ЛАБОРАТОРНА РОБОТА 1

Тема: Створення робочого середовища та основи роботи

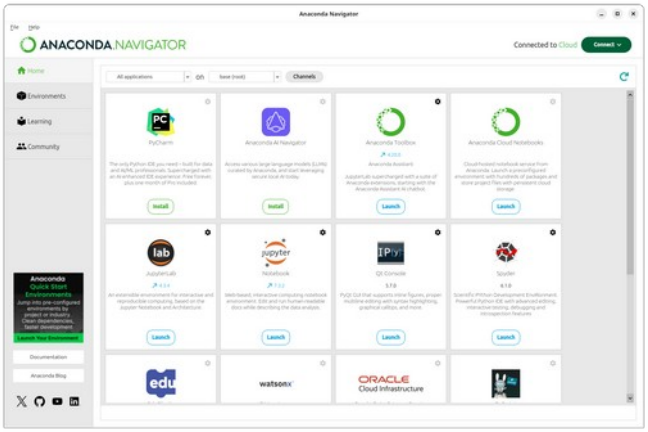
В області аналізу даних та інтерактивних науково-дослідних розрахунків з візуалізацією результатів Python із фреймворками для нього займає одне з перших міст серед багатьох предметно-орієнтованих мов програмування та інструментів такими як R, MATLAB, SAS, Stata, тощо. Розроблені бібліотеки для Python зробили його серйозним конкурентом у вирішенні завдань перетворення та маніпулювання даними. Екосистема Python якнайкраще підходить на роль провідної мови програмування для розробки сучасних фінансових рішень, тому фінансові установи по всьому світу активно застосовують Python та його розгалужену екосистему пакетів аналізу даних, візуалізації та машинного навчання.

Python є об'єктно-орієнтованою, високорівневою мовою програмування з динамічною семантикою. Високорівневі структури даних у поєднанні з динамічною типізацією та динамічним зв'язуванням роблять його зручним інструментом швидкої розробки додатків, а також привабливим засобом написання сценаріїв та мовою інтеграції існуючих компонентів. Зрозумілий синтаксис мови орієнтований на зручність читання коду дозволяє знизити витрати на супровід програм. У Python підтримуються модулі та пакети, що полегшує повторне використання коду та сприяє підвищенню модульності програм. Інтерпретатори, стандартні бібліотеки та велика кількість фреймворків Python є вільно доступними як у вигляді вихідних кодів, так і в бінарній реалізації для більшості комп'ютерних платформ.

Для створення робочого середовища Python для виконання лабораторних робіт, так само, як і його екосистеми для складних задач рівня підприємства, можна використати поетапну інсталяцію за допомогою як звичайних інструментів операційних систем, так і PIP - системи керування пакунками мови Python, яка забезпечує зручний доступ до тисячі пакетів PyPI (https://pypi.org). Наприклад, для інсталяції таких популярних та важливих бібліотек, як: numpy, scipy, pandas, matplotlib, а також середовища виконання jupyter та notebook можна скористатись наступним командним рядком:

$ pip3 install numpy scipy pandas matplotlib jupyter notebook

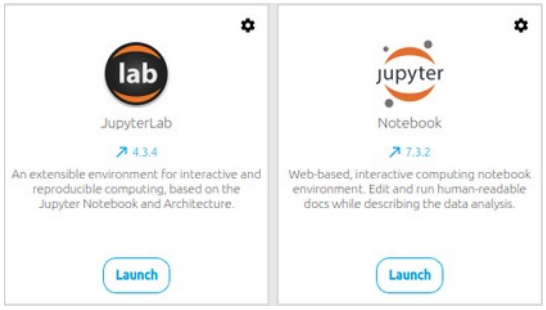
Інший шлях — інсталяція інтегрованого середовища на основі Anaconda (https://www.anaconda.com/download/success). Перевагою цього підходу є отримання підтримки пакетної інтеграції для вирішення різноманітних задач, а також зручна навігація та керування середовищами виконання. Основним інструментом реалізації управління налаштуваннями та пакетами, а також запуску пакетів є Anaconda Navigator.



