 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

**Комп’ютерний практикум №1**

з дисципліни «**Нейронні мережі**»

на тему: «Введення в python та його бібліотеки для обробки та аналізу даних»

**Виконав:**

студент гр. БС-03

Затуловський Г. А.

**Перевірив:**

доц. каф. БМК Федорін І.В.

Зараховано від \_\_\_.\_\_\_.\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис викладача)

Київ-2022

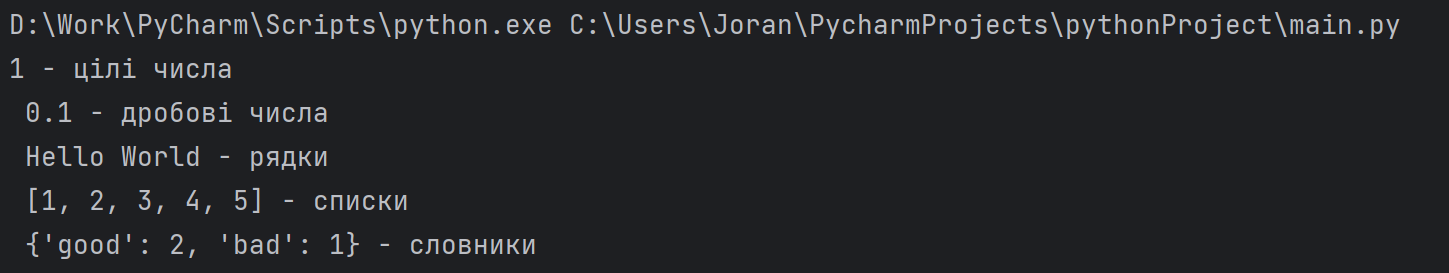
Згідно варіанту № 6:

**Програмна реалізація.**

**Завдання 1 - Основи Python**

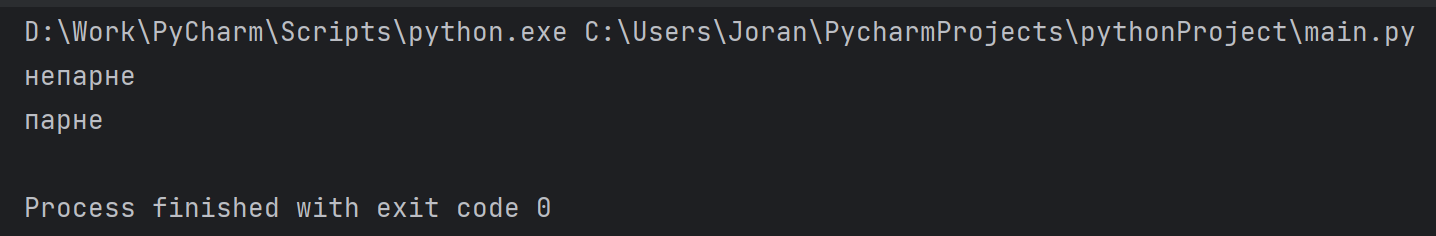
* 1. **Створити змінні різних типів (цілі числа, дробові числа, рядки, списки, словники) та вивести їх значення.**

a = 1 # - цілі числа  
b = 0.1 # - дробові числа  
c = "Hello World" # - рядки  
d = [1,2,3,4,5] # - списки  
f = {"good": 2, "bad":1} # - словники  
print (a,"- цілі числа\n",b,"- дробові числа\n",c,"- рядки\n",d,"- списки\n",f,"- словники\n")

****

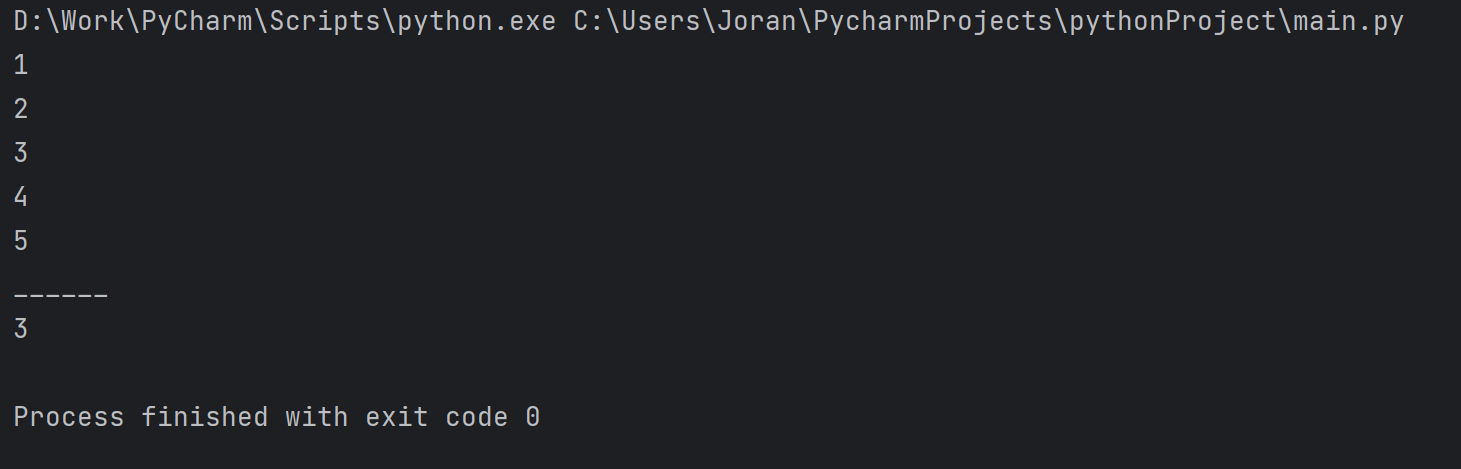
* 1. **Створити умовний оператор, який перевіряє, чи є число парним чи непарним.**

def check(a):  
 if (a % 2 == 1):  
 print("непарне")  
 else:  
 print("парне")  
  
a = 1 # - цілі числа  
b = 0.1 # - дробові числа  
c = "Hello World" # - рядки  
d = [1,2,3,4,5] # - списки  
f = {"good": 2, "bad":1} # - словники  
  
check(a);  
check(d[1])

****

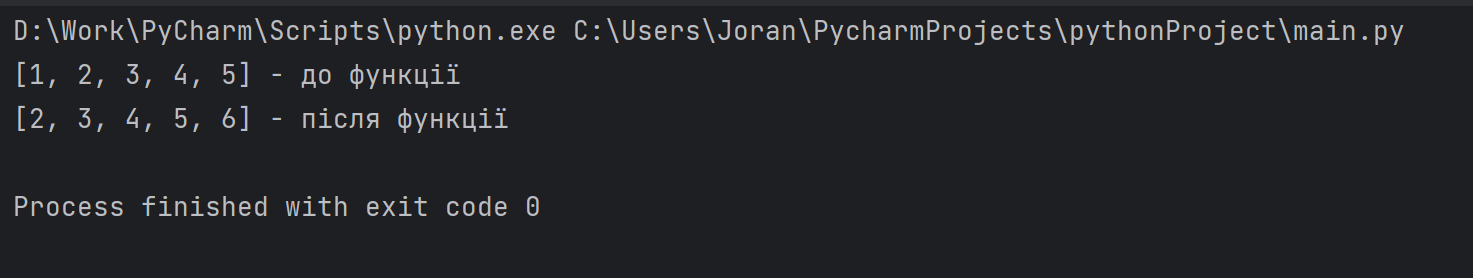
* 1. **Створити цикл for, який перебирає елементи списку, і цикл while, який працює, поки деяке задане умова не є істиною.**

def check(a):  
 if (a % 2 == 1):  
 print("непарне")  
 else:  
 print("парне")  
def for\_ (d):  
 for i in d:  
 print(i)  
  
def while\_ (x):  
 i = 0  
 while(x[i] != 4):  
 i=i+1  
 print(i)  
   
  
d = [1,2,3,4,5] # - списки  
  
for\_(d)  
print('\_\_\_\_\_\_')  
while\_(d)

****

* 1. **Створити функцію, яка приймає вхідні параметри та виконує певні дії з ними (наприклад, виводить на екран, збільшує на 1 тощо).**

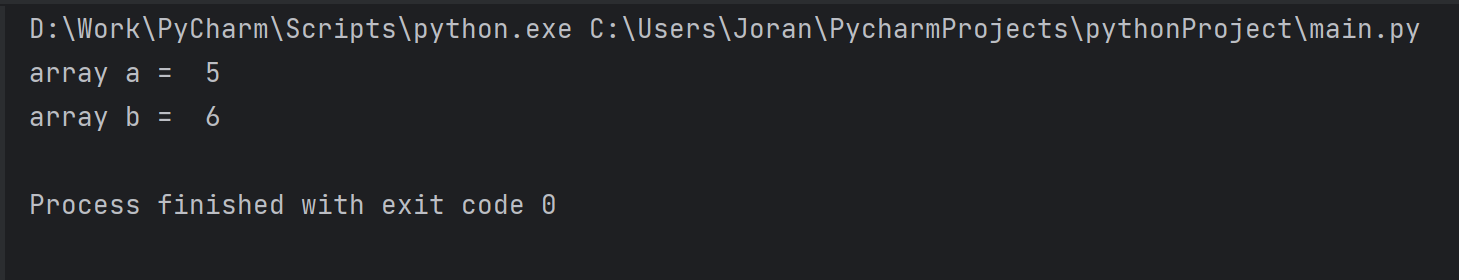
def grow\_one(x):  
 o = 0  
 for i in x:  
 x[o] = 1 + i  
 o = o+1  
  
d = [1,2,3,4,5] # - списки  
print(d,"- до функції")  
grow\_one(d)  
print(d,"- після функції")

****

**Завдання 2 - Використання numpy**

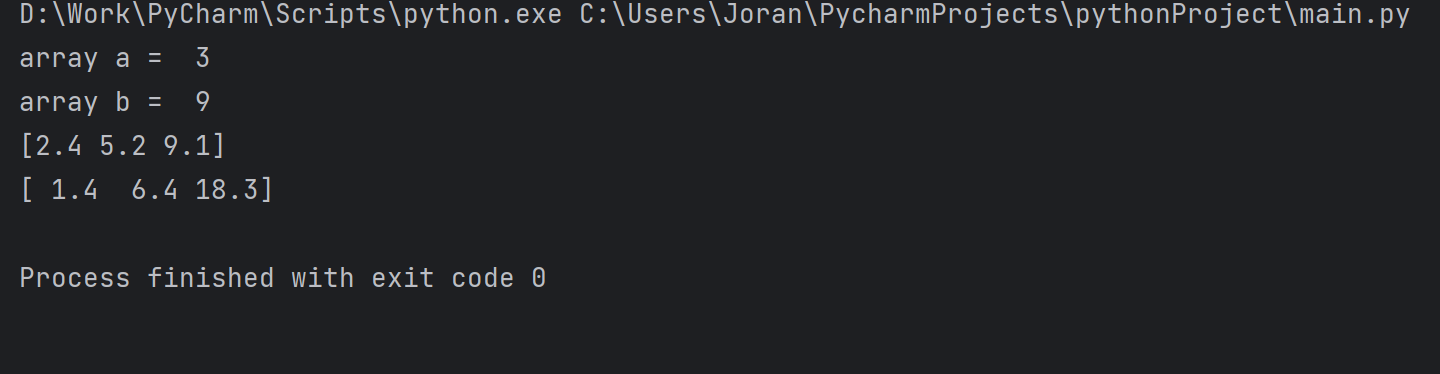
**2.1. Створити масив numpy різних типів (одномірний, двомірний) та вивести їх розмірність.**

import numpy as np  
a = np.array([1,2,3,4,5])  
b = np.array([(1.2,2,4),(3,1.3,6.7)],dtype = float)  
print('array a = ',a.size)  
print('array b = ',b.size)

