Міністерство освіти і науки України

Національний університет «Запорізька політехніка»

Кафедра програмних засобів

ЗВІТ

Дисципліна «Розробка прикладних програм»

Робота №2

Тема «Використання бібліотек Python для високопродуктивних наукових обчислень»

Виконав варіант 19

Студент КНТ-122 Онищенко О. А.

Прийняли

Викладач Дейнега Л. Ю.

2024

МЕТА РОБОТИ

Ознайомитися з основними можливостями бібліотек numpu та scipy і навчитися використовувати їх на практиці для виконання високопродуктивних наукових обчислень.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

1. Обчислити визначений інтеграл

$$\int_{\frac{\pi}{7}}^{\frac{\pi}{4}} \frac{\cos 2x + \sin^2 x}{\sin 3x}$$

2. Побудувати графік фукнції, знайти її мінімум методом Нелдера-Міда з точністю 10

$$f(x, y) = x^2 + y^2 - 4x - y - xy$$

ТЕКСТИ ФАЙЛІВ

run.py

```
import numpy as np
import scipy as sp
from matplotlib import pyplot as plt

def one():
    '''
    Обчислити визначений інтеграл
    '''
    PI=np.pi
    bounds={
        'lower':PI/4,
        'upper':PI/7
    }
```

```
            target_function=lambda
x:(np.cos(2*x)+np.sin(x)**2)/np.sin(3*x)

solution=sp.integrate.quad(target_function,bounds['lower'],bounds['uppe
r'])
            print(solution)
        def two():
            '''
            Побудувати графік функції
            '''
            target_function=lambda x,y:x**2+y**2-4*x-y-x*y
            b=np.arange(-3,3,0.1)
            d=np.arange(-3,3,0.1)
            B,D=np.meshgrid(b,d)
            F=target_function(B,D)

            fig=plt.figure()
            ax=fig.add_subplot(111,projection='3d')
            ax.plot_surface(B,D,F)
            plt.show()
            '''
            Знайти її мінімум методом Нелдера-Міда: точність 10
            '''
            def f(params):
                x,y=params
                return x**2+y**2-4*x-y-x*y
            min_value=sp.optimize.minimize(f,[3,3],method='Nelder-Mead')
            print([float(f'{val:.10f}') for val in
min_value.x.tolist()])
        one()
        two()
```

РЕЗУЛЬТАТИ ВИКОНАННЯ

Вивід у консоль:

```
(-0.24129889066690466, 2.6789558423490586e-15)
[2.999986397, 2.0000283497]
```

