**Модуль 2.1: Дослідницький аналіз даних**

**Що таке дослідницький аналіз даних?**

У даному модулі ми поговоримо про важливу тему у світі даних - exploratory data analysis, або EDA. Це процес дослідження та аналізу даних з метою зрозуміти їх структуру та візуалізувати корисну інформацію, яка може бути прихована в даних. Цей підхід є важливим етапом перед будь-яким аналітичним проектом, оскільки дозволяє виявити патерни та закономірності в даних, а також виявити потенційні проблеми та неточності, які можуть впливати на результати аналізу. У цьому конспекті ми розглянемо основні поняття та інструменти EDA, які допоможуть нам вивчити та зрозуміти наші дані краще.

Одним з основних інструментів первинного дослідження даних є статистика. Ми знаходимо різні статистичні показники наших величин, виявляємо чи є якийсь зв'язок між різними величинами і отримуємо к-сні характеристики цієї аналітики.

**Основи статистики**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B8-%D1%81%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B8)[Статистика](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D1%82%D0%B8%D1%81%D1%82%D0%B8%D0%BA%D0%B0) (https://uk.wikipedia.org/wiki/Статистика) - це наука про збір, аналіз та інтерпретацію даних. Вона дозволяє вивчати явища, які можуть бути виміряні та порівняні за допомогою кількісних методів, таких як розрахунок середнього значення, медіани, стандартного відхилення, кореляції, регресії та інших показників.

Об'єктом досліджень є [**випадкова величина**](https://uk.wikipedia.org/wiki/%D0%92%D0%B8%D0%BF%D0%B0%D0%B4%D0%BA%D0%BE%D0%B2%D0%B0_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B0)

(https://uk.wikipedia.org/wikiВипадкова\_величина) - величина, можливими значеннями якої є результати випробувань чи спостережень явищ або процесів, що носять випадковий характер, наприклад, результати підкидання монети, кубика із числовими позначеннями на його гранях, значення довжини стрибків спортсмена у послідовних спробах тощо. Для нас це буде будь-яка змінна величина у наших даних. Наведемо декілька прикладів:

* кількість кліків на рекламний банер на веб-сайті - залежить від багатьох факторів, таких як зміст банера, місце розташування на сторінці, аудиторія, яка відвідує сайт та інші
* витрати на рекламну кампанію
* ціна на нерухомість

та багато інших. Що по суті є будь-якою змінною величиною, показником, які ми збираємо для нашого датасету.

Ми можемо обчислити багато статистичних показників для випадкової величини - максимальне та мінімальне значення, мода, медіанне значення, математичне очікування(середнє значення), дисперсію, середньоквадратичне відхилення, коваріацію та кореляцію між двома різними випадковими величинами та інші. Якщо мінімальне та максимальне значення для випадкової величини є очевидною характеристикою, то давайте дамо визначення решті важливих показників.

**Мода та медіана**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BC%D0%BE%D0%B4%D0%B0-%D1%82%D0%B0-%D0%BC%D0%B5%D0%B4%D1%96%D0%B0%D0%BD%D0%B0)Мода та медіана - це два основних показники центральної тенденції в статистиці, які дозволяють описати та узагальнити розподіл значень випадкової величини.

Мода - це значення, яке найчастіше зустрічається в розподілі випадкової величини. Іншими словами, мода - це те значення, яке має найбільшу частоту появи в даних: тобто ймовірність цього значення є найбільшим. Наприклад, якщо ми аналізуємо розподіл віку студентів у групі, то мода буде відповідати віку, який зустрічається найбільш часто в цій групі.

Медіана - це значення, яке розділяє впорядковані дані на дві рівні частини. Тобто, якщо ми впорядкуємо дані від найменшого до найбільшого, медіана буде значенням, яке стоїть посередині. Простий приклад ммедіани може бути розподіл кількості годин сну серед учнів школи. Якщо ми впорядкуємо кількість годин сну учнів у порядку зростання, то медіана буде відповідати годинам сну, які стоять у середині впорядкованих даних.

**Математичне очікування**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D0%B5-%D0%BE%D1%87%D1%96%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)[Математичне очікування](Математичне%20очікування) ([https://uk.wikipedia.org/wiki/Математичне сподівання](https://uk.wikipedia.org/wiki/Математичне%20сподівання)) - це середнє значення випадкової величини, яке можна очікувати на основі її розподілу і ймовірностей, пов'язаних з цим розподілом. Іншими словами, це значення, яке можна отримати, якщо провести безліч незалежних випробувань випадкової величини і обчислити середнє арифметичне значення результатів.

Математичне очікування може бути використано для прогнозування результатів випадкових подій або для розуміння поведінки випадкових величин в середньому. Наприклад, якщо ви знаєте розподіл випадкової величини, такої як вартість продукту в інтернет-магазині, і відповідні ймовірності, то ви можете обчислити математичне очікування цієї величини. Це дозволить вам розуміти, яка середня вартість покупки буде у вашого клієнта, що допоможе вам планувати запаси товарів і визначати стратегії продажу.

В математичних обчисленнях математичне очікування може бути обчислене як сума добутків кожного значення випадкової величини на відповідну ймовірність, що це значення виникне. Математичне очікування випадкової величини 𝑋*X* можна обчислити за допомогою наступної формули:

E[𝑋]=M[𝑋]=

де 𝑥𝑖​​ - значення випадкової величини, а 𝑝𝑖​ - відповідна ймовірність виникнення цього значення. Формула вказує на те, що математичне очікування є по суті середнім значенням, оскільки згадаємо, що таке ймовірність:

𝑝𝑖(𝑥𝑖)=

​

де 𝑝𝑖(𝑥𝑖) - ймовірність виникнення значення 𝑥𝑖,𝑛𝑖 - кількість разів, що ми зустріли значення 𝑥𝑖 в нашому датасеті, а 𝑁 - загальна кількість значень досліджуваної випадкової величини.