****

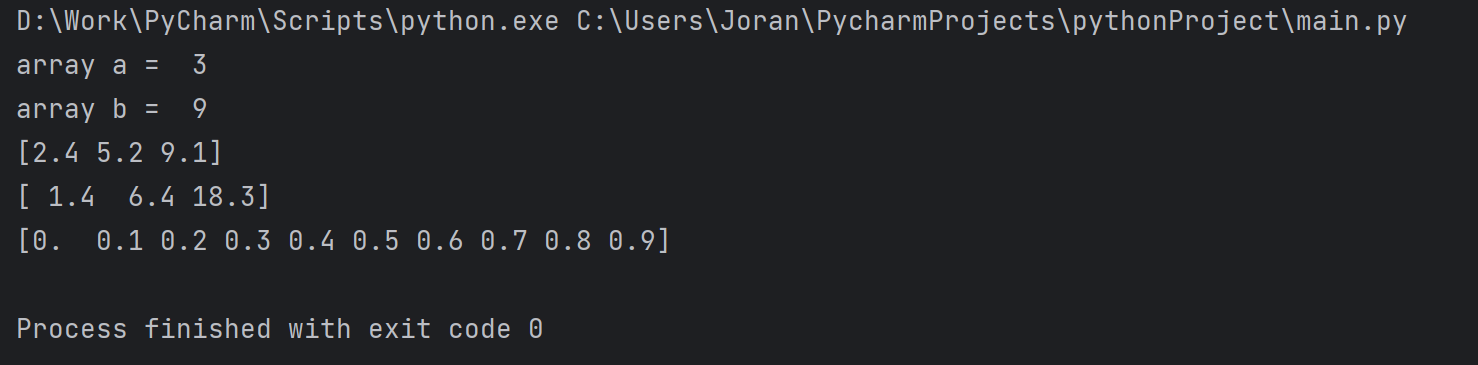
**2.2. Виконати базові арифметичні операції над масивами (додавання, множення).**

import numpy as np  
a = np.array([1,2,3])  
b = np.array([(1.2,2,4),(2,1.3,6.7),(1.4,3.2,6.1)],dtype = float)  
print('array a = ',a.size)  
print('array b = ',b.size)  
  
print(a+b[2])  
print(a\*b[2])

****

**2.3. Створити масив значень, що розташовані в діапазоні від 0 до 1 з кроком 0.1.**

import numpy as np  
c = np.arange(0, 1,0.1)  
print(c)

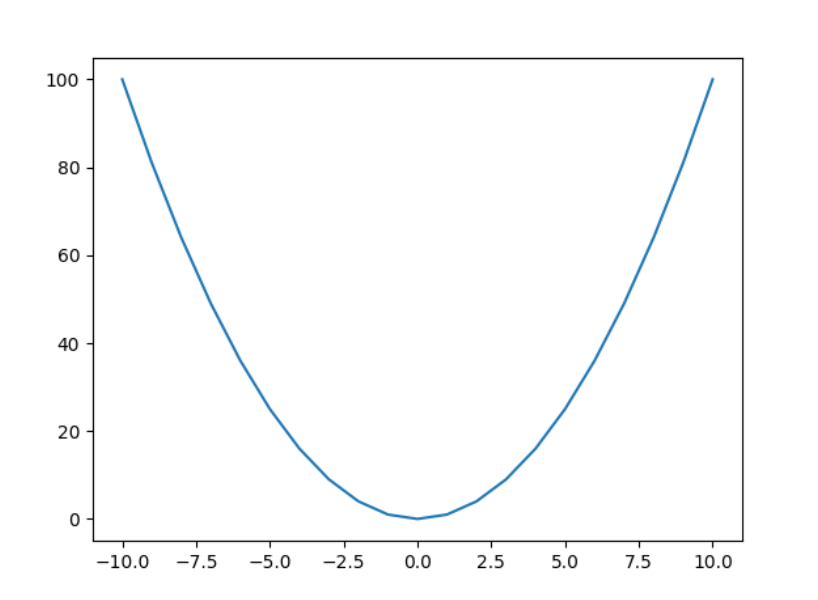
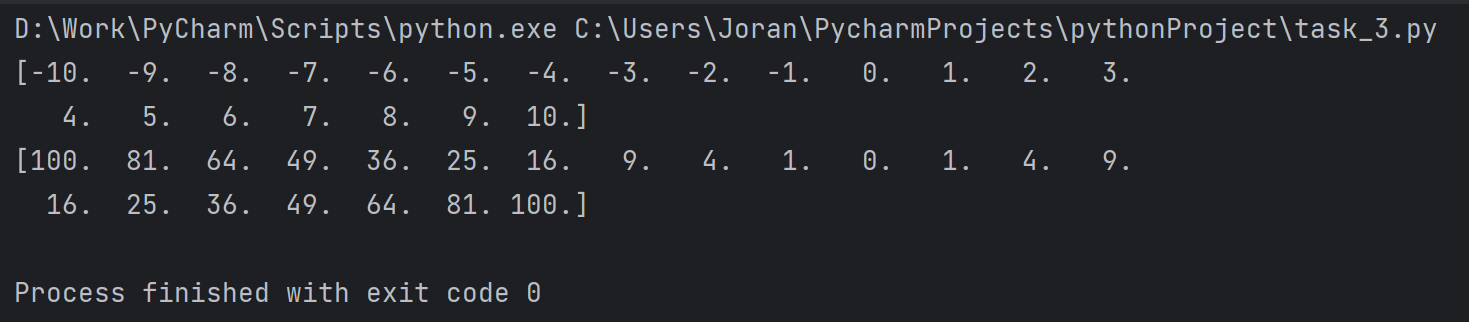
****

**2.4. Обчислити скалярний добуток двох одномірних масивів.**

**Завдання 3 - Використання matplotlib**

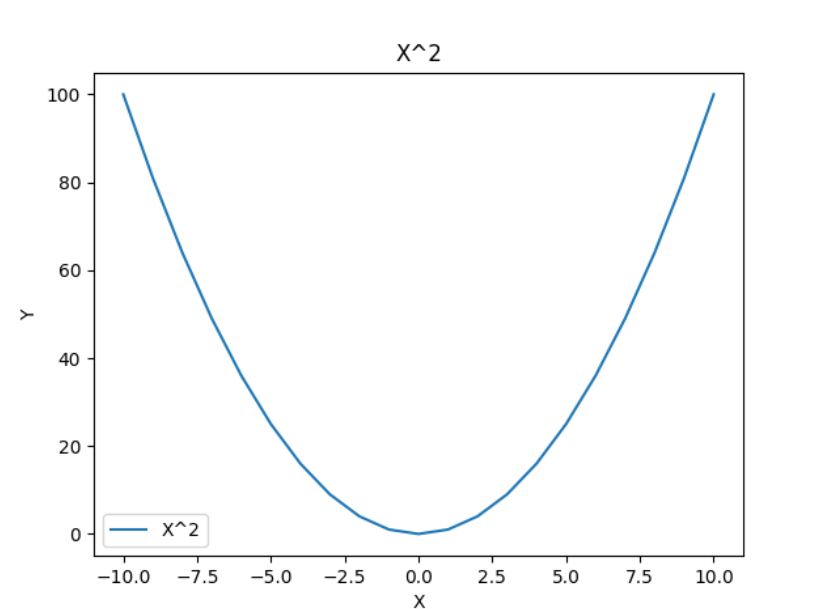
3.1. Створити графік функції y = x^2 на проміжку від -10 до 10.

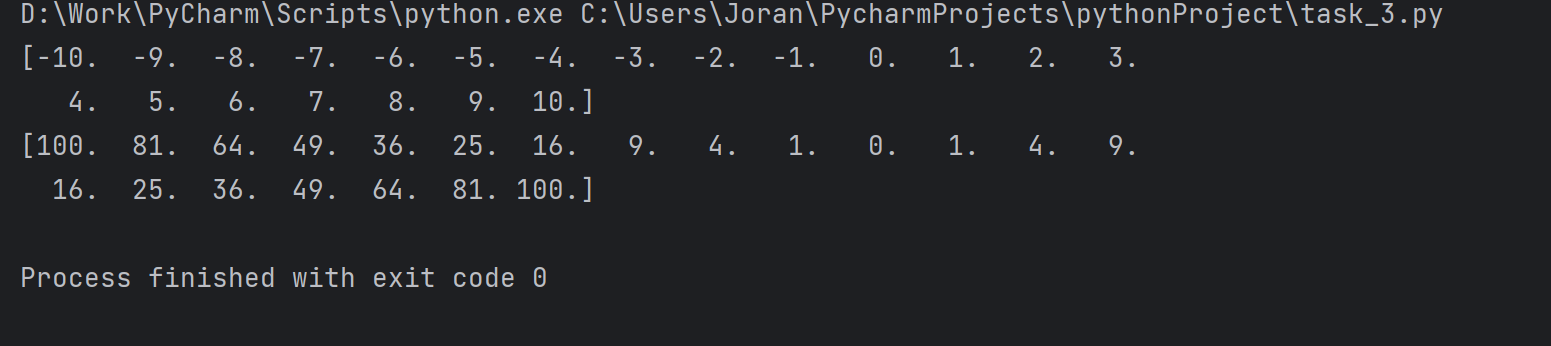
import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.arange(-10, 10.1,1)  
y = x\*\*2  
print(x)  
print(y)  
plt.plot(x,y,label="X^2")  
  
plt.show()

3.2. Додати заголовки до осей та заголовок графіку.

