Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Лабораторна робота №1

Аналіз даних з використанням мови Python

Тема: Базове знайомство з бібліотекою NumPy

Виконав

студент групи ІП-11:

Панченко С. В.

Перевірив:

Тимофєєва Ю. С

Київ 2023

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Ознайомитись з основними можливостями роботи з масивами бібліотеки NumPy, визначити статистичні характеристики.

2 ЗАВДАННЯ

Створити програму, яка:

1. Генерує випадкові і невипадкові масиви різними способами, зазначеними в теоретичних відомостях.
2. Демонструє звернення до елементів масиву за допомогою індексів, в тому числі від’ємних; виділення підмасивів як одновимірних, так і багатовимірних масивів.
3. Демонструє основні арифметичні операції над масивами, а також роботу методів reduce, accumulate, outer.
4. Вираховує статистичні характеристики, а саме, мінімальне і максимальне значення, вибіркові середнє, дисперсію, середньоквадратичне відхилення, медіану та 25 та 75 персентилі, величини ширина пелюстки (petal_width) з набору даних щодо квіток ірису (iris.csv).

Оформити звіт. Звіт повинен містити:

— титульний лист;

— код програми;

— результати виконання коду;

Продемонструвати роботу програми та відповісти на питання стосовно теоретичних відомостей та роботи програми.

3 ВИКОНАННЯ

3.1 Створення масивів

Імпортуємо модуль NumPy та покажемо різні можливості створення масивів.

Покажемо можливість створення цілочисельної матриці 3x3.

```
In [1]: import numpy as np
        np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

Out[1]: array([[1, 2, 3],
               [4, 5, 6],
               [7, 8, 9]])
```

Рисунок 3.1 - Створення матриці 3x3

Покажемо можливість створення матриці різних типів, наприклад, float та int.

```
In [3]: a_int = np.array([1, 2, 3, 4], dtype='float')
        a_float = np.array([1, 2, 3, 4], dtype='int32')
        print(a_int)
        print(a_float)

        [1. 2. 3. 4.]
        [1 2 3 4]
```

Рисунок 3.2 - Створення масивів типів float та int32

NumPy має можливість автоматичного створення масивів, покажемо це.

Покажемо можливість створення масиву, задаючи початкове, кінцеве значення та крок.

```
In [4]: np.arange(1,8,3)

Out[4]: array([1, 4, 7])
```

Рисунок 3.3 - Створення масиву, де початок - 1, кінець - 8, крок - 3

Продемонструємо можливість створення масиву цілочисельних одиниць за допомогою методу ones.

```
In [5]: np.ones(9, dtype=int)

Out[5]: array([1, 1, 1, 1, 1, 1, 1, 1, 1])
```

Рисунок 3.4 - Одновимірний масив одиниць з 9-ти елементів

Продемонструємо можливість створення двовимірного масиву нулів за допомогою методу zeros.

```
In [6]: np.zeros((3, 4), dtype='float')

Out[6]: array([[0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.]])
```

Рисунок 3.5 - Матриця 3x4 нулів типу float

Створимо масив з 6-ти значень, що рівномірно розподілені між 0 та 1.

```
In [12]: np.linspace(0, 1, 6)

Out[12]: array([0. , 0.2, 0.4, 0.6, 0.8, 1. ])
```

Рисунок 3.6 - Одновимірний масив з шести значень, що рівномірно розподілені між 0 а 1

Створимо масив 4x4 з рівномірно розподілених випадкових чисел від 0 до 1.

```
In [13]: np.random.random((4, 4))

Out[13]: array([[0.13033435, 0.74283195, 0.68012242, 0.10431953],
                [0.9159478 , 0.24454688, 0.86073132, 0.78828324],
                [0.94786684, 0.34052187, 0.75032444, 0.4807424 ],
                [0.57201738, 0.75682698, 0.33011881, 0.1185312 ]])
```

Рисунок 3.7 - Масив 4x4 з рівномірно розподілених випадкових чисел від 0 до 1

Створимо матрицю 4x2 випадкових цілих чисел від -10 до 10. Для цього використаємо метод numpy.random.randint.

```
In [15]: np.random.randint(-10, 10, (4, 2))

Out[15]: array([[ -5,   4],
                [-10,   4],
                [  6,  -6],
                [ -4,   2]])
```

Рисунок 3.8 - Матриця 4x2 випадкових цілих чисел від -10 до 10

Створимо масив з 7-ми порожніх елементів, у яких буде випадковий вміст комірок пам'яті.

```
In [16]: np.empty(5)

Out[16]: array([0.2, 0.4, 0.6, 0.8, 1. ])
```