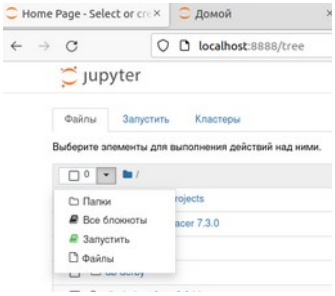
Для роботи з кодом Python можна використати, наприклад, середовище Jupyter Notebook. Для його запуску в командному рядку задаємо наступні команди:

```
(base) vitaliy@vitaliy-HP-EliteBook-8460p:~$ jupyter notebook
```

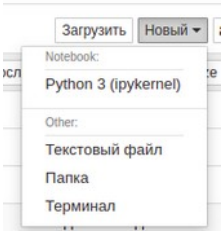
За допомогою Anaconda Navigator запуск виконується або JupyterLab, або Jupyter Notebook:



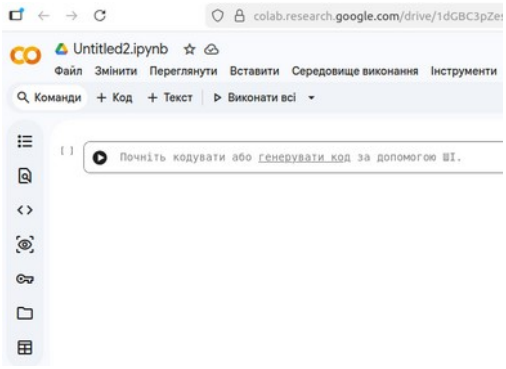
За замовчуванням запускається основний браузер, наприклад, FireFox або Chrome, в якому до адресний рядка заноситься відповідна адреса:



Створюємо новий блокнот та задаємо йому ім’я:



Також для створення програм на Python та вивчення його фреймворків можна використовувати Google Colab (https://colab.google/). Це сервіс Jupyter Notebook, який не потребує особливих налаштувань для використання та надає безкоштовний доступ до обчислювальних ресурсів, що включають графічні та технологічні процесори.



Для роботи з даними в середовищі Python широко використовується бібліотека pandas (Python Data Analysis Library). Для її інсталяції в середовище виконання слід виконати наступну команду:

~$ pip3 install pandas

Очікувано, що будуть додаватись ще декілька бібліотек, які забезпечують pandas. Бібліотеки, які попередньо вже було встановлено, перевіряються на актуальність версій, при необхідності виконується оновлення. Для вивчення основних концепцій pandas необхідні певні об'єми даних, для чого буде використовуватись популярний відкритий ресурс Yahoo Finance. Для цього необхідно встановити open-source бібліотеку yfinance:

~$ pip3 install yfinance

За допомогою yfinance вирішуються такі задачі: завантаження цін акцій (Open, High, Low, Close, Volume) за будь-який період; перегляд балансових звітів, звітів про прибутки та збитки, рух грошових коштів; відстеження виплат дивідендів акціонерам та дроблення акцій; завантаження інформації про компанію; отримання прогнозів аналітиків та рекомендацій (buy/sell).
Для завантаження даних з Yahoo Finance також необхідно під'єднати бібліотеку pandas:

```
[2]: import pandas as pd
     import yfinance as yf
```

Будь-яка інформація, що підлягає аналізу повинна зберігатись у файлі певного типу, крім того, якщо аналіз виконується за допомогою pandas, то вона повинна відповідати певній структурі даних. В pandas є два типи структур даних: Series та DataFrame.
Series є одновимірною структурою даних - «одновимірна ndarray», з мітками осей (включаючи часові ряди). Мітки не обов'язково мають бути унікальними, але повинні мати тип хешування. Об'єкт підтримує індексування на основі цілих чисел та міток та надає методи для виконання операцій із індексом. Статистичні методи з ndarray було замінено, щоб автоматично виключити відсутні дані (наразі представлені як NaN).

