

# Data Tidying and Cleaning

Preparing Data for Knowledge Extraction



**Yordan Darakchiev**  
Technical Trainer



**SoftUni**



**Software University**

<https://softuni.bg>

# Have a Question?

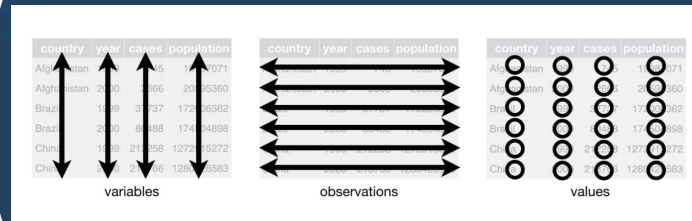
[sli.do](https://sli.do)

**#DataScience**

# Table of Contents

1. Messy and Tidy Data
  - Tidying up Messy Data
2. Operations on Datasets
3. Cleaning Data
  - Validation
  - Transformation
  - Error Correction
  - Features
4. Data Tidying and Cleaning as a Process





# Data Tidying

Arranging Data in a Meaningful Manner

- Most important rules when creating (or using) datasets
  - Columns – attributes (features, variables)
  - Rows – observations
  - Cells – values (one observation of one feature)
  - All other data is called **messy data**
- Empirical rule for testing whether a dataset is tidy
  - Adding one more observation should create one new row
    - No new columns
    - No multiple rows
    - No partial rows, no changes to other rows
- pandas allows us to read, tidy up and transform datasets
  - Data modelling requires a tidy and clean dataset in order to work well (garbage in – garbage out)

## What we want

country	year	cases	population
Afghanistan	2000	266	20195360
Brazil	1999	31737	17206362
Brazil	2000	80488	17404898
China	1999	212258	1272115272
China	2000	212256	128025853