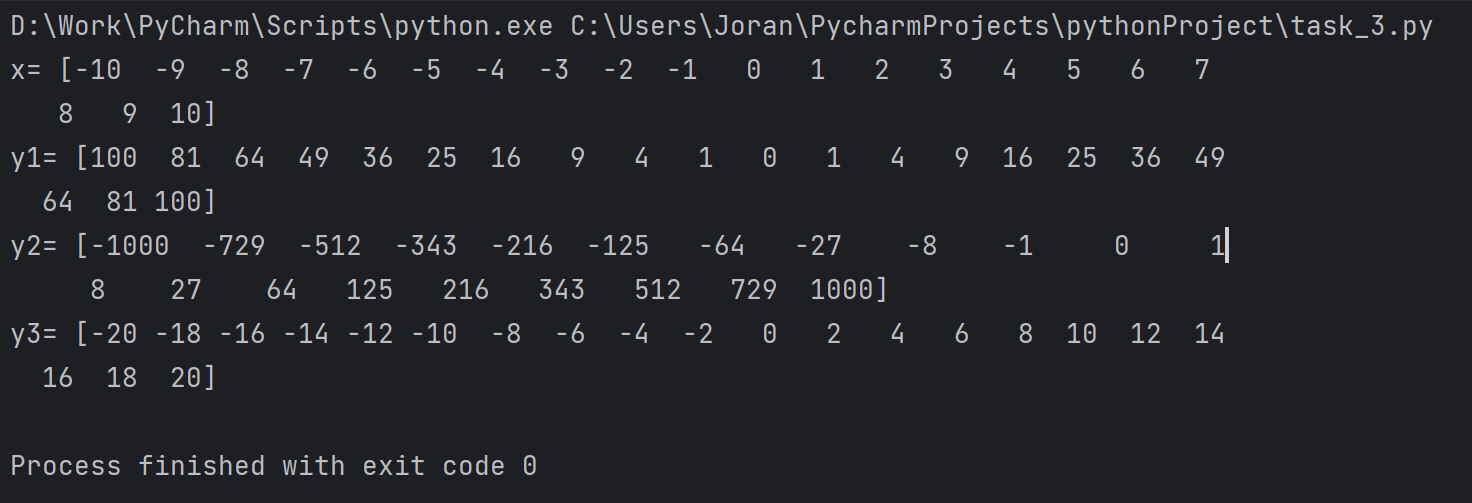
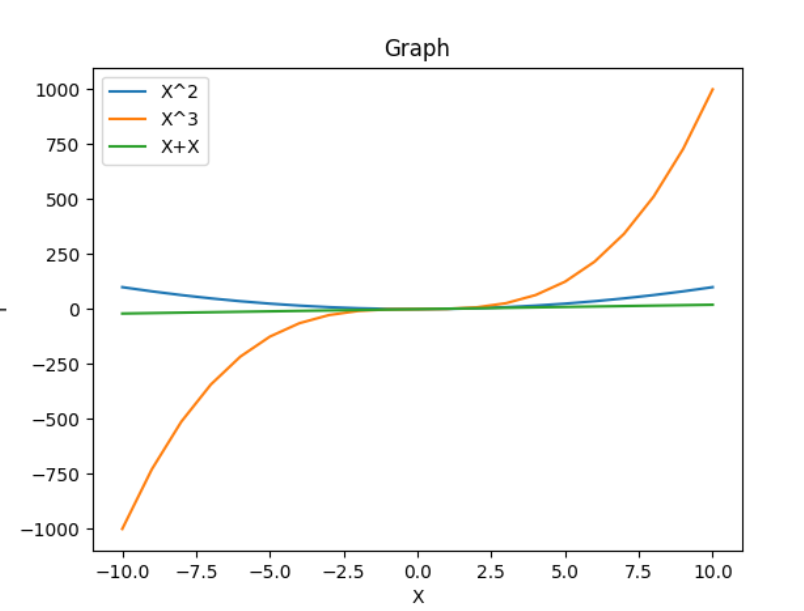
import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.arange(-10, 10.1,1)  
y = x\*\*2  
print(x)  
print(y)  
plt.plot(x,y,label="X^2")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.title("X^2")  
plt.legend()  
plt.show()





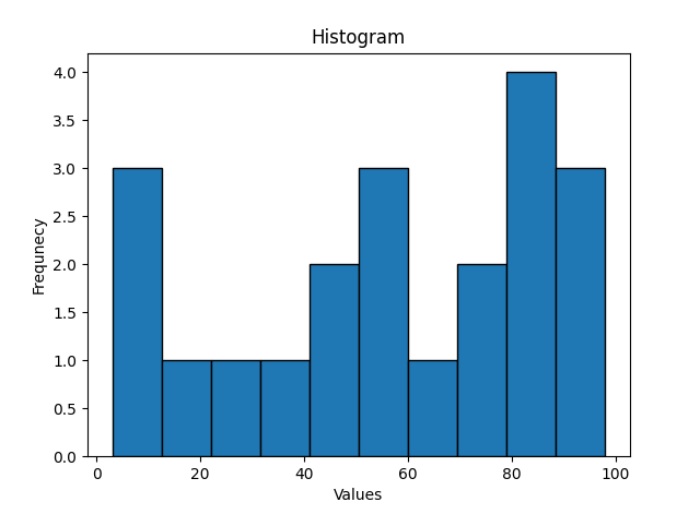
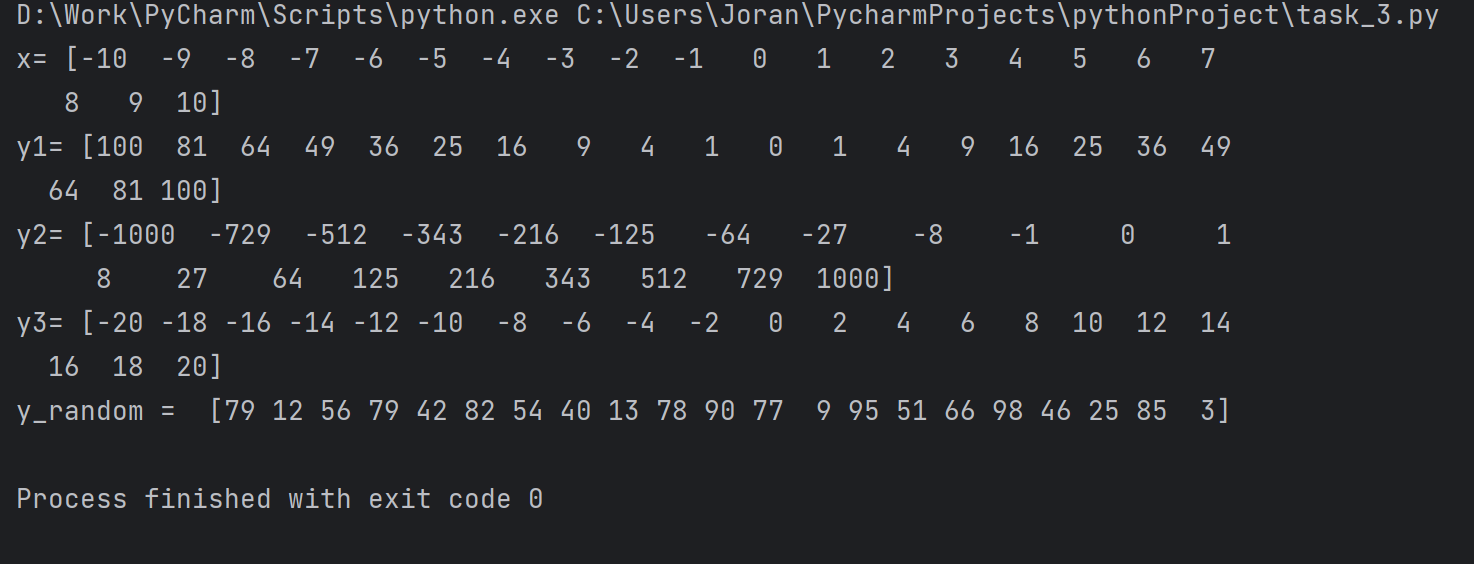
3.3. Накласти на одному графіку декілька кривих (наприклад, y = x^2, y = x^3) з різними колірними мітками, додати легенду.

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.arange(-10, 11,1)  
y = x\*\*2  
y2 = x\*\*3  
y3 = x+x  
print("x=",x)  
print("y1=",y)  
print("y2=",y2)  
print("y3=",y3)  
plt.plot(x,y,label="X^2")  
plt.plot(x,y2,label="X^3")  
plt.plot(x,y3,label="X+X")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.title("Graph")  
plt.legend()  
plt.show()



3.4. Створити гістограму за допомогою matplotlib на основі випадкового масиву значень.

import matplotlib.pyplot as plt  
import numpy as np  
  
x = np.arange(-10, 11,1)  
y = x\*\*2  
y2 = x\*\*3  
y3 = x+x  
print("x=",x)  
print("y1=",y)  
print("y2=",y2)  
print("y3=",y3)  
plt.plot(x,y,label="X^2")  
plt.plot(x,y2,label="X^3")  
plt.plot(x,y3,label="X+X")  
plt.xlabel("X")  
plt.ylabel("Y")  
plt.title("Graph")  
plt.legend()  
plt.show()  
y\_random = np.random.randint(100,size=len(x))  
print("y\_random = ",y\_random)  
plt.hist(y\_random,bins=10,edgecolor ='black')  
plt.xlabel("Values")  
plt.ylabel("Frequnecy")  
plt.title("Histogram")  
  
plt.show()

**Для завдань 4 та 5 зробити вказані для набору даних відповідно до варіанту:**

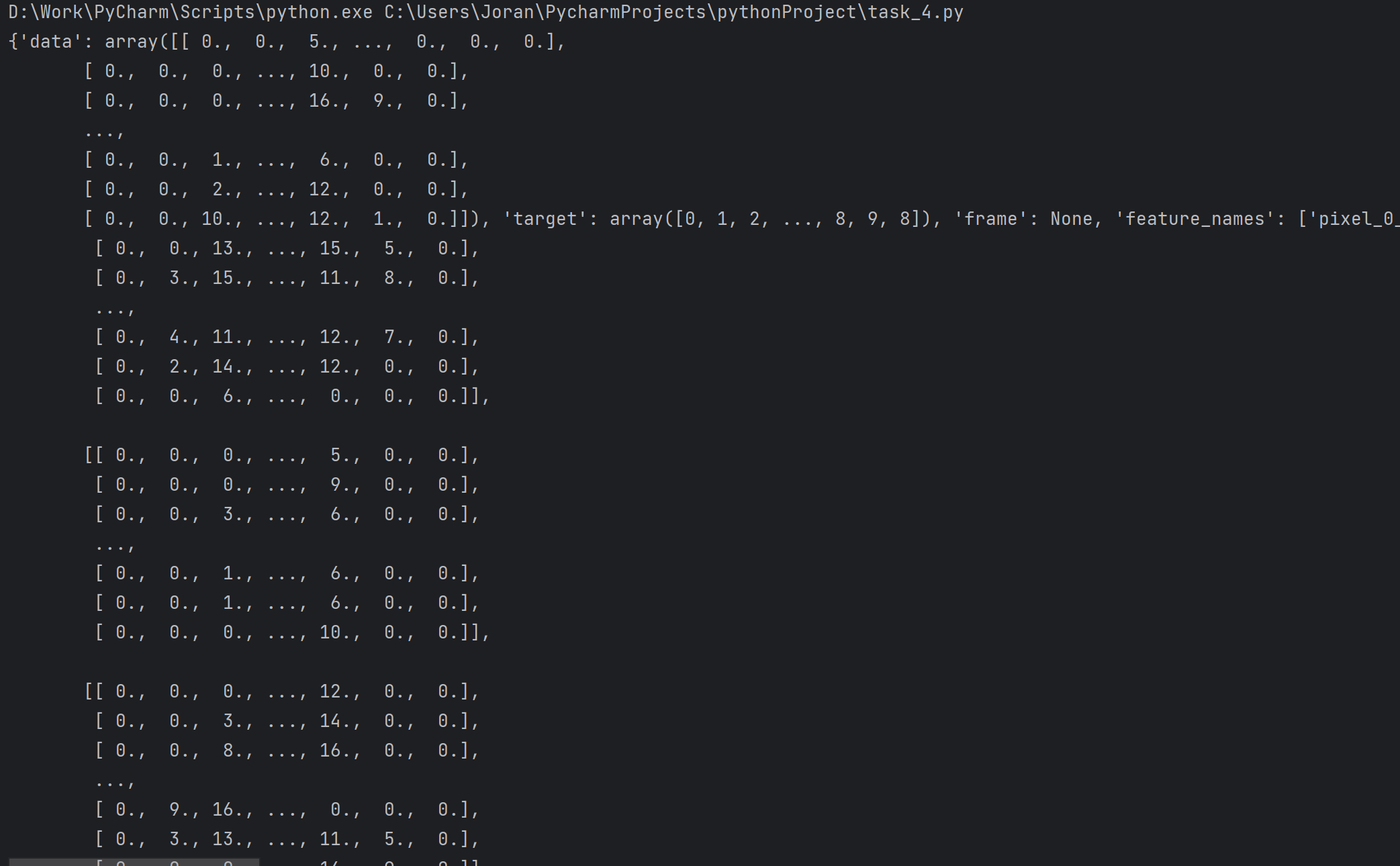
* [1, 7, 13, 19] Iris Dataset: Цей набір даних містить інформацію про 150 квітів ірису, які поділені на три види. Для кожного квіту представлені чотири ознаки: довжина та ширина чашолистиків і пелюсток.
* [2, 8, 14, 20] Wine Dataset: Цей набір даних містить хімічну інформацію про 178 вин з трьох різних сортів. Для кожного вина представлено 13 хімічних ознак.
* [3, 9, 15, 21] Breast Cancer Dataset: Цей набір даних містить ознаки, виведені з зображень клітин, що взяті з людської грудної тканини, і ці ознаки використовуються для прогнозування, чи є пухлина доброякісною або злоякісною.
* [4, 10, 16, 22] Boston Housing Dataset: Цей набір даних містить інформацію про житлові будинки в Бостоні, включаючи середнє значення будинків і 13 атрибутів, які можуть впливати на ціну.
* [5, 11, 17, 23] Diabetes Dataset: Цей набір даних використовується для прогнозування кількості показників прогресування діабету за рік на основі базових ознак пацієнтів.
* [6, 12, 18, 24] Digits Dataset: Це набір даних з зображень рукописних цифр, який може бути використаний для навчання моделей розпізнавання зображень.