Тоді якщо ми підставимо це визначення у формулу математичного очікування, то отримаємо такий результат:

E[𝑋]=M[𝑋]= = = ​

де 𝑥𝑗 - значення випадкової величини з можливими повторами, тобто точно так, як і у нас в датасеті. А даний вираз і є середнім значенням, і його часто позначають як 𝜇𝑥​​.

**Стандартне відхилення**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B5-%D0%B2%D1%96%D0%B4%D1%85%D0%B8%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F)

От ми розібрались з тим, що у будь-якої випадкової величини 𝑋 є певне очікуване значення 𝜇𝑥​​, а значить ми можемо порахувати, наскільки в загалоньму кожне конкретне значення може відхилятись від нашого очікуваного, тобто середнього значення. І є декілька величин, якими можна охарактеризувати відхилення.

[**Дисперсія**](https://uk.wikipedia.org/wiki/%D0%94%D0%B8%D1%81%D0%BF%D0%B5%D1%80%D1%81%D1%96%D1%8F_%D0%B2%D0%B8%D0%BF%D0%B0%D0%B4%D0%BA%D0%BE%D0%B2%D0%BE%D1%97_%D0%B2%D0%B5%D0%BB%D0%B8%D1%87%D0%B8%D0%BD%D0%B8) (https://uk.wikipedia.org/wikiДисперсія\_випадкової\_величини) - це міра розсіяння значень випадкової величини відносно середнього значення розподілу. Більші значення дисперсії свідчать про більші відхилення значень випадкової величини від центру розподілу. Часом ще її називають варіація, тому що це від англійського слова variance. Обчислити дисперсію можна за даною формулою:

𝜎2≡D(𝑋)=E[(𝑋−𝜇)2]=∑𝑥(𝑥−𝜇)2𝑝(𝑥)

Для нашого ж набору даних, можемо переписати значення дисперсії:

𝜎2≡D(𝑋)=𝑁(𝑥𝑖−𝜇)2/𝑁

А якщо ми візьмемо від даної величини корінь, то отримаємо [стандартне відхилення](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B5_%D0%B2%D1%96%D0%B4%D1%85%D0%B8%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F) (https://uk.wikipedia.org/wiki/Стандартне\_відхилення).



**Кореляція**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BA%D0%BE%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D1%8F)Попередні статистичні показники характеризували нам лише одну випадкову величину. Проте, якщо ми хочемо виявити, чи є якийсь статистичний зв'язок між двома різними випадковими величинами, ми повинні ввести додаткові означення.

[Коваріа́ція](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%B2%D0%B0%D1%80%D1%96%D0%B0%D1%86%D1%96%D1%8F) (<https://uk.wikipedia.org/wiki/Коваріація>) — це міра спільної мінливості двох випадкових змінних. Якщо більші значення однієї змінної здебільшого відповідають більшим значенням іншої, й те саме виконується для менших значень, тобто змінні схильні демонструвати подібну поведінку, то коваріація є додатною. В протилежному випадку, коли більші значення однієї змінної здебільшого відповідають меншим значенням іншої, тобто змінні схильні демонструвати протилежну поведінку, коваріація є від'ємною.

Обчислити коваріацію можна за даною формулою:

cov(𝑋,𝑌)=E[(𝑋−E[𝑋])(𝑌−E[𝑌])]

Або якщо спростити:



Числове значення коваріації досить складно інтерпретувати, тому ми можемо поділити коваріацію на стандартні відхилення випадкових величин 𝑋*X* та 𝑌*Y* - тоді ми отримаємо [кореляцію](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D1%8F) (https://uk.wikipedia.org/wiki/ Кореляція) між цими величинами:

 ​

У контексті Data Science, кореляція дозволяє оцінити, наскільки сильно залежні між собою дві змінні, тобто, якщо одна змінна збільшується, чи призводить це до збільшення або зменшення іншої змінної.

Коефіцієнт кореляції вимірює ступінь лінійної залежності між двома змінними. Його значення в межах від -1 до 1, де значення 1 вказує на повну позитивну кореляцію (тобто коли збільшення однієї змінної призводить до збільшення іншої змінної), а значення -1 вказує на повну негативну кореляцію (тобто коли збільшення однієї змінної призводить до зменшення іншої змінної). Значення 0 вказує на відсутність кореляції між змінними.

**Бібліотека Pandas**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%B1%D1%96%D0%B1%D0%BB%D1%96%D0%BE%D1%82%D0%B5%D0%BA%D0%B0-pandas)Pandas (Panel data) - це широко поширена бібліотека мовою Python з відкритим вихідним кодом для обробки і аналізу даних. На відміну від основної структури даних бібліотеки NumPy, головний об'єкт pandas DataFrame може містити неоднорідні типи даних (числа з плаваючою комою, цілі числа, рядки, дати тощо), які можна структурувати у вигляді ієрархії та індексувати.

Бібліотека pandas надає дві структури для роботи з даними: Series і DataFrame.

Series - це маркована одновимірна структура даних, її можна уявити, як таблицю з одним рядком (або стовпцем). З Series можна працювати як із звичайним масивом (звертатися за номером індексу), і як із асоційованим масивом, у цьому випадку можна використовувати ключ для доступу до елементів даних.

DataFrame - це двовимірна маркована структура. Вона схожа на звичайну таблицю і це виявляється у способі її створення та у роботі з її елементами.

