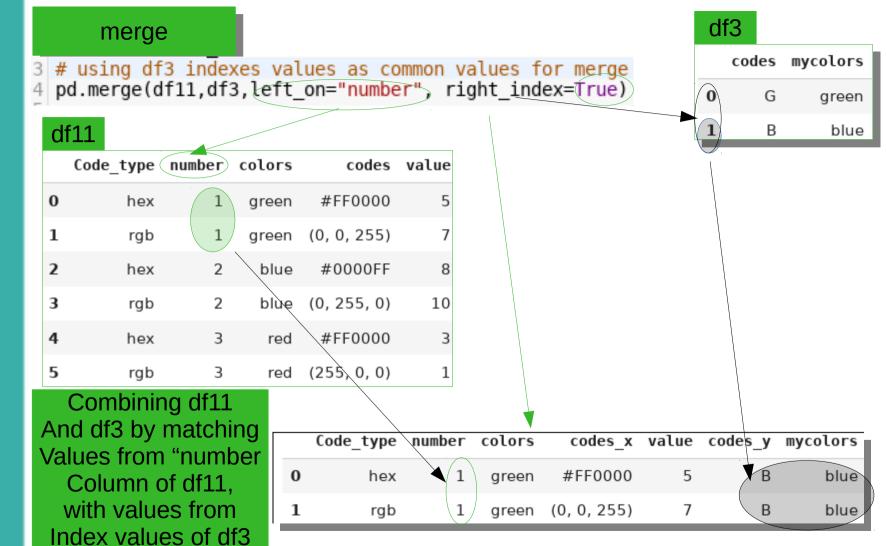




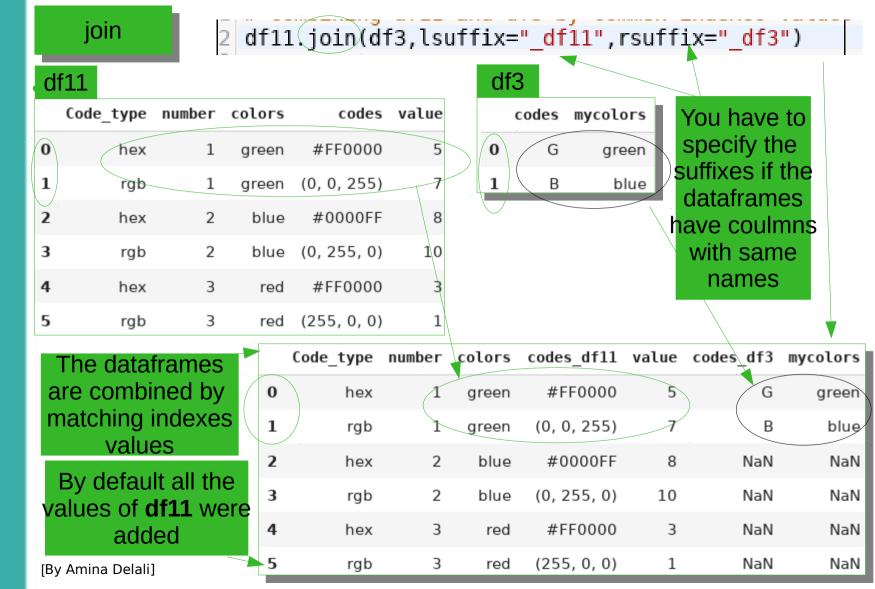


The suffixes argument used to customize the suffixes added to columns with the same name