Загальний формат класу:

class pandas.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)

де data - масив, ітерабельні, довідникові або скалярні дані; index — одновимірний масив, значення якого повинні бути хешованими і мати таку ж довжину, що і дані; dtype — опціональний параметр, що має строковий, numpy.dtype або ExtensionDtype тип даних для вихідної серії; name — необов'язковий параметр, який має строковий тип даних та задає ім'я серії; copy — буліанівський тип (за замовчуванням false) і вказує на копіювання вхідних даних.
Достатньо простими прикладами внесення даних до Series є:
з використанням масиву

```
Ввод [14]: d = [100,200,300,400,500,600,700,800,900]
           test_set_series = pd.Series(data=d)

           test_set_series

Out[14]: 0    100
         1    200
         2    300
         3    400
         4    500
         5    600
         6    700
         7    800
         8    900
         dtype: int64
```

з використанням довідника

```
Ввод [13]: d = {'1': 101,'2': 202,'3': 303,'4': 404,'5': 505,'6': 606,'7': 707,'8': 808,'9': 909}
           test_set_series = pd.Series(data=d, index=['1','2','3','4','5','6','7','8','9'])

           test_set_series

Out[13]: 1    101
         2    202
         3    303
         4    404
         5    505
         6    606
         7    707
         8    808
         9    909
         dtype: int64
```

DataFrame — двовимірна структура, яка складається із стовпців та рядків. У стовпців є імена, а рядки мають індекси. Загальний вигляд класу:

class pandas.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)

де data — масив (структурований або однорідний), Iterable, словник або DataFrame; index — індекс або одновимірний масив; columns - індекс або одновимірний масив; dtype — тип даних; copy — буліанівський тип, що позначає копіювання вхідних даних.
Простий приклад застосування DataFrame:

```
Ввод [21]: d = {'user_id':[1000001,1000002,1000003,1000004,1000005],
                'phone_type':['android','ios','error','error','ios'],
                'source':['invited_a_friend','invited_a_friend',
                          'invited_a_friend','invited_a_friend','invited_a_friend'],
                'free':[5.0,4.0,37.0,0.0,6.0],
                'super':[0.0,0.0,0.0,0.0,0.0]}
           df = pd.DataFrame(data=d)
           df
```

Out[21]:

	user_id	phone_type	source	free	super
0	1000001	android	invited_a_friend	5.0	0.0
1	1000002	ios	invited_a_friend	4.0	0.0
2	1000003	error	invited_a_friend	37.0	0.0
3	1000004	error	invited_a_friend	0.0	0.0
4	1000005	ios	invited_a_friend	6.0	0.0

Для завантаження даних з Yahoo Finance використаємо наступний рядок:

```
df_yahoo = yf.download('AAPL',start='2000-01-01',end='2010-12-31',progress=False)
```

де AAPL — біржовий тікер компанії Apple Inc.;
start та end — визначають діапазон отримання даних.

Тип даних, до якого відноситься df_yahoo, визначаємо наступним чином:

```
[4]: type(df_yahoo)

[4]: pandas.core.frame.DataFrame
```

Також можна визначити тип індекса:

```
[8]: type(df_yahoo.index)

[8]: pandas.core.indexes.datetimes.DatetimeIndex
```

Таким чином, **df_yahoo** — це pandas.DataFrame з часовим індексом (**DatetimeIndex**), що містить фінансові дані.
Значення індекса виводимо наступним чином:

```
print(df_yahoo.index)

DatetimeIndex(['2000-01-03', '2000-01-04', '2000-01-05', '2000-01-06',
               '2000-01-07', '2000-01-10', '2000-01-11', '2000-01-12',
               '2000-01-13', '2000-01-14',
               ...
               '2010-12-16', '2010-12-17', '2010-12-20', '2010-12-21',
               '2010-12-22', '2010-12-23', '2010-12-27', '2010-12-28',
               '2010-12-29', '2010-12-30'],
              dtype='datetime64[ns]', name='Date', length=2766, freq=None)
```

Стовпці отриманих даних мають тип:

```
type(df_yahoo.columns)

pandas.core.indexes.multi.MultiIndex
```

а їх значення:

```
print(df_yahoo.columns)

MultiIndex([( 'Close', 'AAPL'),
            (  'High', 'AAPL'),
            (   'Low', 'AAPL'),
            (  'Open', 'AAPL'),
            ('Volume', 'AAPL')],
           names=['Price', 'Ticker'])
```

