НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Лабораторна робота № 1

з дисципліни «Прикладні задачі машинного навчання»

Тема: «Введення в data science»

Перевірив:
Нестерук А. О.

Виконав:
Ал Хадам М. Р.

Київ – 2023

Завдання:

1. **На сайті http://www.ukrstat.gov.ua/ обрати дані які для Вас є цікавими, можна використати будь-який ресурс з відкритими даними.**
2. **Знайти математичне сподівання, медіану, моду, дисперсію, середньоквадратичне відхилення (поясніть їх зміст)**
3. **Візуалізувати завантажені дані за допомогою гістограми**
4. **Для цих даних проробити всі дії з пункту колекції Series і DataFrame бібліотеки pandas**
5. **Прочитати набір даних катастрофи «Титаніка»**
6. **Завантажити набір даних катастрофи «Титаніка» за URL-адресою**
7. **Переглянути рядки набору даних катастрофи «Титаніка»**
8. **Налаштувати назви стовпців**
9. **Провести простий аналіз даних**
10. **Побудувати гістограму віку пасажирів**

Виконання:

1.

При виконанні роботи було використано набір даних, присвячений загальної кількості зароблених грошей різних кіберспортивних спортсменів в різних дисциплінах. Імпортуємо бібліотеку Pandas та завантажимо дані:

```
In 27
import pandas as pd

In 28
df_teams = pd.read_csv('assets/highest_earning_teams.csv')
df_players = pd.read_csv('assets/highest_earning_players.csv')

In 29
df_teams.head()
```

Out 29 — 5 rows × 6 columns

	TeamId	TeamName	TotalUSDPrize	TotalTournaments	Game	Genre
0	760	San Francisco Shock	3.10e+06	7	Overwatch	First-Person Shooter
1	776	London Spitfire	1.59e+06	13	Overwatch	First-Person Shooter
2	768	New York Excelsior	1.57e+06	18	Overwatch	First-Person Shooter
3	773	Philadelphia Fusion	1.19e+06	15	Overwatch	First-Person Shooter
4	766	Seoul Dynasty	1.13e+06	6	Overwatch	First-Person Shooter

```
In 30
df_players.head()
```

Out 30 — 5 rows × 8 columns

	PlayerId	NameFirst	NameLast	CurrentHandle	CountryCode	TotalUSDPrize	Game	Genre
0	3883	Peter	Rasmussen	dupreeh	dk	1.82e+06	Counter-Strike: Global Offensive	First-Person Shooter
1	3679	Andreas	Højsleth	Xyp9x	dk	1.80e+06	Counter-Strike: Global Offensive	First-Person Shooter
2	3885	Nicolai	Reedtz	dev1ce	dk	1.79e+06	Counter-Strike: Global Offensive	First-Person Shooter
3	3672	Lukas	Rossander	gla1ve	dk	1.65e+06	Counter-Strike: Global Offensive	First-Person Shooter
4	17800	Emil	Reif	Magisk	dk	1.42e+06	Counter-Strike: Global Offensive	First-Person Shooter

Виконаємо первинну обробку даних (видалимо з кожного датасету по непотрібному стовпцю зовнішніх ключів):

```
In 122
df_teams = df_teams.drop('TeamId', axis=1)
df_teams
```

Out 122 — 1-10 of 928 rows × 5 columns

	TeamName	TotalUSDPrize	TotalTournaments	Game	Genre
0	San Francisco Shock	3.10e+06	7	Overwatch	First-Person Shooter
1	London Spitfire	1.59e+06	13	Overwatch	First-Person Shooter
2	New York Excelsior	1.57e+06	18	Overwatch	First-Person Shooter
3	Philadelphia Fusion	1.19e+06	15	Overwatch	First-Person Shooter
4	Seoul Dynasty	1.13e+06	6	Overwatch	First-Person Shooter
5	Vancouver Titans	9.50e+05	4	Overwatch	First-Person Shooter
6	Shanghai Dragons	7.55e+05	5	Overwatch	First-Person Shooter
7	Los Angeles Gladiators	7.10e+05	13	Overwatch	First-Person Shooter
8	Atlanta Reign	5.96e+05	9	Overwatch	First-Person Shooter
9	Los Angeles Valiant	5.35e+05	6	Overwatch	First-Person Shooter

```
In 123
df_players = df_players.drop('PlayerId', axis=1)
df_players
```

