Lesson #06
Data Cleaning Basic

March 2020

1. Given a sorted array of integers, how can you find the location of a particular integer x?
2. How can you quickly computer 2^x ?
3. What is the value of 15^18^15 ?

KANTAR
Information is Beautiful
Awards 2019

About News Awards Entry Showcase Sponsor

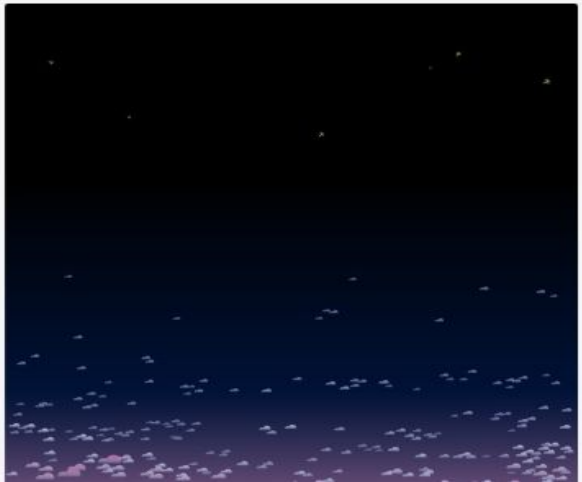
Awards Challenges

http://bit.do/datascienceviz

All 2019 2018 2017 2016 2015 2014 2013 2012

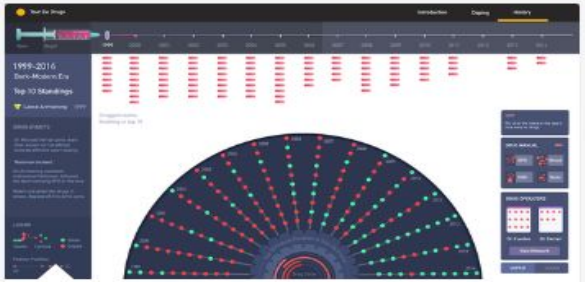
All ★ Winners Silver Gold Shortlist Longlist Bronze Rising Star Impressive Individual Best-Non-English-Language Outstanding Outfit Student Community Most Beautiful

All Arts, Entertainment & Culture Humanitarian Leisure, Games & Sport Maps, Places & Spaces News & Current Affairs People, Language & Identity Politics & Global Visualization & Information Design Science & Technology Unusual



Super Kamiokande

This stunning piece of digital journalism is the result of a successful collaboration between the ABC's Tokyo bureau and the Story Lab team here in Australia. Super Kamiokande is a giant science...



Tour De Drugs - Data Narrative

The Tour de France is an annual men's multiple stage bicycle race primarily held in France. Due to the intense nature of the sport, doping has been long associated with the race. While doping in...

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The **Pandas** library is built on NumPy and provides easy-to-use **data structures** and **data analysis** tools for the Python programming language.

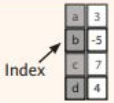
Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

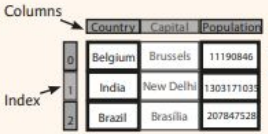
Series

A **one-dimensional** labeled array capable of holding any data type




```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame



A **two-dimensional** labeled data structure with columns of potentially different types

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

```
>>> s['b']
-5
```
Get one element

```
>>> df[1:]
  Country     Capital  Population
1   India  New Delhi  1303171035
2  Brazil   Brasilia   207847528
```
Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

```
>>> df.iloc[[0],[0]]
'Belgium'
```
Select single value by row & column

```
>>> df.iat([0],[0])
'Belgium'
```

By Label

```
>>> df.loc[[0], ['Country']]
'Belgium'
```
Select single value by row & column labels

```
>>> df.at([0], ['Country'])
'Belgium'
```

By Label/Position

```
>>> df.ix[2]
Country      Brazil
Capital    Brasilia
Population 207847528
```
Select single row of subset of rows