**Об'єкт Series**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BE%D0%B1%D1%94%D0%BA%D1%82-series)У найпростішій формі об'єкт Series можна створити так само, як одновимірний масив NumPy:

import pandas as pd

mountain\_height = pd.Series([2061, 2035.8, 2028.5, 2022.5, 2016.4])

print(mountain\_height)

0 2061.0

1 2035.8

2 2028.5

3 2022.5

4 2016.4

dtype: float64

Об'єкту Series можна присвоїти рядок імені name і тип даних dtype. У прикладі вище індекс не був явно встановлений, і тоді для індексування за замовчуванням використовується послідовність цілих чисел (починаючи з 0).

Ми можемо вказати явну індексацію елементів, передаючи послідовність як аргумент index, або коли створюємо об'єкт Series зі словника.

import pandas as pd

mountains\_height = pd.Series(

data=[2061, 2035.8, 2028.5, 2022.5, 2016.4],

index=["Goverla", "Brebenskyl", "Pip\_Ivan", "Petros", "Gutin\_Tomnatik"],

name="Height, m",

dtype=float,

)

print(mountains\_height)

Goverla 2061.0

Brebenskyl 2035.8

Pip\_Ivan 2028.5

Petros 2022.5

Gutin\_Tomnatik 2016.4

Name: Height, m, dtype: float64

Це дозволяє зручно посилатися на елементи послідовності Series з використанням міток індексу, замість цілих чисел, у тому числі і на окремі елементи послідовності:

print(mountains\_height[0]) # 2061.0

print(mountains\_height["Goverla"]) # 2061.0

Можна уявити елементи в іншому порядку:

print(mountains\_height[["Pip\_Ivan", "Goverla", "Gutin\_Tomnatik"]])

Pip\_Ivan 2028.5

Goverla 2061.0

Gutin\_Tomnatik 2016.4

Name: Height, m, dtype: float64

Вирізання елементів у стилі Python працює. Можливо навіть використання нотації, схожої на стиль вирізання, але в цьому випадку зазначений кінцевий елемент включається в групу, що вирізається:

import pandas as pd

mountains\_height = pd.Series(

data=[2061, 2035.8, 2028.5, 2022.5, 2016.4],

index=["Goverla", "Brebenskyl", "Pip\_Ivan", "Petros", "Gutin\_Tomnatik"],

name="Height, m",

dtype=float,

)

print(mountains\_height[1:3])

print(mountains\_height["Brebenskyl":"Petros"])

Brebenskyl 2035.8

Pip\_Ivan 2028.5

Name: Height, m, dtype: float64

Brebenskyl 2035.8

Pip\_Ivan 2028.5

Petros 2022.5

Name: Height, m, dtype: float64

Можна звертатися до рядку даних як до атрибуту послідовності Series:

print(mountains\_height.Petros) # 2022.5

print(mountains\_height.Brebenskyl) # 2035.8

Можливе виконання числових операцій з даними Series з використанням векторизації, як і для масивів NumPy.

Операції порівняння та фільтрації об'єкта Series за допомогою логічних операцій створюють новий об'єкт Series

print(mountains\_height > 2030)

print(mountains\_height[mountains\_height > 2030])

Goverla True

Brebenskyl True

Pip\_Ivan False

Petros False

Gutin\_Tomnatik False

Name: Height, m, dtype: bool

Goverla 2061.0

Brebenskyl 2035.8

Name: Height, m, dtype: float64

Операції перевірки на існування елемента в Series використовують індекс, а не значення:

print("Goverla" in mountains\_height) # True

Сортування виконується за індексом або за значеннями, дивлячись, що необхідно для конкретного випадку. Використовуються методи Series.sort\_index та Series.sort\_values відповідно. Методи за замовчуванням повертають новий об'єкт Series, але за допомогою аргументу inplace=True можна оновити вихідний об'єкт.

Аргумент ascending визначає порядок сортування за замовчуванням, містить значення True

sort\_index = mountains\_height.sort\_index()

print(sort\_index)

Brebenskyl 2035.8

Goverla 2061.0

Gutin\_Tomnatik 2016.4

Petros 2022.5

Pip\_Ivan 2028.5

Name: Height, m, dtype: float64

mountains\_height.sort\_values(inplace=True, ascending=False)

print(mountains\_height)

Goverla 2061.0

Brebenskyl 2035.8

Pip\_Ivan 2028.5

Petros 2022.5

Gutin\_Tomnatik 2016.4

Name: Height, m, dtype: float64

Елементи NaN (не число) в об'єкті Series можна замінити на задане значення, використовуючи метод fillna:

import pandas as pd

mountains\_height = pd.Series(

{"Goverla": 2061, "Brebenskyl": 2035.8, "Pip\_Ivan": 2028.5},

index=["Goverla", "Brebenskyl", "Pip\_Ivan", "Petros", "Gutin\_Tomnatik"],

name="Height, m",

dtype=float,

)

print(mountains\_height)

mountains\_height.fillna(0, inplace=True)

print(mountains\_height)

Goverla 2061.0

Brebenskyl 2035.8

Pip\_Ivan 2028.5

Petros NaN

Gutin\_Tomnatik NaN

Name: Height, m, dtype: float64

Goverla 2061.0

Brebenskyl 2035.8

Pip\_Ivan 2028.5

Petros 0.0

Gutin\_Tomnatik 0.0

Name: Height, m, dtype: float64

**Об'єкт DataFrame**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BE%D0%B1%D1%94%D0%BA%D1%82-dataframe)Як ми вже говорили об'єкт DataFrame - ця двовимірна таблиця даних.

import pandas as pd

contacts = pd.DataFrame(

{

"name": [

"Allen Raymond",

"Chaim Lewis",

"Kennedy Lane",

"Wylie Pope",

"Cyrus Jackson",

],

"email": [

"nulla.ante@vestibul.co.uk",

"dui.in@egetlacus.ca",

"mattis.Cras@nonenimMauris.net",

"est@utquamvel.net",

"nibh@semsempererat.com",

],

"phone": [

"(992) 914-3792",

"(294) 840-6685",

"(542) 451-7038",

"(692) 802-2949",

"(501) 472-5218",

],

"favorite": [False, False, True, False, True],

},

index=[1, 2, 3, 4, 5],

)

print(contacts)

name email phone favorite

1 Allen Raymond nulla.ante@vestibul.co.uk (992) 914-3792 False

2 Chaim Lewis dui.in@egetlacus.ca (294) 840-6685 False

3 Kennedy Lane mattis.Cras@nonenimMauris.net (542) 451-7038 True

4 Wylie Pope est@utquamvel.net (692) 802-2949 False

5 Cyrus Jackson nibh@semsempererat.com (501) 472-5218 True

Для роботи з елементами DataFrame розглянемо підходи, що найчастіше використовуються, які представлені у таблиці нижче.