variables

observations

values

## What we get instead

	A	B	C	D	E	F	G	H	I	J
1	Company I	Website U	First Name	Last Name	Email	Street	Add City	Postal / Zip	State / Prc	Country
2	Vandelay I	http://van	George	Costanza	g.costanza	3661 Hinkl	Binghamtc	13901	New York	USA
3	Thatherto	http://king	Heather	Hall	heather.h	1683 Jasper	Edmonton	T5J 3N6	Alberta	Canada
4	Western E	http://ww	Erich	Thomson	eric.t@we	745 Stroog	Atlanta	30305	Georgia	USA
5	Mammoth	http://tiny	Michael	Blake	michael.bl	4984 Clair	Morgan	76671	Texas	USA
6	Powell Mc	http://ww	Harbert	Powell	harbert@	1125 Cros	Michigan	48607	Detroit	USA
7	Urban Son	http://ww	Garcia Est	Orosco	garcia@ur	16221 E 4C	Denver	80239	Colorado	USA
8	Wayne Enl	http://way	Bruce	Wayne	b.wayne@	3881 Mich	Chicago	60631	Illinois	USA
9	Kramerica	http://ww	Cosmo	Kramer	ckramer@	22 Marion	New York	66059	New York	America
10	Bluth Com	http://thel	Bluth	George	george@b	36 Southw	Fairfield	60042	New Jerse	USA
11	Flowers B	http://ww	Elliott	Irene	irene@ire	2197 Tatol	Buffalo Gr	60089	Illinois	USA
12	Spacely Sp	http://spa	Spacely	Cosmo	cosmo@sj	89 Davids	New York	66412	New York	USA
13	Dunder Mi	http://ww	Dudner	Bob	bob@dunc	25 Spring	C Waterloo	N3E 0R4	Ontario	Canada
14	Sixty Seco	http://ww	Brown	Darren	darren@si	35 Brentw	San Franci	94210	California	USA
15	Charles To	http://ww	Angel	Charles	c.angel@t	20 Dora	Cr The Bay	94212	California	USA
16	Spade and	http://ww	Archer	Christoph	c.a@spade	68 Cambri	Waterloo	N2N 2J8	Ontario	Canada
17	Atlantic N	http://ww	Thomas	Won	won@atla	94 Walter	Vancouver	K2C 4CJ	BC	Canada
18	Tessier-As	http://ww	Tessier	Marie	marie@te	2 Bourke	C Melbourne	3418	Australia	
19	Southern C	http://ww	Van Houte	Kirk	kirk@allie	67 Gadd A	Springfield	63012	Illinois	USA
20	Initech	http://ww	Charles	Deborah	d.charles@	35 Quintin	Waterloo	N3E 0R4	Ontario	Canada
21	McMahon	http://ww	Tate	Charles	tate@man	87 Wright	Palo Alto	94304	California	USA
22	Widget Inc	http://ww	Wilbert	Gregg	gregg@wik	3094 Boul	Quebec	G1R 1B8	Quebec	Canada
23	Acme Corp	http://ww	Harris	Mike	mike@acn	59 South	S London	N1 9EW	UK	
24	LexCorp	http://ww	Luthor	Lex	lex@lex.c	29 Fergus	Toronto	N2N 2K2	Ontario	Canada

```
<data endTime="2014-04-23 23:27:29" generated="2014-04-24 00:08:14" handle="46608654" radarStatus="online" startTime="2014-04-23 23:27:00" type="data">
noise,20140423232700,4,4,COMMUNITY_NOISE,48.5 noise,20140423232700,60,60,COMMUNITY_NOISE,54.3 noise,20140423232700,3,3,COMMUNITY_NOISE,50.2 noise,20140423232700,6,6,COMMUNITY_NOISE,46.9
noise,20140423232700,61,61,COMMUNITY_NOISE,44.5 noise,20140423232700,2,2,COMMUNITY_NOISE,35 noise,20140423232701,4,4,COMMUNITY_NOISE,49.4 noise,20140423232701,60,60,COMMUNITY_NOISE,52.3
noise,20140423232701,3,3,COMMUNITY_NOISE,49.3 noise,20140423232701,6,6,COMMUNITY_NOISE,47.2 noise,20140423232701,61,61,COMMUNITY_NOISE,44.5 noise,20140423232701,2,2,COMMUNITY_NOISE,34.2
noise,20140423232702,4,4,COMMUNITY_NOISE,48.8 noise,20140423232702,60,60,COMMUNITY_NOISE,50.7 noise,20140423232702,3,3,COMMUNITY_NOISE,49.3 noise,20140423232702,6,6,COMMUNITY_NOISE,46.5
noise,20140423232702,61,61,COMMUNITY_NOISE,44.7 noise,20140423232702,2,2,COMMUNITY_NOISE,34.6 noise,20140423232703,4,4,COMMUNITY_NOISE,49.3 noise,20140423232703,60,60,COMMUNITY_NOISE,50.4
noise,20140423232703,3,3,COMMUNITY_NOISE,48.9 noise,20140423232703,6,6,COMMUNITY_NOISE,48.8 noise,20140423232703,61,61,COMMUNITY_NOISE,45.6 noise,20140423232703,2,2,COMMUNITY_NOISE,37.8
</data>
```

BioLegato: bldna - DNA sequence tasks

File Edit Documentatio DNARNA RNA\_Struc Similarity Database Pattern Alignment Primers Help

KJ939334 agcagatattacatacagtgaaatcactaattaaagccatggagaagaaatcactagctggcttatgcttcctcttcttctgtt

NM\_001112307 ggacgagagggttttttgatataaaatgagtggtgactatcacacttgaaataaggccacaacgctagatacaatttttataagtag

KJ551546 cacaaactccaccagtagcaaaagccagtagcagcgaagcagagtttaggtcgctccagccggccgtttcacaaatgtggag

KJ551545 aggaagtagaggttcacatcccccttttaaccccttctaccacctatcaaatgaatggcgctcatcacacaagttcttccctgccgt

KJ551544 gtacaaatcccaagataaaggtgtacattttacaggggtgcgttttagcagatatacaaaataaggaaagaaatggagtcatacacaaagct

KJ551543 cttgaaaggcttaattctgtctatacaaatctctttagcttttcataggaaaatctcatggatctctccatgaagggtgtgtgtggt

KJ551542 ctcccaagcaccctgaattccaagctgtagtggaaccgatccatggagggttcacgcagggtctctctcggggcgtctctccatcc

KJ551541 atccgtctaccatagaaaccagctcgaaagccgaaggctttcacattctaccatacaaatctcttcatcttccaaagcaaccgacac