```
>>> df.ix[:,'Capital']
0     Brussels
1    New Delhi
2     Brasilia
```
Select a single column of subset of columns

```
>>> df.ix[1,'Capital']
'New Delhi'
```
Select rows and columns

Boolean Indexing

```
>>> s[~(s > 1)]
```
Series s where value is not >1

```
>>> s[(s < -1) | (s > 2)]
```
s where value is <-1 or >2

```
>>> df[df['Population']>1200000000]
```
Use filter to adjust DataFrame

Setting

```
>>> s['a'] = 6
```
Set index a of Series s to 6

Dropping

```
>>> s.drop(['a', 'c'])
```
Drop values from rows (axis=0)

```
>>> df.drop('Country', axis=1)
```
Drop values from columns(axis=1)

Sort & Rank

```
>>> df.sort_index()
```
Sort by labels along an axis

```
>>> df.sort_values(by='Country')
```
Sort by the values along an axis

```
>>> df.rank()
```
Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

```
>>> df.shape
```
(rows,columns)

```
>>> df.index
```
Describe index

```
>>> df.columns
```
Describe DataFrame columns

```
>>> df.info()
```
Info on DataFrame

```
>>> df.count()
```
Number of non-NA values

Summary

```
>>> df.sum()
```
Sum of values

```
>>> df.cumsum()
```
Cummulative sum of values

```
>>> df.min()/df.max()
```
Minimum/maximum values

```
>>> df.idxmin()/df.idxmax()
```
Minimum/Maximum index value

```
>>> df.describe()
```
Summary statistics

```
>>> df.mean()
```
Mean of values

```
>>> df.median()
```
Median of values

Applying Functions

```
>>> f = lambda x: x*2
>>> df.apply(f)
```
Apply function

```
>>> df.applymap(f)
```
Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
a    10.0
b     NaN
c     5.0
d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
a    10.0
b    -5.0
c     5.0
d     7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> df.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> df.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

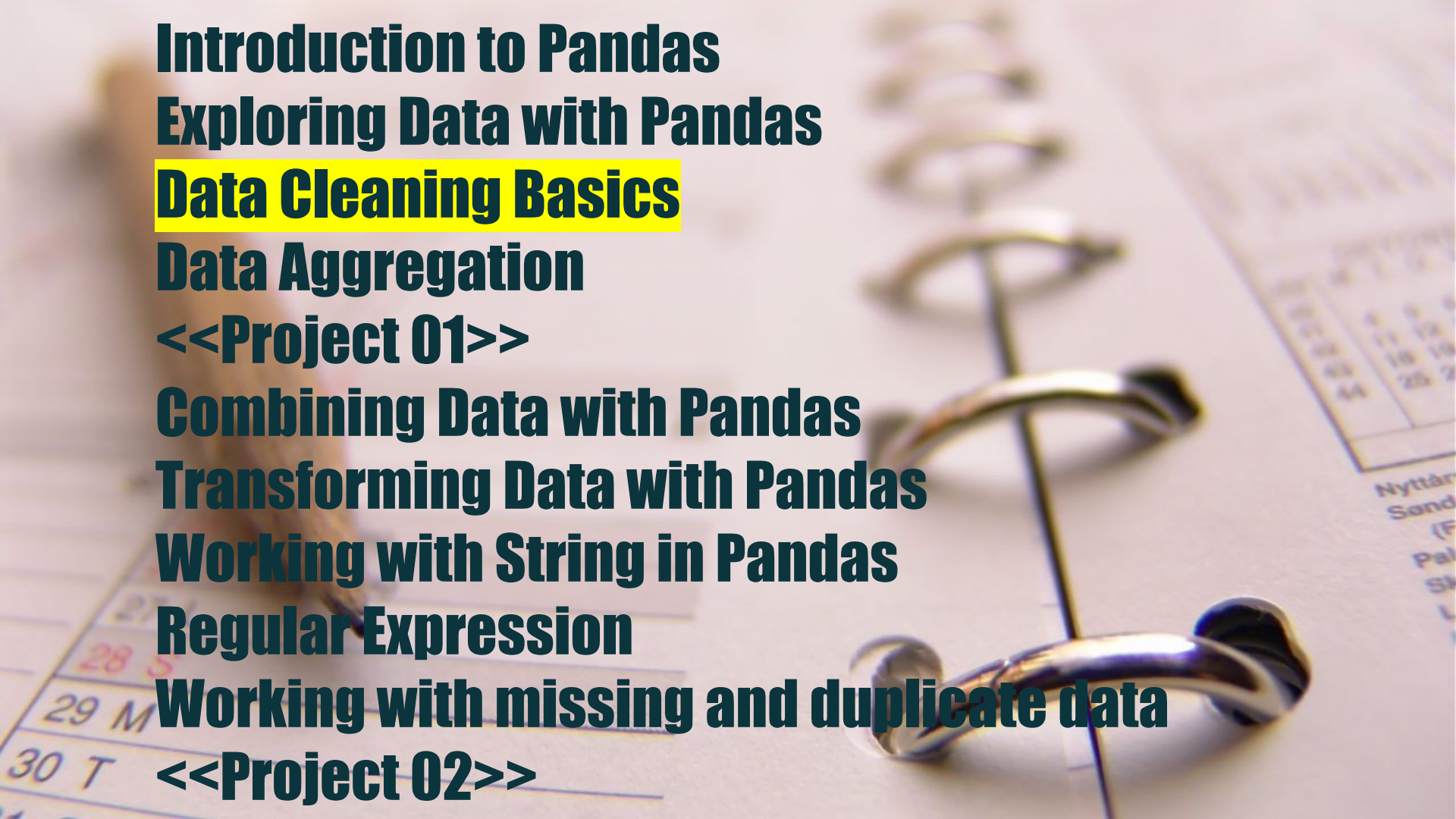
```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