Рисунок 3.9 - Масив з 7-ми порожніх елементів

3.2 Індексація елементів

Покажемо доступ елементів за індексом. Створимо масив елементів, передаючи список цілих чисел. Знайдемо другий елемент масиву.

```
In [19]: a = np.array([1, 2, 3, 4, 5, 6])
         a[1]

Out[19]: 2
```

Рисунок 3.10 - Другий елемент масиву

Знайдемо останній елемент масиву.

```
In [20]: a[-1]

Out[20]: 6
```

Рисунок 3.11 - Останній елемент масиву

Створимо матрицю, передавши список списків цілих чисел. Виділимо одновимірний підмасив, передавши індекс 0 та : для захвати усіх чисел рядка.

```
In [21]: a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
         a[0, :]

Out[21]: array([1, 2, 3])
```

Рисунок 3.12 - Перший рядок матриці

Виділимо двовимірний підмасив, передавши проміжок рядків :2 та стовпців :2.

```
In [22]: a[:2, :2]

Out[22]: array([[1, 2],
                [4, 5]])
```

Рисунок 3.13 - Матриця 2x2 з перших двох елементів перших двох рядків

3.3 Арифметичні операції з масивами

Покажемо додавання (+, np.add), віднімання (-, np.subtract), множення (, np.multiply), ділення (/, np.divide), піднесення до ступеню (*, np.power), ділення за модулем (%, np.mod), зміна знаку на протилежний (-, np.negative).

```
In [23]: a = np.array([1, 2, 3, 4], dtype='float')
         print(a)
         print(a + 4)
         print(a - 4)
         print(a * 4)
         print(a / 4)
         print(a ** 2)
         print(a % 2)
         print(-a)

[1. 2. 3. 4.]
[5. 6. 7. 8.]
[-3. -2. -1.  0.]
[ 4.  8. 12. 16.]
[0.25 0.5  0.75 1.  ]
[ 1.  4.  9. 16.]
[1. 0. 1. 0.]
[-1. -2. -3. -4.]
```

Рисунок 3.14 - Основні арифметичні операції над масивами

Застосуємо операцію reduce, багаторазово застосовує задану операцію до елементів масиву, поки не залишиться один результат. На даному прикладі бачимо, що операція add виконується доти, поки не будуть додані усі елементи в загальну їхню суму.

```
In [25]: a = np.arange(0, 6)
         np.add.reduce(a)

Out[25]: 15
```

Рисунок 3.15 - Застування reduce на операції add

Застосуємо операцію accumulate, яка працює аналогічно, але зберігає проміжні результати. Тут ми побачимо залежність суми від кількості доданих елементів.

```
In [26]: np.add.accumulate(a)

Out[26]: array([ 0,  1,  3,  6, 10, 15])
```

Рисунок 3.16 - Застування accumulate на операції add

Застосуємо операцію outer, яка видає результат застосування операції до всіх пар елементів. Застосуємо функцію multiply та покажемо усі можливі добутки за допомогою матриці.

```
In [27]: np.multiply.outer(a, a)

Out[27]: array([[ 0,  0,  0,  0,  0,  0],
                [ 0,  1,  2,  3,  4,  5],
                [ 0,  2,  4,  6,  8, 10],
                [ 0,  3,  6,  9, 12, 15],
                [ 0,  4,  8, 12, 16, 20],
                [ 0,  5, 10, 15, 20, 25]])
```

Рисунок 3.17 - Матриця усіх можливих добутків елементів між собою

3.4 Виведення простих статистичних характеристик

Зчитаємо датасет iris.csv та виведемо статистичні характеристики, а саме, мінімальне і максимальне значення, вибіркові середнє, дисперсію, середньоквадратичне відхилення, медіану та 25 та 75 персентилі, величини ширини пелюстки (petal_width).

```
In [34]: import pandas as pd
         data = pd.read_csv('data/iris.csv')
         petal = data['petal_length'].values
         print(f'Min: {petal.min()}')
         print(f'Max: {petal.max()}')
         print(f'Mean: {petal.mean()}')
         print(f'Std: {petal.std()}')
         print(f'Variance: {petal.var()}')
         print(f'Median: {np.median(petal)}')
         print(f'Quantile 0.25: {np.quantile(petal, 0.25)}')
         print(f'Quantile 0.75: {np.quantile(petal, 0.75)}')

         Min: 1.0
         Max: 6.9
         Mean: 3.758666666666666
         Std: 1.7585291834055212
         Variance: 3.092424888888889
         Median: 4.35
         Quantile 0.25: 1.6
         Quantile 0.75: 5.1
```

Рисунок 3.18 - Статистичні характеристики величини ширини пелюстки (petal_width)

4 ВИСНОВОК

Під час виконання даної лабораторної роботи я здобув базові навички роботи з масивами, виконання арифметичних операцій та вираховування статистичних характеристик за допомогою пакету NumPy у мові програмування Python.