Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем і технологій

Лабораторна робота №1

Прикладні задачі машинного навчання

“Введення в data science”

Виконав
студент групи ІК – 33:
Вересоцький А. Ю.

Перевірив:
асистент кафедри ІСТ
Нестерук А.О.

Київ 2024

Завдання:

1. На сайті http://www.ukrstat.gov.ua/ обрати дані які для Вас є цікавими. Або можна використати будь-який інший ресурс з відкритими даними, та завантажте дані.

Для виконання лабораторної роботи я обрав дані опитування щодо музичного смаку респондентів та самооцінки їх психічного здоров'я з онлайн ресурсу kaggle. Опитування передбачало оцінку рівня тривожності, депресії, безсоння та ОКР.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```
✓ [173] < 10 ms

```
df = pd.read_csv('survey.csv')
```
✓ [174] 10ms

```
df
```
✓ [175] 11ms

1-10 736 rows × 33 cols

	Timestamp	Age	Primary streaming service	Hours per day	While working	Instrumental
0	8/27/2022 19:29:02	18.0	Spotify	3.0	Yes	Yes
1	8/27/2022 19:57:31	63.0	Pandora	1.5	Yes	No
2	8/27/2022 21:28:18	18.0	Spotify	4.0	No	No
3	8/27/2022 21:40:40	61.0	YouTube Music	2.5	Yes	No
4	8/27/2022 21:54:47	18.0	Spotify	4.0	Yes	No
5	8/27/2022 21:56:50	18.0	Spotify	5.0	Yes	Yes
6	8/27/2022 22:00:29	18.0	YouTube Music	3.0	Yes	Yes
7	8/27/2022 22:18:59	21.0	Spotify	1.0	Yes	No
8	8/27/2022 22:33:05	19.0	Spotify	6.0	Yes	No
9	8/27/2022 22:44:03	18.0	I do not use a streaming service.	1.0	Yes	No

2. Виконати первинну обробку даних.

```
genre_columns = [col for col in df.columns if col.startswith('Frequency')]
df = df.drop(columns=genre_columns)
```
✓ [176] < 10 ms

```
# Перетворення "Yes"/"No" у 1/0
binary_cols = ["While working", "Instrumentalist", "Composer", "Exploratory", "Foreign languages"]
df[binary_cols] = df[binary_cols].map(lambda x: 1 if x == "Yes" else 0)
```
✓ [177] < 10 ms

```
print(df.isnull().sum())     df
```
✓ [178] < 10 ms

```
Age                        1
Primary streaming service  1
Hours per day              0
While working              0
Instrumentalist            0
Composer                   0
Fav genre                  0
Exploratory                0
Foreign languages          0
BPM                      107
Anxiety                    0
Depression                 0
Insomnia                   0
```

```
df = df.dropna()
```
✓ [179] < 10 ms

Видаляємо непотрібні колонки, значення яких ми не будемо аналізувати.
Перетворюємо відповіді “Yes”/”NO” на 1/0 щоб в подальшому легше будувати гістограми.
Також позбавляємось від рядків, в яких присутні пусті значення.

```python
# Діапазони допустимих значень
valid_ranges = {
    "Age": (0, 120),
    "BPM": (1, 400),
    "Hours per day": (0, 24),
    "Anxiety": (0, 10),
    "Depression": (0, 10),
    "Insomnia": (0, 10),
    "OCD": (0, 10)
}

# Фільтрація значень у допустимих межах
for column, (min_val, max_val) in valid_ranges.items():
    df = df[(df[column] >= min_val) & (df[column] <= max_val)]
```

✓ [180] < 10 ms

Перевіряємо, щоб числові значення знаходилися у допустимих для них межах.

3. Знайти математичне сподівання, медіану, моду, дисперсію, середньоквадратичне відхилення (поясніть їх зміст).

```python
# Статистичний аналіз колонок
numeric_columns = ['Age', 'Hours per day', 'BPM', 'Anxiety', 'Depression', 'Insomnia', 'OCD']

# Створюємо словник для зберігання статистик
stats = {}

for column in numeric_columns:
    stats[column] = {
        'Математичне сподівання': df[column].mean(),
        'Медіана': df[column].median(),
        'Мода': df[column].mode().iloc[0],
        'Дисперсія': df[column].var(),
        'Середньоквадратичне відхилення': df[column].std()
    }

# Виводимо статистики
for column, metrics in stats.items():
    print(f"\nСтатистика для {column}:")
    for metric, value in metrics.items():
        print(f"{metric}: {value:.2f}")
```

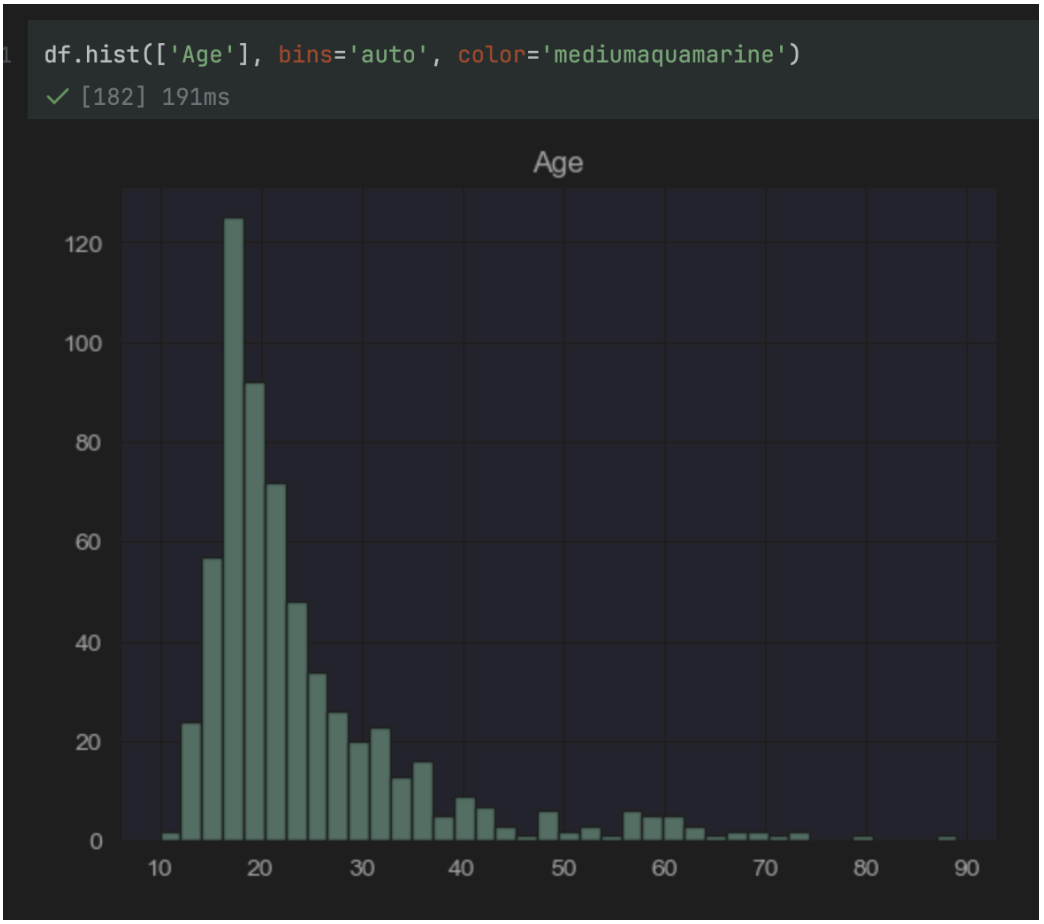
✓ [181] < 10 ms

```
Статистика для Age:
Математичне сподівання: 24.73
Медіана: 21.00
Мода: 18.00
Дисперсія: 135.82
Середньоквадратичне відхилення: 11.65

Статистика для Hours per day:
Математичне сподівання: 3.72
Медіана: 3.00
Мода: 2.00
Дисперсія: 9.63
Середньоквадратичне відхилення: 3.10
```