Розглянемо приклади цих операцій для DataFrame contacts

Вибір стовпця:

print(contacts["name"])

1 Allen Raymond

2 Chaim Lewis

3 Kennedy Lane

4 Wylie Pope

5 Cyrus Jackson

Name: name, dtype: object

Вибір рядка за міткою:

print(contacts.loc[1])

name Allen Raymond

email nulla.ante@vestibul.co.uk

phone (992) 914-3792

favorite False

Name: 1, dtype: object

Вибір рядка за індексом:

print(contacts.iloc[1])

name Chaim Lewis

email dui.in@egetlacus.ca

phone (294) 840-6685

favorite False

Name: 2, dtype: object

Зріз за рядками:

print(contacts[0:2])

name email phone favorite

1 Allen Raymond nulla.ante@vestibul.co.uk (992) 914-3792 False

2 Chaim Lewis dui.in@egetlacus.ca (294) 840-6685 False

Вибір рядків, які відповідають умові:

print(contacts[contacts["favorite"]])

name email phone favorite

3 Kennedy Lane mattis.Cras@nonenimMauris.net (542) 451-7038 True

5 Cyrus Jackson nibh@semsempererat.com (501) 472-5218 True

**Читання та запис об'єктів Series та DataFrame**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%87%D0%B8%D1%82%D0%B0%D0%BD%D0%BD%D1%8F-%D1%82%D0%B0-%D0%B7%D0%B0%D0%BF%D0%B8%D1%81-%D0%BE%D0%B1%D1%94%D0%BA%D1%82%D1%96%D0%B2-series-%D1%82%D0%B0-dataframe)

**Робота з даними у форматі CSV**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0-%D0%B7-%D0%B4%D0%B0%D0%BD%D0%B8%D0%BC%D0%B8-%D1%83-%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D1%96-csv)Ми вже неодноразово в процесі навчання зустрічалися з форматом CSV (Comma-Separated Values - значення, розділені комами). Він є одним із найбільш популярних форматів для зберігання табличних даних. Нагадаємо, що він являє собою текстовий документ, в якому елементи даних, найчастіше, розділені комами (можливі й інші символи на кшталт ; тощо), а рядки файлу є записами в таблиці.

Для читання текстових файлів з даними у форматі CSV в об'єкті DataFrame існує метод read\_csv. Метод приймає велику кількість можливих [аргументів](https://pandas.pydata.org/pandas-docs/dev/reference/api/pandas.read_csv.html) ([https://pandas.pydata.org/pandas-docs/dev/reference/api/ pandas.read\_csv.html](https://pandas.pydata.org/pandas-docs/dev/reference/api/%20pandas.read_csv.html)), але ми розберемо тільки основні, які стануть у пригоді нам у роботі:

* filepath\_or\_buffer — шлях до зчитуваного файлу. Є обов'язковим параметром. Шлях може вказувати як локальний файл, так і URL для завантажування даних з мережі;
* sep — роздільник стовпців. За замовчуванням кома ,, але можна використовувати значення:
* /s+ для стовпців, розділених пробільними символами,
* \t для роздільників-табуляцій
* None, щоб бібліотека pandas сама спробувала логічно визначити символ-роздільник.
* delimiter — псевдонім для аргументу sep;
* header  — номери рядків (індекси), які використовуються для імен стовпців. За замовчуванням header=0, що говорить про використання першого рядка для імен стовпців. Іноді у файлі немає заголовка, тоді потрібно присвоїти значення header=None і визначити імена стовпців в аргументі names;
* names — послідовність неповторюваних імен стовпців, які використовуватимуться під час читання файлу;
* nrows — кількість рядків, які зчитуються з файлу. Буває необхідно під час читання декількох рядків з дуже великого файлу для тестування або дослідження даних, що містяться в ньому;
* comment — визначає символ, наприклад #, який при виявленні на початку рядка повідомляє про те, що весь рядок необхідно пропустити;
* skip\_blank\_lines — за замовчуванням встановлено значення True – пропуск порожніх рядків у вхідному файлі. При встановленні значення False порожні рядки інтерпретуються як рядки із значень NaN;
* delim\_whitespace — можна встановити для цього аргумента значення True, замість визначення аргумента sep='\s+', щоб встановити, що стовпці даних розділяються символами пробілів.