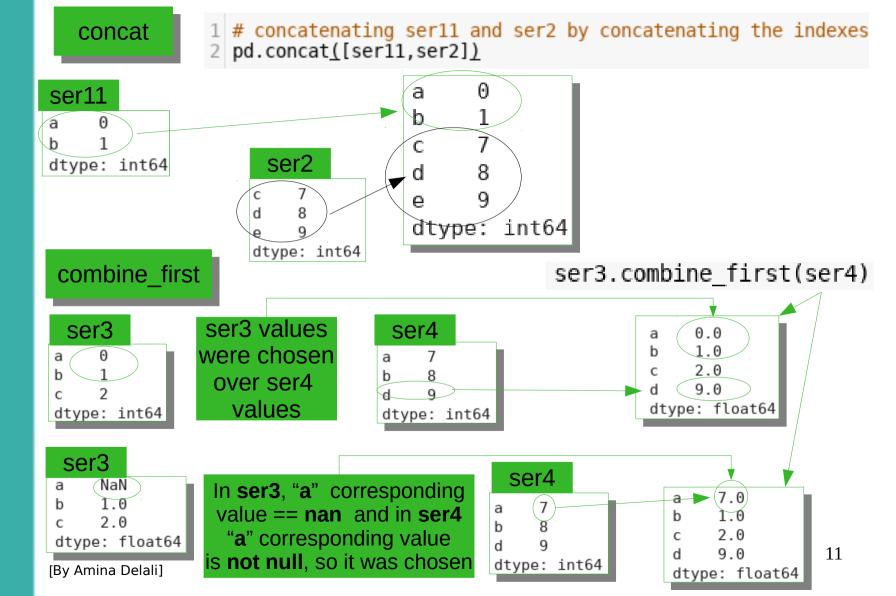








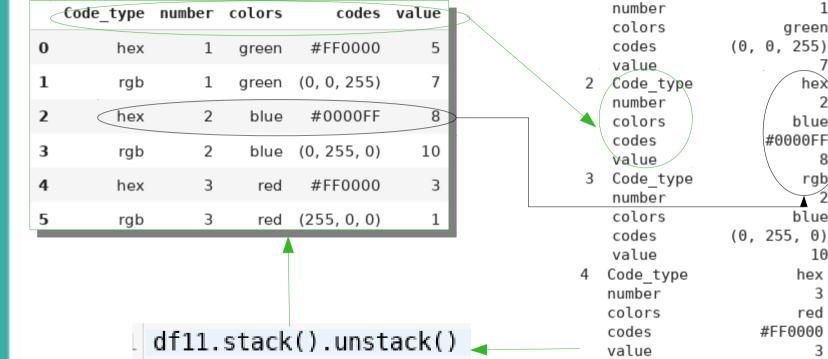
## 3- Reshaping and pivoting





## and Reshaping ivoting 0

- stack & unstack df11.stack()
- **stack**: pivot **columns** label to **rows** indexes
- unstack: pivot rows indexes to columns labels 1