Обраховуємо значення математичного сподівання, медіани, моди, дисперсії та середньоквадратичного відхилення для всіх числових колонок.

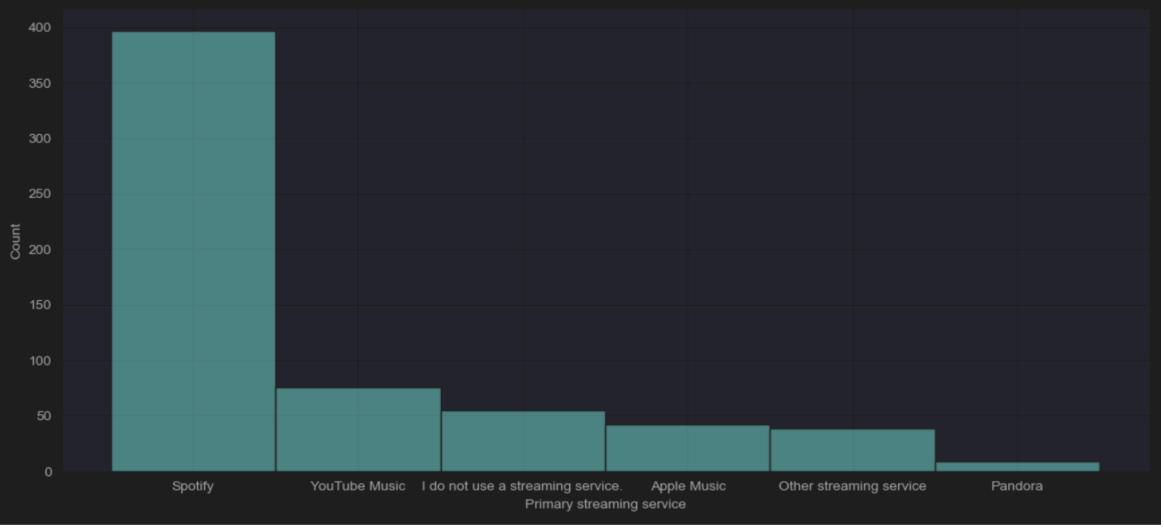
Математичне сподівання представляє собою середнє значення серед набору значень.
Медіана – середнє значення при розташуванні значень в порядку сортування.
Мода – значення, яке найбільш часто зустрічається.
Дисперсія – міра розсіювання даних від середніх значень.
Стандартне відхилення є квадратним коренем дисперсії. Чим менше дисперсія і стандартне відхилення, тим ближче значення даних до математичного сподівання і тим менше загальне «розпорошення»

4. Створити гістограми для всіх можливих параметрів даних.



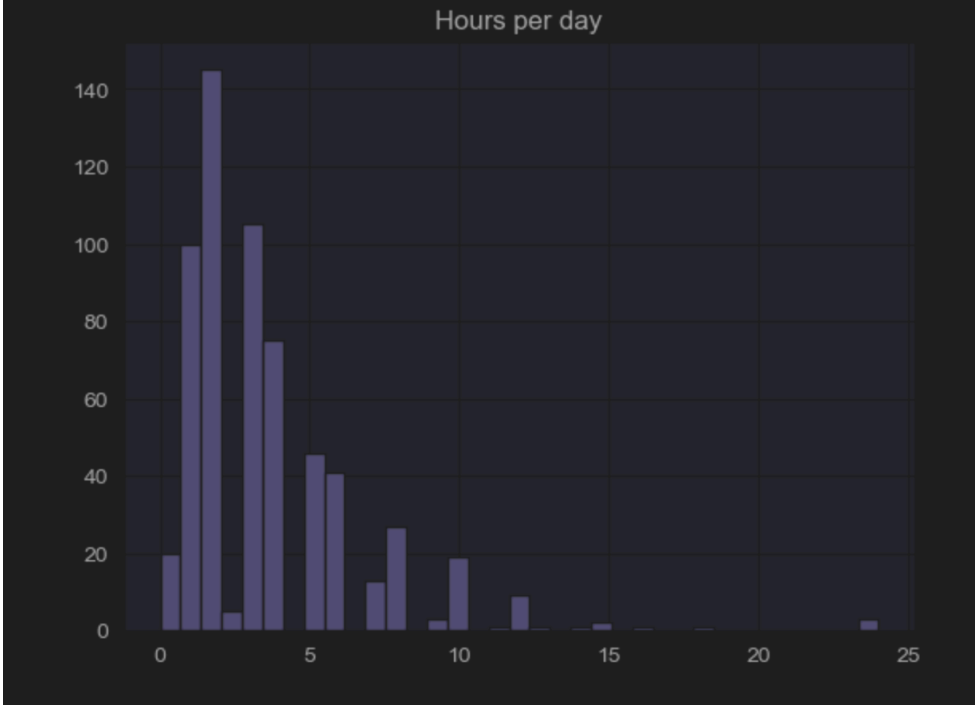
Створюємо гістограму віку респондентів.

```
plt.figure(figsize=(14, 6))
sns.histplot(df['Primary streaming service'], color='darkturquoise')
plt.show()
```
✓ [183] 62ms



Створюємо гістограму, що показує розподіл стримінгових сервісів, якими користуються респонденти.