Наприклад, щоб прочитати текстовий файл "users.csv" наступної структури:

name,email,phone,favorite

Allen Raymond,nulla.ante@vestibul.co.uk,(992) 914-3792,False

Chaim Lewis,dui.in@egetlacus.ca,(294) 840-6685,False

Kennedy Lane,mattis.Cras@nonenimMauris.net,(542) 451-7038,True

Wylie Pope,est@utquamvel.net,(692) 802-2949,False

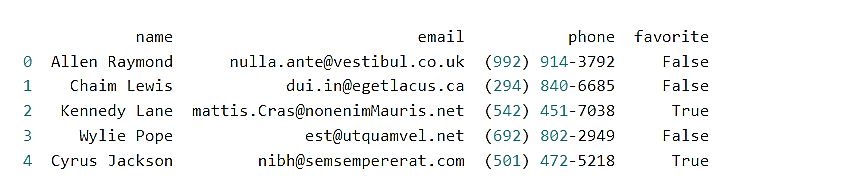
Cyrus Jackson,nibh@semsempererat.com,(501) 472-5218,True

Необхідно виконати наступний код:

import pandas as pd

users = pd.read\_csv("users.csv")

print(users)



Щоб записати дані у файл CSV — необхідно використовувати метод об'єкта DataFrame to\_csv, який зберігає дані відповідно до значень параметрів:

* path\_or\_buf — ім'я файлу;
* sep — символ роздільник полів (за замовчуванням кома ',');
* na\_rep — рядок для заміни відсутніх даних (за замовчуванням порожній рядок);
* columns — послідовність, що ідентифікує виведені стовпці;
* header — за замовчуванням значення True, що означає, що імена стовпців повинні виводитися. Можна встановити значення False або список імен стовпців;
* index — за замовчуванням значення True, що означає, що індекси повинні зберігатись;

Приклад:

import pandas as pd

contacts = pd.DataFrame(

{

"name": [

"Allen Raymond",

"Chaim Lewis",

"Kennedy Lane",

"Wylie Pope",

"Cyrus Jackson",

],

"email": [

"nulla.ante@vestibul.co.uk",

"dui.in@egetlacus.ca",

"mattis.Cras@nonenimMauris.net",

"est@utquamvel.net",

"nibh@semsempererat.com",

],

"phone": [

"(992) 914-3792",

"(294) 840-6685",

"(542) 451-7038",

"(692) 802-2949",

"(501) 472-5218",

],

"favorite": [False, False, True, False, True],

},

index=[1, 2, 3, 4, 5],

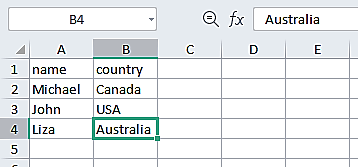
)

contacts.to\_csv("data.csv", index=False)

Ми отримаємо той самий файл, який читали у попередньому прикладі.

**Робота з файлами Microsoft Excel**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0-%D0%B7-%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%D0%BC%D0%B8-microsoft-excel)

Бібліотека pandas має можливість зчитувати вміст файлів Excel з розширеннями .xls та .xlsx в об'єкт DataFrame за допомогою методу pd.read\_excel.



**Файл persons.xlsx**

Примітка: можливо буде необхідно встановити додаткові пакети як openpyxl

import pandas as pd

persons = pd.read\_excel("persons.xlsx")

print(persons)

name country

0 Michael Canada

1 John USA

2 Liza Australia

Щоб зберегти вміст об'єкта DataFrame у файл Excel на одному листі необхідно скористатися методом to\_excel:

import pandas as pd

contacts = pd.DataFrame(

{

"name": [

"Allen Raymond",

"Chaim Lewis",

"Kennedy Lane",

"Wylie Pope",

"Cyrus Jackson",

],

"email": [

"nulla.ante@vestibul.co.uk",

"dui.in@egetlacus.ca",

"mattis.Cras@nonenimMauris.net",

"est@utquamvel.net",

"nibh@semsempererat.com",

],

"phone": [

"(992) 914-3792",

"(294) 840-6685",

"(542) 451-7038",

"(692) 802-2949",

"(501) 472-5218",

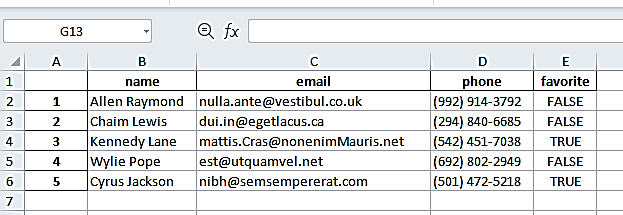
],

"favorite": [False, False, True, False, True],

},

index=[1, 2, 3, 4, 5],)

contacts.to\_excel('contacts.xlsx', sheet\_name='Contacts')



**Результат запису у файл contacts.xlsx**

Примітка: для запису даних на кілька листів, необхідно створити об'єкт pd.ExcelWriter та викликати метод to\_excel для кожного об'єкта.

**Робота з файлами JSON**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0-%D0%B7-%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%D0%BC%D0%B8-json)Для читання даних у форматі JSON використовується метод read\_json.

Параметри методу, що найчастіше використовуються:

* path\_or\_buf — шлях до файлу на диску, URL або JSON рядок, вміст якого - коректний JSON.
* orient — орієнтація, значення за замовчуванням None
* typ — Тип структури pandas: 'series' - це Series, 'frame' - DataFrame. За замовчуванням значення frame

Параметр orient відповідає, щоб JSON, що завантажується, міг бути перетворений на структуру даних pandas, а значить він повинен мати певний вигляд. Можливі значення orient та відповідний їм JSON:

* split це словник із наступною структурою

{

"columns": ["name", "country"],

"index": [1, 2, 3],

"data": [

["Michael", "Canada"],

["John", "USA"],

["Liza", "Australia"]

]

}

Надалі виведення всіх прикладів буде однаковим, код відрізнятиметься лише параметрами методу orient і ми їх будемо пропускати.

import pandas as pd

employees = pd.read\_json("./json/split.json", orient="split")

print(employees)

name country

1 Michael Canada

2 John USA

3 Liza Australia

* records список зі структурою

[

{

"name": "Michael",

"country": "Canada"

},

{

"name": "John",

"country": "USA"

},

{

"name": "Liza",

"country": "Australia"

}

]

* index словник зі структурою

{

"1": {

"name": "Michael",

"country": "Canada"

},

"2": {

"name": "John",

"country": "USA"

},

"3": {

"name": "Liza",

"country": "Australia"

}

}

* columns словник зі структурою

{

"name": { "1": "Michael", "2": "John", "3": "Liza" },

"country": { "1": "Canada", "2": "USA", "3": "Australia" }

}

* values масив значень.