KJ551540 gcacagcagctccatcactccaccaaaccttgatacactcataccaagcagtaaatcgatgtcgatggagggttcacgcaagcccag

KJ551539 cccagctccaagggttttcacattctaccatacaaatcttcatcttccaaaggaaaccgccatgggtctgtccgcgaagggtatcata

KJ551538 atcatacaaatccatcttgaaaggcttttaattgttcatagaaccttctcaattttcacaggaaaactagatggatcgggtccatc

KJ551537 caccacagcacacaaatccacccggtagtaacaaagccagtagcgtagcgaagcagagaggttttaggtcgctccagccggg

# Tidy and Messy Data

- A very good [paper](#) on tidy data
- Example: several datasets
  - Same information, different ease of use

	country	year	cases	population
1	Afghanistan	1999	745	19987071
2	Afghanistan	2000	2666	20595360
3	Brazil	1999	37737	172006362
4	Brazil	2000	80488	174504898
5	China	1999	212258	1272915272
6	China	2000	213766	1280428583

	country	year	rate
1	Afghanistan	1999	745/19987071
2	Afghanistan	2000	2666/20595360
3	Brazil	1999	37737/172006362
4	Brazil	2000	80488/174504898
5	China	1999	212258/1272915272
6	China	2000	213766/1280428583

Tidy dataset

	country	year	key	value
1	Afghanistan	1999	cases	745
2	Afghanistan	1999	population	19987071
3	Afghanistan	2000	cases	2666
4	Afghanistan	2000	population	20595360
5	Brazil	1999	cases	37737
6	Brazil	1999	population	172006362
7	Brazil	2000	cases	80488
8	Brazil	2000	population	174504898
9	China	1999	cases	212258
10	China	1999	population	1272915272
11	China	2000	cases	213766
12	China	2000	population	1280428583



1. The table header contains values
  - Identify the variables and distribute (unpivot) the values
  - Read the `pew.csv` dataset
    - Distribution of income by religion
  - Show the first 5 values (use the `head()` function)
    - Also see the number of variables and observations (`shape`)
    - This will also ensure that you've read the dataset correctly
    - **Variables:** religion, income, frequency
  - Transform the dataset to make it tidy ([docs](#))

```
pew = pd.read_csv("pew.csv")
pew_tidy = pew.melt(
    id_vars = ["religion"], # Identifier variables (all others are "unpivoted")
    var_name = "income", # Variable
    value_name = "frequency" # Value
)
```



## 2. Multiple variables stored in one column

- Identify and split the variables into separate columns
- Read the tb.csv dataset
  - Tuberculosis cases
  - m04, m514, m1524, etc. contain two variables (gender and age)
    - male, 0-4 years old; male, 5-14 years old, etc.
    - There's also a problem with missing values (NaN)
- Tidying process
  - First, melt all columns (they are values and should not be)
  - Next, split the column names and extract the gender and age information
  - Add the new info to the dataset
  - Remove all missing values

# Messy to Tidy Data

```
def process_age_group(age_group):
    ages = {"04": "0-4", "65": "65+", "u": "unknown"}
    if age_group in ages:
        return ages[age_group]
    else:
        # Put a dash before the last two digits
        return f"{age_group[:-2]}-{age_group[-2:]}"

tb = tb.melt(
    id_vars = ["iso2", "year"], var_name = "sex_and_age", value_name = "cases")

tb["sex"] = tb.sex_and_age.str.get(0)
tb["age_group"] = tb.sex_and_age.str.slice(1)
tb = tb.drop(columns = "sex_and_age")

tb.age_group = tb.age_group.apply(process_age_group)

# Tidy up the column and row order
tb = tb[["iso2", "year", "sex", "age_group", "cases"]]
tb = tb.sort_values(["iso2", "year"])
```

3. Variables are stored in both rows and columns
  - Identify and split the variables
  - Read the weather.csv dataset
    - Daily weather records in Mexico in 2010
    - d1, d2, etc. are the days of a month; tmin and tmax should be columns
      - Make a new column with the date: [date, tmin, tmax]
  - Tidying process
    - Melt all days
    - Create days based on date, month and year
    - Pivot the tmin and tmax columns