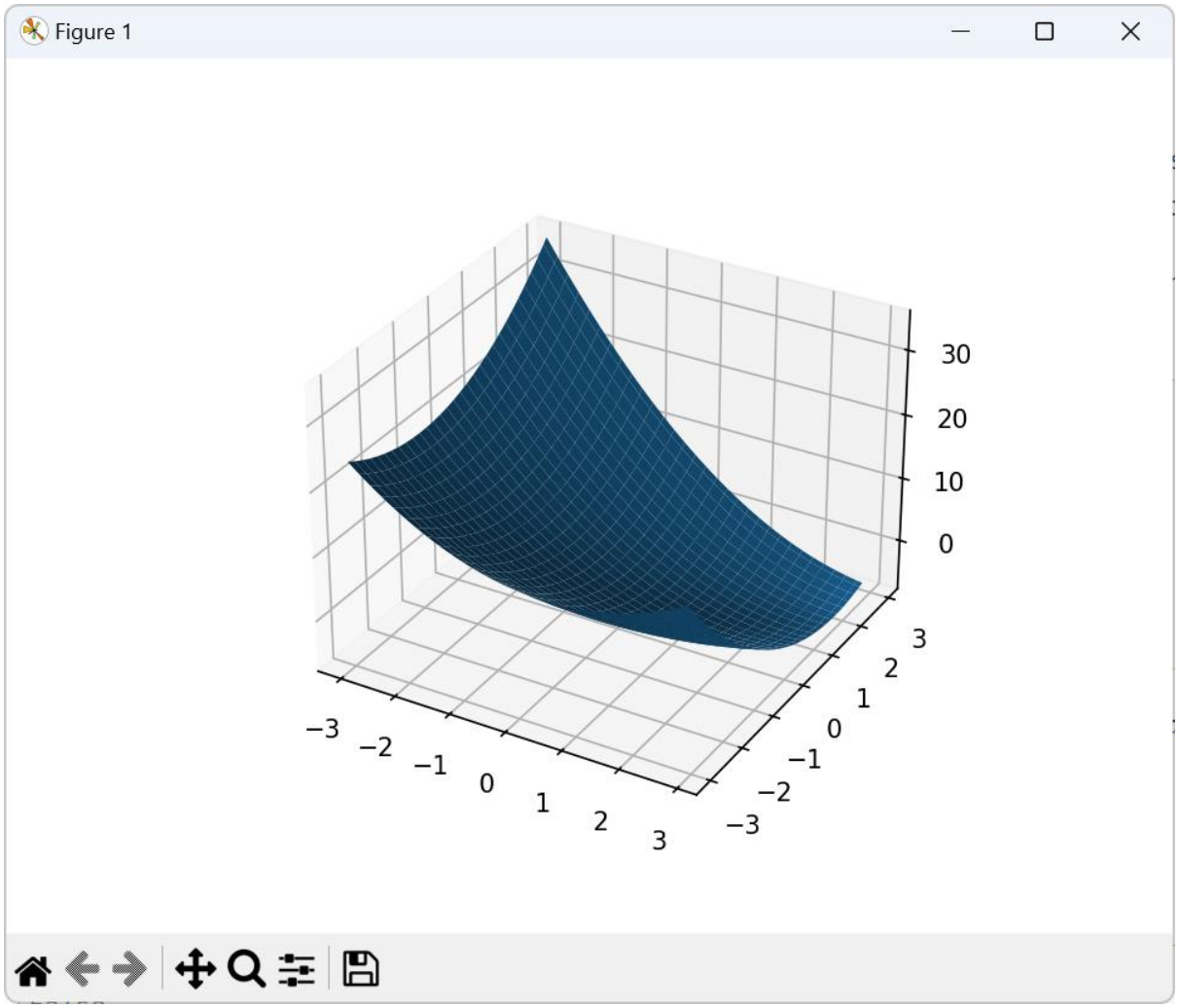

Рисунок 1.1 – Графік функції

КОНТРОЛЬНІ ПИТАННЯ

Слава ІСУСУ ХРИСТУ

Засоби бібліотеки numpy

Бібліотека numpy дає засоби роботи з масивами: тип даних `ndarray`. Аби використати бібліотеку потрібно встановити її: у терміналі

```
pip install numpy
```

Далі імпортувати її: у .py файлі

```
import numpy as np
```

Аби використати масив можна передати методу `array` список:

```
a=np.array([3,7,12,40])
print(a)
> [ 3  7 12 40]
```

Аби взяти елемент масиву типу `ndarray` можна використати індекси:

```
print(a[-1])
> 40
```

Масиви типу `ndarray` також можна змінювати як і списки:

```
a[-1]=12
print(a)
> [ 3  7 12 12]
```

Джерело

Засоби бібліотеки scipy

Спеціальні функції (scipy.special)

Інтеграція (scipy.integrate)

Оптимізація (scipy.optimize)

Інтерполяція (scipy.interpolate)

Перетворення Фур'є (scipy.fft)

Обробка сигналів (scipy.signal)

Лінійна алгебра (scipy.linalg)

Розріджені масиви (scipy.sparse)

Розріджені задачі на власні значення з ARPACK

Процедури стиснення розріджених графів (scipy.sparse.csgraph)

Просторові структури даних та алгоритми (scipy.spatial)

Статистика (scipy.stats)

Багатовимірна обробка зображень (scipy.ndimage)

Файловий ввід/вивід (scipy.io)