Для виводу отриманого набору даних виконуємо:

```
print(df_yahoo)

Price          Close      High       Low      Open     Volume
Ticker          AAPL      AAPL      AAPL      AAPL       AAPL
Date
2000-01-03  0.839280  0.843498  0.762428  0.786328  535796800
2000-01-04  0.768521  0.829439  0.758680  0.811633  512377600
2000-01-05  0.779767  0.828971  0.772269  0.777892  778321600
2000-01-06  0.712287  0.802260  0.712287  0.795700  767972800
2000-01-07  0.746027  0.757274  0.716037  0.723534  460734400
...              ...       ...       ...       ...        ...
2010-12-23  9.705103  9.751589  9.692207  9.747091  223157200
2010-12-27  9.737493  9.760286  9.642721  9.682610  249816000
2010-12-28  9.761187  9.796876  9.748890  9.774382  175924000
2010-12-29  9.755788  9.790577  9.750089  9.783679  163139200
2010-12-30  9.706903  9.762385  9.688608  9.761486  157494400

[2766 rows x 5 columns]
```

Таким чином, структура df_yahoo є типовою структурою DataFrame, де індекс: DatetimeIndex — дати торгів (тип pandas.Timestamp), а колонки містять фінансові дані самих торгів:

Колонка	Тип даних	Бізнес-інтерпретація
Open	float64	Ціна відкриття
High	float64	Максимум за день
Low	float64	Мінімум за день
Close	float64	Ціна закриття
Volume	int64	Обсяг торгів

Ці дані є історичним біржовим котируванням AAPL у вигляді часового ряду. На їх основі можна показати певні повноцінні операції статистичного аналізу. На практиці, такий аналіз є основою для формування певних бізнес-висновків.
Описова статистика (descriptive statistics) для часовго ряду ціни закриття акцій Apple (AAPL) за обраний період отримується за допомогою наступної функції:

```
df_yahoo['Close'].describe()
```

Ticker	AAPL
count	2766.000000
mean	2.356126
std	2.421508
min	0.196741
25%	0.343547
50%	1.288865
75%	3.821609
max	9.761187

де count = 2766 - кількість спостережень, (приблизно 11 років × ~252 торгові дні на рік);
mean = 2.356126 - середнє арифметичне значення ціни закриття (для фінансових часових рядів mean не є “типовою” ціною, тому, що дані нестаціонарні і є довгостроковий тренд зростання;
std = 2.421508 - стандартне відхилення — міра розсіювання;
min = 0.196741 - мінімальна зафіксована ціна закриття;
25% = 0.343547 (Q1) - перший квартиль, 25% днів мали ціну ≤ 0.34 $;
50% = 1.288865 - медіана або центральне значення, половина значень менша за 1.29 $;
75% = 3.821609 (Q3) - третій квартиль, 75% значень ≤ 3.82 $;
max = 9.761187 - максимальна ціна закриття ≈ 9.76 $, припадає на кінець періоду і відображає зростання бізнесу Apple у довгостроковій перспективі.

Для завантаження .csv файла з даними в pandas використовується функція read_csv(). Якщо є розмічений .csv файл, наприклад Fish.csv, то додавання рядка

pd.read_csv('Fish.csv',delimiter=',')

дозволяє отримати дані з цього файлу

Ввод [23]:
```
pd.read_csv('Fish.csv',delimiter=',')
```

Out[23]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340
...	...	...	...	...	...	...	...
154	Smelt	12.2	11.5	12.2	13.4	2.0904	1.3936
155	Smelt	13.4	11.7	12.4	13.5	2.4300	1.2690
156	Smelt	12.2	12.1	13.0	13.8	2.2770	1.2558
157	Smelt	19.7	13.2	14.3	15.2	2.8728	2.0672
158	Smelt	19.9	13.8	15.0	16.2	2.9322	1.8792

159 rows × 7 columns

Також дані з файлу можна зберегти у деякій змінній, що дозволяє їх виводити без додаткового читання:

Ввод [24]:
```
fish_parameters = pd.read_csv('Fish.csv',delimiter=',')
fish_parameters
```