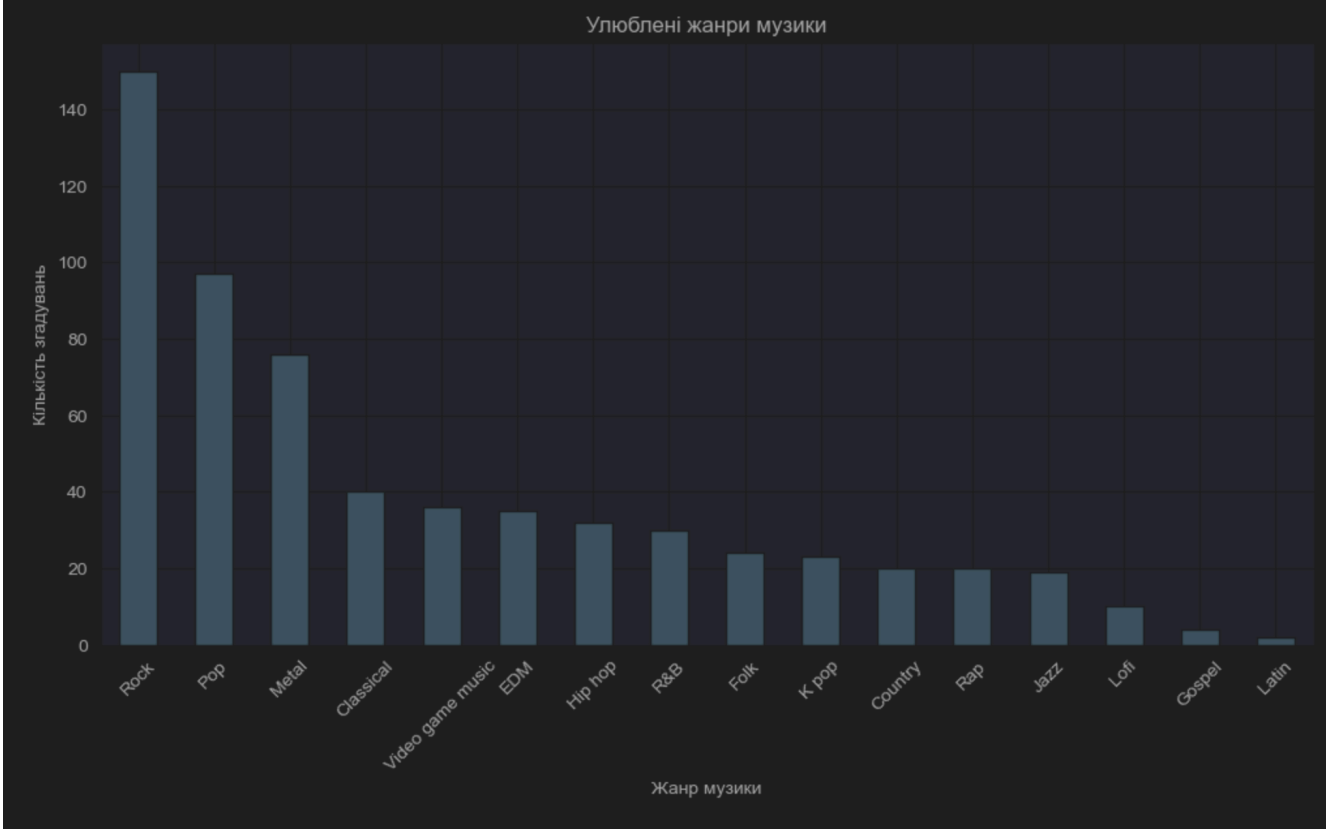
```
df.hist(['Hours per day'], bins='auto', color='slateblue')
```
✓ [184] 62ms



Створюємо гістограму розподілу часу, протягом якого респонденти слухають музику за весь день.

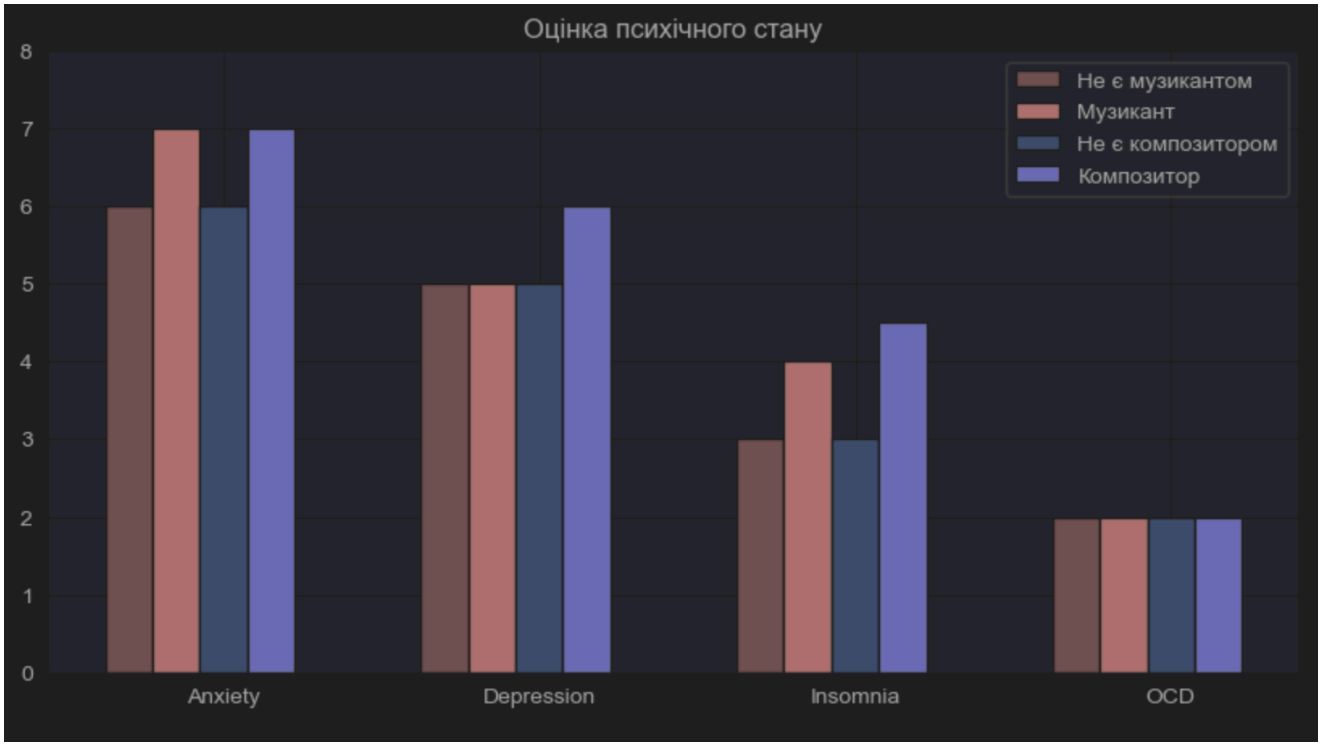
```python
# Побудова гістограми частотності жанрів
plt.figure(figsize=(12, 6))
df["Fav genre"].value_counts().plot(kind="bar", color="skyblue")
plt.xlabel("Жанр музики")
plt.ylabel("Кількість згадувань")
plt.title("Улюблені жанри музики")
plt.xticks(rotation=45)
plt.show()
```