Джерело

Призначення бібліотеки matplotlib

Надання інструментів графічного подання даних для легшого аналізу та розуміння

Джерело

Типи графіків через matplotlib

1. Звичайний графік

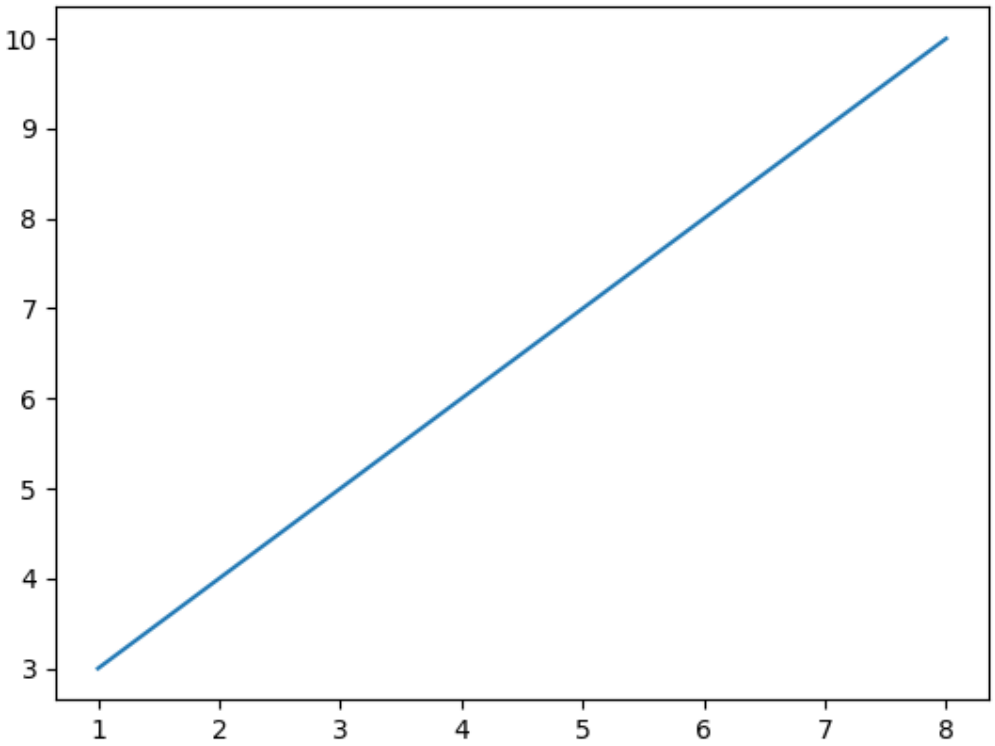

Рисунок 2.1 – Вигляд звичайного графіку

2. Підграфік