Out 123 — 1-10 of 1000 rows × 7 columns

	NameFirst	NameLast	CurrentHandle	CountryCode	TotalUSDPrize	Game	Genre
0	Peter	Rasmussen	dupreeh	dk	1.82e+06	Counter-Strike: Global Offensive	First-Person Shooter
1	Andreas	Højsleth	Xyp9x	dk	1.80e+06	Counter-Strike: Global Offensive	First-Person Shooter
2	Nicolai	Reedtz	dev1ce	dk	1.79e+06	Counter-Strike: Global Offensive	First-Person Shooter
3	Lukas	Rossander	gla1ve	dk	1.65e+06	Counter-Strike: Global Offensive	First-Person Shooter
4	Emil	Reif	Magisk	dk	1.42e+06	Counter-Strike: Global Offensive	First-Person Shooter
5	Jakey	Yip	Stewie2k	us	1.09e+06	Counter-Strike: Global Offensive	First-Person Shooter
6	Epitácio	de Melo	TACO	br	1.06e+06	Counter-Strike: Global Offensive	First-Person Shooter
7	Fernando	Alvarenga	fer	br	1.06e+06	Counter-Strike: Global Offensive	First-Person Shooter
8	Gabriel	Toledo	FalleN	br	1.06e+06	Counter-Strike: Global Offensive	First-Person Shooter
9	Marcelo	David	coldzera	br	1.02e+06	Counter-Strike: Global Offensive	First-Person Shooter

2.

Знайдемо математичне сподівання загальної кількості турнірів у певної команди:

```
In 145  1  round(df_teams['TotalTournaments'].mean(), 4)

Out 145     31.6961
```

Знайдемо медіану (значення, яке розділяє масив загальної кількості турнірів для певної команди на дві рівні частини: половина значень масиву більша за медіану, а інша половина – менша):

```
In 178  1  df_teams['TotalTournaments'].median()

Out 178     11.0
```

Знайдемо моду (це значення, яке зустрічається найбільшу кількість разів у масиві даних):

```
In 179  1  df_teams['TotalTournaments'].mode()

Out 179  |< < 1 row > >|  Length: 1, dtype: int64
```

	TotalTournaments
0	1

Для знаходження дисперсії та середньоквадратичного відхилення використаємо модуль statistics та знайдемо дисперсію (міра розсіяння числових даних у вибірці):

```
In 181  1  import statistics
        2
        3  # variance
        4  round(statistics.pvariance(df_teams['TotalTournaments']), 4)

Out 181     3726.2396
```

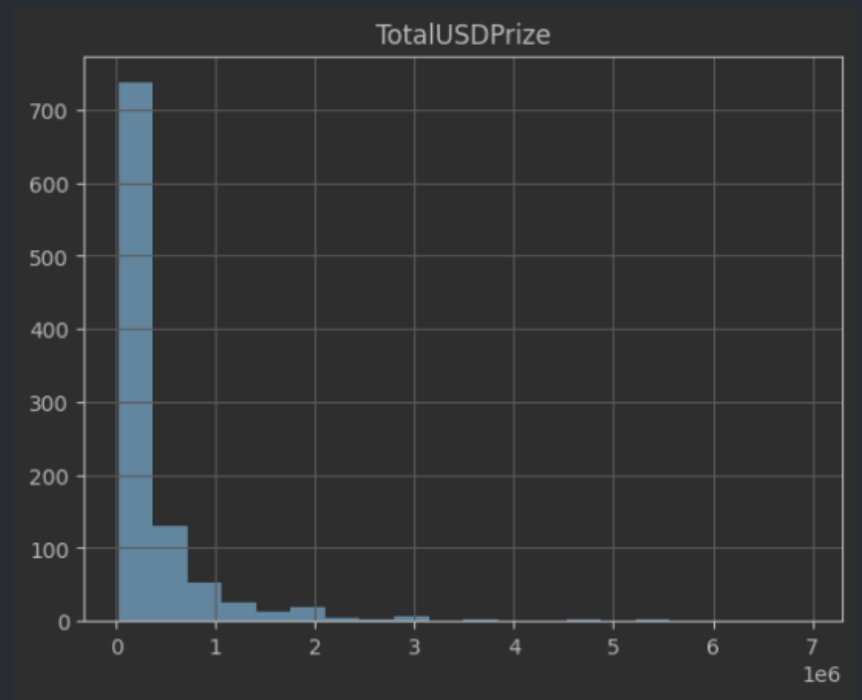
Знайдемо середньоквадратичне відхилення (квадратний корінь з вище знайденної дисперсії):

```
In 182  1  # standard deviation
        2  round(statistics.pstdev(df_teams['TotalTournaments']), 4)

Out 182    61.0429
```

3.