Out[24]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340
...	...	...	...	...	...	...	...
154	Smelt	12.2	11.5	12.2	13.4	2.0904	1.3936
155	Smelt	13.4	11.7	12.4	13.5	2.4300	1.2690
156	Smelt	12.2	12.1	13.0	13.8	2.2770	1.2558
157	Smelt	19.7	13.2	14.3	15.2	2.8728	2.0672
158	Smelt	19.9	13.8	15.0	16.2	2.9322	1.8792

159 rows × 7 columns

За допомогою певних функцій можна вивести 5 перших або 5 останніх рядків:

Ввод [25]:
```
fish_parameters.head()
```

Out[25]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

Ввод [26]:
```
fish_parameters.tail()
```

Out[26]:

	Species	Weight	Length1	Length2	Length3	Height	Width
154	Smelt	12.2	11.5	12.2	13.4	2.0904	1.3936
155	Smelt	13.4	11.7	12.4	13.5	2.4300	1.2690
156	Smelt	12.2	12.1	13.0	13.8	2.2770	1.2558
157	Smelt	19.7	13.2	14.3	15.2	2.8728	2.0672
158	Smelt	19.9	13.8	15.0	16.2	2.9322	1.8792

Також можна вивести дані з певних колонок, наприклад:

```
Ввод [27]: fish_parameters[['Species','Width']]
```

Out[27]:

	Species	Width
0	Bream	4.0200
1	Bream	4.3056
2	Bream	4.6961
3	Bream	4.4555
4	Bream	5.1340
...	...	...
154	Smelt	1.3936
155	Smelt	1.2690
156	Smelt	1.2558
157	Smelt	2.0672
158	Smelt	1.8792

159 rows × 2 columns

У наведеному прикладі зовнішні дужки повідомляють pandas, що необхідно вибрати стовпці, а внутрішні вказують список імен стовпців. Порядок імен впливає на результат виводу.
Інколи в проектах аналітичного прогнозування необхідно отримати об’єкти Series разом з DataFrames. Це можна зробити за допомогою одного із способів:

- fish_parameters.Species
- fish_parameters['Species']

```
Ввод [28]: fish_parameters.Species
Out[28]: 0      Bream
         1      Bream
         2      Bream
         3      Bream
         4      Bream
                ...
         154    Smelt
         155    Smelt
         156    Smelt
         157    Smelt
         158    Smelt
         Name: Species, Length: 159, dtype: object
```

```
Ввод [30]: fish_parameters['Species']
Out[30]: 0      Bream
         1      Bream
         2      Bream
         3      Bream
         4      Bream
                ...
         154    Smelt
         155    Smelt
         156    Smelt
         157    Smelt
         158    Smelt
         Name: Species, Length: 159, dtype: object
```

Фільтрація даних виконується у два етапи: на першому кожний рядок отримує буліанівське значення щодо відповідності умові фільтрації, а на наступному - виводяться усі рядки, що отримали значення true. Наприклад:

```
Ввод [32]: fish_parameters.Species == 'Bream'
Out[32]: 0       True
         1       True
         2       True
         3       True
         4       True
                ...
         154    False
         155    False
         156    False
         157    False
         158    False
         Name: Species, Length: 159, dtype: bool
```

```
Ввод [33]: fish_parameters[fish_parameters.Species == 'Bream']
```

Out[33]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340
5	Bream	450.0	26.8	29.7	34.7	13.6024	4.9274
6	Bream	500.0	26.8	29.7	34.5	14.1795	5.2785
7	Bream	390.0	27.6	30.0	35.0	12.6700	4.6900
8	Bream	450.0	27.6	30.0	35.1	14.0049	4.8438
9	Bream	500.0	28.5	30.7	36.2	14.2266	4.9594
10	Bream	475.0	28.4	31.0	36.2	14.2628	5.1042
11	Bream	500.0	28.7	31.0	36.2	14.3714	4.8146
12	Bream	500.0	29.1	31.5	36.4	13.7592	4.3680
13	Bream	340.0	29.5	32.0	37.3	13.9129	5.0728

Функції фільтрації можна поєднувати. Наприклад:

```
Ввод [36]: fish_parameters[fish_parameters.Species == 'Bream'][['Species','Width']]
```

Out[36]:

	Species	Width
0	Bream	4.0200
1	Bream	4.3056
2	Bream	4.6961
3	Bream	4.4555
4	Bream	5.1340
5	Bream	4.9274
6	Bream	5.2785
7	Bream	4.6900
8	Bream	4.8438
9	Bream	4.9594
10	Bream	5.1042
11	Bream	4.8146
12	Bream	4.3680
13	Bream	5.0728
14	Bream	5.1708

У наведеному прикладі виконується первинна фільтрація за Species, а далі виводяться тільки стовпці з назвами Species та Width.

