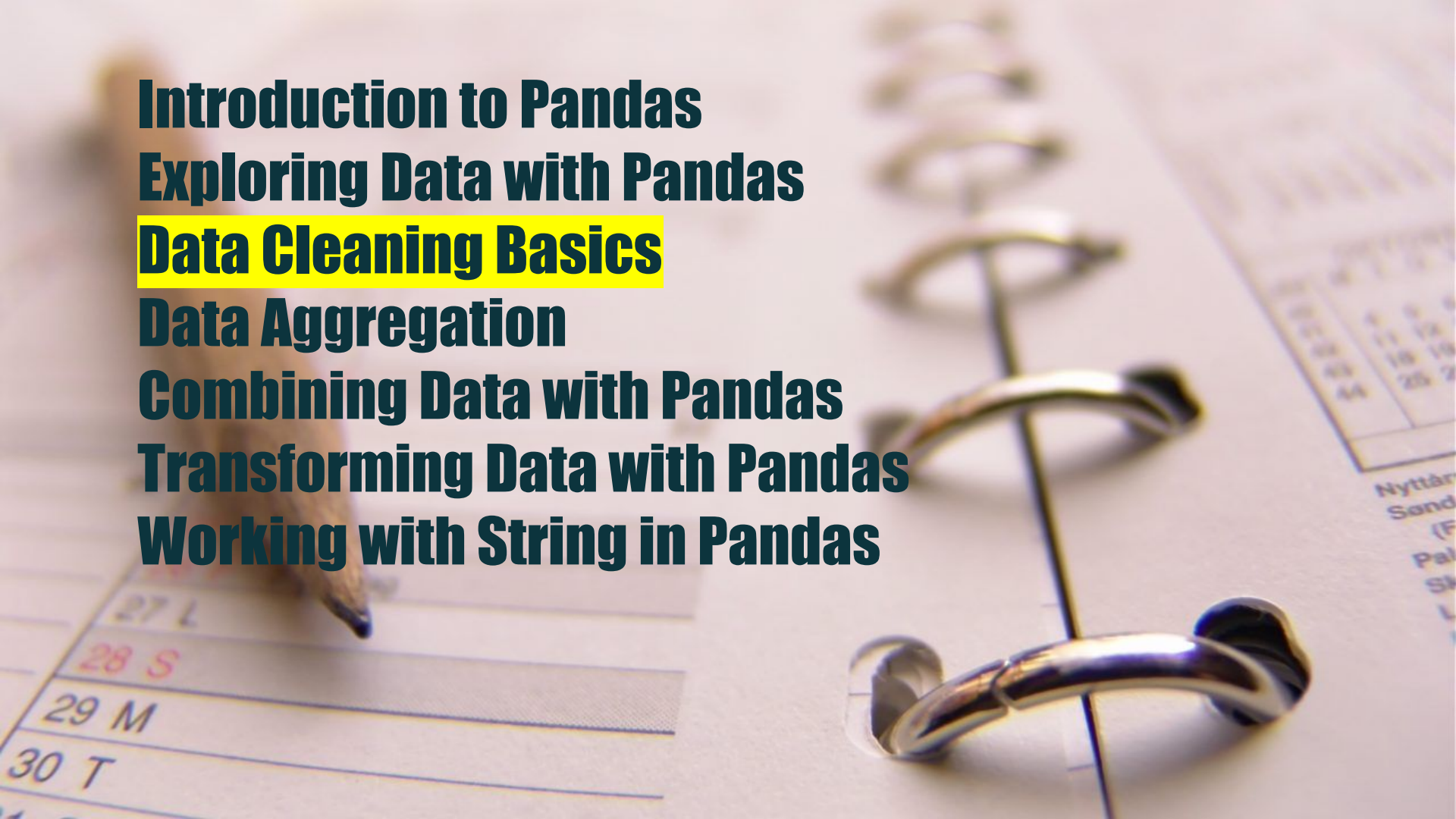