Візуалізуємо завантажені дані за допомогою гістограми:

```
In 42   1  df_players.hist(bins=20)

Out 42     array([[<AxesSubplot: title={'center': 'TotalUSDPrize'}>]], dtype=object)
```



4. Виконаємо усі дії з пункту колекції Series і DataFrame бібліотеки pandas:

Створимо Series, використавши дані зі стовпця ‘total_prize’ попереднього датафрейму:

```
In 35  1  total_prize = df_players['TotalUSDPrize']
       2  total_prize
```

Out 35 — 1-10 — Length: 1000, dtype: float64

	TotalUSDPrize
0	1.82e+06
1	1.80e+06
2	1.79e+06
3	1.65e+06
4	1.42e+06
5	1.09e+06
6	1.06e+06
7	1.06e+06
8	1.06e+06
9	1.02e+06

А зараз створимо Series за допомогою функції range():

Серія з однаковими значеннями

```
In 37  1  test = pd.Series(99, range(7))
       2  test
```

Out 37 — 7 rows — Length: 7, dtype: int64

	data
0	99
1	99
2	99
3	99
4	99
5	99
6	99

Звернемося до 99-го елементу:

Звертання за індексом

```
In 39  1  total_prize[99]
```

```
Out 39  232167.42
```

Обчислимо описові статистики:

Описова статистика

```
In 45  1  total_prize.describe()
```

Out 45 — 8 rows — Length: 8, dtype: float64

	TotalUSDPrize
count	1.00e+03
mean	3.98e+05
std	6.91e+05
min	2.42e+04
25%	8.38e+04
50%	1.68e+05
75%	3.94e+05
max	6.95e+06

Створимо колекцію з нестандартними індексами:

Нестандартні індекси

```
In 46  1  test = pd.Series([10, 20, 30, 40], index=['ind10', 'ind20', 'ind30', 'ind40'])
       2  test
```

Out 46 — 4 rows — Length: 4, dtype: int64

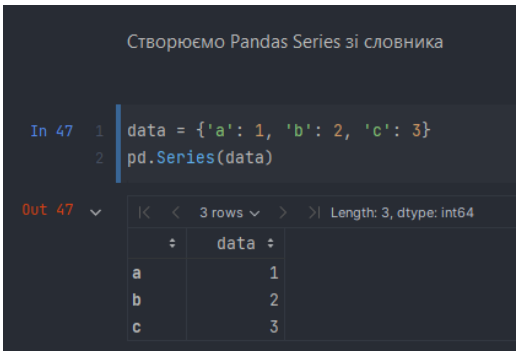
	data
ind10	10
ind20	20
ind30	30
ind40	40

Звернемося до елементу з використанням нестандартного індексу:

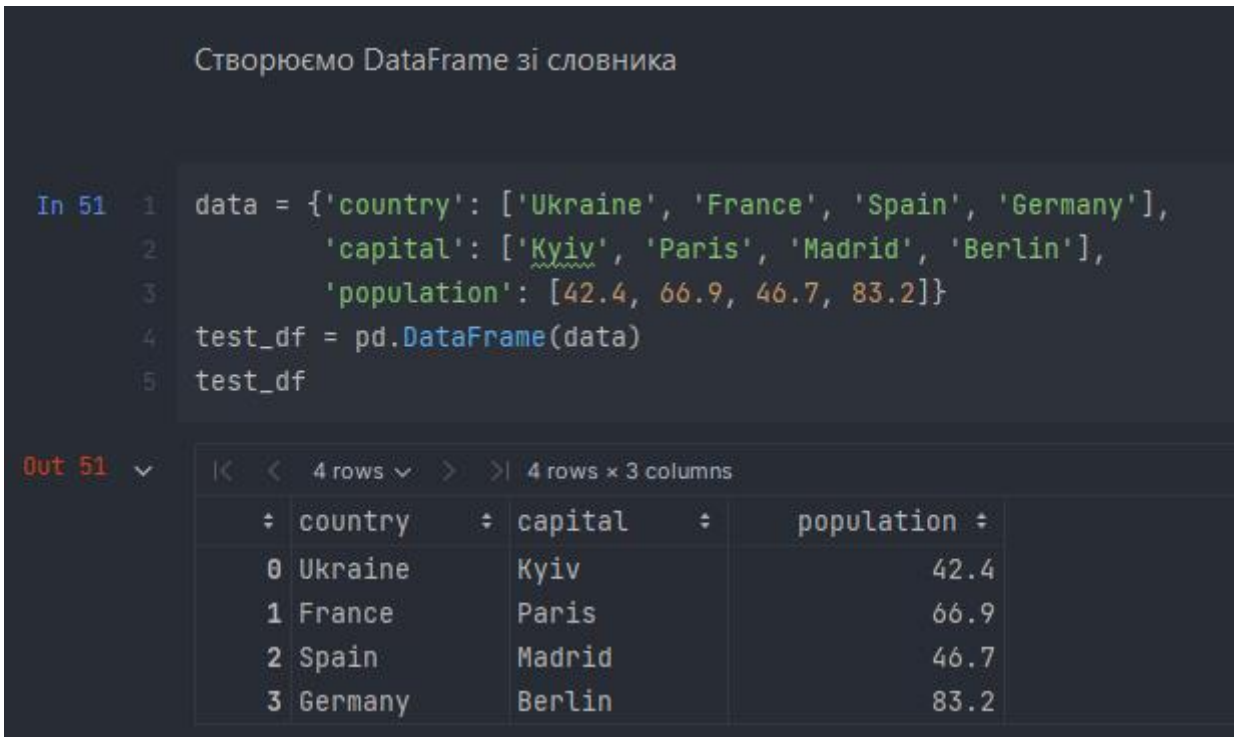
```
In 48  1  test['ind10']
```

```
Out 48  10
```

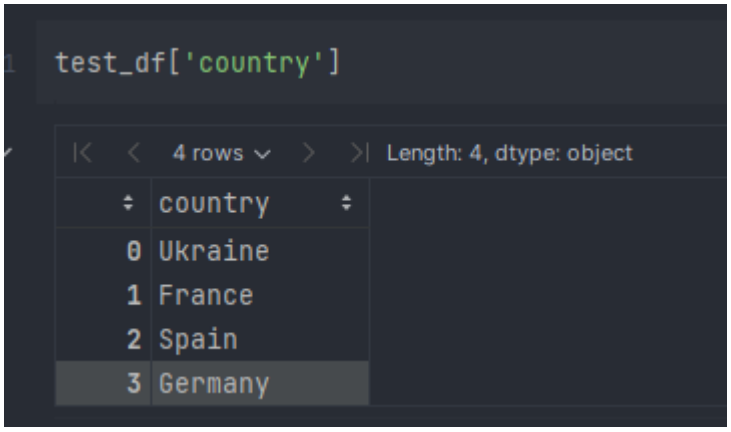
Використаємо в якості ініціалізатора словник:



Створимо колекцію DataFrame на базі словника:



Звернемося до стовпця ‘country’:



Виберемо рядки за допомогою loc та iloc:

```
loc / iloc

In 56
test_df.index = ['ind1', 'ind2', 'ind3', 'ind4']
test_df
```

Out 56 — 4 rows × 3 columns

	country	capital	population
ind1	Ukraine	Kyiv	42.4
ind2	France	Paris	66.9
ind3	Spain	Madrid	46.7
ind4	Germany	Berlin	83.2

```
In 57
test_df.loc['ind1']
```

Out 57 — 3 rows, Length: 3, dtype: object

	ind1
country	Ukraine
capital	Kyiv
population	42.4

```
In 58
test_df.iloc[0]
```

Out 58 — 3 rows, Length: 3, dtype: object

	ind1
country	Ukraine
capital	Kyiv
population	42.4

Використаємо зріз за індексами:

```
In 59  1  test_df.loc['ind1': 'ind3']
```

Out 59 — 3 rows × 3 columns

	country	capital	population
ind1	Ukraine	Kyiv	42.4
ind2	France	Paris	66.9
ind3	Spain	Madrid	46.7

```
In 60  1  test_df.iloc[0: 3]
```