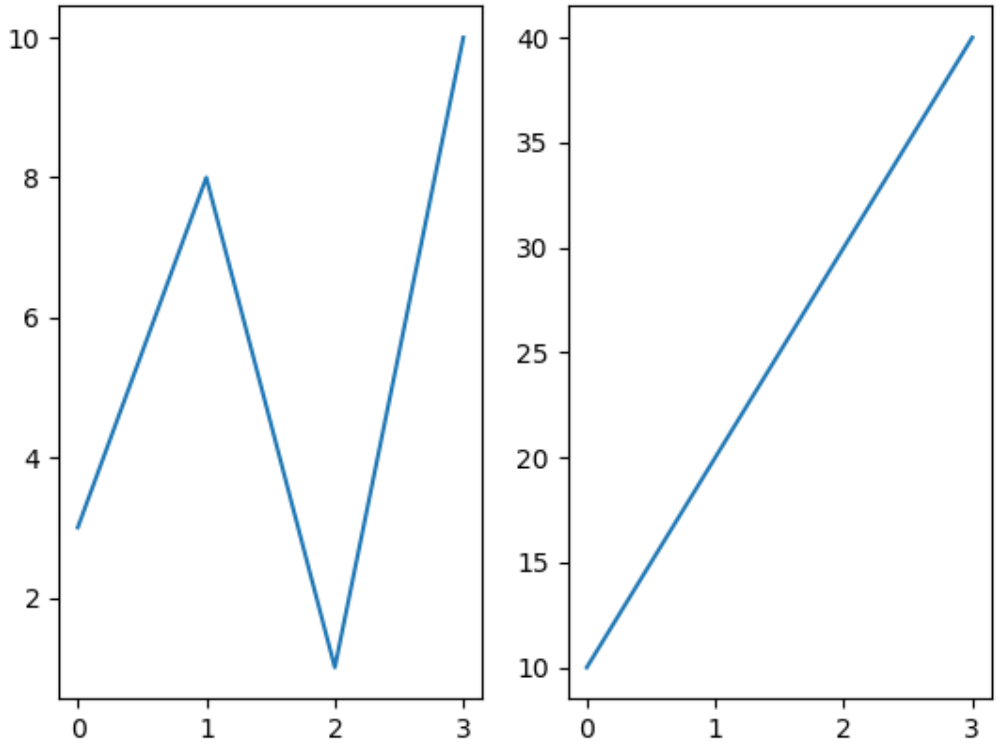

Рисунок 2.2 – Вигляд підграфіку

3. Стовпчаста діаграма

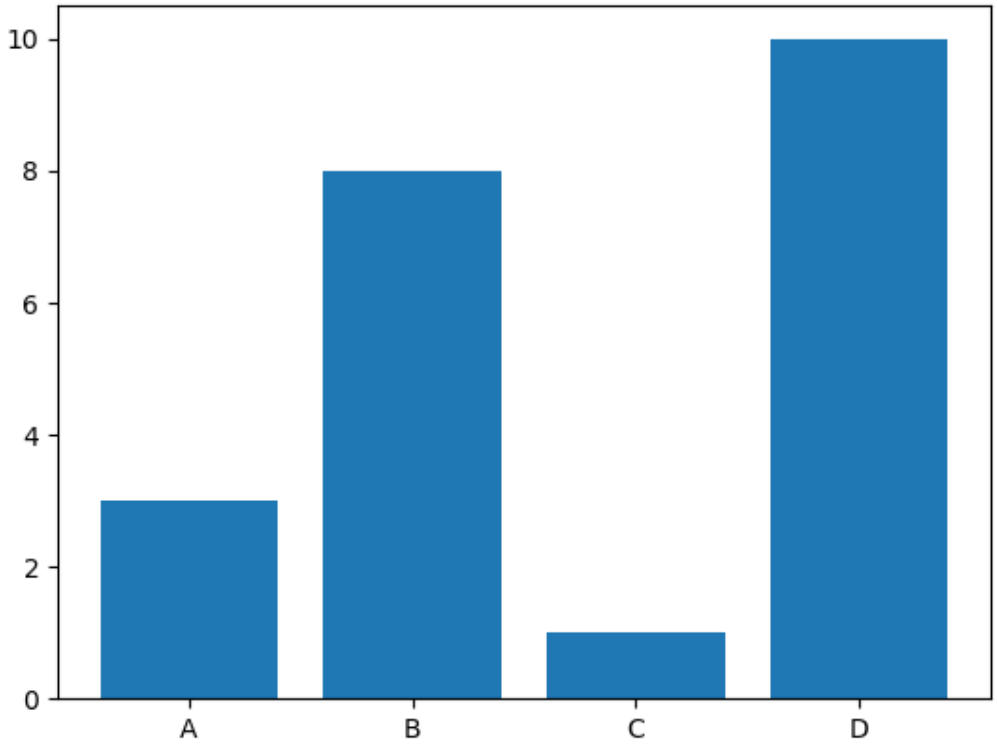

Рисунок 2.3 – Вигляд стовпчастої діаграми

4. Горизонтальна стовпчаста діагарма

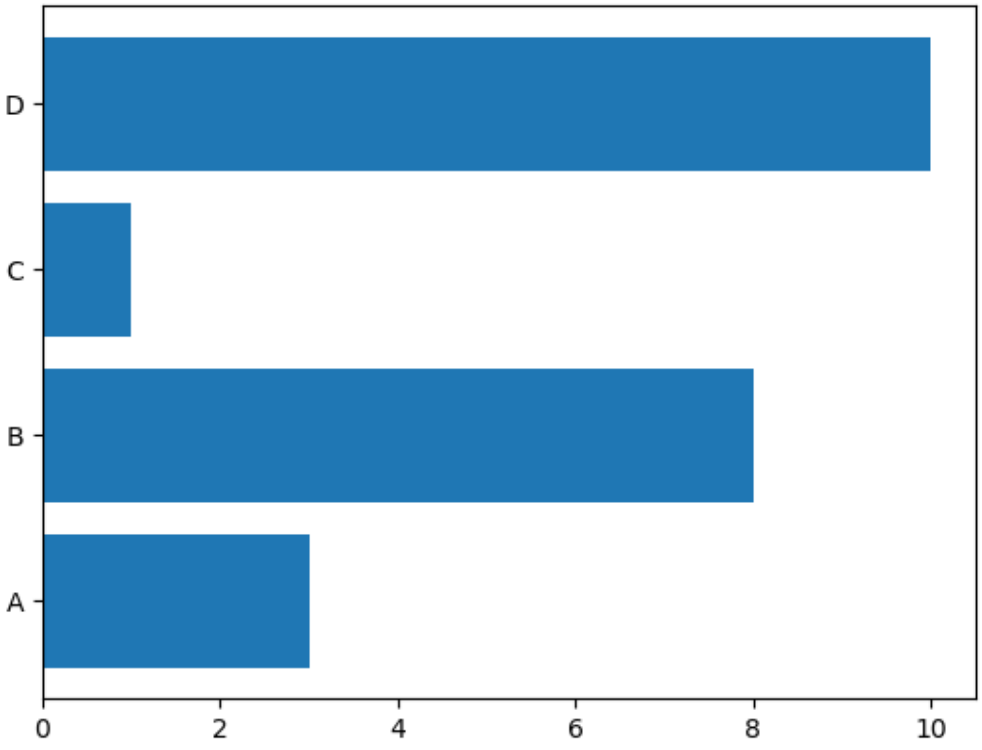

Рисунок 2.4 – Вигляд горизонтальної стовпчастої діаграми