Функція count() дозволяє підрахувати кількість значень у кожному стовпчику:

```
Ввод [37]: fish_parameters.count()
Out[37]: Species    159
         Weight     159
         Length1    159
         Length2    159
         Length3    159
         Height     159
         Width      159
         dtype: int64
```

У такому вигляді ці дані не мають сенсу для більшості стовпців тому, що вони пов’язані, але підходять саме для стовпця Species — підраховують кількість позицій. Тому застосування count() разом з функцією фільтрації дає краще за сенсом результат:

```
fish_parameters[['Species']].count()
Species    159
```

і, якщо потрібно розширити фільтрацію для визначення конкретного типу позицій, наприклад для “Bream”, то можна отримати наступне:

```
fish_parameters[fish_parameters.Species == 'Bream'][['Species']].count()
Species    35
```

Функція sum() дозволяє виконати підрахунок деякого загального показника, а саме його суму. У таблиці прикладу такою величиною може бути “Weight”. Наприклад, для обчислення загальної ваги за усіма позиціями застосовуємо функцію sum() разом із фільтром “Weight”:

```
Ввод [15]: fish_parameters[['Weight']].sum()
Out[15]: Weight    63333.9
```

Також можна розширити фільтрацію для конкретизації підрахунку показника “Weight” за типом позиції у таблиці:

```
fish_parameters[fish_parameters.Species == 'Bream'][['Weight']].sum()
```

```
Weight    21624.0
```

Для знаходження в таблиці позицій з найменшим або найбільшим значенням використовуються функції min() та max():

```
fish_parameters.Weight.min()
```

```
0.0
```

```
fish_parameters.Weight.max()
```

```
1650.0
```

Тепер, якщо потрібно знайти ці позиції можна або прямо задати значення параметру для порівняння

```
Ввод [42]: fish_parameters[fish_parameters.Weight == 0]
```

Out[42]:

	Species	Weight	Length1	Length2	Length3	Height	Width
40	Roach	0.0	19.0	20.5	22.8	6.4752	3.3516

```
Ввод [43]: fish_parameters[fish_parameters.Weight == 1650]
```

Out[43]:

	Species	Weight	Length1	Length2	Length3	Height	Width
144	Pike	1650.0	59.0	63.4	68.0	10.812	7.48

або додати функції min() та max() до фільтрації та порівняння

```
Ввод [44]: fish_parameters[fish_parameters.Weight == fish_parameters.Weight.min()]
```

Out[44]:

	Species	Weight	Length1	Length2	Length3	Height	Width
40	Roach	0.0	19.0	20.5	22.8	6.4752	3.3516

```
Ввод [45]: fish_parameters[fish_parameters.Weight == fish_parameters.Weight.max()]
```

Out[45]:

	Species	Weight	Length1	Length2	Length3	Height	Width
144	Pike	1650.0	59.0	63.4	68.0	10.812	7.48

Для деяких параметрів інколи потрібно визначати середньостатистичні параметри, наприклад, середню величину та медіану. Для цього використовуються функції mean() та median():

```
Ввод [46]: fish_parameters.Weight.mean()
Out[46]: 398.3264150943396

Ввод [47]: fish_parameters.Weight.median()
Out[47]: 273.0

Ввод [48]: fish_parameters.Width.mean()
Out[48]: 4.417485534591194

Ввод [49]: fish_parameters.Width.median()
Out[49]: 4.2485
```

Ці функції корисно поєднувати з певними завданнями, наприклад, знайти усі позиції, що мають значення більше за середнє по показнику “Weight”:

```
fish_parameters[fish_parameters.Weight > fish_parameters.Weight.mean()]
```