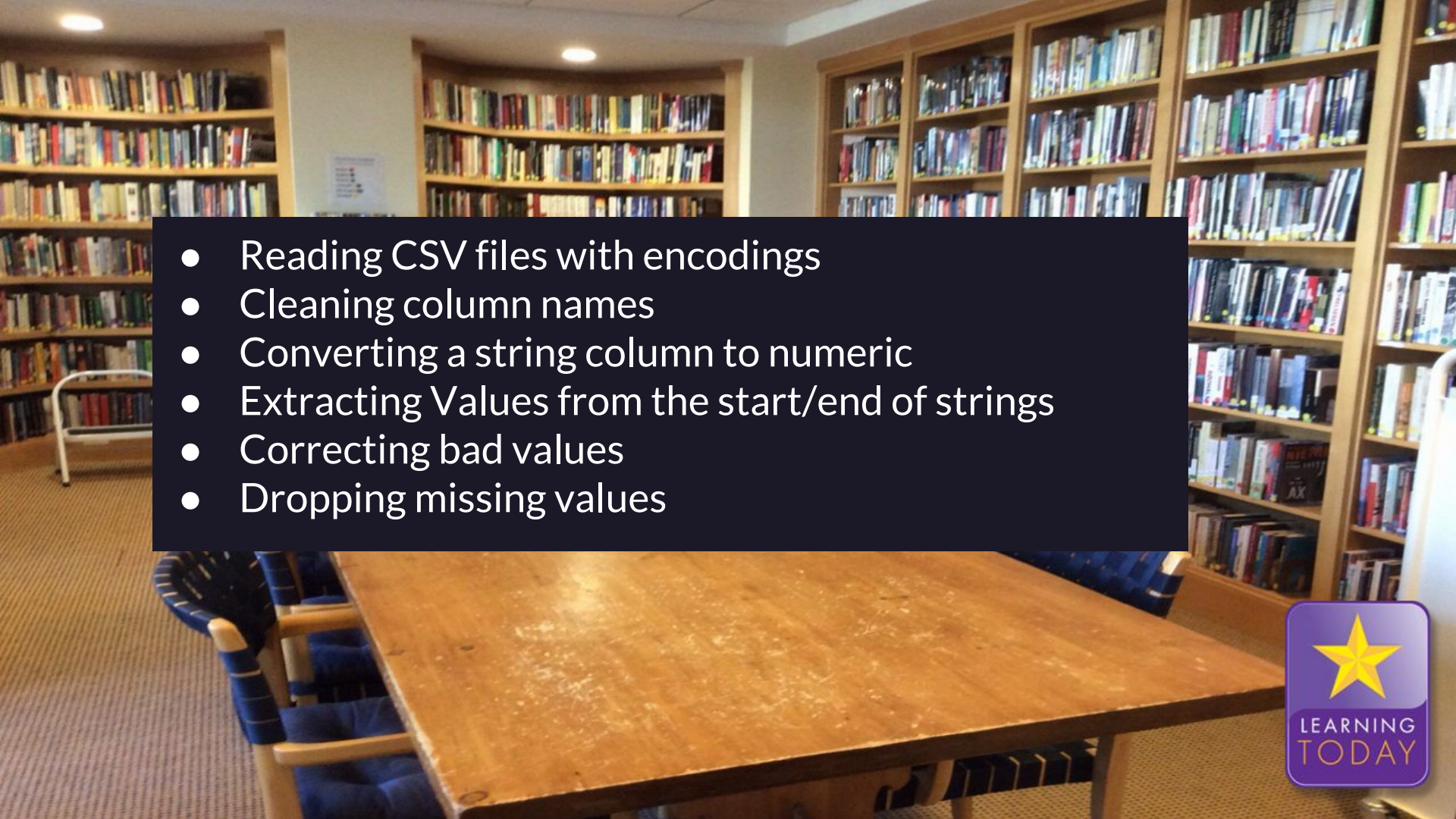
Lesson #05
Data Cleaning Basic