**Завдання 4 - Використання pandas**

* 1. Створіть pandas DataFrame з вбудованого набору даних з бібліотеки sklearn.

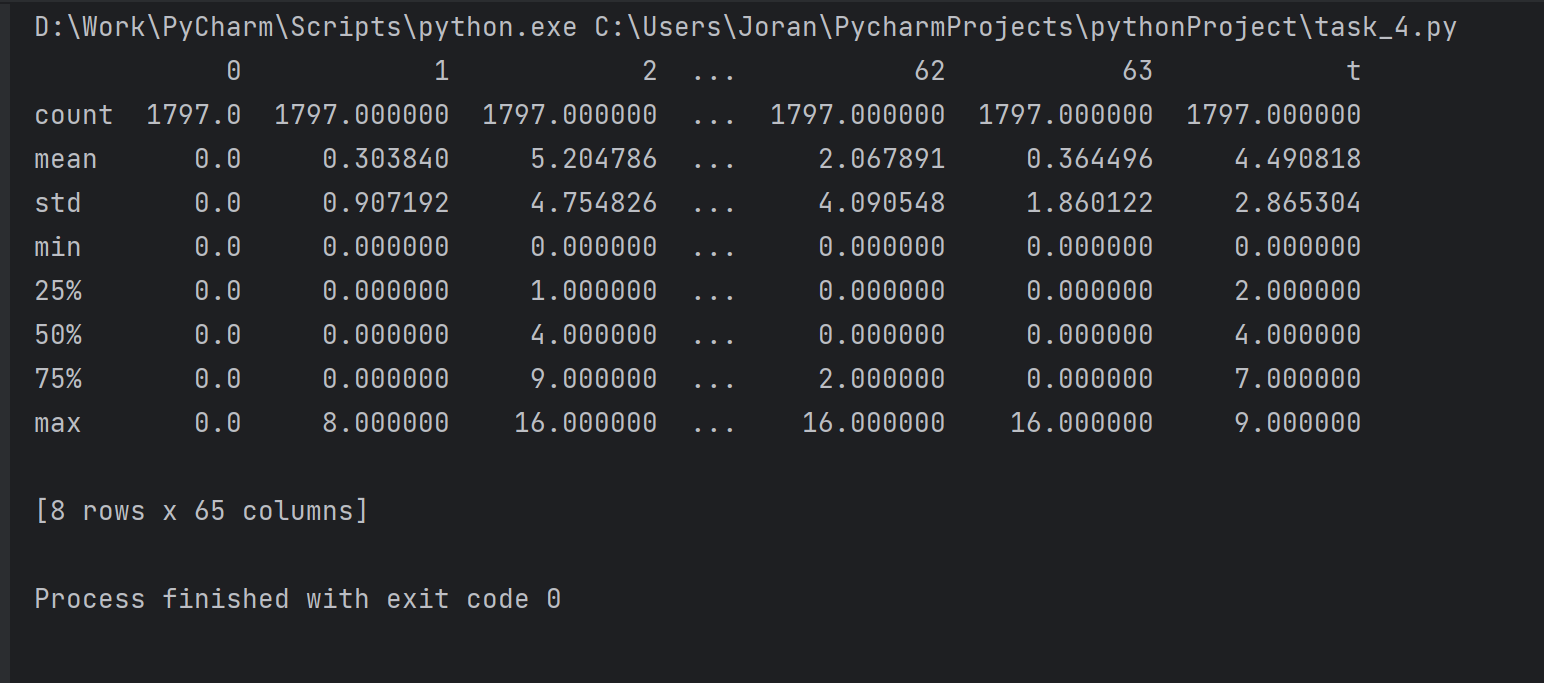
Першим кроком є імпортування даних за допомогою команди from sklearn.datasets import load\_*dataset* (відповідно до варіанту завдань).

from sklearn.datasets import load\_digits  
datas = load\_digits()  
print(datas)



* 1. Виведіть перші 5 записів у DataFrame.

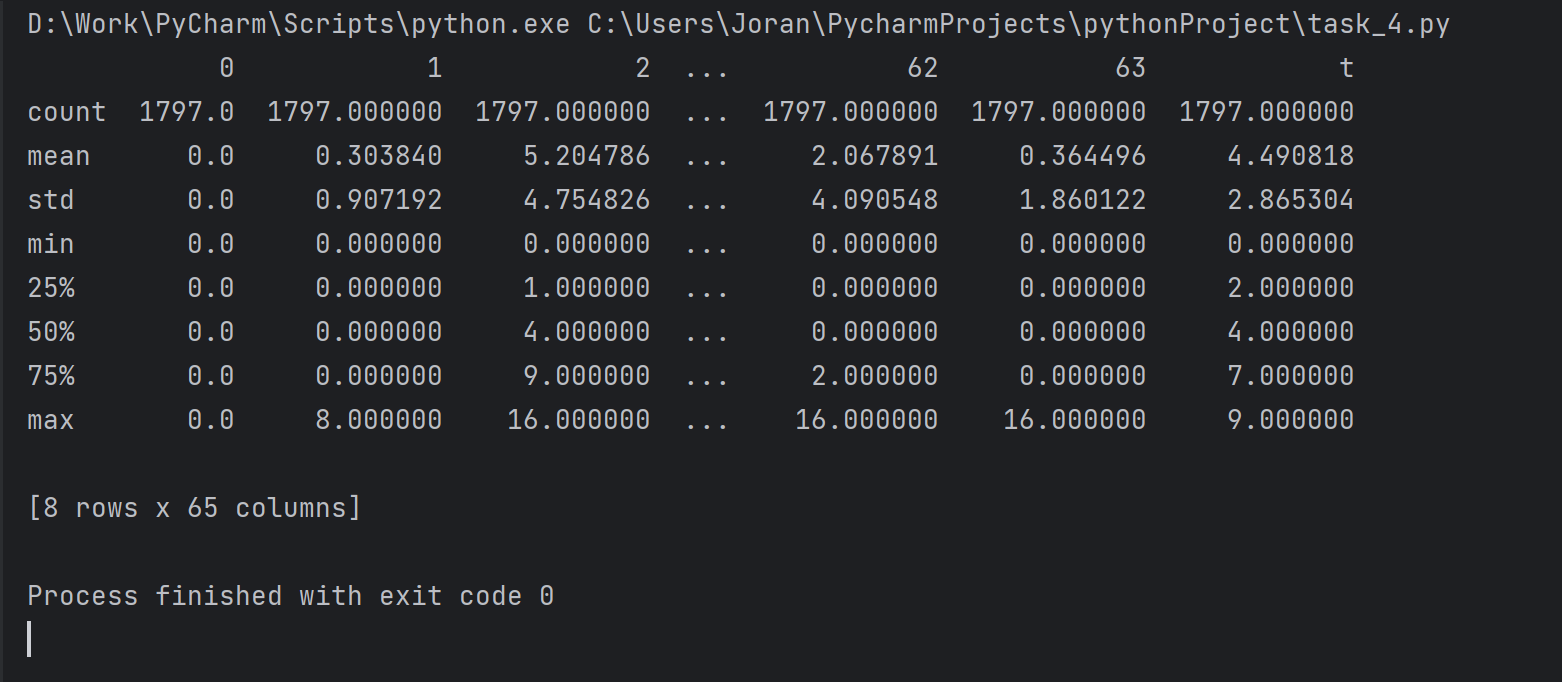
1. import numpy as np  
   import pandas as pd  
   import matplotlib as plot  
     
   from sklearn.datasets import load\_digits  
   datas = load\_digits()  
     
   df = pd.DataFrame(datas.data)  
   #print (df[0:5])  
   df['t'] = datas.target  
   print(df.describe())



* 1. Виведіть статистичну інформацію про DataFrame за допомогою метода

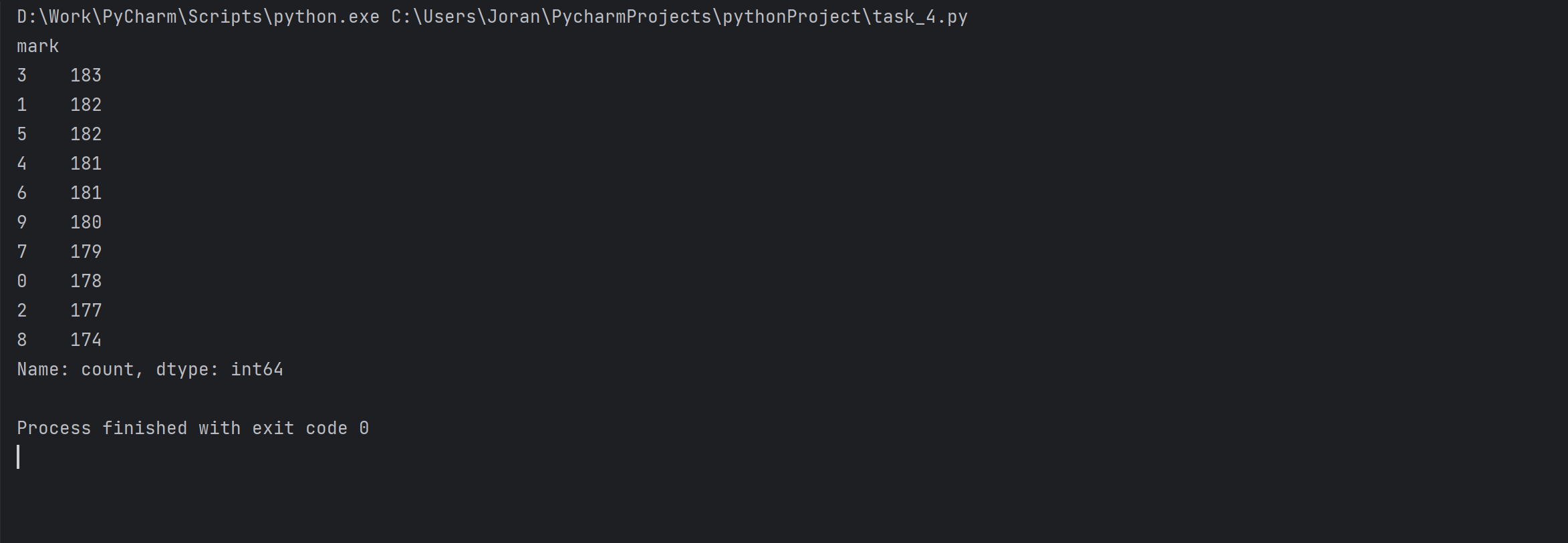
.describe().

import numpy as np  
import pandas as pd  
import matplotlib as plot  
  
from sklearn.datasets import load\_digits  
datas = load\_digits()  
  
df = pd.DataFrame(datas.data)  
#print (df[0:5])  
  
print(df.describe())



* 1. Виведіть інформацію про кількість записів у кожному класі (використайте метод .value\_counts() на стовпці з мітками класів).