Out 60 — 3 rows × 3 columns

	country	capital	population
ind1	Ukraine	Kyiv	42.4
ind2	France	Paris	66.9
ind3	Spain	Madrid	46.7

А для вибору конкретних рядків використаємо синтаксис списку:

```
In 61  1  test_df.loc[['ind1', 'ind3']]
```

Out 61 — 2 rows × 3 columns

	country	capital	population
ind1	Ukraine	Kyiv	42.4
ind3	Spain	Madrid	46.7

```
In 62  1  test_df.iloc[[0, 3]]
```

Out 62 — 2 rows × 3 columns

	country	capital	population
ind1	Ukraine	Kyiv	42.4
ind4	Germany	Berlin	83.2

Створимо новий датафрейм для більш зручної демонстрації логічного індексування:

```python
test_df = pd.DataFrame({
    'Name': ['John', 'Alice', 'Bob', 'Emily', 'David'],
    'Age': [25, 30, 22, 28, 35],
    'City': ['New York', 'London', 'Paris', 'Tokyo', 'Sydney'],
    'Salary': [50000, 60000, 45000, 70000, 55000]
}, index=['ind1', 'ind2', 'ind3', 'ind4', 'ind5'])
test_df
```

5 rows | 5 rows × 4 columns

	Name	Age	City	Salary
ind1	John	25	New York	50000
ind2	Alice	30	London	60000
ind3	Bob	22	Paris	45000
ind4	Emily	28	Tokyo	70000
ind5	David	35	Sydney	55000

Виберемо усі значення у яких поле вік більше 25:

```python
test_df[test_df['Age'] > 25]
```

3 rows | 3 rows × 4 columns

	Name	Age	City	Salary
ind2	Alice	30	London	60000
ind4	Emily	28	Tokyo	70000
ind5	David	35	Sydney	55000

Або значення, які одночасно старші за 25 та мають зарплатню менше 70000:

```
In 55  1  test_df[(test_df['Age'] > 25) & (test_df['Salary'] < 70000)]
```

Out 55 — 2 rows | 2 rows × 4 columns

	Name	Age	City	Salary
ind2	Alice	30	London	60000
ind5	David	35	Sydney	55000

Звернемося до конкретного осередку DataFrame по рядку і стовпцю:

```
In 56  1  test_df.at['ind1', 'Age']

Out 56    25
```

Використаємо метод describe(), щоб отримати описову статистику:

```
In 57  1  test_df.describe()
```

Out 57 — 8 rows × 2 columns

	Age	Salary
count	5.00	5.00
mean	28.00	56000.00
std	4.95	9617.69
min	22.00	45000.00
25%	25.00	50000.00
50%	28.00	55000.00
75%	30.00	60000.00
max	35.00	70000.00

Транспонуємо наш датафрейм, скориставшись атрибутом T:

```
In 58  1  test_df.T
```

Out 58 — 4 rows × 5 columns

	ind1	ind2	ind3	ind4	ind5
Name	John	Alice	Bob	Emily	David
Age	25	30	22	28	35
City	New York	London	Paris	Tokyo	Sydney
Salary	50000	60000	45000	70000	55000

Та отримаємо його описову статистику:

```
In 59  1  test_df.T.describe()
```

Out 59 — 4 rows × 5 columns

	ind1	ind2	ind3	ind4	ind5
count	4	4	4	4	4
unique	4	4	4	4	4
top	John	Alice	Bob	Emily	David
freq	1	1	1	1	1

Відсортуємо рядки датафрейму за індексами:

```
In 60   1  test_df.sort_index(ascending=False)
```

Out 60

5 rows × 4 columns

	Name	Age	City	Salary
ind5	David	35	Sydney	55000
ind4	Emily	28	Tokyo	70000
ind3	Bob	22	Paris	45000
ind2	Alice	30	London	60000
ind1	John	25	New York	50000

Або за стовпцями, вказавши параметр axis=1:

```
In 61   1  test_df.sort_index(axis=1, ascending=False)
```

Out 61

5 rows × 4 columns

	Salary	Name	City	Age
ind1	50000	John	New York	25
ind2	60000	Alice	London	30
ind3	45000	Bob	Paris	22
ind4	70000	Emily	Tokyo	28
ind5	55000	David	Sydney	35

Відсортуємо датафрейм за значенням стовпця ‘age’

```
In 62   1  test_df.sort_values(by='Age')
```