5. Гістограма

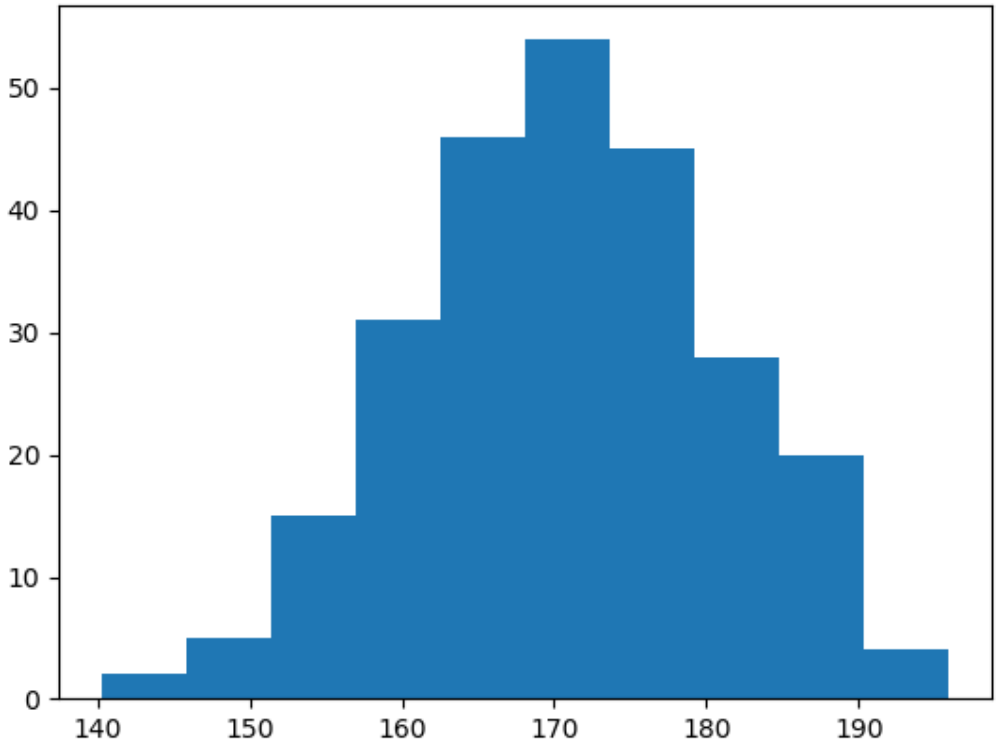

Рисунок 2.5 – Вигляд гістограми

6. Кругова діаграма

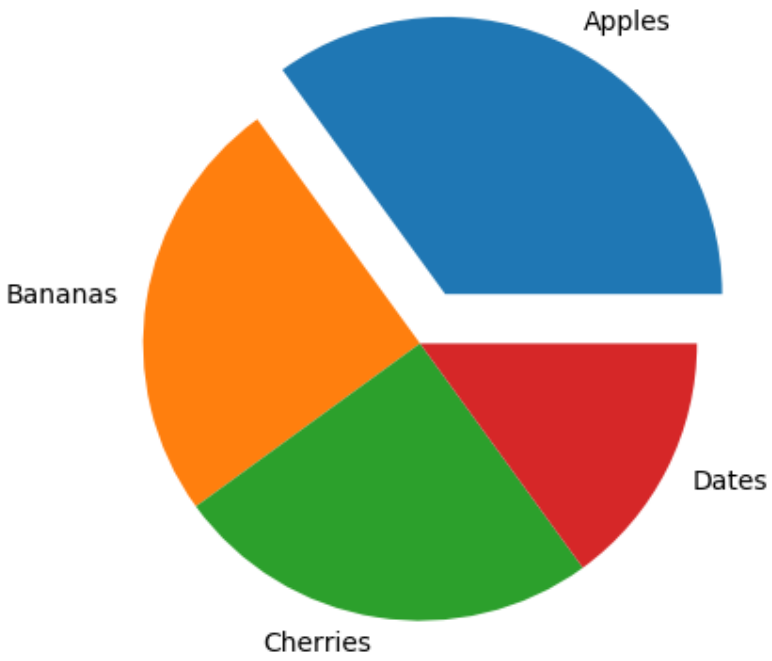

Рисунок 2.6 – Вигляд кругової діаграми

Налаштування графіку matplotlib

Метод `matplotlib.pyplot.plot` приймає параметри значень x та y.