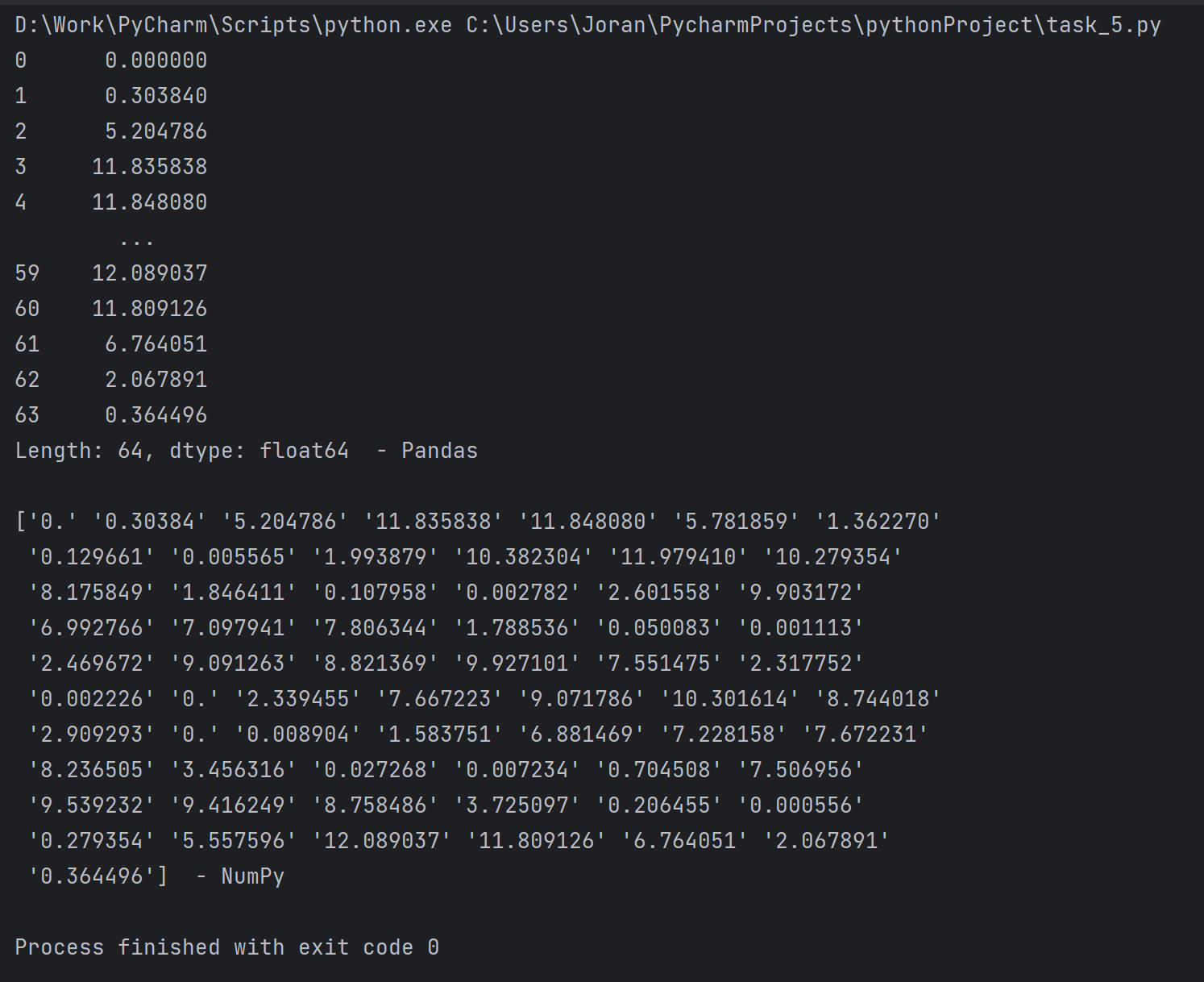
1. import numpy as np  
   import pandas as pd  
   import matplotlib as plot  
     
   from sklearn.datasets import load\_digits  
   datas = load\_digits()  
     
   df = pd.DataFrame(datas.data)  
   #print (df[0:5])  
     
   #print(df.describe())  
   df['mark'] = datas.target  
   print(df['mark'].value\_counts())



**Завдання 5 - Аналіз даних за допомогою scipy, matplotlib і pandas**

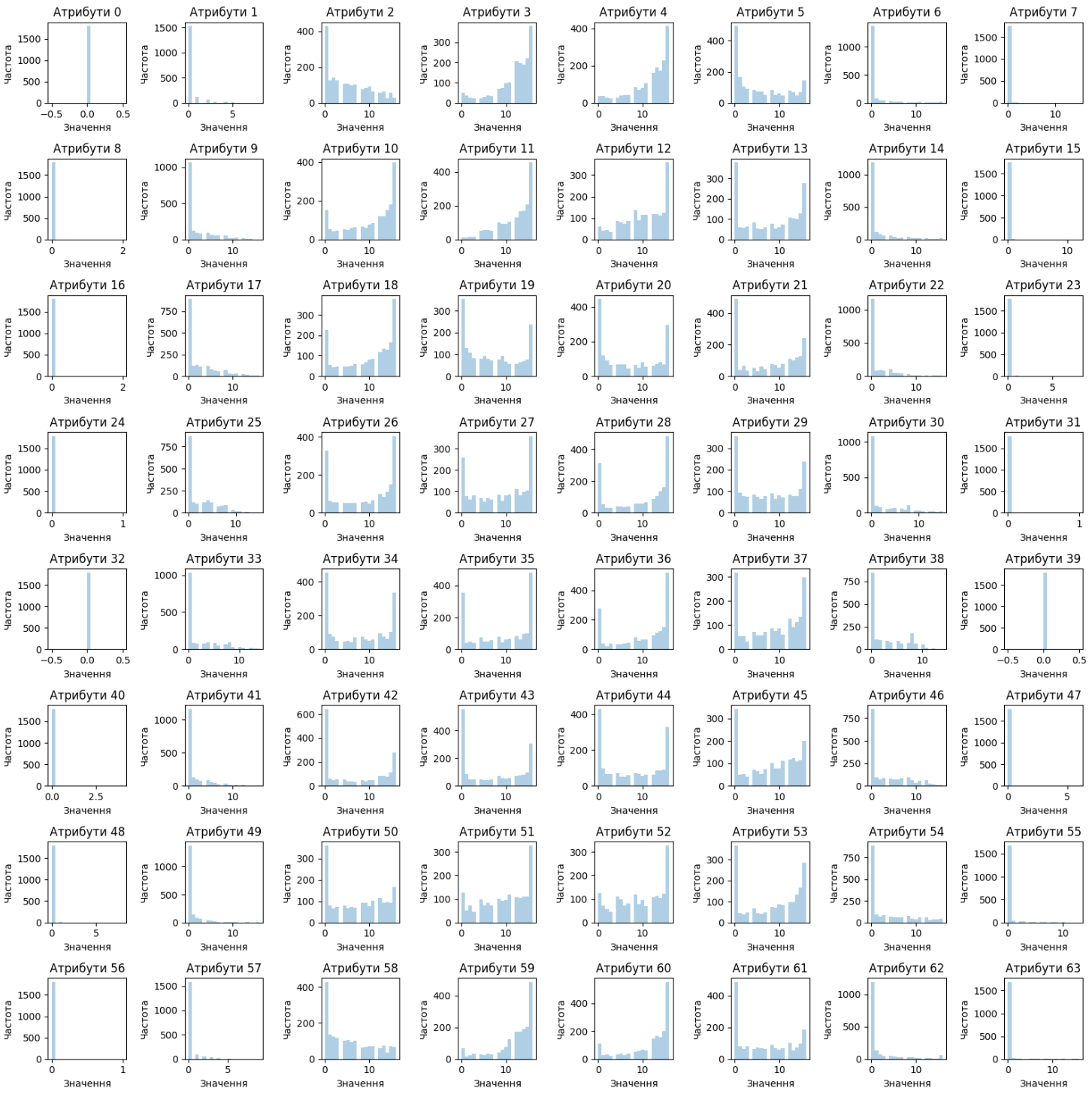
* 1. Обчисліть середні значення для кожного з атрибутів в наборі даних (використовуйте numpy).

1. import numpy as np  
   import pandas as pd  
   import matplotlib as plot  
   from sklearn.datasets import load\_digits  
   datas = load\_digits()  
   df = pd.DataFrame(datas.data)  
     
   print(df.mean() , ' - Pandas\n')  
     
   datas\_mean = np.array(df.mean)  
   d = np.array(datas.data)  
   d\_mean = np.mean(d,axis=0)  
   d\_mean\_correct =np.array([np.format\_float\_positional(x, precision=6) for x in d\_mean ])  
   print(d\_mean\_correct, ' - NumPy')