Out 62

5 rows × 4 columns

	Name	Age	City	Salary
ind3	Bob	22	Paris	45000
ind1	John	25	New York	50000
ind4	Emily	28	Tokyo	70000
ind2	Alice	30	London	60000
ind5	David	35	Sydney	55000

5. Прочитаємо та 6. Завантажимо набір даних катастрофи Титаніка:

Titanic

```
In 31  1  import pandas as pd

In 32  1  titanic = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/carData/TitanicSurvival.csv')
       2  titanic
```

Out 32 — 1-10 | 1309 rows × 5 columns

	name	survived	sex	age	class
0	Allen, Miss. Elisabeth Walton	yes	female	29.00	1st
1	Allison, Master. Hudson Trevor	yes	male	0.92	1st
2	Allison, Miss. Helen Loraine	no	female	2.00	1st
3	Allison, Mr. Hudson Joshua Crei	no	male	30.00	1st
4	Allison, Mrs. Hudson J C (Bessi	no	female	25.00	1st
5	Anderson, Mr. Harry	yes	male	48.00	1st
6	Andrews, Miss. Kornelia Theodos	yes	female	63.00	1st
7	Andrews, Mr. Thomas Jr	no	male	39.00	1st
8	Appleton, Mrs. Edward Dale (Cha	yes	female	53.00	1st
9	Artagaveytia, Mr. Ramon	no	male	71.00	1st

7. Переглянемо рядки набору даних, використавши функції head() і tail():

```
In 34  1  titanic.head()
```

Out 34 — 5 rows | 5 rows × 5 columns

	Unnamed: 0	survived	sex	age	passengerClass
0	Allen, Miss. Elisabeth Walton	yes	female	29.00	1st
1	Allison, Master. Hudson Trevor	yes	male	0.92	1st
2	Allison, Miss. Helen Loraine	no	female	2.00	1st
3	Allison, Mr. Hudson Joshua Crei	no	male	30.00	1st
4	Allison, Mrs. Hudson J C (Bessi	no	female	25.00	1st

```
In 35  1  titanic.tail()
```

Out 35 — 5 rows | 5 rows × 5 columns

	Unnamed: 0	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

8. Налаштуємо назви стовпців:

```
In 36  1  titanic.columns = ['name', 'survived', 'sex', 'age', 'class']

In 37  1  titanic.head()
```

Out 37 — 5 rows × 5 columns

	name	survived	sex	age	class
0	Allen, Miss. Elisabeth Walton	yes	female	29.00	1st
1	Allison, Master. Hudson Trevor	yes	male	0.92	1st
2	Allison, Miss. Helen Loraine	no	female	2.00	1st
3	Allison, Mr. Hudson Joshua Crei	no	male	30.00	1st
4	Allison, Mrs. Hudson J C (Bessi	no	female	25.00	1st

9. Проведемо простий аналіз даних:

Спочатку виокремимо пасажирів, яким вдалося вижити:

```
In 41  1  survived = titanic[titanic['survived'] == 'yes']
       2  survived
```

Out 41 — 500 rows × 5 columns

	name	survived	sex	age	class
0	Allen, Miss. Elisabeth Walton	yes	female	29.00	1st
1	Allison, Master. Hudson Trevor	yes	male	0.92	1st
5	Anderson, Mr. Harry	yes	male	48.00	1st
6	Andrews, Miss. Kornelia Theodos	yes	female	63.00	1st
8	Appleton, Mrs. Edward Dale (Cha	yes	female	53.00	1st
11	Astor, Mrs. John Jacob (Madelei	yes	female	18.00	1st
12	Aubart, Mme. Leontine Pauline	yes	female	24.00	1st
13	Barber, Miss. Ellen Nellie	yes	female	26.00	1st
14	Barkworth, Mr. Algernon Henry W	yes	male	80.00	1st
17	Baxter, Mrs. James (Helene DeLa	yes	female	50.00	1st

- Визначимо наймолодшого пасажира серед них:

```
youngest = survived[survived['age'] == survived['age'].min()]
youngest
```

1 row × 5 columns

	name	survived	sex	age	class
763	Dean, Miss. Elizabeth Gladys M	yes	female	0.17	3rd

- Найстаршого:

```
oldest = survived[survived['age'] == survived['age'].max()]
oldest
```