[

["Michael", "Canada"],

["John", "USA"],

["Liza", "Australia"]

]

0 1

0 Michael Canada

1 John USA

2 Liza Australia

Але необхідно знати, що залежно від значення typ, ми можемо використовувати певні значення orient. Якщо typ='series', то orient може бути split, records або index, якщо typ='frame', то orient розширює свої значення до списку: split, records, index, columns, values.

При записі даних у форматі JSON необхідна зворотна операція перетворення даних. Для цього використовується метод to\_json

import pandas as pd

data = {

"name": {"1": "Michael", "2": "John", "3": "Liza"},

"country": {"1": "Canada", "2": "USA", "3": "Australia"}

}

employees = pd.DataFrame(data)

employees.to\_json("employees.json", orient="split")

Структура файлу employees.json вийде наступною:

{

"columns": ["name", "country"],

"index": ["1", "2", "3"],

"data": [

["Michael", "Canada"],

["John", "USA"],

["Liza", "Australia"]

]

}

**Збір даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%B7%D0%B1%D1%96%D1%80-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)

Інструментарій для парсингу даних у Python великий, але бібліотека pandas має власний метод read\_html. Його можна використовувати для парсингу веб-сторінок з метою вилучення даних, що містяться в HTML-таблицях. Якщо таблиці правильно оформлені, то метод дозволяє достатньо ефективно виконувати необхідну роботу.

Основні параметри.

* io — URL або файловий об'єкт, з якого зчитується HTML-код для парсингу.
* match — необов'язковий рядок для пошуку в таблиці, тобто обробляються і повертаються тільки таблиці, що містять цей рядок.
* header — індексний рядок, який використовуватиметься для заголовків стовпців. За замовчуванням None – використовуються заголовки комірок у форматі HTML <th>, якщо вони присутні.
* attrs — словник атрибутів HTML для визначення потрібної таблиці.
* thousands — символ-роздільник, що використовується для поділу груп розрядів у великих числах (за замовчуванням кома).
* decimal — символ, який використовується для позначення десяткової крапки (за замовчуванням крапка)

import pandas as pd

tmp = pd.read\_html("https://statisticstimes.com/tech/top-computer-languages.php", attrs={"id": "table\_id1"})

print(tmp[0].head())

Шукана таблиця в HTML розмітці має атрибут id зі значенням table\_id1

Aug 2021 Change Programming language Share Trends

0 1 NaN Python 29.93 % -2.2 %

1 2 NaN Java 17.78 % +1.2 %

2 3 NaN JavaScript 8.79 % +0.6 %

3 4 NaN C# 6.73 % +0.2 %

4 5 ↑ C/C++ 6.45 % +0.7 %

**Робота з даними. Очищення та обстеження даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%80%D0%BE%D0%B1%D0%BE%D1%82%D0%B0-%D0%B7-%D0%B4%D0%B0%D0%BD%D0%B8%D0%BC%D0%B8-%D0%BE%D1%87%D0%B8%D1%89%D0%B5%D0%BD%D0%BD%D1%8F-%D1%82%D0%B0-%D0%BE%D0%B1%D1%81%D1%82%D0%B5%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)**Додавання**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%B4%D0%BE%D0%B4%D0%B0%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)Щоб додати дані до структури Series, потрібно вказати новий індекс для об'єкта та задати значення елементу.

import pandas as pd

data = pd.Series([1, 2, 3])

data[3] = 4

print(data)

0 1

1 2

2 3

3 4

dtype: int64

У DataFrame можна додавати як додаткові стовпці, так і цілі записи.

Додавання стовпця:

import pandas as pd

data = {

"name": {"1": "Michael", "2": "John", "3": "Liza"},

"country": {"1": "Canada", "2": "USA", "3": "Australia"}

}

employees = pd.DataFrame(data)

employees["age"] = [25, 32, 19]

print(employees)

name country age

1 Michael Canada 25

2 John USA 32

3 Liza Australia 19

Використовуючи метод append, можна додати об'єкт Series як рядок.

import pandas as pd

data = {

"name": {"1": "Michael", "2": "John", "3": "Liza"},

"country": {"1": "Canada", "2": "USA", "3": "Australia"},

"age": {"1": 25, "2": 32, "3": 19}

}

employees = pd.DataFrame(data)

new\_employee = pd.Series(["Jhon", "Denmark", 23], ["name", "country", "age"])

employees = employees.append(new\_employee, ignore\_index=True)

print(employees)

name country age

0 Michael Canada 25

1 John USA 32

2 Liza Australia 19

3 Jhon Denmark 23

**Видалення**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%B2%D0%B8%D0%B4%D0%B0%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F)Для видалення даних використовується метод drop. Методу як аргумент передається список міток для видалення. При цьому поточна структура не змінюється, а повертається новий об'єкт Series, в якому відсутні вказані мітки.