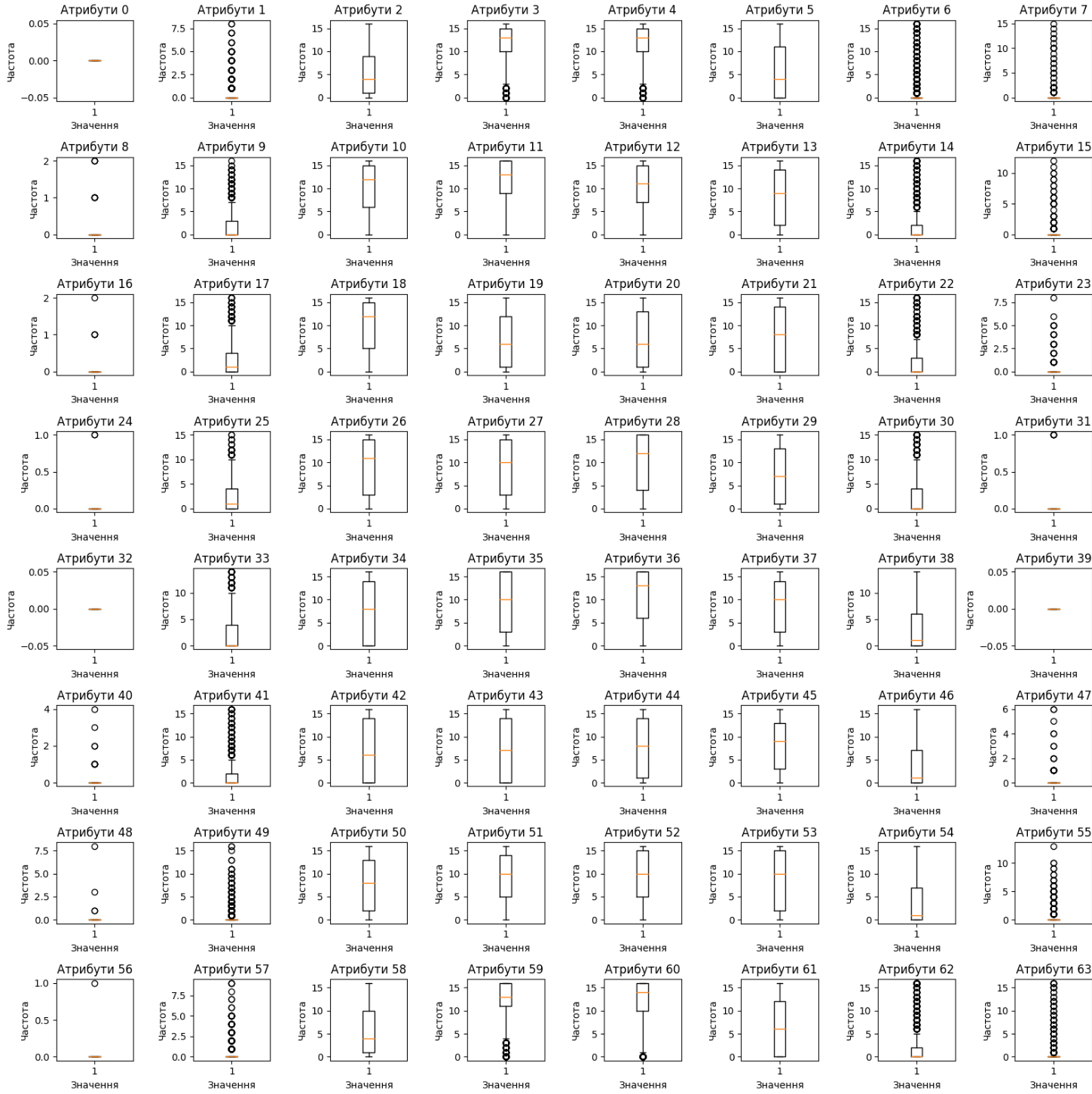
* 1. Побудуйте гістограми для кожного з атрибутів за допомогою matplotlib. Ви можете використовувати цикл for, щоб це зробити в одній команді.

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plot  
from sklearn.datasets import load\_digits  
datas = load\_digits()  
df = pd.DataFrame(datas.data)  
  
print(df.mean() , ' - Pandas\n')  
print(df[1].value\_counts())  
datas\_mean = np.array(df.mean)  
d = np.array(datas.data)  
d\_mean = np.mean(d,axis=0)  
d\_mean\_correct =(np.array([np.format\_float\_positional(x, precision=6) for x in d\_mean ]))  
print(d\_mean\_correct, ' - NumPy')  
  
fig, axes = plot.subplots(nrows=8, ncols=8, figsize=(16, 16))  
#print(datas.data)  
#print(df)  
  
for i in range(8):  
 for j in range(8):  
 ax = axes[i, j]  
 attribute = datas.data[:, i \* 8 + j]  
 ax.hist(attribute, bins=20, alpha=0.35)  
 ax.set\_title(f'Атрибути {i \* 8 + j}')  
 ax.set\_xlabel('Значення')  
 ax.set\_ylabel('Частота')  
  
plot.tight\_layout()  
plot.show()



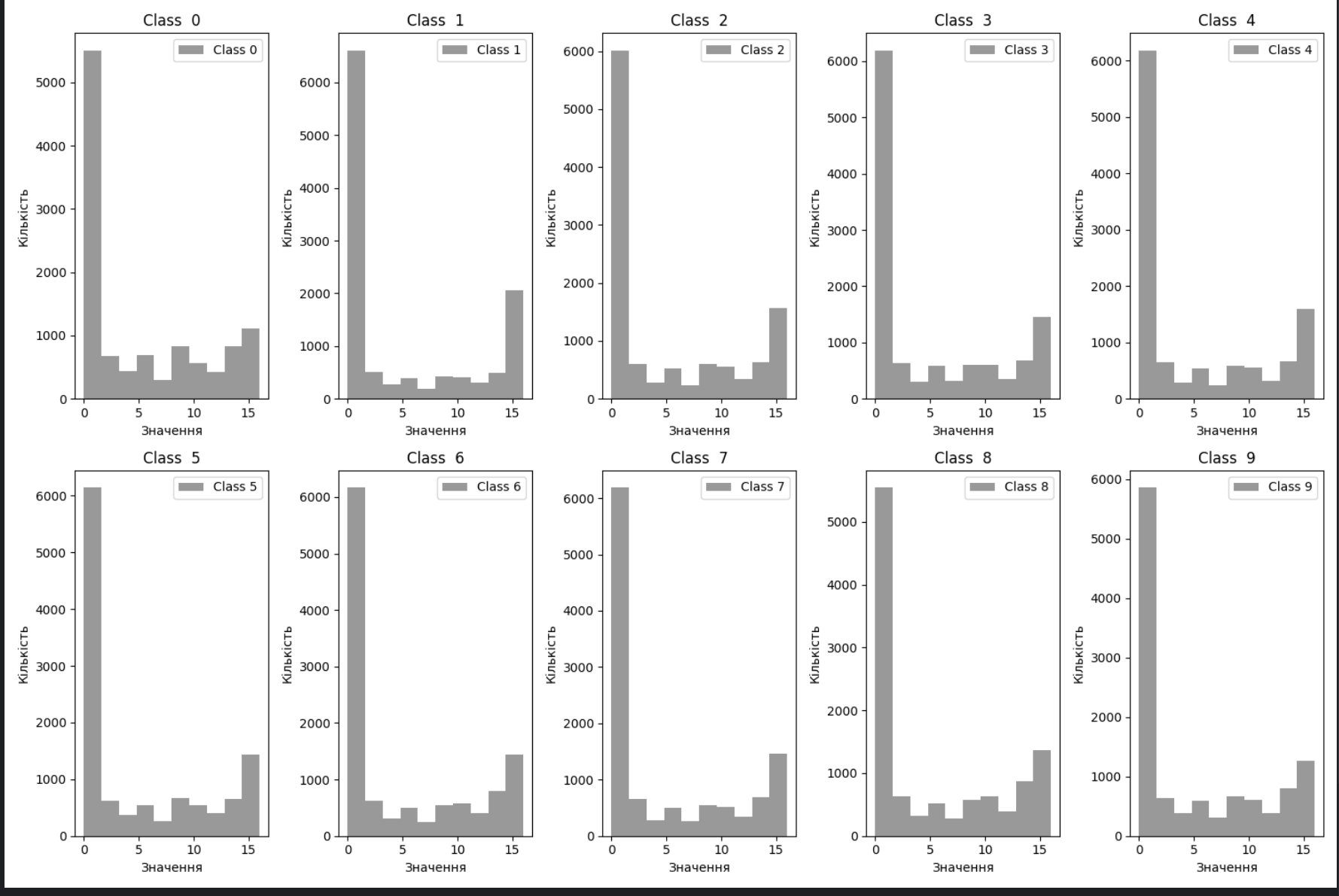
* 1. Побудуйте бокс-плоти для кожного з атрибутів в наборі даних. Для цього ви можете використати pandas метод .boxplot().

import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plot  
from sklearn.datasets import load\_digits  
datas = load\_digits()  
df = pd.DataFrame(datas.data)  
  
print(df.mean() , ' - Pandas\n')  
print(df[1].value\_counts())  
datas\_mean = np.array(df.mean)  
d = np.array(datas.data)  
d\_mean = np.mean(d,axis=0)  
d\_mean\_correct =(np.array([np.format\_float\_positional(x, precision=6) for x in d\_mean ]))  
print(d\_mean\_correct, ' - NumPy')  
  
fig, axes = plot.subplots(nrows=8, ncols=8, figsize=(16, 16))  
#print(datas.data)  
#print(df)  
  
for i in range(8):  
  
 for j in range(8):  
 ax = axes[i, j]  
 attribute = datas.data[:, i \* 8 + j]  
 #ax.hist(attribute, bins=20, alpha=0.35)  
 ax.boxplot(attribute)  
 ax.set\_title(f'Атрибути {i \* 8 + j}')  
 ax.set\_xlabel('Значення')  
 ax.set\_ylabel('Частота')  
  
plot.tight\_layout()  
plot.show()



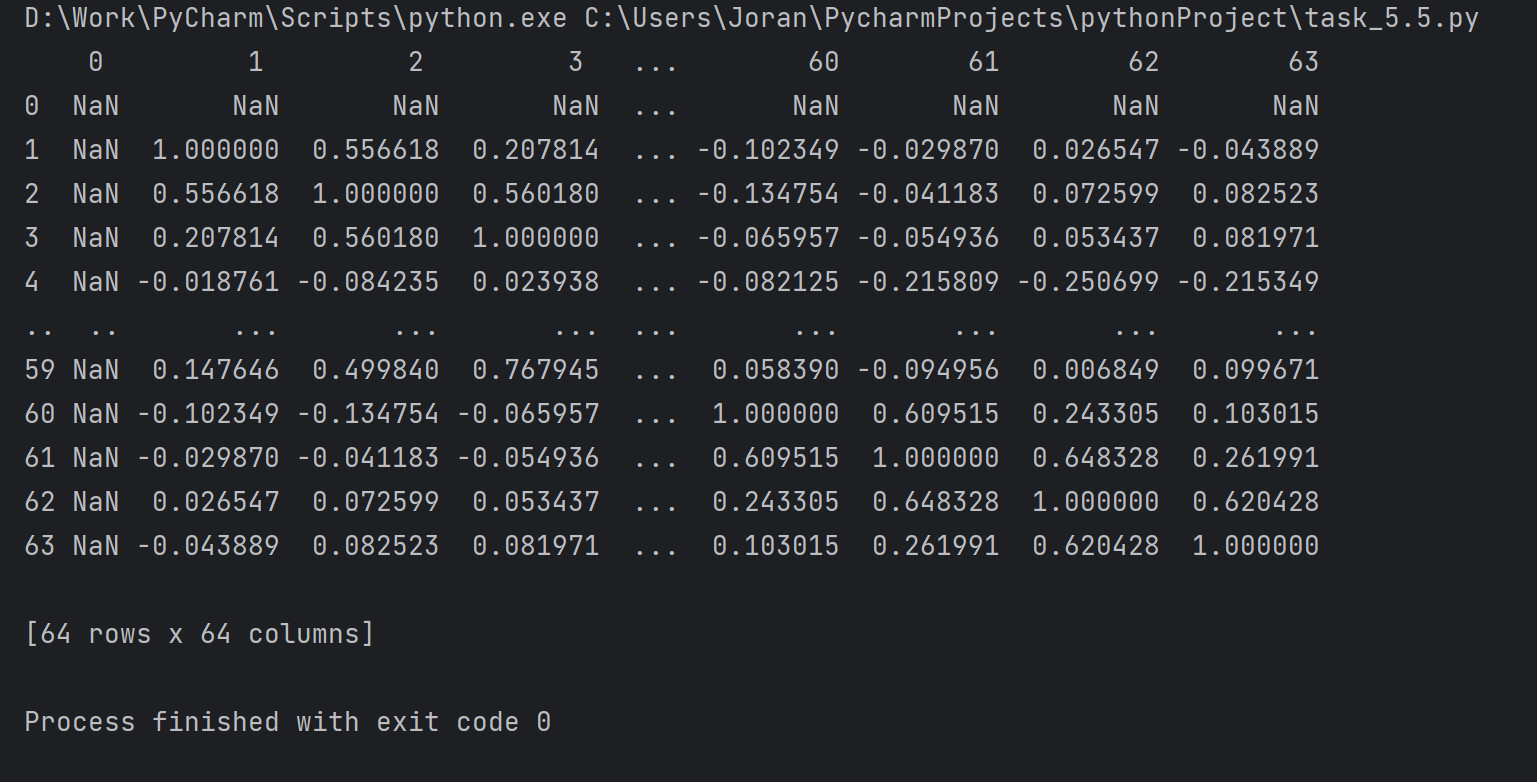
* 1. Порівняйте розподіли атрибутів між різними класами. Це можна зробити, відобразивши гістограми або бокс-плоти для кожного класу окремо. Зверніть увагу на будь-які відмінності, які ви можете спостерігати.