# Messy to Tidy Data

```
weather_data = weather_data.melt(
    id_vars = ["id", "year", "month", "element"], var_name = "day")
weather_data.day = weather_data.day.str.slice(1).astype(int)

# Remove missing / invalid days (e.g., 31st April) and dates with no records
weather_data = weather_data.dropna()
weather_data["date"] = pd.to_datetime(weather_data[["year", "month", "day"]])
weather_data = weather_data.drop(columns = ["year", "month", "day"])

# Pivot the elements back to their own columns
weather_data = weather_data.pivot_table(
    index = ["id", "date"], columns = "element", values = "value")

# Pivoting returns a multi-indexed element, go back to a flat DataFrame
weather_data = weather_data.reset_index()
weather_data.columns.name = ""
weather_data = weather_data[["id", "date", "tmin", "tmax"]]
```

## 4. One type in multiple tables

- Merge the tables into one
  - Read all tables, add the new columns
  - Often the filename should be in its own column (if it's important)
  - Melt and tidy if necessary

## 5. Multiple types in one table

- Split into more tables
  - If necessary, introduce relations (similar to a relational database)
- Each table should be responsible for one type of measurement
- \* Read the billboard.csv dataset and apply those transformations



# Operations on Datasets

Basic Tools to Get Started Working with Messy Data

- Selecting only some rows (aka **selection**)
- First / last n records (observations)

```
weather_data.head(10)  
weather_data.tail() # 5 by default
```

- Random n records

```
weather_data.sample(n = 10)  
weather_data.sample() # 1 random record by default
```

- Smallest / largest n records in each column

```
weather_data.nsmallest(3, "tmax")  
weather_data.nlargest(3, "tmax")
```

- Subsetting by a Boolean expression (predicate)
  - Returns only rows where the expression returns True

```
weather_data[weather_data.tmax > 30]
```



- Selecting only some columns (aka **projection**)

- Single column (returns a Series object)

```
weather_data["tmax"]  
weather_data.tmax # Possible in most cases
```

- More than one column (returns a DataFrame object)

```
weather_data[["tmin", "tmax"]]
```

- Combining filters

```
weather_data[weather_data.date > "2010-08-01"][["date", "tmax"]]  
weather_data.loc[weather_data.date > "2010-08-01", ["date", "tmax"]]
```

- A note on Boolean expressions

- and, or, not are &, |, ~
- **Always** put parentheses around the individual expressions

```
weather_data[  
    (weather_data.date > "2010-08-01") & (weather_data.date < "2010-09-01")]
```

- These methods work by columns
  - If multiple columns are passed, they are applied to each column individually

```
print("Count:", weather_data.tmin.count()) # number of non-null values
print("Min:", weather_data.tmin.min())
print("Max:", weather_data.tmin.max())
print("Mean:", weather_data.tmin.mean())
print("Median:", weather_data.tmin.median())
print("Standard deviation:", weather_data.tmin.std())
```

- Grouping
  - Splits the data into several groups based on the values of a column
  - We have to apply a method after grouping
    - Or iterate over the groups (using a for-loop)
  - Example: Average number of people for each income group

```
pew_tidy.groupby("income").mean()
```



Data Cleaning

# Cleaning Data

You've Got the Data... Now What?

- No common way of doing this
- We need to rely on intuition and some common patterns
  - Tidy up the dataset
    - You must know the dataset documentation first
  - Treat nulls / NaNs: either remove them or replace them
    - Replacing values might be **dangerous**
    - If done properly, it will affect the data in a positive way
  - Identify and fix errors (also **dangerous**)
  - Melt and pivot datasets
  - Merge (join) and separate datasets
  - Subset variables and / or observations
  - Summarize and group variables
  - *Pandas Cheat Sheet*

# Example: Weather Data

- Since there's no common way of cleaning, we'll explore and clean a dataset, showing steps and examples as we go
- Dataset (weather data, courtesy of [synesthesiam@github](#))
- Read the dataset (you don't need to download it)
  - See how many variables and observations are there
  - Display the first and last few rows to get a sense of the data
  - Check the data types (to see if something's wrong with the reading)
    - E.g., numbers recognized as strings
  - See a subset of the columns
  - Summarize (describe) the dataset