Якщо потрібно змінити вихідну структуру, то можна передати методу аргумент inplace зі значенням True:

numbers = pd.Series([1, 2, 3, 4, 5])

numbers.drop([1,3], inplace=True)

print(numbers)

0 1

2 3

4 5

dtype: int64

Для видалення елементів зі структури DataFrame також застосовується метод drop

import pandas as pd

data = {

"name": {"1": "Michael", "2": "John", "3": "Liza"},

"country": {"1": "Canada", "2": "USA", "3": "Australia"},

"age": {"1": 25, "2": 32, "3": 19}

}

employees = pd.DataFrame(data)

employees = employees.drop(["2"])

print(employees)

name country age

1 Michael Canada 25

3 Liza Australia 19

Щоб видаляти стовпці, необхідно вказати вісь через параметр axis за замовчуванням axis=0, що означає роботу з рядками. Якщо вказати axis=1, то це дозволить видаляти стовпці:

import pandas as pd

data = {

"name": {"1": "Michael", "2": "John", "3": "Liza"},

"country": {"1": "Canada", "2": "USA", "3": "Australia"},

"age": {"1": 25, "2": 32, "3": 19}

}

employees = pd.DataFrame(data)

employees = employees.drop(["age"], axis=1)

print(employees)

name country

1 Michael Canada

2 John USA

3 Liza Australia

**Об'єднання**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BE%D0%B1%D1%94%D0%B4%D0%BD%D0%B0%D0%BD%D0%BD%D1%8F)Для об'єднання структур pandas використовується метод concat

Основні параметри:

* objs — структура Series або DataFrame.
* axis — значення 0 об'єднання здійснюється по рядках, 1 - по стовпцях.
* join — значення за замовчуванням: 'outer'. Якщо 'outer', підсумкова структура буде результатом об'єднання (логічне АБО) переданих структур, 'inner' - підсумкова структура буде результатом перетину (логічне І) переданих структур.
* ignore\_index — логічний вираз має значення за замовчуванням False. Якщо значення True, то не використовується значення індексу в процесі об'єднання, а якщо False - використовується.

Розглянемо приклади.

import pandas as pd

data1 = {

"name": {"1": "Michael", "2": "John"},

"country": {"1": "Canada", "2": "USA"},

"age": {"1": 25, "2": 32}

}

employees1 = pd.DataFrame(data1)

data2 = {

"name": {"3": "Liza", "4": "Jhon"},

"country": {"3": "Australia", "4": "Denmark"},

"age": {"3": 19, "4": 23}

}

employees2 = pd.DataFrame(data2)

employees = pd.concat([employees1, employees2])

print(employees)

name country age

1 Michael Canada 25

2 John USA 32

3 Liza Australia 19

4 Jhon Denmark 23

Можна об'єднати по стовпчикам.

import pandas as pd

data1 = {

"name": {"1": "Michael", "2": "John", "3": "Liza", "4": "Jhon"},

"country": {"1": "Canada", "2": "USA", "3": "Australia", "4": "Denmark"}

}

employees1 = pd.DataFrame(data1)

data2 = {

"age": {"1": 25, "2": 32, "3": 19, "4": 23}

}

employees2 = pd.DataFrame(data2)

employees = pd.concat([employees1, employees2], axis=1)

print(employees)

Результат буде той самий.

name country age

1 Michael Canada 25

2 John USA 32

3 Liza Australia 19

4 Jhon Denmark 23

Також до складу pandas входить метод [merge](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.merge.html) (<https://pandas.pydata.org/docs/> reference/api/pandas.DataFrame.merge.html) для об'єднання даних, що використовує підхід Database-style.

**Очищення**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%BE%D1%87%D0%B8%D1%89%D0%B5%D0%BD%D0%BD%D1%8F)Ідеальних даних не буває. У ході будь-яких експериментальних досліджень, виходять набори даних з некоректними або відсутніми значеннями.

Бібліотека pandas надає різноманітні методи для обробки подібних відсутніх значень, включаючи функції для видалення або заміни некоректних точок даних на середні значення або значення за замовчуванням.

Метод dropna повертає новий об'єкт DataFrame з рядками, що містять лише ненульові дані:

import numpy as np

import pandas as pd

data = pd.DataFrame([[1, 2, 3], [4, np.nan, 6], [7, np.nan, np.nan]])

print(data)

0 1 2

0 1 2.0 3.0

1 4 NaN 6.0

2 7 NaN NaN

Можна відкинути всі рядки зі значеннями NaN

data = data.dropna()

print(data)

0 1 2

0 1 2.0 3.0

Можливо відкинути лише рядки (або стовпці), що складаються повністю зі значень NaN. У цьому випадку в метод dropna передається аргумент how='all' замість how='any', що використовується за замовчуванням.

Якщо необхідно замінити, а не відкинути дані, ми знаємо, що можна використовувати метод fillna. Наприклад, для заміни відсутніх даних у стовпцях середніми значеннями за цими стовпцями:

import numpy as np

import pandas as pd

data = pd.DataFrame([[1, 2, 3], [4, np.nan, 6], [7, np.nan, np.nan]])

data = data.fillna({0: data[0].mean(), 1: data[1].mean(), 2: data[2].mean()})

print(data)

0 1 2

0 1 2.0 3.0

1 4 2.0 6.0

2 7 2.0 4.5

Якщо необхідно замінювати конкретні значення, а не NaN існує схожий метод [replace] (<https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.replace.html>)

Для видалення дублюючих даних можна використовувати метод [drop\_duplicates](https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop_duplicates.html) (https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.drop\_duplicates.html)

import pandas as pd

data = {

"name": ["Michael", "Steve", "Liza", "Jhon", "Liza", "Jhon"],

"country": ["Canada", "USA", "Australia", "Denmark", "Australia", "Denmark"],

"age": [25, 32, 19, 23, 19, 23]

}

employees = pd.DataFrame(data)

employees = employees.drop\_duplicates()

print(employees)