✓ [261] 101ms



Створюємо гістограму улюблених жанрів за кількістю їх згадувань респондентами.

```python
labels = ['Anxiety', 'Depression', 'Insomnia', 'OCD']
x = np.arange(len(labels))
width = 0.15

fig, ax = plt.subplots(figsize=(10, 5))

# Вибираємо тільки числові значення перед підрахунком медіани
num_df = df.select_dtypes(include=[np.number])

b1 = ax.bar(x-2*width, num_df[df.Instrumentalist == 0].median()[-4:], width, color='indianred', label="Не є музикантом")
b2 = ax.bar(x-width, num_df[df.Instrumentalist == 1].median()[-4:], width, color='darkred', label="Музикант")
b3 = ax.bar(x, num_df[df.Composer == 0].median()[-4:], width, color='cornflowerblue', label="Не є композитором")
b4 = ax.bar(x+width, num_df[df.Composer == 1].median()[-4:], width, color='darkblue', label="Композитор")

ax.set_ylim([0, 8])
ax.set_title('Оцінка психічного стану')
ax.set_xticks(x, labels)
ax.legend()

plt.show()
✓ [186] 64ms
```



Створюємо гістограму, що групує людей за тим, чи вони грають на музичних інструментах та чи є композиторами та визначає медіану їхніх оцінок психічного здоров'я.

5. Для цих даних проробити всі дії з пункту колекції Series і DataFrame бібліотеки pandas.

Створення Series з індексами за замовчуванням

```
1  age = pd.Series(df.Age.head(10).values)
   ✓ [187] < 10 ms
```

Виведення колекції Series

```
1  age
   ✓ [188] < 10 ms
```

10 rows Length: 10, dtype: float64

	<unnamed>
0	18.0
1	61.0
2	18.0
3	18.0
4	18.0
5	21.0
6	19.0
7	18.0
8	19.0
9	19.0

Створення колекції Series з однаковими значеннями

```
1  pd.Series(19.0, range(3))
   ✓ [189] < 10 ms
```

3 rows Length: 3, dtype: float64

	<unnamed>
0	19.0
1	19.0
2	19.0

Звернення до елементів Series

```
1  age[0]
   ✓ [190] < 10 ms
   np.float64(18.0)
```

Обчислення описових статистик для Series

```
1 age.count()
✓ [191] < 10 ms
  np.int64(10)

1 age.mean()
✓ [192] < 10 ms
  np.float64(22.9)

1 age.min()
✓ [193] < 10 ms
  np.float64(18.0)

1 age.max()
✓ [194] < 10 ms
  np.float64(61.0)

1 age.std()
✓ [195] < 10 ms
  np.float64(13.42013412749664)
```

```
1 age.describe()
✓ [196] < 10 ms
```

8 rows · Length: 8, dtype: float64

	123 <unnamed>
count	10.0
mean	22.9
std	13.42013412749664
min	18.0
25%	18.0
50%	18.5
75%	19.0
max	61.0

Створення колекції Series з нестандартними індексами

Для призначення нестандартних індексів використовується ключовий аргумент index:

```
age = pd.Series(df.Age.reset_index(drop=True).loc[10:14].values,
 index=['Miles', 'Kevin', 'Molly', 'Kylie', 'Mark'])
✓ [197] < 10 ms
```

```
age
✓ [198] < 10 ms
```

5 rows Length: 5, dtype: float64

	<unnamed>
Miles	18.0
Kevin	16.0
Molly	16.0
Kylie	17.0
Mark	15.0

Словники як ініціалізатор

Якщо колекція Series ініціалізується словником, то ключі стають індексами Series, а значення - значеннями елементів Series:

```
age = pd.Series({'Miles': 18, 'Kevin': 16, 'Molly': 16, 'Kylie': 17,
 'Mark': 15})
✓ [199] < 10 ms
```

```
age
✓ [200] < 10 ms
```

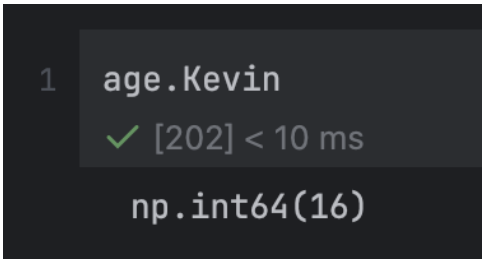
5 rows Length: 5, dtype: int64

	<unnamed>
Miles	18
Kevin	16
Molly	16
Kylie	17
Mark	15

Звернення до елементів Series з використанням нестандартних індексів

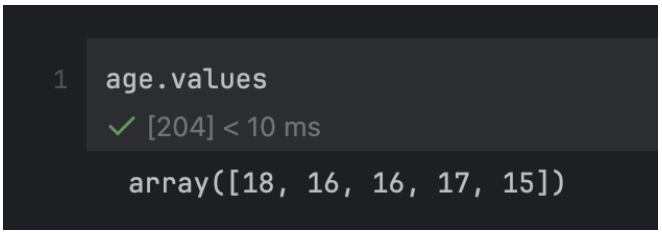
Вказуємо значення нестандартного індексу в квадратних дужках:

Також до значень можна звертатись через точку, адже елементи автоматично стають атрибутами словнику



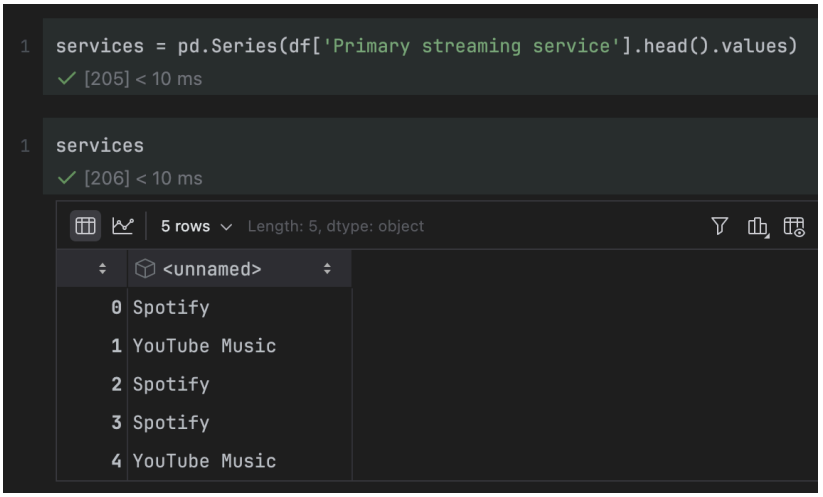
Колекція Series також містить вбудовані атрибути. Наприклад, атрибут dtype повертає тип елемента базової колекції array:

Атрибут values повертає базову колекцію array:

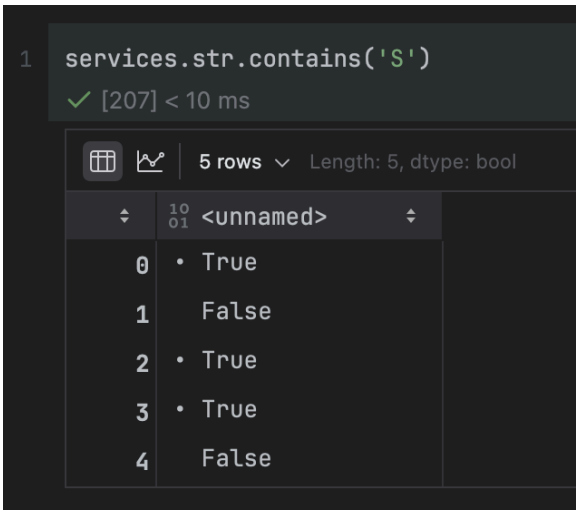


Створення колекції Series із строковими елементами

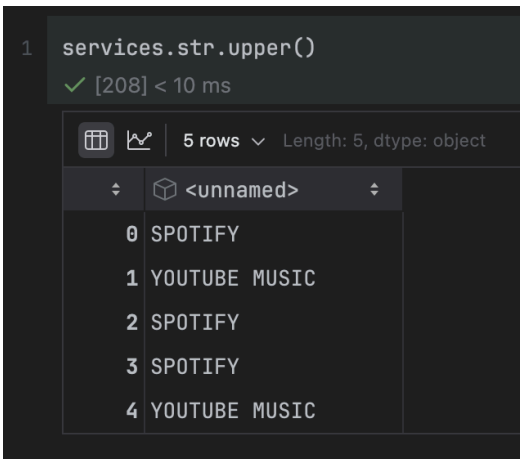
Спочатку створимо колекцію Series з рядками:



Тепер викличемо метод contains для кожного елемента, щоб визначити, чи містять елементи колекції букву 'S' верхнього регістра:



При виклику методу upper, усі елементи колекції змінюють свій регістр на верхній



Створення DataFrame на базі словника

```
mental_state_dict = {'Miles': [6, 4, 7, 0], 'Kevin': [8, 8, 4, 3],
 'Molly': [5, 7, 10, 0], 'Kylie': [7, 5, 0, 3], 'Mark': [7, 3, 0, 2]}
[209]
```

```
mental_state = pd.DataFrame(mental_state_dict)
[210]
```

```
mental_state
[211]
```

4 rows · 4 rows × 5 cols · Static Output

	Miles	Kevin	Molly	Kylie	Mark
0	6	8	5	7	7
1	4	8	7	5	3
2	7	4	10	0	0
3	0	3	0	3	2

Налаштування індексів DataFrame з використанням атрибута index

```
pd.DataFrame(mental_state_dict, index=['Anxiety', 'Depression',
 'Insomnia', 'OCD'])
✓ [42] < 10 ms
```

4 rows · 4 rows × 5 cols · Static Output

	Miles	Kevin	Molly	Kylie	Mark
Anxiety	6	8	5	7	7
Depression	4	8	7	5	3
Insomnia	7	4	10	0	0
OCD	0	3	0	3	2

```
mental_state.index = ['Anxiety', 'Depression', 'Insomnia', 'OCD']
✓ [43] 12ms
```

```
mental_state
✓ [44] < 10 ms
```

4 rows · 4 rows × 5 cols · Static Output

	Miles	Kevin	Molly	Kylie	Mark
Anxiety	6	8	5	7	7
Depression	4	8	7	5	3
Insomnia	7	4	10	0	0
OCD	0	3	0	3	2

Звернення до стовпців DataFrame

```
mental_state['Molly']
[214]

Anxiety        5
Depression     7
Insomnia      10
OCD            0
Name: Molly, dtype: int64
```

Аналогічно до колекції Series, значення вважаються атрибутами, тому:

```
mental_state.Kevin
✓ [133] < 10 ms
```

4 rows Length: 4, dtype: int64

	Kevin
Anxiety	8
Depression	8
Insomnia	4
OCD	3

Вибір рядків з використанням атрибутів loc і iloc

Атрибут loc використовується для звернення до рядків за текстовою міткою (значенням)

```
mental_state.loc['Anxiety']
✓ [134] < 10 ms
```

5 rows Length: 5, dtype: int64

	Anxiety
Miles	6
Kevin	8
Molly	5
Kylie	7
Mark	7

До рядків також можна звертатися по цілочисельним індексам, що починається з 0, за допомогою атрибута iloc

```
mental_state.iloc[1]
```
✓ [135] < 10 ms