1 row | 1 rows × 5 columns

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

- Середній вік:

```
average_age = survived['age'].mean()
average_age
```

```
28.918228103072597
```

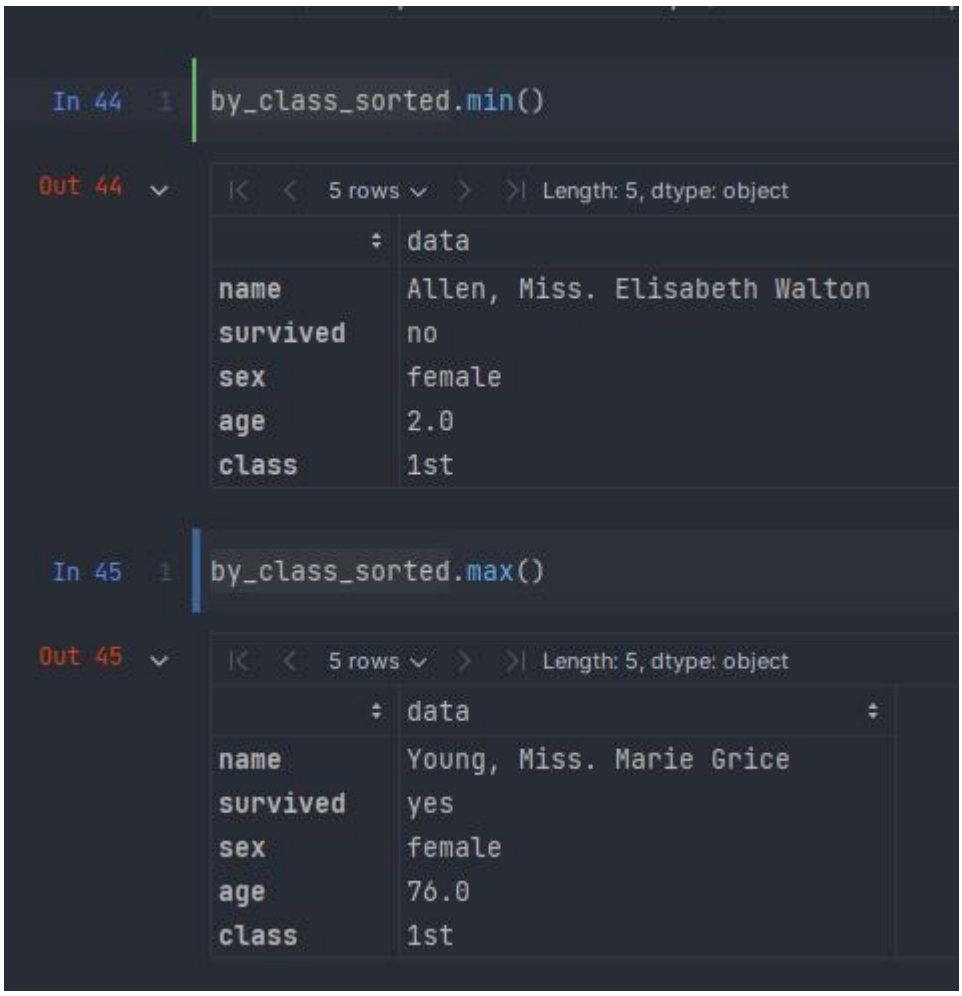
Знайдемо жінок-пасажирів першого класу, яким вдалося вижити, та відсортуємо їх за віком:

```
In 66  by_class_sorted = titanic[(titanic['class'] == '1st') & (titanic['sex'] == 'female')].sort_values(by='age', ascending=False)
       by_class_sorted
```