March 2019


Introduction to Pandas
Exploring Data with Pandas
Data Cleaning Basics
Data Aggregation
Combining Data with Pandas
Transforming Data with Pandas
Working with String in Pandas
27 L
28 S
29 M
30 T



- Reading CSV files with encodings
- Cleaning column names
- Converting a string column to numeric
- Extracting Values from the start/end of strings
- Correcting bad values
- Dropping missing values

Update from repository

```
git clone https://github.com/ivanovitchm/datascience_one_2019_1
```

Or ....

```
git pull
```


Dataset - Laptops computers

Dataset - Laptops computers

	Manufacturer	Model Name	Category	Screen Size	Screen	CPU	RAM	Storage	GPU	Operating System	Operating System Version	Weight	Price (Euros)
0	Apple	MacBook Pro	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS	NaN	1.37kg	1339,69
1	Apple	Macbook Air	Ultrabook	13.3"	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS	NaN	1.34kg	898,94
2	HP	250 G6	Notebook	15.6"	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS	NaN	1.86kg	575,00
3	Apple	MacBook Pro	Ultrabook	15.4"	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS	NaN	1.83kg	2537,45
4	Apple	MacBook Pro	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS	NaN	1.37kg	1803,60

Reading CSV files with encodings

Character representation of binary in three encodings

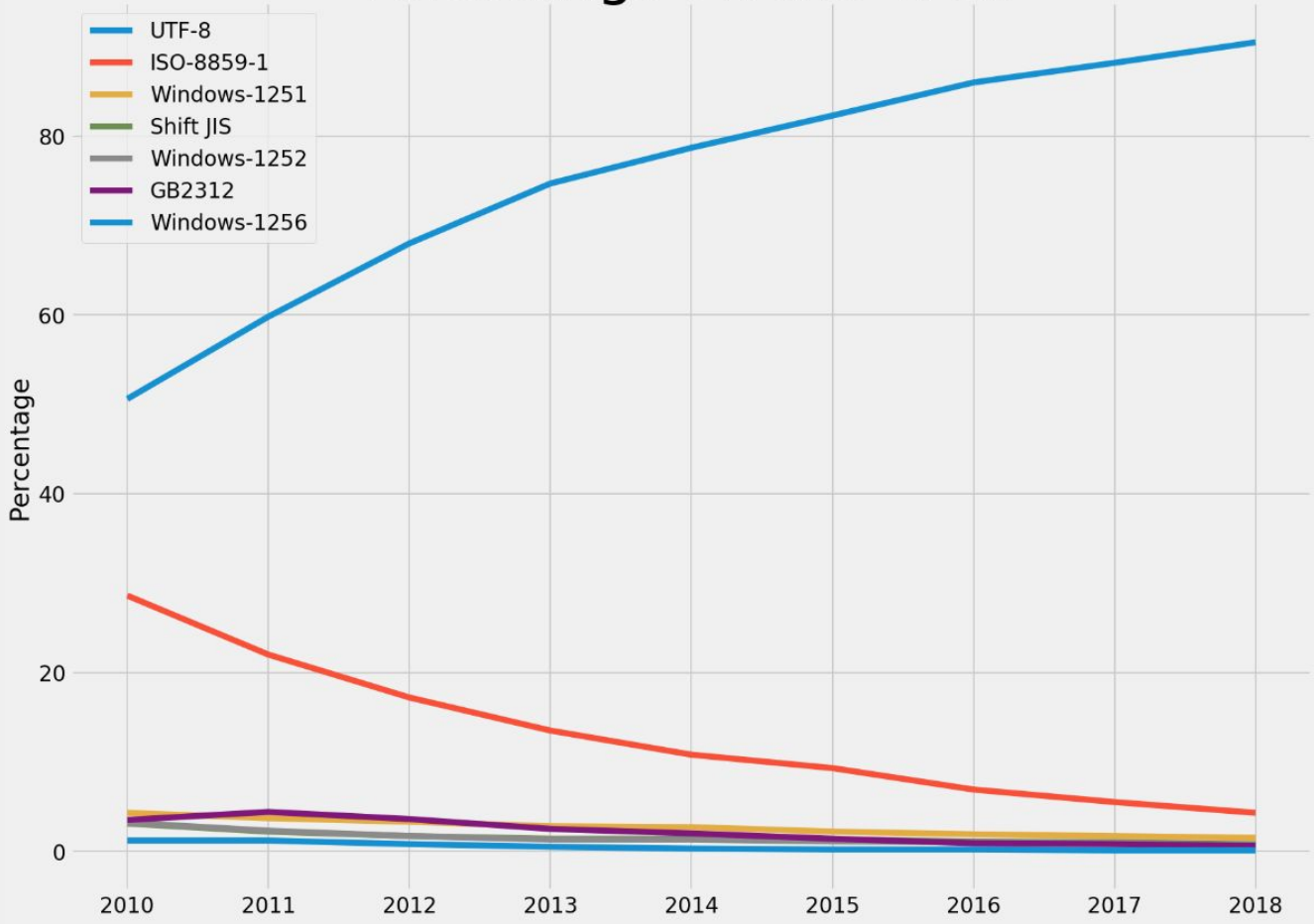
Binary Representation	*Encoding*	*Characters*
`11000100 01000010`	Latin-1	ÄB
`11000100 01000010`	Mac Roman	ƒB
`11000100 01000010`	GB18030	腂