5 rows Length: 5, dtype: int64

	Depression
Miles	4
Kevin	8
Molly	7
Kylie	5
Mark	3

Вибір рядків з використанням атрибутів loc і iloc

При використанні з атрибутом loc сегментів, що містять мітки, заданий діапазон включає верхній індекс

```
mental_state.loc['Anxiety':'Insomnia']
```
✓ [136] < 10 ms

3 rows 3 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Anxiety	6	8	5	7	7
Depression	4	8	7	5	3
Insomnia	7	4	10	0	0

При використанні з атрибутом iloc сегментів, що містять цілочисельні індекси, заданий діапазон не включає верхній індекс [1; 4)

```
mental_state.iloc[1:4]
```
✓ [137] < 10 ms

3 rows 3 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Depression	4	8	7	5	3
Insomnia	7	4	10	0	0
OCD	0	3	0	3	2

Щоб вибрати конкретні рядки, використовуємо синтаксис списку в поєднанні з loc або iloc:

```
mental_state.loc[['Anxiety', 'Insomnia']]
```
✓ [138] < 10 ms

2 rows · 2 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Anxiety	6	8	5	7	7
Insomnia	7	4	10	0	0

```
mental_state.iloc[[1, 3]]
```
✓ [139] < 10 ms

2 rows · 2 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Depression	4	8	7	5	3
OCD	0	3	0	3	2

Вибір підмножин рядків і стовпців

Вибираємо показники рядків починаючи з депресії закінчуючи ОКР тільки для Кевіна та Моллі:

```
mental_state.loc['Depression':'OCD', ['Kevin', 'Molly']]
```
✓ [140] < 10 ms

3 rows · 3 rows × 2 cols

	Kevin	Molly
Depression	8	7
Insomnia	4	10
OCD	3	0

Вибираємо тільки показники тривожності та ОКР для всіх починаючи від Моллі закінчуючи Кайлі:

```
mental_state.iloc[[0, 3], 2:4]
```
✓ [141] < 10 ms

2 rows · 2 rows × 2 cols

	Molly	Kylie
Anxiety	5	7
OCD	0	3

Логічне індексування

Pandas перевіряє кожне значення в датафреймі ментального стану і, якщо воно більше або дорівнює 5, включає в нову колекцію DataFrame.

```
mental_state[mental_state >= 5]
✓ [142] < 10 ms
```

4 rows × 4 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Anxiety	6.0	8.0	5.0	7.0	7.0
Depression	NaN	8.0	7.0	5.0	NaN
Insomnia	7.0	NaN	10.0	NaN	NaN
OCD	NaN	NaN	NaN	NaN	NaN

Звернення до конкретного осередку DataFrame по рядку і стовпцю

Атрибути at і iat колекції DataFrame можуть використовуватися для отримання окремого значення з DataFrame. Як і loc і iloc, атрибут at використовує мітки, а iat - цілочисельні індекси.

```
mental_state.at['Anxiety', 'Molly']
✓ [143] < 10 ms
  np.int64(5)

mental_state.iat[0, 4]
✓ [144] < 10 ms
  np.int64(7)
```

Також можна присвоїти чергові значення конкретних елементів.

```
mental_state.at['Anxiety', 'Molly'] = 4
✓ [145] < 10 ms

mental_state.at['Anxiety', 'Molly']
✓ [146] < 10 ms
  np.int64(4)

mental_state.iat[0, 4] = 8
✓ [147] < 10 ms

mental_state.iat[0, 4]
✓ [148] < 10 ms
  np.int64(8)
```

Описова статистика

Колекції Series і DataFrame містять метод describe, який обчислює основні характеристики описової статистики для даних і повертає їх у формі DataFrame.

```
mental_state.describe()
```
✓ [149] < 10 ms

8 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
count	4.0	4.0	4.0	4.0	4.0
mean	4.25	5.75	5.25	3.75	3.25
std	3.095695936834452	2.6299556396765835	4.272001872658765	2.9860788111948193	3.4034296427770228
min	0.0	3.0	0.0	0.0	0.0
25%	3.0	3.75	3.0	2.25	1.5
50%	5.0	6.0	5.5	4.0	2.5
75%	6.25	8.0	7.75	5.5	4.25
max	7.0	8.0	10.0	7.0	8.0

Для управління точністю та іншими настройками за замовчуванням може використовуватися функція pandas set_option:

```
pd.set_option('display.precision', 2)
```
✓ [150] < 10 ms

```
mental_state.describe()
```
✓ [151] < 10 ms

8 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
count	4.0	4.0	4.0	4.0	4.0
mean	4.25	5.75	5.25	3.75	3.25
std	3.095695936834452	2.6299556396765835	4.272001872658765	2.9860788111948193	3.4034296427770228
min	0.0	3.0	0.0	0.0	0.0
25%	3.0	3.75	3.0	2.25	1.5
50%	5.0	6.0	5.5	4.0	2.5
75%	6.25	8.0	7.75	5.5	4.25
max	7.0	8.0	10.0	7.0	8.0

Ймовірно, для оцінок найважливішою характеристикою є математичне сподівання.

```
mental_state.mean()
```
✓ [152] < 10 ms

5 rows Length: 5, dtype: float64

	<unnamed>
Miles	4.25
Kevin	5.75
Molly	5.25
Kylie	3.75
Mark	3.25

Транспонування DataFrame з використанням атрибута T

Щоб швидко транспонувати DataFrame, тобто поміняти місцями рядки і стовпці, можна скористатися атрибутом T:

```
mental_state.T
✓ [153] < 10 ms
```

5 rows 5 rows × 4 cols

	Anxiety	Depression	Insomnia	OCD
Miles	6	4	7	0
Kevin	8	8	4	3
Molly	4	7	10	0
Kylie	7	5	0	3
Mark	8	3	0	2

Припустимо, замість того щоб обчислити зведену статистику по респондентам, ви хочете отримати її за психічними розладами:

```
mental_state.T.describe()
✓ [154] < 10 ms
```

8 rows 8 rows × 4 cols

	Anxiety	Depression	Insomnia	OCD
count	5.0	5.0	5.0	5.0
mean	6.6	5.4	4.2	1.6
std	1.6733200530681511	2.073644135332772	4.381780460041329	1.5165750888103102
min	4.0	3.0	0.0	0.0
25%	6.0	4.0	0.0	0.0
50%	7.0	5.0	4.0	2.0
75%	8.0	7.0	7.0	3.0
max	8.0	8.0	10.0	3.0