Якщо значення x не зазначено, будуть використані стандартні: діапазон з кроком 1.

Третім аргументом можна зазначити marker: змінює маркер кожної точки графіку:

```
plt.plot(ypoints, marker = 'o')
```

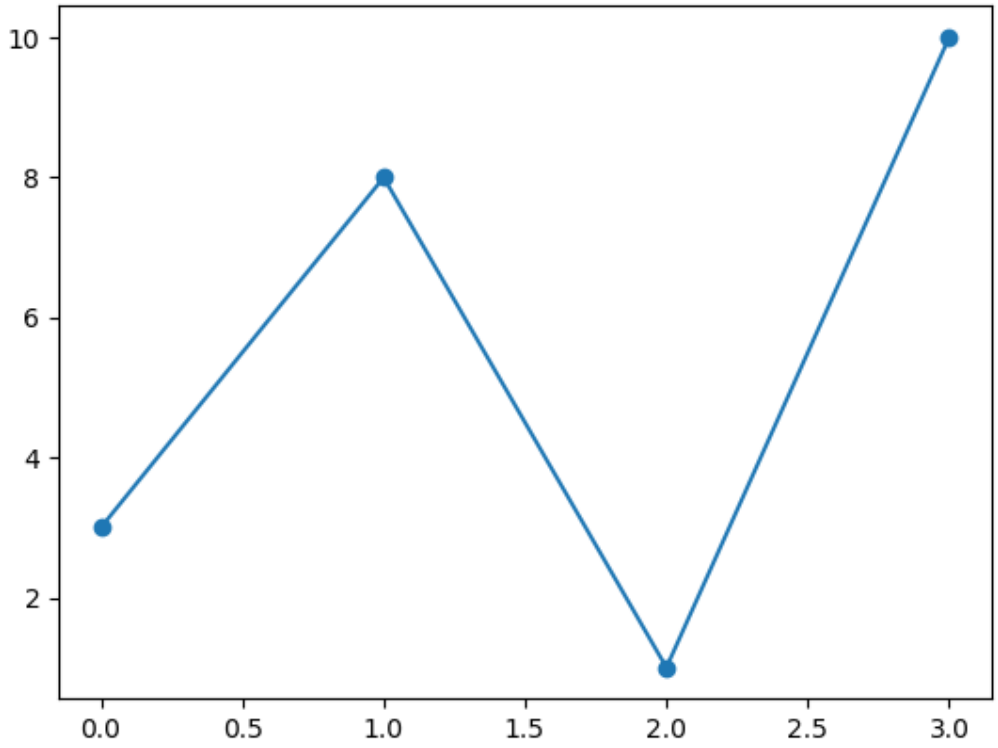

Рисунок 2.7 – Вигляд маркеру o

Різні типи маркерів.