# Example: Weather Data

- The column names don't look good
    - Make them "pythonic" (lowercase\_with\_underscores)
      - This will make selecting them easier (weather.mean\_temp)
- ```
weather.columns = ["date", "max_temp", "mean_temp", "min_temp", "max_dew",  
                  "mean_dew", "min_dew", "max_humidity", "mean_humidity",  
                  "min_humidity", "max_pressure", "mean_pressure",  
                  "min_pressure", "max_visibility", "mean_visibility",  
                  "min_visibility", "max_wind", "mean_wind", "max_gusts",  
                  "precipitation", "cloud_cover", "events", "wind_dir"]
```
- What are the ranges of data?
    - E. g. temperature, pressure, humidity
    - Use the min() and max() methods
  - \* Try to explore the data a bit
    - Plot a few histograms and / or boxplots to see the distributions

# Example: Weather Data

- Convert the dates to a datetime object
  - To make performing time-dependent analysis easier

```
weather.date = pd.to_datetime(weather.date)
```

- If needed, use `apply()` to perform a function on every row

```
from datetime import datetime
def string_to_date(date_string):
    return datetime.strptime(date_string, "%Y-%m-%d")
```

```
weather.date = weather.date.apply(string_to_date)
```

- It's even better to use dates as indices (when we need to subset date ranges or perform other time-dependent tasks)

```
weather = weather.set_index("date") # or use inplace = True
```

```
print(weather.loc[pd.to_datetime("2012-08-19")])
# or weather.loc["2012-08-19"], or any other formatting
```

- Also see why precipitation is not a float and edit it



# Example: Weather Data

- Remove or replace missing values
  - In this case, replacing is better because removing takes away an entire row

```
weather_with_events = weather.dropna(subset = ["events"])  
weather.events = weather.events.fillna("") # Better
```

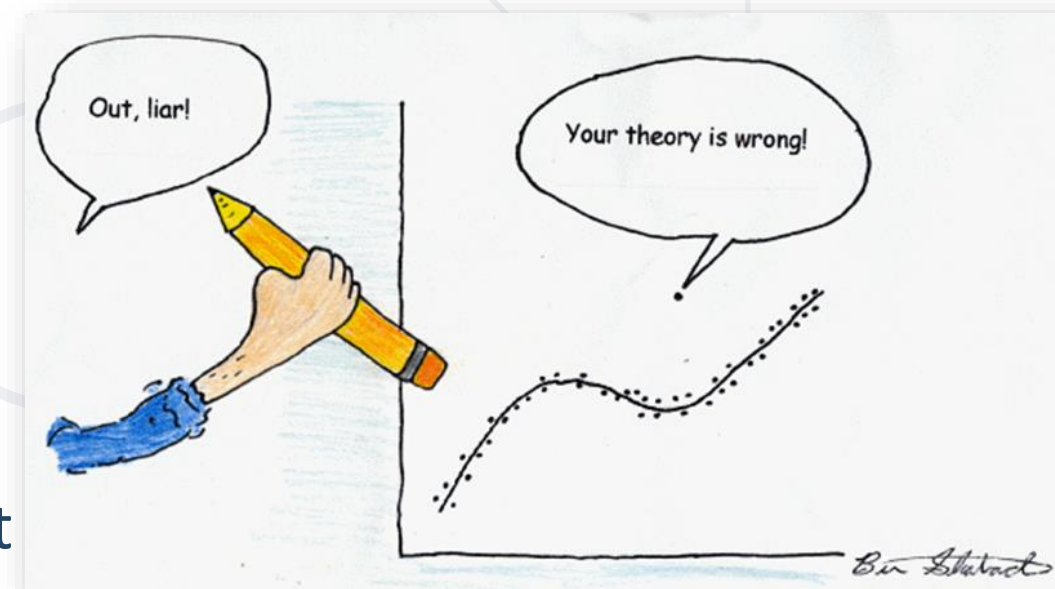
- Try to see how variables interact – group the data
  - E.g., by cloud cover and events
  - Print the number of days when each combination of {cover, events} occurred

```
for (cover, events), group_data in weather.groupby(["cloud_cover", "events"]):  
    print(f"Cover: {cover}, Events: {events}, Count: {len(group_data)}")  
# Or: weather.groupby(["cloud_cover", "events"]).size()
```