import matplotlib.pyplot as plt  
from sklearn.datasets import load\_digits  
digits = load\_digits()  
datas = digits.data  
images = digits.images  
target = digits.target  
  
  
  
class\_samples = {}  
for class\_l in range(10):  
 class\_samples[class\_l] = images[target == class\_l]  
  
print(class\_samples)  
print(len(class\_samples[2]))  
  
plt.figure(figsize=(15, 10))  
  
for class\_label in range(10):  
 samples\_for\_class = class\_samples[class\_label]  
  
 attribute = samples\_for\_class.reshape(-1)  
 print('==============')  
 print(attribute)  
 print(len(attribute),f'Number {class\_label}')  
 plt.subplot(2, 5, class\_label + 1)  
 plt.hist(attribute, bins=10, alpha=0.4,color='black', label=f'Class {class\_label}')  
 plt.xlabel('Значення')  
 plt.ylabel('Кількість')  
 plt.title(f'Class {class\_label}')  
 plt.legend()  
  
plt.tight\_layout()  
plt.show()



* 1. Обчисліть кореляційну матрицю для атрибутів в наборі даних. Ви можете використати метод .corr() в pandas DataFrame.

import pandas as pd  
from sklearn.datasets import load\_digits  
  
digits = load\_digits()  
df = pd.DataFrame(digits.data)  
correlation = df.corr()  
  
print(correlation)



**Висновок:** Ми ознайомитись з основами Python та додатковими бібліотеками для наукового обчислення - numpy, scipy, matplotlib.

**Відповідь на контрольні питання.**

1. **Які основні відмінності Python від інших мов програмування? Що таке відступи в Python?**

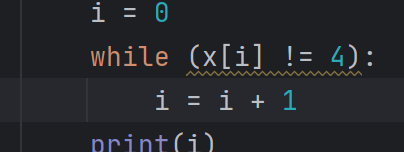
• Python є інтерпретованою мовою, що означає, що виконується без попередньої компіляції.

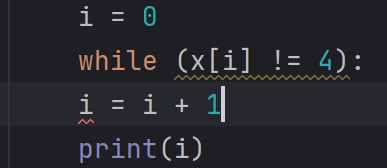
• Python має чіткий та зрозумілий синтаксис, який зробив його популярним для початківців та експертів.

• Python має багатий набір вбудованих типів даних та величезну кількість сторонніх бібліотек.

• Підтримує динамічну типізацію даних

відступи в Python - Python використовує відступ для позначення блоку коду.

Правильно: 

Неправильно: 

1. **Що таке цикли в Python? Поясніть різницю між циклами for та while.**

Цикл – це код у тілі виконується лише доти, доки дотримано умову.

У while умова задається явно. Для перебирається кожен елемент колекції.

1. **Опишіть роботу умовних конструкцій в Python. Як вони використовуються?**

Умовні конструкції в Python - це спосіб виконання різних блоків коду в залежності від заданої умови. Умовні конструкції використовуються для керування логікою програми і прийняття рішень в залежності від виконання певних умов.

1. **Що таке список в Python? Як можна створити список? Як можна змінити елементи списку?**

Списки в Python – це впорядковані колекції елементів. Вони можуть містити будь-який тип даних і підтримують додавання та видалення елементів.

1. lst = [1, 2, 3, 4, 5]   
   print(lst[0]) # виводить перший елемент списку   
   lst.append(6) # додає 6 в кінець списку
2. **Які основні типи даних у Python? Наведіть приклади кожного типу.**

* int (цілі числа): Наприклад, 5, -10, 1000.
* float (дробові числа): Наприклад, 3.14, 0.01, -2.5.
* str (рядки): Наприклад, "Hello, World!", 'Python'.
* list (списки): Колекція об'єктів, яка дозволяє зберігати дані впорядковано та зі змінюваною довжиною.
* bool (булеві значення): Має лише два можливих значення - True (правда) та False (хибно).
* dict (словники): Колекція пар ключ-значення, де кожен ключ унікальний.

a = 1 # - цілі числа

b = 0.1 # - дробові числа  
c = "Hello World" # - рядки  
d = [1,2,3,4,5] # - списки  
f = {"good": 2, "bad":1} # - словники

is\_true = True # булеві

1. **Що таке NumPy? Які основні функції бібліотеки?**

NumPy (Numerical Python) – це фундаментальний пакет для наукових обчислень в Python. Він містить, зокрема, потужний N-вимірний масив об'єктів та інструменти для роботи з цими масивами.

Функції, які надає NumPy

* функції для виконання операцій над масивами;
* функції введення і виведення даних;
* базові математичні функції;
* лінійна алгебра;
* робота з поліномами;
* статистика;
* генерація випадкових чисел і розподілів;
* дискретне перетворення Фур'є;
* базові фінансові функції;
* функції для роботи з датою і часом;
* логічні функції;
* функції для роботи з рядками;
* функції для роботи з множинами;
* функції пошуку, сортування та підрахунку елементів;
* віконні функції.
* функції створення масивів;

1. **Як створити масив в NumPy? Як змінити форму масиву?**

Створення масиву в NumPy

import numpy as np # Імпортуємо бібліотеку NumPy   
  
# Створення одновимірного масиву  
arr1 = np.array([1, 2, 3, 4, 5])  
  
# Створення двовимірного масиву (матриці)  
arr2 = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])  
  
# Створення масиву з послідовності чисел  
arr3 = np.arange(0, 10, 2) # Масив чисел від 0 до 10 (не включаючи) з кроком 2  
  
# Створення масиву з нулів або одиниць  
zeros\_array = np.zeros((3, 4)) # Масив з нулів розміром 3x4  
ones\_array = np.ones((2, 2)) # Масив з одиниць розміром 2x2

Змінення форми масиву

arr1 = np.array([1, 2, 3, 4, 5])  
  
# Зміна форми масиву на двовимірний (матрицю)  
arr2 = arr1.reshape(5, 1)

1. **Що таке pandas? Які основні типи даних в pandas?**

Pandas – це відкрита бібліотека Python, що надає високопродуктивні, гнучкі структури даних і інструменти аналізу даних. Особливо відома своїми структурами даних DataFrame та Series.

DataFrame: DataFrame - це двовимірна таблиця з даними, яка складається з рядків та стовпців. Вона надає індекси для рядків та назви стовпців, що дозволяє звертатися до даних за назвами стовпців. DataFrame подібний до таблиці в базі даних або ексель-аркушу.

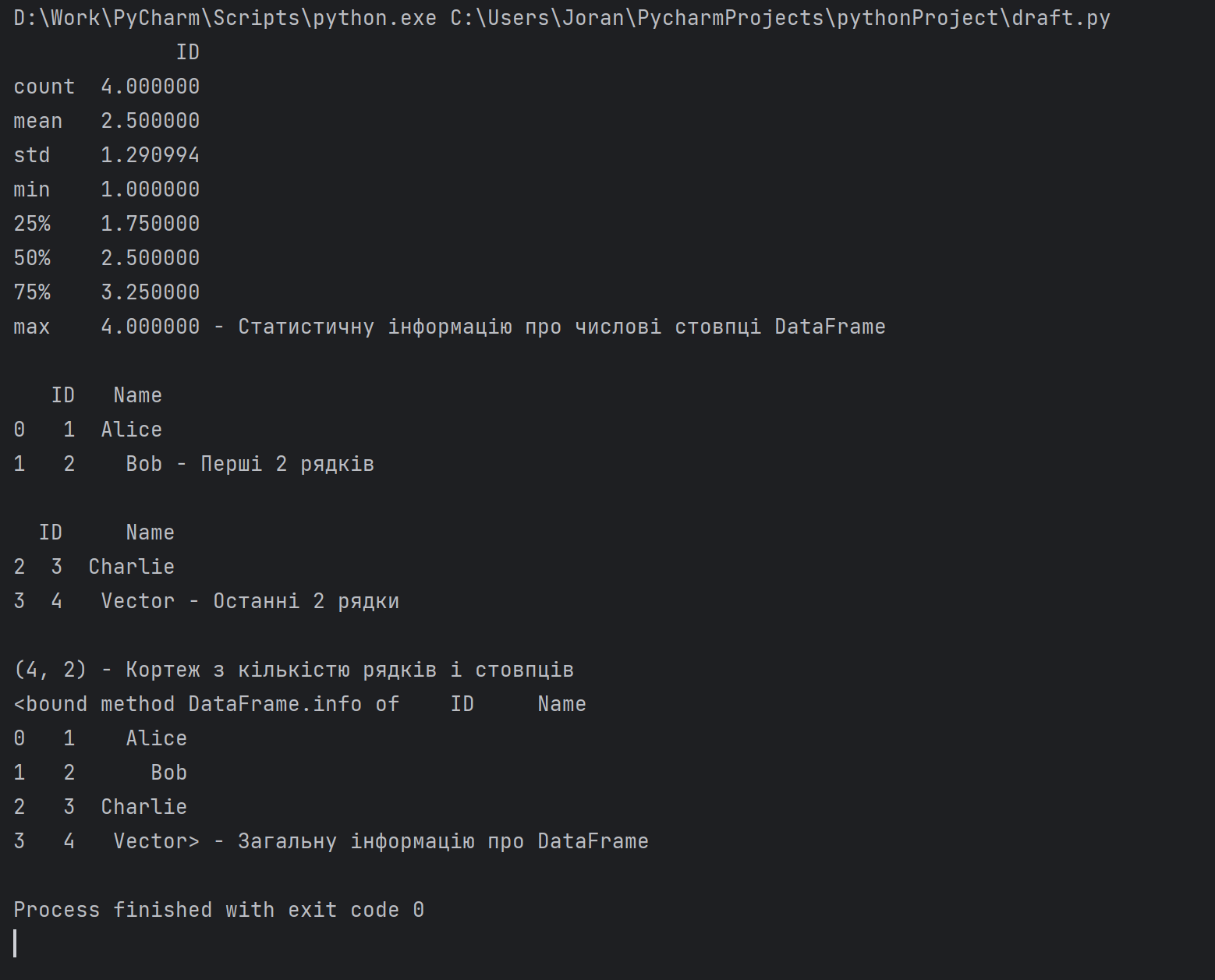
Series: Series - це одновимірний масив даних, який може містити дані одного типу (наприклад, числа або рядки). У Series також є індекси, які дозволяють легко звертатися до даних.