Інший аргумент – fmt: змінює вигляд графіку у форматі маркер/лінія/колір:

```
plt.plot(ypoints, 'o:r')
```

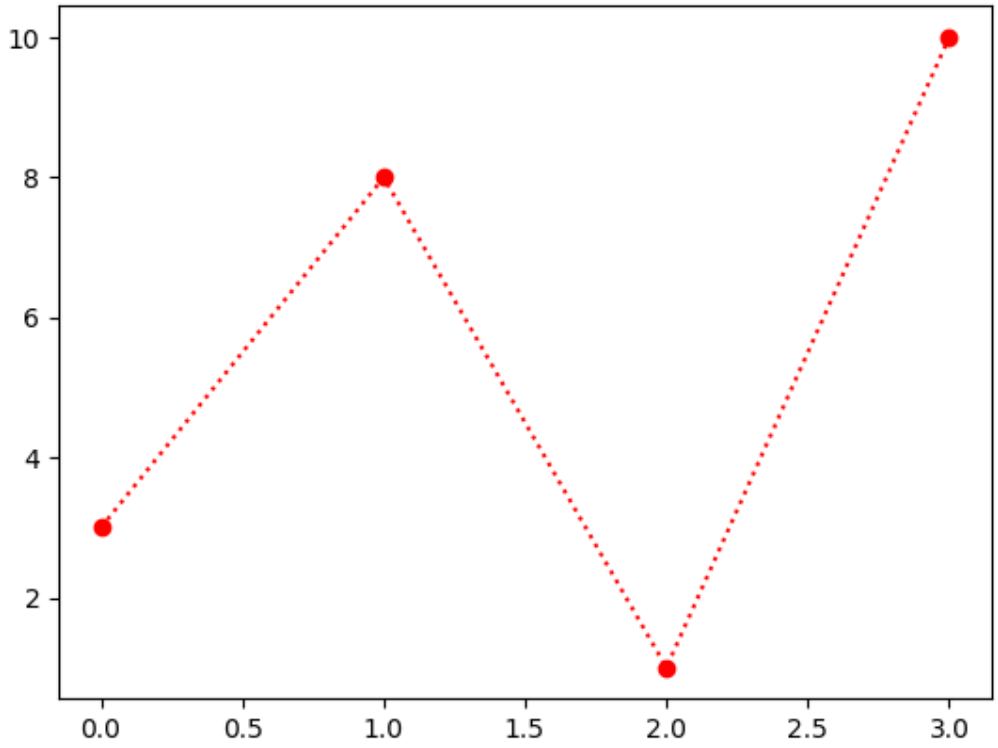

Рисунок 2.8 – Вигляд налаштованого графіку типу o:r

Різні типи ліній.

Різні значення кольорів.

Ще один аргумент – ms: змінює розмір маркера:

```
plt.plot(ypoints, marker = 'o', ms = 20)
```