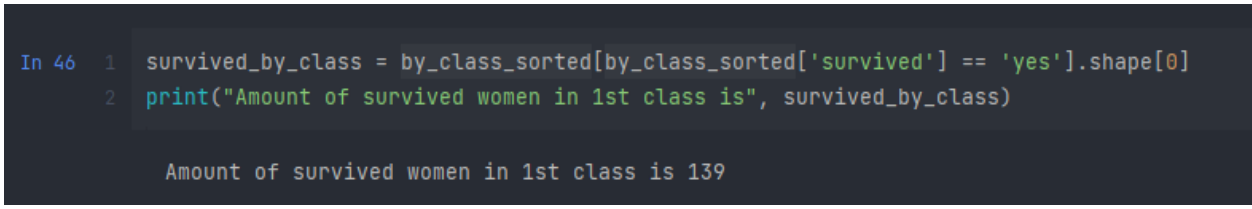
Out 66 — 1-10 | 144 rows × 5 columns

	name	survived	sex	age	class
61	Cavendish, Mrs. Tyrell William	yes	female	76.0	1st
78	Compton, Mrs. Alexander Taylor	yes	female	64.0	1st
83	Crosby, Mrs. Edward Gifford (Ca	yes	female	64.0	1st
6	Andrews, Miss. Kornelia Theodos	yes	female	63.0	1st
286	Straus, Mrs. Isidor (Rosalie Id	no	female	63.0	1st
284	Stone, Mrs. George Nelson (Mart	yes	female	62.0	1st
43	Bucknell, Mrs. William Robert (	yes	female	60.0	1st
304	Warren, Mrs. Frank Manley (Anna	yes	female	60.0	1st
116	Fortune, Mrs. Mark (Mary McDoug	yes	female	60.0	1st
42	Brown, Mrs. John Murray (Caroli	yes	female	59.0	1st

Знайдемо наймолодшу та найстаршу серед них:

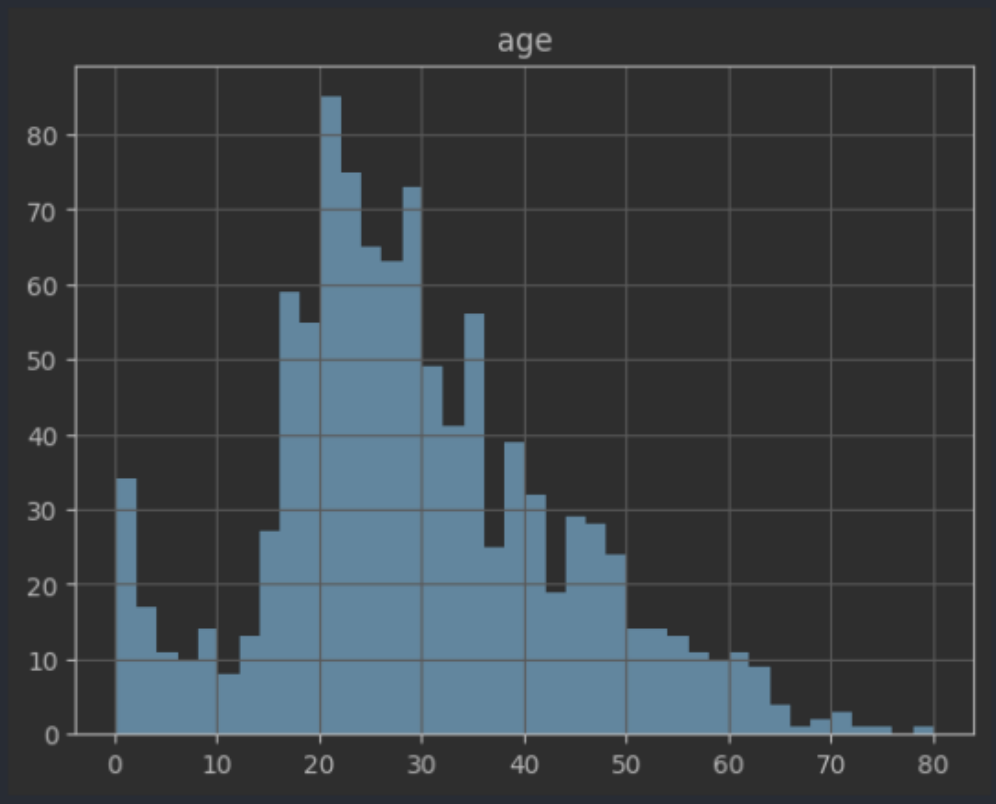


Загальна кількість вижившихжінок, які були пасажирками першого класу:



11. Побудуємо гістограму віку пасажирок:

```
In 47   1  titanic.hist(bins=40)

Out 47     array([[<AxesSubplot: title={'center': 'age'}>]], dtype=object)
```



Висновок

Виконуючи цю лабораторну роботу я ознайомився з бібліотеками pandas та matplotlib. Також було проведено невеликий аналіз даних, присвячених катастрофі лайнера «Титанік».