Щоб переглянути середні оцінки всіх респондентів кожного психічного розладу, викличте mean для атрибута T:

```
mental_state.T.mean()
✓ [155] < 10 ms
```

4 rows Length: 4, dtype: float64

	<unnamed>
Anxiety	6.6
Depression	5.4
Insomnia	4.2
OCD	1.6

Сортування рядків за індексами

Відсортуємо рядки колекції. За замовчуванням сортування виконується по зростанню.

```
mental_state.sort_index(ascending=False)
```
✓ [156] < 10 ms

4 rows × 4 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
OCD	0	3	0	3	2
Insomnia	7	4	10	0	0
Depression	4	8	7	5	3
Anxiety	6	8	4	7	8

Сортування за індексами стовпців

Тепер відсортуємо стовпці по зростанню (зліва направо) по назвам стовпців.

```
mental_state.sort_index(axis=1)
```
✓ [157] < 10 ms

4 rows × 4 rows × 5 cols

	Kevin	Kylie	Mark	Miles	Molly
Anxiety	8	7	8	6	4
Depression	8	5	3	4	7
Insomnia	4	0	0	7	10
OCD	3	3	2	0	0

Сортування за значеннями стовпців

Припустимо, ви хочете переглянути оцінки безсоння за спаданням, щоб імена респондентів слідували від найбільшої оцінки до найменшої. Виклик методу sort_values виглядає так:

```
mental_state.sort_values(by='Insomnia', axis=1, ascending=False)
```
✓ [158] < 10 ms

4 rows × 4 rows × 5 cols

	Molly	Miles	Kevin	Kylie	Mark
Anxiety	4	6	8	7	8
Depression	7	4	8	5	3
Insomnia	10	7	4	0	0
OCD	0	0	3	3	2

Можливо, імена респондентів і оцінки стане зручніше читати при виведенні в стовпець, так що впорядкувати можна і транспоновану колекцію DataFrame.

```
mental_state.T.sort_values(by='Anxiety', ascending=False)
```
✓ [159] < 10 ms

5 rows · 5 rows × 4 cols

	Anxiety	Depression	Insomnia	OCD
Kevin	8	8	4	3
Mark	8	3	0	2
Kylie	7	5	0	3
Miles	6	4	7	0
Molly	4	7	10	0

Не виключено, що результати інших тестів окрім ОКР виводити взагалі не потрібно. Об'єднаймо операцію вибору з сортуванням:

```
mental_state.loc['OCD'].sort_values(ascending=False)
```
✓ [160] < 10 ms

5 rows · Length: 5, dtype: int64

	OCD
Kevin	3
Kylie	3
Mark	2
Miles	0
Molly	0

Копіювання і сортування на місці

За замовчуванням sort_index і sort_values повертають копію вихідної колекції DataFrame. DataFrame також можна впорядкувати на місці. Для цього передайте ключовий аргумент inplace = True функції sort_index або sort_values.

```
mental_state
```
✓ [161] < 10 ms

4 rows · 4 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Insomnia	7	4	10	0	0
Anxiety	6	8	4	7	8
Depression	4	8	7	5	3
OCD	0	3	0	3	2

```
mental_state.sort_values(by='Miles', ascending=False, inplace=True)
```
✓ [162] < 10 ms

```
mental_state
```
✓ [163] < 10 ms

4 rows · 4 rows × 5 cols

	Miles	Kevin	Molly	Kylie	Mark
Insomnia	7	4	10	0	0
Anxiety	6	8	4	7	8
Depression	4	8	7	5	3
OCD	0	3	0	3	2

6. Прочитати набір даних катастрофи «Титаніка»

Набір даних катастрофи «Титаніка» належить до числа найпопулярніших наборів даних машинного навчання і доступний в багатьох форматах, включаючи CSV: https://vincentarelbundock.github.io/Rdatasets/datasets.html

7. Завантажити набір даних катастрофи «Титаніка» за URL-адресою

```
titanic = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/carData/TitanicSurvival.csv')
✓ [164] 148ms
```

8. Переглянути рядки набору даних катастрофи «Титаніка»

Для економії місця переглянемо перші і останні п'ять рядків за допомогою методів head і tail колекції DataFrame.

```
pd.set_option('display.precision', 2)
✓ [165] < 10 ms
```

```
titanic.head()
✓ [166] < 10 ms
```

5 rows 5 rows × 5 cols

	rownames	survived	sex	age	passengerClass
0	Allen, Miss. Elisabeth W…	yes	female	29.0	1st
1	Allison, Master. Hudson …	yes	male	0.916700006	1st
2	Allison, Miss. Helen Lor…	no	female	2.0	1st
3	Allison, Mr. Hudson Josh…	no	male	30.0	1st
4	Allison, Mrs. Hudson J C…	no	female	25.0	1st

```
titanic.tail()
✓ [167] < 10 ms
```

5 rows 5 rows × 5 cols

	rownames	survived	sex	age	passengerClass
1304	Zabour, Miss. Hileni	no	female	14.5	3rd
1305	Zabour, Miss. Thamine	no	female	NaN	3rd
1306	Zakarian, Mr. Mapriededer	no	male	26.5	3rd
1307	Zakarian, Mr. Ortin	no	male	27.0	3rd
1308	Zimmerman, Mr. Leo	no	male	29.0	3rd

9. Налаштувати назви стовпців

Назва першого стовпчика в наборі даних виглядає досить дивно ('Unnamed: 0'). Цю проблему можна вирішити налаштуванням імен стовпців. Замінімо 'Unnamed: 0' на 'name' і скоротіть 'passengerClass' до 'class':

```
titanic.columns = ['name', 'survived', 'sex', 'age', 'class']
✓ [168] < 10 ms
```

```
titanic.head()
✓ [169] < 10 ms
```

5 rows · 5 rows × 5 cols

	name	survived	sex	age	class
0	Allen, Miss. Elisabeth W…	yes	female	29.0	1st
1	Allison, Master. Hudson …	yes	male	0.916700006	1st
2	Allison, Miss. Helen Lor…	no	female	2.0	1st
3	Allison, Mr. Hudson Josh…	no	male	30.0	1st
4	Allison, Mrs. Hudson J C…	no	female	25.0	1st

10. Провести простий аналіз даних

Визначимо наймолодшого пасажира, найстаршого та який був середній вік пасажирів.

```
titanic.loc[titanic.age == titanic.age.min()]
✓ [170] < 10 ms
```