1. **Як можна створити DataFrame в pandas? Як отримати інформацію про DataFrame?**

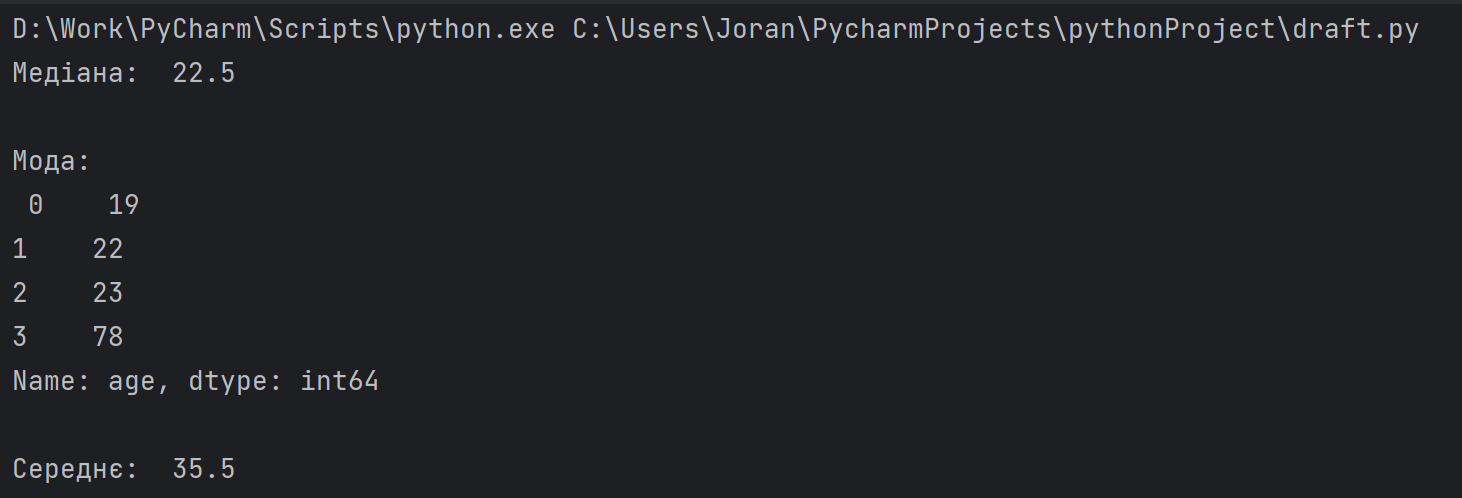
Створення DataFrame в pandas

import pandas as pd  
import numpy as np  
# Створення DataFrame зі списку списків  
data = [[1, 'Alice'], [2, 'Bob'], [3, 'Charlie'],[4,'Vector']]  
df1 = pd.DataFrame(data, columns=['ID', 'Name'])  
  
# Створення DataFrame зі словника  
data = {'ID': [1, 2, 3, 4 ], 'Name': ['Alice', 'Bob', 'Charlie','Vector']}  
df2 = pd.DataFrame(data)  
  
# Створення DataFrame з NumPy масиву  
  
data = np.array([[1, 'Alice'], [2, 'Bob'], [3, 'Charlie'],[4,'Vector']])  
df3 = pd.DataFrame(data, columns=['ID', 'Name'])  
  
  
  
print(df1.describe(),'- Cтатистичну інформацію про числові стовпці DataFrame\n')  
print(df2.head(2),'- Перші 2 рядків\n')  
print(df3.tail(2),'- Останні 2 рядки\n')  
print(df1.shape,'- Кортеж з кількістю рядків і стовпців ')  
print(df2.info,'- Загальну інформацію про DataFrame ')

отримання інформацію про DataFrame



1. **Як обчислити основні статистики в pandas DataFrame, такі як середнє, медіана, мода?**
2. df = pd.DataFrame(data)  
     
   print('Медіана: ',df['age'].median(),'\n')  
   print(f'Мода: \n',df['age'].mode(),'\n')  
   print(f'Середнє: ',df['age'].mean(),'\n')

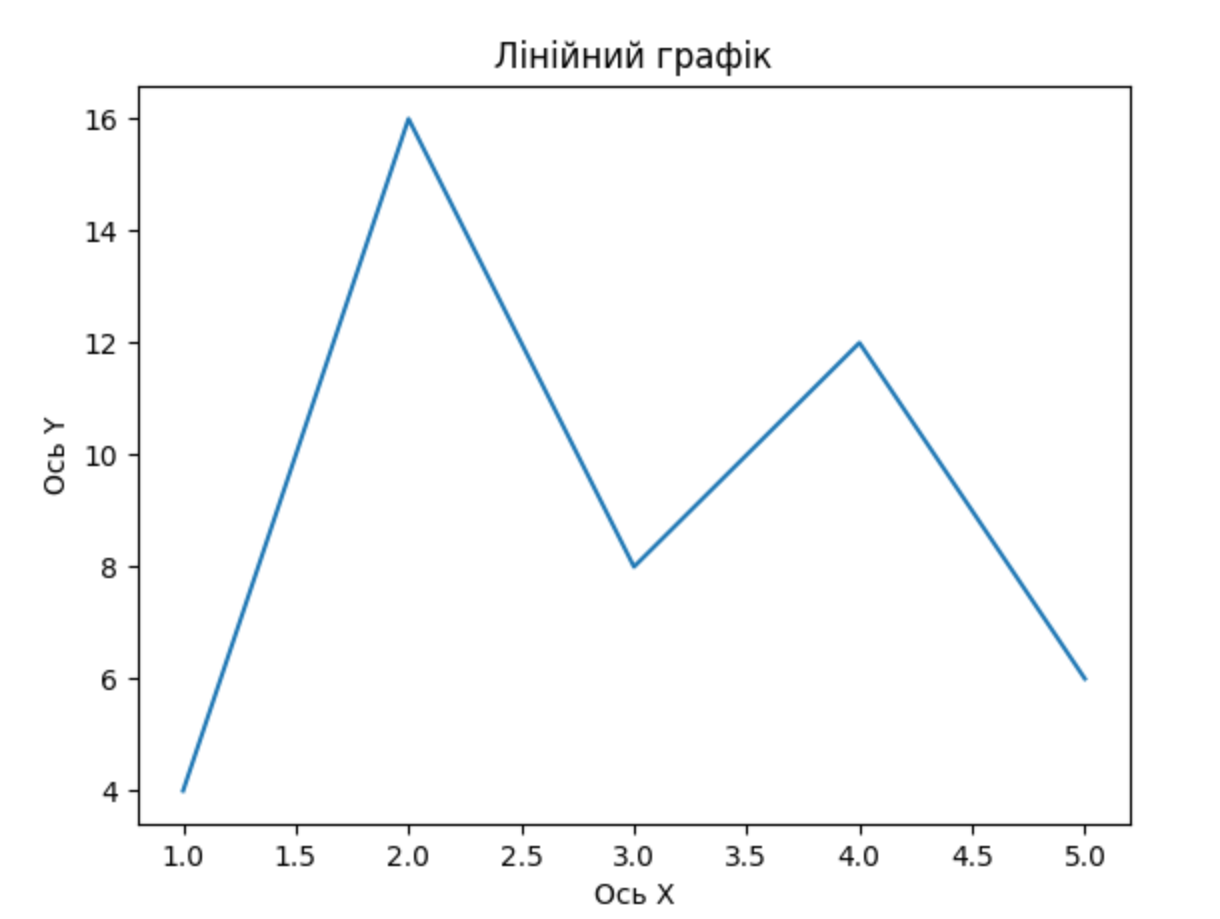
****

1. **Що таке matplotlib? Для чого вона використовується?**

Matplotlib – це бібліотека Python для створення статичних, анімованих та інтерактивних візуалізацій у Python.

1. **Як побудувати лінійний графік в matplotlib? Як додати назви до осей і заголовок графіку?**

x = [1, 2, 3, 4, 5]  
y = [4, 16, 8, 12, 6]  
  
plt.plot(x, y)  
plt.xlabel('Ось X')  
plt.ylabel('Ось Y')  
plt.title('Лінійний графік')  
plt.show()

****

1. **Як побудувати гістограму, box-plot або scatter plot в matplotlib?**

гістограму

data = np.random.randn(1000)  
plt.hist(data, bins=30)  
plt.show()

scatter plot

x = np.random.rand(100)  
y = np.random.rand(100)  
plt.scatter(x, y)  
plt.show()

box-plot

1. plt.boxplot(data)  
   plt.show()
2. **Що таке scikit-learn? Які основні задачі можна вирішити з допомогою цієї бібліотеки?**

Scikit-learn – це бібліотека Python для машинного навчання. Вона надає широкий спектр алгоритмів навчання, включаючи класифікацію, регресію, кластеризацію та “редукцію” розмірності.

1. **Які вбудовані датасети доступні в scikit-learn? Наведіть приклади.**

* Iris dataset: Цей класичний набір даних (надалі – датасет) включає інформацію про 150 зразків ірису, включаючи довжину та ширину чашолистиків та пелюсток ірису для кожного зразка, а також вид ірису.
* Digits dataset: Цей датасет містить зображення від 0 до 9, кожне зображення 8x8 пікселів. Цей датасет використовується в задачах класифікації.
* Wine dataset: Датасет містить хімічні характеристики різних видів вина.
* Breast cancer dataset: Даний датасет містить характеристики клітин, які використовуються для прогнозування доброякісних або злоякісних пухлин.
* Boston Housing dataset: Датасет містить інформацію про різні характеристики будинків в Бостоні, США, і їх ціну. Цей датасет використовується в задачах регресії.
* Diabetes dataset: Даний датасет включає інформацію про пацієнтів із цукровим діабетом, включаючи їх вік, стать, індекс маси тіла, середній артеріальний тиск та шість показників сироватки крові.

1. **Як можна завантажити вбудований датасет з scikit-learn? Як отримати доступ до даних і міток цільового класу?**

Завантаженням датасету digits.

Після завантаження датасету ви можете отримати доступ до даних та міток цільового класу наступним чином:

from sklearn.datasets import load\_digits  
digits = load\_digits()  
  
  
datas = digits.data  
  
targets = digits.target  
print('Завантаження датасету -',datas)  
print('Мітки цільового класу -',targets)

1. **Які типи графіків можна створити за допомогою matplotlib?**

За допомогою matplotlib можна створити:

* Лінійний графік (Line plot): Це основний тип графіку, який використовується для відображення зміни значень в часі. Лінійні графіки зазвичай використовуються для візуалізації трендів в часі.
* Нестандартні типи графіків, такі як ступінчасті графіки (step plot).
* Scatter Plot (діаграма розсіювання): Графік для візуалізації точкових даних, корисний для виявлення залежностей та кластерів.
* Histograms (гістограми): Використовується для візуалізації розподілу числових даних та оцінки їхньої частоти.
* Bar Chart (гістограми): Графік для візуалізації категоріальних даних, таких як кількість елементів у кожній категорії.
* Pie Chart (Кругова діаграма): Кругова діаграма використовується для відображення пропорцій між частинами цілого.
* Поверхневі графіки (Surface Plots): Використовуються для візуалізації тривимірних даних на поверхні.
* Теплова карта (Heatmap): Використовується для візуалізації матриць даних, де кожен елемент представлений кольором.
* Box Plot (ящик з вусами): Діаграма для візуалізації статистичних параметрів розподілу даних.
* Violin Plot (скрипковий графік): Комбінує інформацію з гістограми та ящика з вусами для відображення розподілу даних.
* Субплоти (subplots): Для створення декількох графіків на одному рисунку можна використовувати метод subplot.