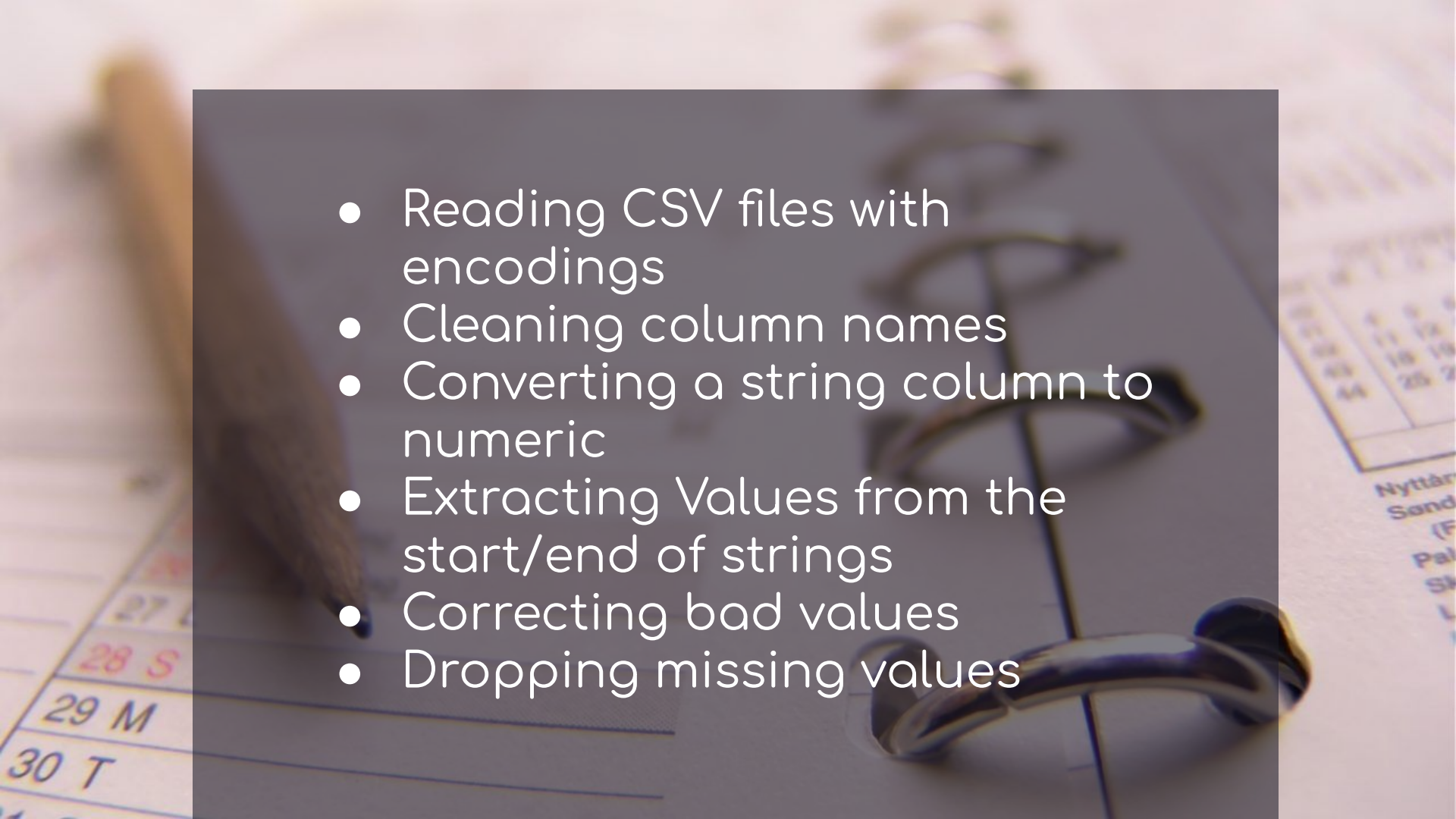
Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> df.to_sql('myDf', engine)
```


Introduction to Pandas
Exploring Data with Pandas
Data Cleaning Basics
Data Aggregation
<<Project 01>>
Combining Data with Pandas
Transforming Data with Pandas
Working with String in Pandas
Regular Expression
Working with missing and duplicate data
<<Project 02>>



- 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/datascience2020
```

