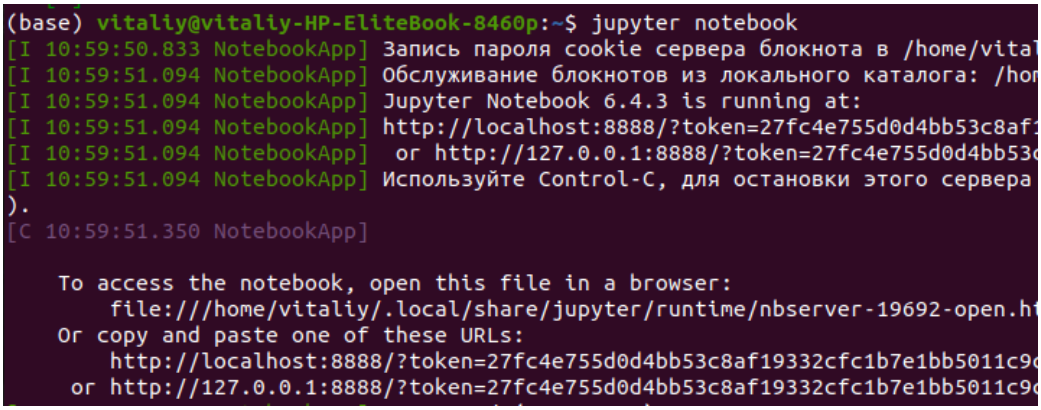
ЛАБОРАТОРНА РОБОТА 1

Тема: Основи роботи з Pandas

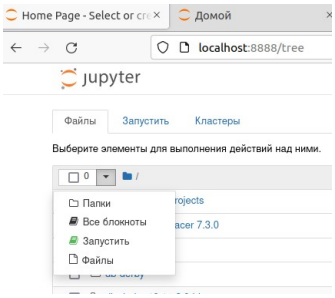
Для виконання лабораторної роботи необхідно інсталювати пакет Python3, а також такі бібліотеки та середовища, як: numpy, scipy, pandas, matplotlib, jupyter та notebook. На платформі Linux, після встановлення Python можна скористатись наступним рядком:

$ pip3 install numpy scipy pandas matplotlib jupyter notebook

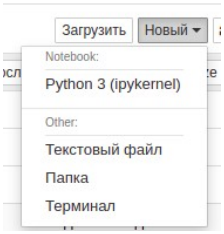
Pandas — це одна із популярних бібліотек Python для аналітики та роботи з даними. Pandas дозволяє працювати з двомірними таблицями даних в Python. Після інсталяції numpy та Pandas можна приступити до роботи, для чого необхідно підключитись до серверу та запустити Jupyter:



Для роботи з Jupyter Notebook можна використати встановлений браузер, наприклад, FireFox або Chrome, для чого до адресного рядка занести адресу, що було вказано при запуску разом з token:



Створюємо новий блокнот та задаємо йому ім’я:



Для подальшої роботи необхідно до створеного Jupyter Notebook додати бібліотеки numpy та pandas

```
Ввод [1]: import numpy as np
          import pandas as pd
```

Тепер до них можна буде звертатись через np та pd.

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 .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: python, numpy scipy pandas matplotlib jupyter notebook.
2. Запустіть jupyter notebook та створіть новий Notebook.
3. За своїм варіантом csv-файлу виконайте приклади, що наведено у теоретичній частині лабораторної роботи.
4. Підготуйте звіт з виконання практичних завдань, у якому розмістіть скриншоти виконання прикладів та надайте пояснення до них.