1 row · 1 rows × 5 cols

	name	survived	sex	age	class
763	Dean, Miss. Elizabeth Gl…	yes	female	0.166700006	3rd

```
titanic.loc[titanic.age == titanic.age.max()]
✓ [171] < 10 ms
```

1 row · 1 rows × 5 cols

	name	survived	sex	age	class
14	Barkworth, Mr. Algernon …	yes	male	80.0	1st

```
titanic.age.mean()
✓ [172] < 10 ms
np.float64(29.881134512434034)
```

Статистику по пасажирам які вижили:

```
titanic[titanic.survived == 'yes'].describe()
```

✓ [173] < 10 ms

8 rows · 8 rows × 1 cols

	age
count	427.0
mean	28.918228103072597
std	15.061481385020018
min	0.166700006
25%	20.0
50%	28.0
75%	38.0
max	80.0

Відсортуймо всіх жінок з кают 1-го класу, знайдемо наймолодшу та найстаршу серед них.

```
women_first_class = titanic[(titanic["sex"] == "female") & (titanic["class"] == "1st")]
```

✓ [174] < 10 ms

```
women_first_class.loc[women_first_class.age == women_first_class.age.min()]
```

✓ [175] < 10 ms

1 row · 1 rows × 5 cols

	name	survived	sex	age	class
2	Allison, Miss. Helen Lor…	no	female	2.0	1st

```
women_first_class.loc[women_first_class.age == women_first_class.age.max()]
```

✓ [176] < 10 ms

1 row · 1 rows × 5 cols

	name	survived	sex	age	class
61	Cavendish, Mrs. Tyrell W…	yes	female	76.0	1st

Кількість вижившних серед жінок першого класу:

```
women_first_class[women_first_class["survived"] == "yes"].shape[0]
```

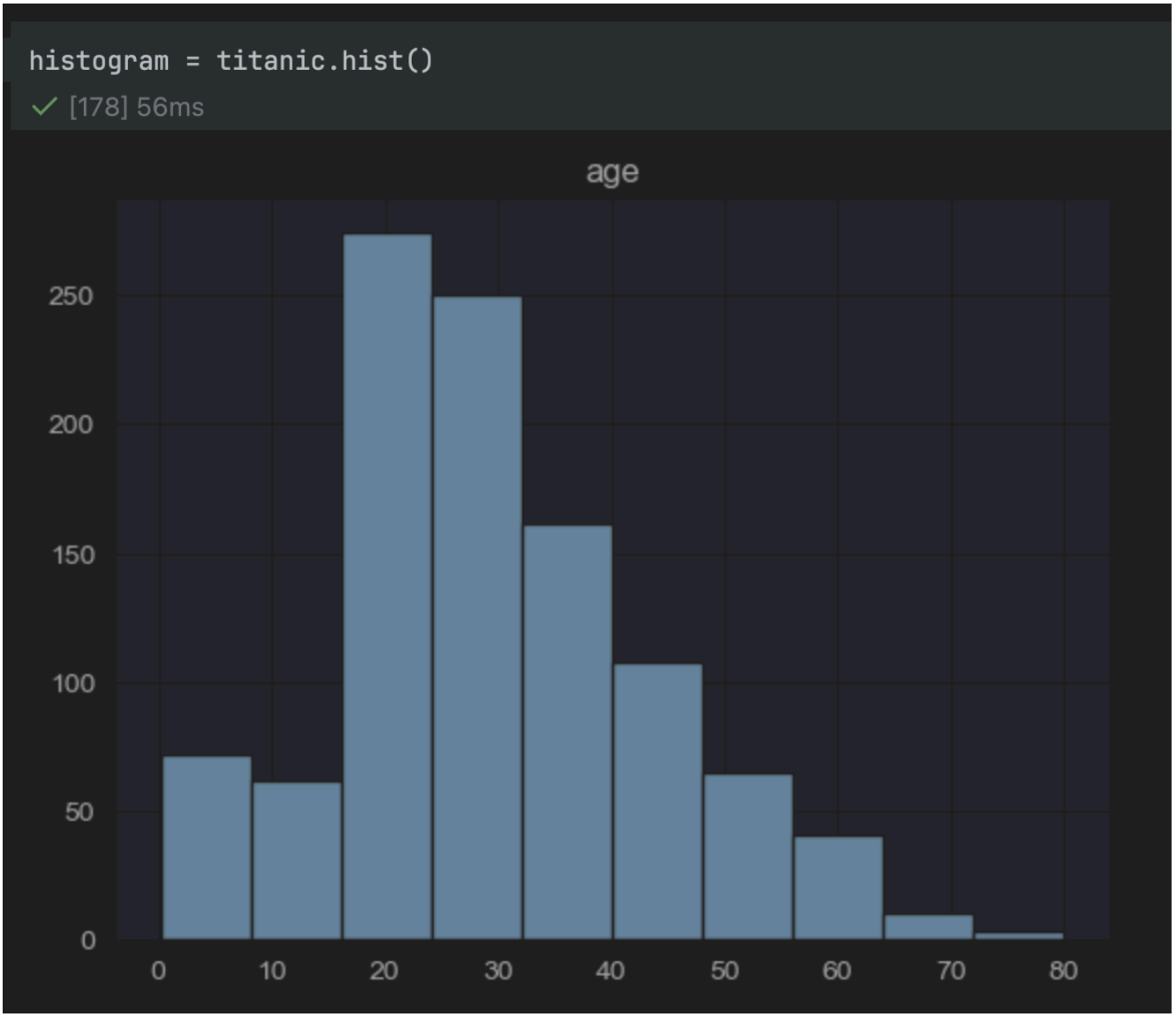
✓ [177] < 10 ms

```
139
```

11. Побудувати гістограму віку пасажирів

Щоб переглянути гістограми по кожному числовому стовпці даних, можемо викликати **hist** для своєї колекції DataFrame.

Оскільки набір даних катастрофи «Титаніка» містить тільки один числовий стовпець даних, тому побудуймо діаграму, що покаже гістограму для розподілу вікових груп:



Висновок: Під час виконання лабораторної роботи я ознайомився з широким спектром функцій та можливостей бібліотек *statistics* та *pandas*. Особливу увагу приділив роботі з колекціями Series та DataFrame, детально вивчивши методи для їхнього опрацювання. Навчився переглядати, аналізувати й змінювати дані, використовуючи різні вбудовані функції. Окрім цього, дізнався більше про форматування даних, що значно покращило моє розуміння принципів роботи з цими колекціями.