Binary representation of characters in three encodings

Characters	*Encoding*	*Binary Representation*
Føö	Latin-1	`01000110 11111000 11110110`
Føö	Mac Roman	`01000110 10111111 10011010`
Føö	UTF-8	`01000110 11000011 10111000 11000011 10110110`

Source: David C. Zentgraf

Historical Usage of Encodings on the Web



```
import pandas as pd

laptops = pd.read_csv("laptops.csv")
laptops.head()
```

```
        laptops = pd.read_csv("laptops.csv",encoding="Latin-1")
```

```
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_tokens()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_with_dtype()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._string_convert()

pandas/_libs/parsers.pyx in pandas._libs.parsers._string_box_utf8()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe9 in position 4: invalid continuation byte
```

Cleaning column names

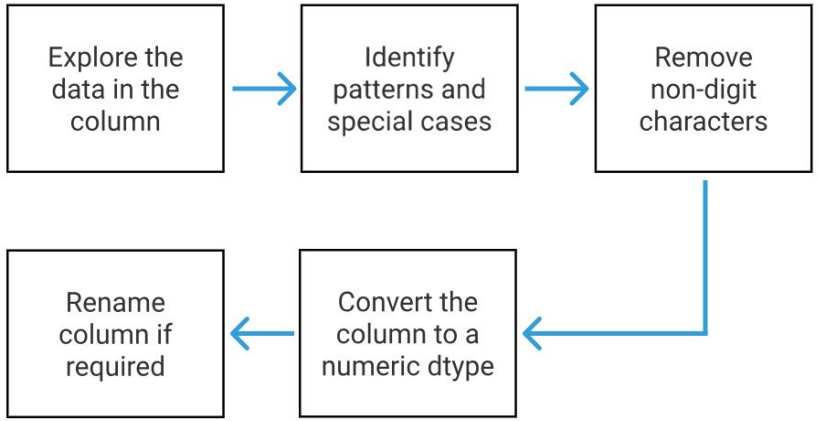
```
['Manufacturer',
 'Model Name',
 'Category',
 'Screen Size',
 'Screen',
 'CPU',
 'RAM',
 ' Storage',
 'GPU',
 'Operating System',
 'Operating System Version',
 'Weight',
 'Price (Euros)']
```

```
def clean_col(col):
    col = col.strip()
    col = col.replace("(","")
    col = col.replace(")","")
    col = col.lower()
    return col

laptops.columns = [clean_col(c) for c in laptops.columns]
laptops.columns.tolist()
```

Converting a string column to numeric

	category	screen_size	screen
0	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600
1	Ultrabook	13.3"	1440x900
2	Notebook	15.6"	Full HD 1920x1080
3	Ultrabook	15.4"	IPS Panel Retina Display 2880x1800
4	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600



Converting a string column to numeric

```
1 laptops["screen_size"] = laptops["screen_size"].str.replace('"','')
2 laptops["screen_size"].unique()
```

```
array(['13.3', '15.6', '15.4', '14.0', '12.0', '11.6', '17.3', '10.1',
       '13.5', '12.5', '13.0', '18.4', '13.9', '12.3', '17.0', '15.0',
       '14.1', '11.3'], dtype=object)
```

```
1 laptops["screen_size"] = laptops["screen_size"].astype(float)
2 print(laptops["screen_size"].dtype)
3 laptops["screen_size"].unique()
```

```
float64
array([13.3, 15.6, 15.4, 14. , 12. , 11.6, 17.3, 10.1, 13.5, 12.5, 13. ,
       18.4, 13.9, 12.3, 17. , 15. , 14.1, 11.3])
```

#tip

```
1
2 laptops["weight"] = (laptops["weight"]
3                       .str.replace("kg","")
4                       .astype(float)
5                      )
```

Extracting Values from the start of strings

```
laptops["gpu"].head()
```

0	Intel Iris Plus Graphics 640
1	Intel HD Graphics 6000
2	Intel HD Graphics 620
3	AMD Radeon Pro 455
4	Intel Iris Plus Graphics 650

```
(laptops["gpu"]
    .head()
    .str.split()
)
```

0	[Intel, Iris, Plus, Graphics, 640]
1	[Intel, HD, Graphics, 6000]
2	[Intel, HD, Graphics, 620]
3	[AMD, Radeon, Pro, 455]
4	[Intel, Iris, Plus, Graphics, 650]

Extracting Values from the start of strings

```
(laptops["gpu"]
    .head()
    .str.split(n=1)
)
```