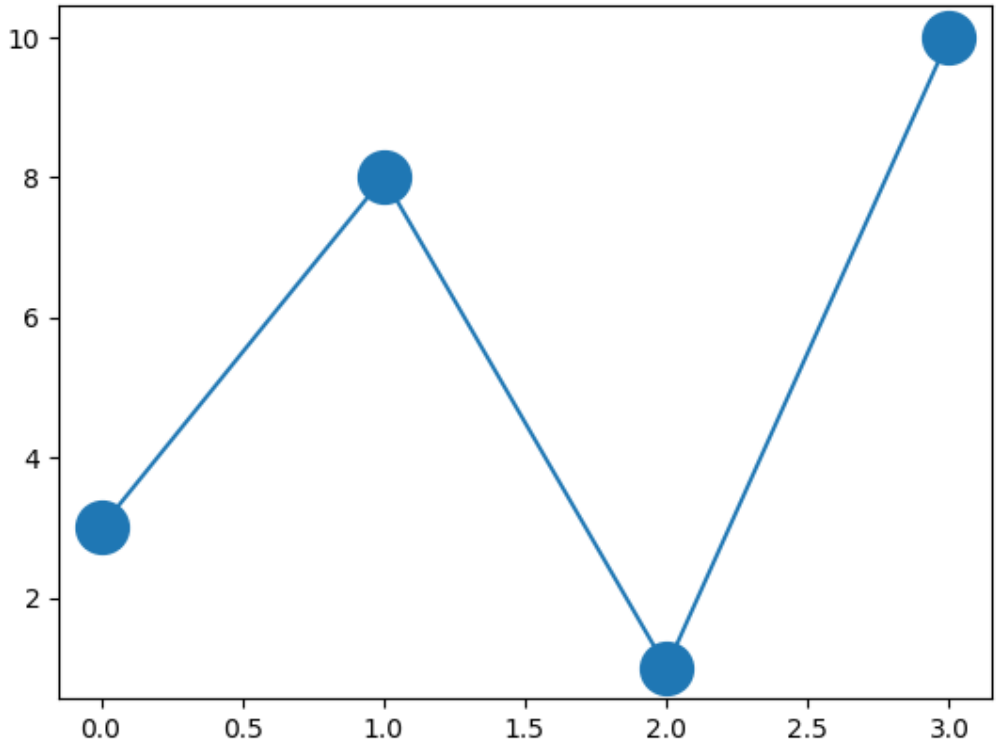

Рисунок 2.9 – Вигляд графіку з розміром маркера 20

Джерело, інше

Побудова кількох графіків у одному вікні matplotlib

Для побудови підграфіку (або відображення кількох графіків у одному вікні) можна використати метод `matplotlib.pyplot.subplot`:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(1, 2, 1)
plt.plot(x,y)
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.show()
```

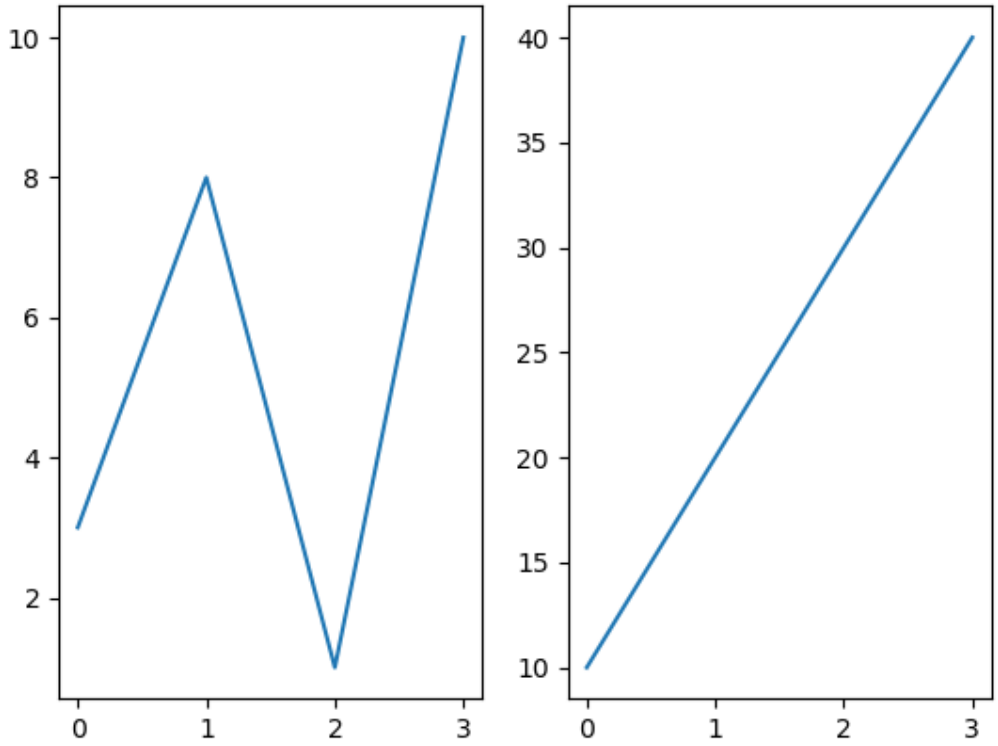

Рисунок 3.1 – Вигляд двох графіків у одному вікні

Метод `subplot` приймає три параметри:

1. Номер рядку на якому розташувати графік
2. Номер стовпця на якму розташувати графік
3. Номер графіку

```
plt.subplot(1, 2, 1) # графікбуде на 1 ряду у 2 стовпці і це 1-ий графік
```

Джерело