0 Michael Canada 25

1 Steve USA 32

2 Liza Australia 19

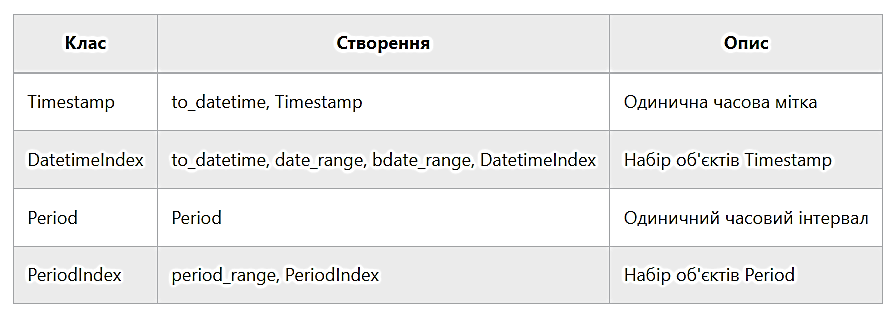
3 Jhon Denmark 23

**Часові ряди**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D1%87%D0%B0%D1%81%D0%BE%D0%B2%D1%96-%D1%80%D1%8F%D0%B4%D0%B8)Популярним напрямом у сфері роботи з даними є аналіз часових рядів.

Бібліотека pandas містить набір інструментів, що дозволяє проводити маніпуляції з даними, представленими у вигляді часових рядів.

**Класи pandas для роботи з часовими рядами**



В pandas розрізняють часові мітки та часові інтервали.

Часова мітка — це конкретне значення дати та часу, наприклад: 2021-09-10 3:39:05. Часовий інтервал передбачає наявність неповної часової мітки та маркера, який визначає період, наприклад мітка 2021-09 матиме маркер M, що означає — місяць. DatetimeIndex — це клас, що дозволяє зберігати масив часових міток. PeriodIndex - клас, що зберігає масив часових інтервалів. Функціонал створення часових міток та часових інтервалів дуже великий, і можливість його розглянути виходить за межі цього конспекту.

Для створення часової мітки можна скористатися конструктором Timestamp.

import pandas as pd

date = pd.Timestamp("2021-09-10")

print(date) # 2021-09-10 00:00:00

Або методом to\_datetime.

import pandas as pd

date = pd.to\_datetime("2021-09-10 2:54:13")

print(date) # 2021-09-10 2:54:1

Усі об'єкти класів можуть бути використані як індекси для структур даних pandas.

Наприклад, створимо DatetimeIndex — ряд часових міток з відсіченням в один день тривалістю вісім днів, змінна date. На основі отриманого об'єкту створимо структуру Series з денною температурою у місті Полтава у ці дні.

import pandas as pd

date = pd.date\_range(start='2021-09-01', freq='D', periods=8)

temperature = pd.Series([23, 17, 17, 16, 15, 14, 17, 20], index=date)

print(temperature)

2021-09-01 23

2021-09-02 17

2021-09-03 17

2021-09-04 16

2021-09-05 15

2021-09-06 14

2021-09-07 17

Для визначення кроку діапазону параметр freq може містити наступні значення.



**Візуалізація даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)Для візуалізації даних у бібліотеці pandas існує метод plot, який викликається в об'єкта структури Series або DataFrame. Якщо використовувати цей метод без параметрів, то буде побудований лінійний графік. За вигляд графіка відповідає аргумент kind і, залежно від його значення, змінюватиметься форма графічного представлення даних. Можливі значення цього параметра наведені в таблиці нижче.



import pandas as pd

import matplotlib.pyplot as plt

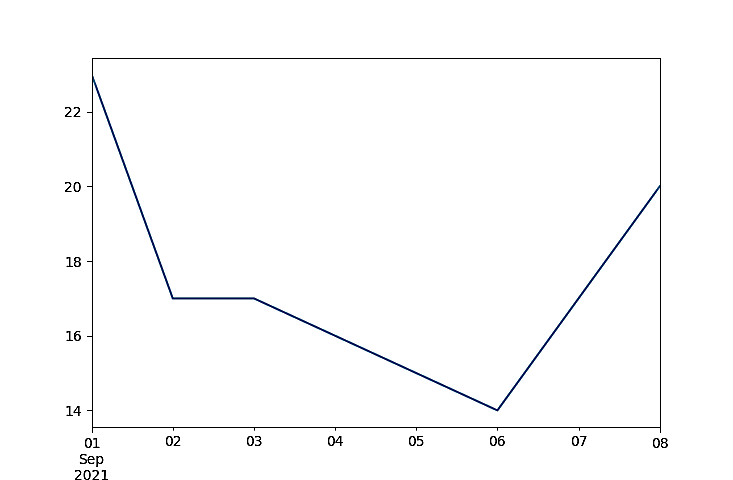
date = pd.date\_range(start='2021-09-01', freq='D', periods=8)

temperature = pd.Series([23, 17, 17, 16, 15, 14, 17, 20], index=date)

temperature.plot()

plt.show()

Щоб побачити побудову, ми імпортували бібліотеку matplotlib і викликали метод plt.show(). Детальніше ми познайомимося з цією бібліотекою у наступному модулі.



**Графік температур**

**Додаткові джерела**

[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main#%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BA%D0%BE%D0%B2%D1%96-%D0%B4%D0%B6%D0%B5%D1%80%D0%B5%D0%BB%D0%B0)

Ми розглянули основні можливості та моменти роботи з пакетом Pandas. Для закріплення матеріалу рекомендуємо прочитати ще наступний цикл статей:

* [Звісно офіційна документація](https://pandas.pydata.org/) (https://pandas.pydata.org/)
* [Трішки практики на частково безкоштовному ресурсі](https://www.w3schools.com/python/pandas/default.asp)
* (https://www.w3schools.com/python/pandas/default.asp)
* [Приклад Data Science завдань та рішень в pandas](https://youtu.be/cc0HOiKN_ac)
* (https://youtu.be/cc0HOiKN\_ac)
* [Шпаргалка по бібліотеці Pandas](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf)
* (https://pandas.pydata.org/Pandas\_Cheat\_Sheet.pdf)