Or ....

```
git pull
```


Dataset - Laptops computers

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`

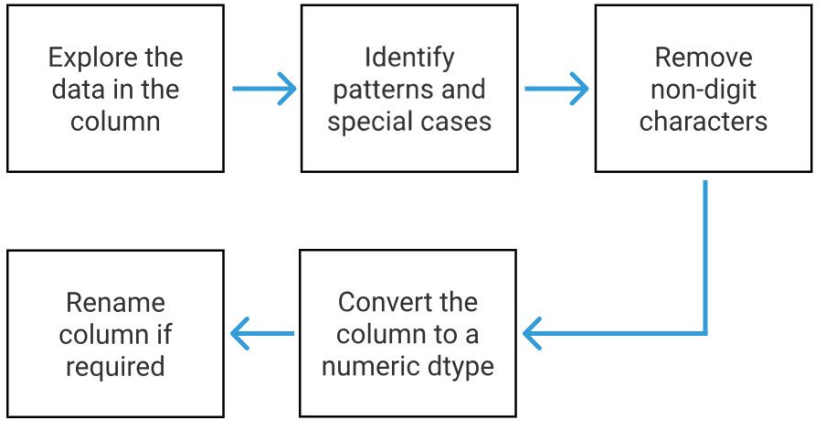
Source: David C. Zentgraf

Cleaning column names

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

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



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

```
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

```
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

	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

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

Lecture 06 Data Cleaning Basics