0	[Intel, Iris Plus Graphics 640]
1	[Intel, HD Graphics 6000]
2	[Intel, HD Graphics 620]
3	[AMD, Radeon Pro 455]
4	[Intel, Iris Plus Graphics 650]

```
(laptops["gpu"]
    .head()
    .str.split(n=1, expand=True)
)
```

	0	1
0	Intel	Iris Plus Graphics 640
1	Intel	HD Graphics 6000
2	Intel	HD Graphics 620
3	AMD	Radeon Pro 455
4	Intel	Plus Graphics 650

Extracting Values from the end of strings

```
1 print(laptops["screen"].unique().shape)
2 print(laptops["screen"].unique()[:10])
```

```
(40,)
['IPS Panel Retina Display 2560x1600' '1440x900' 'Full HD 1920x1080'
 'IPS Panel Retina Display 2880x1800' '1366x768'
 'IPS Panel Full HD 1920x1080' 'IPS Panel Retina Display 2304x1440'
 'IPS Panel Full HD / Touchscreen 1920x1080'
 'Full HD / Touchscreen 1920x1080' 'Touchscreen / Quad HD+ 3200x1800']
```

Extracting Values from the end of strings

```
sentences
```

0	Joe's favorite color is orange
1	Lisa's umbrella is purple
2	Rashid's new shirt is blue
3	Joanne's new puppy is black
4	Carrie's soccer team wears red

```
sentenes.str.rsplit(n=1,expand=True)
```

	0	1
0	Joe's favorite color is	orange
1	Lisa's umbrella is	purple
2	Rashid's new shirt is	blue
3	Joanne's new puppy is	black
4	Carrie's soccer team wears	red

Extracting Values from the end of strings

```
laptops.loc[:9, "screen"].str.rsplit(n=1,expand=True)
```

	0	1
0	IPS Panel Retina Display	2560x1600
1	1440x900	None
2	Full HD	1920x1080
3	IPS Panel Retina Display	2880x1800
4	IPS Panel Retina Display	2560x1600
5	1366x768	None
6	IPS Panel Retina Display	2880x1800
7	1440x900	None
8	Full HD	1920x1080
9	IPS Panel Full HD	1920x1080

```
screen_res = laptops["screen"].str.rsplit(n=1, expand=True)

# giving the columns string labels makes them easier to work with
screen_res.columns = ["A", "B"]

# for rows where the value of column "B" is null, fill in the
# value found in column "A" for that row
screen_res.loc[screen_res["B"].isnull(), "B"] = screen_res["A"]

laptops["screen_resolution"] = screen_res["B"]
```

Correcting bad values

```
1 s = pd.Series(["pair","oranje","bananna","oranje","oranje","oranje"])
```

```
1 corrections = {
2     "pair": "pear",
3     "oranje": "orange",
4     "bananna": "banana"
5 }
6
7 s = s.map(corrections)
8 print(s)
```

```
0      pear
1    orange
2    banana
3    orange
4    orange
5    orange
dtype: object
```

re-run

```
s.map(corrections)
```

```
0    NaN
1    NaN
2    NaN
3    NaN
4    NaN
5    NaN
dtype: object
```

Dropping missing values

```
print(df.dropna())
```

	A	B	C	D
w	6.0	3.0	7.0	4.0
y	4.0	3.0	7.0	7.0

	A	B	C	D
w	6.0	3.0	7.0	4.0
x	6.0	2.0	NaN	7.0
y	4.0	3.0	7.0	7.0
z	2.0	5.0	NaN	1.0

```
print(df.dropna(axis=1))
```

	A	B	D
w	6.0	3.0	4.0
x	6.0	2.0	7.0
y	4.0	3.0	7.0
z	2.0	5.0	1.0

Challenge: extracting storage information

```
1 laptops.loc[76:81, "storage"]
```

```
76                  2TB HDD
77    128GB SSD +   1TB HDD
78                  1TB HDD
79    128GB SSD +   1TB HDD
80                256GB SSD
81                512GB SSD
Name: storage, dtype: object
```

	storage_1_capacity_gb	storage_1_type	storage_2_capacity_gb	storage_2_type
76	2000.0	HDD	NaN	None
77	128.0	SSD	1000.0	HDD
78	1000.0	HDD	NaN	None
79	128.0	SSD	1000.0	HDD
80	256.0	SSD	NaN	None
81	512.0	SSD	NaN	None

Lesson_05 Data Cleaning Basics