- Plot data
  - Next time

- If needed, perform transformations
  - Math operations: log, square root, addition, multiplication, etc.
    - Be careful as you'll get results in different dimensions
  - Normalizing scores (such as using Z-scores) is recommended in most cases
    - It's much better for ML algorithms to have data of similar scales
    - You can do that manually or use a library (such as [\*sklearn.preprocessing\*](#))
  - By convention, calculated columns are added to the dataset
- **Describe all operations as you're doing them**
  - Describe what you're doing and why
    - Useful to check your work later (or allow others to do that)
  - If needed, save the resulting dataset into a file
    - Supply your data transformation log with it
    - Provide a dataset description

- **Outliers** – values which are far from their expected range
  - Or having a very low probability of happening (assuming a model)
- Many possible cases
  - Wrong data entry (e.g. an adult weighing 5kg might be 50kg or something else)
  - Wrong assumptions (the data is correct, our view isn't)
- What to do?
  - Inspect the data point
  - Try to figure out what happened
    - If needed, remove the row or try to replace the value
  - Try a transformation
  - If possible, perform analysis with and without the outlier(s) and compare your results



- The quality of our results depends strongly on the features we use
  - "Garbage in – garbage out"
- **Dimensionality reduction**
  - Reducing the number of variables (features)
  - We can do this manually or use algorithms
  - **Feature selection**
    - Selecting only columns that are useful
  - **Feature extraction**
    - Transforming non-structured to structured data
      - Examples: images, audio, text
    - Getting meaningful features
- **Feature engineering**
  - Using our knowledge of the data to create meaningful features
    - Involves a lot of brainstorming and testing

# Next Steps (Optional)

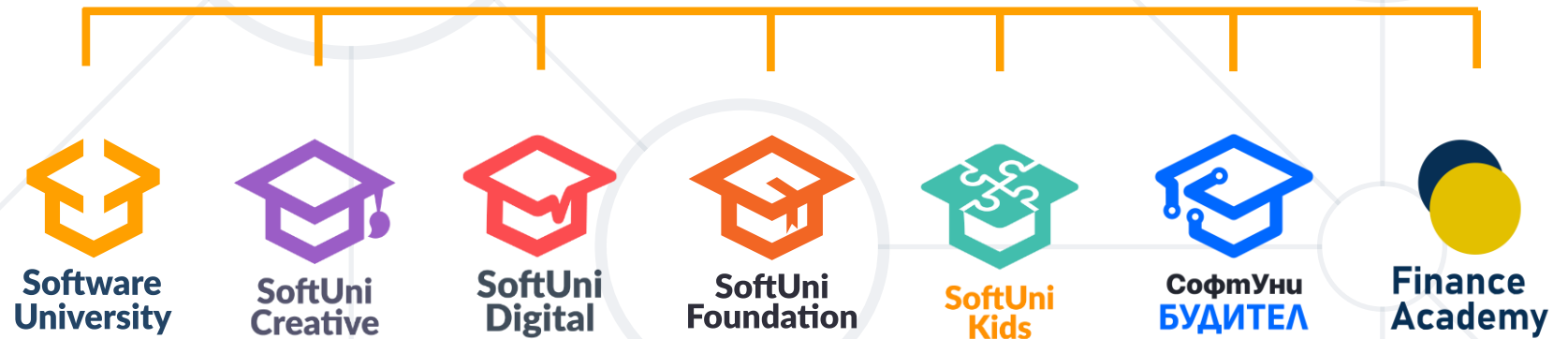
- Have a look at scikit-learn's "*Dataset Transformations*" module
  - It describes the most common operations
    - Data cleaning
    - Dimensionality reduction
    - Feature extraction
- There are many algorithms based on
  - Data types (e.g., text or numerical data, labelled vs. not labelled)
  - Model types (how we want to present our data, e.g., linear model)
  - Algorithm types (e.g., finding similar news articles, recommending movies to users, classifying, etc.)
- No "hard and fast rule", use your intuition
  - Knowing more tools / models / algorithms -> better performance

# Summary

- Messy and Tidy Data
  - Tidying up Messy Data
- Operations on Datasets
- Cleaning Data
  - Validation
  - Transformation
  - Error Correction
  - Features
- Data Tidying and Cleaning as a Process



# Questions?





# SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