Code type

Code type

Code type

number

colors

codes

value

dtvpe: object

number colors

codes

value

hex

rgb

areen

hex

blue #0000FF

rgb

blue

hex

red

rgb

red

#FF0000

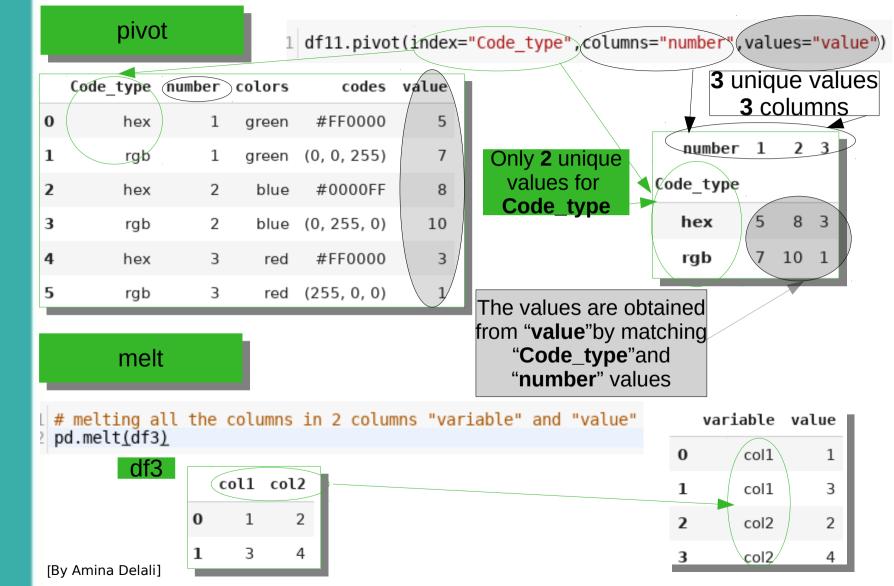
(255, 0,

10

green

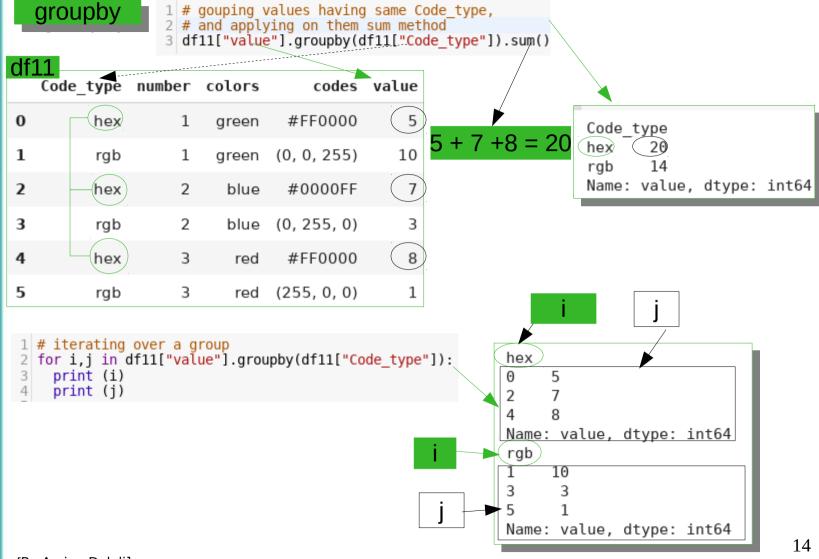
#FF0000

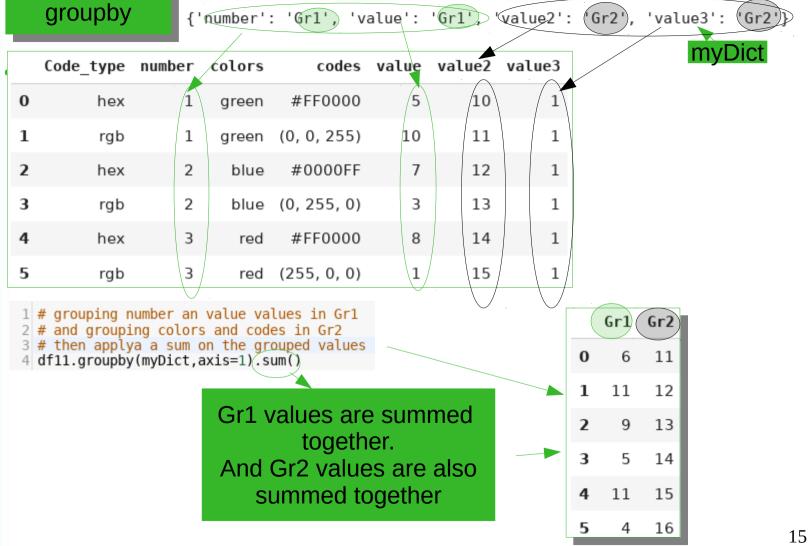
## 3- Reshaping and pivoting





# Group









We could just write: groupby("Code\_type")

## Columns kept columns

my\_group2=df11.groupby(df11["Code\_type"],as\_index=False)
2 # "Code\_type" is no longer an index, but a column
3 my\_group2.mean()

		NE	Remains a column					
Code_type numb			value	value2	value3			
0	hex	2.0	6.666667	12.0	1.0			
1	rab	2.0	4.666667	13.0	1.0			

Domaine a column

16



## aggregation erations Other

	Code_type	number	colors	codes	value	value2	value3
0	hex	1	green	#FF0000	5	10	2
1	rgb	1	green	(0, 0, 255)	10	11	31
2	hex	2	blue	#0000FF	7	12	62
3	rgb	2	blue	(0, 255, 0)	3	13	156
4	hex	3	red	#FF0000	8	14	230
5	rgb	3	red	(255, 0, 0)	1	15	1000

The data values in "value3" column are grouped by intervals created by **cut**(same length) and **qcut** (same size)

### cut & qcut

```
# grouping value3 by intervals of the same lenght
```

intervals=pd(cut)(df11.value3,3)

my\_group4= df11["value3"].groupby(intervals)

4 my group4.count()

We see that the number values in each interval is

different from the others

1 # grouping value3 by intervals of the same size

2 # same number of values in each interval intervals=pd(qcut(df11.value3,3)

my group5= df11["value3"].groupby(intervals)

5 my group5.count()

We see that the number values in each interval are all the same == 2

value3 (1.002, 334.667] (334.667, 667.333] (667.333, 1000.0] Name: value3, dtype: int64

> value3 (1.999, 51.667] (51.667, 180.667] (180.667, 1000.0]

Name: value3, dtype: int64





### crosstab

```
# for each value in column "number",
# crosstab will calculate the frequnecies
# of each unique value in "colors"

pd.crosstab(df11.number,df11.colors)
```

 colors
 blue
 green
 red

 number
 0
 2
 0

 2
 2
 0
 0

 3
 0
 0
 2

For "number" value **==1**, corresponds: **2** values **== green** and **0** value in **blue** and **red** in "colors"

df	11						
	Code_type	number	colors	codes	value	value2	value3
0	hex	1	green	#FF0000	5	10	2
1	rgb	1	green	(0, 0, 255)	10	11	31
2	hex	2	blue	#0000FF	7	12	62
3	rgb	2	blue	(0, 255, 0)	3	13	156
4	hex	3	red	#FF0000	8	14	230
5	rgb	3	red	(255, 0, 0)	1	15	1000



## References

- Datawatch. What is data wrangling? On-line at https://www.datawatch.com/what-is-data-wrangling/. Accessed on 31-10-2018.
- Wes McKinney. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. O'Reilly Media, Inc, 2018.
- pydata.org. Pandas documentation. On-line at https://pandas.pydata.org/. Accessed on 19-10-2018.



## Thank you!

FOR ALL YOUR TIME