	Species	Weight	Length1	Length2	Length3	Height	Width
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340
5	Bream	450.0	26.8	29.7	34.7	13.6024	4.9274
6	Bream	500.0	26.8	29.7	34.5	14.1795	5.2785
8	Bream	450.0	27.6	30.0	35.1	14.0049	4.8438
9	Bream	500.0	28.5	30.7	36.2	14.2266	4.9594
...	...	...	...	...	...	...	...
140	Pike	950.0	48.3	51.7	55.1	8.9262	6.1712
141	Pike	1250.0	52.0	56.0	59.7	10.6863	6.9849
142	Pike	1600.0	56.0	60.0	64.0	9.6000	6.1440
143	Pike	1550.0	56.0	60.0	64.0	9.6000	6.1440
144	Pike	1650.0	59.0	63.4	68.0	10.8120	7.4800

63 rows × 7 columns

При аналізі даних часто приходиться використовувати їх сегментацію, наприклад, групування та агрегацію на основі значень у стовпчиках. Наприклад, знайдемо середню величину для параметру “Width” за типами позицій, для чого застосуємо наступне:

```
Ввод [52]: fish_parameters.groupby('Species').Width.mean()

Out[52]: Species
         Bream        5.427614
         Parkki       3.220736
         Perch        4.745723
         Pike         5.086382
         Roach        3.657850
         Smelt        1.340093
         Whitefish    5.473050
```

Функція groupby('Species') дозволяє згрупувати дані за вказаним стовпчиком, а потім для кожної групи виконати підрахунки за вказаним параметром з використанням заданої функції.
Також корисну інформацію можна отримати при застосуванні групування разом с аналізом мінімальних або максимальних значень за певним параметром:

```
Ввод [53]: fish_parameters.groupby('Species').Weight.min()

Out[53]: Species
         Bream        242.0
         Parkki        55.0
         Perch          5.9
         Pike         200.0
         Roach          0.0
         Smelt          6.7
         Whitefish    270.0
         Name: Weight, dtype: float64

Ввод [54]: fish_parameters.groupby('Species').Weight.max()

Out[54]: Species
         Bream        1000.0
         Parkki        300.0
         Perch        1100.0
         Pike         1650.0
         Roach         390.0
         Smelt          19.9
         Whitefish    1000.0
         Name: Weight, dtype: float64
```

Також слід звернути увагу на тип об’єкту, що повертається при використанні методів. Так, наприклад, якщо застосувати поєднання функцій та методу групування у наступному вигляді

fish_parameters.groupby('Species').min().Weight

то в результаті повертається об’єкт класу Series:

```
Ввод [56]: fish_parameters.groupby('Species').min().Weight
Out[56]: Species
         Bream        242.0
         Parkki        55.0
         Perch          5.9
         Pike         200.0
         Roach          0.0
         Smelt          6.7
         Whitefish    270.0
         Name: Weight, dtype: float64
```

А якщо застосувати для цих даних

```
fish_parameters.groupby('Species').min()[['Weight']]
```

то повертається об’єкт класу DataFrame:

```
Ввод [55]: fish_parameters.groupby('Species').min()[['Weight']]
```

Out[55]:

Species	Weight
Bream	242.0
Parkki	55.0
Perch	5.9
Pike	200.0
Roach	0.0
Smelt	6.7
Whitefish	270.0

Практичні завдання

1. Виконайте інсталяцію необхідних пакетів.
2. Запустіть jupyter notebook та створіть новий Notebook.
3. За своїм варіантом виконайте завантаження даних з Yahoo Finance за останні 10 років:

Варіант	**Біржевий тікер**	**Назва компанії**
1	MSFT	Microsoft
2	GOOG, GOOGL	Alphabet
3	AMZN	Amazon
4	META	Meta
5	TSLA	Tesla
6	NVDA	NVIDIA

4. Отримайте описову статистику ціни закриття акцій для компанії свого варіанту.
5. За своїм варіантом csv-файлу виконайте приклади, що наведено у теоретичній частині лабораторної роботи.
6. Підготуйте звіт з виконання практичних завдань, у якому розмістіть скриншоти виконання прикладів та надайте пояснення до них.