Лабораторна робота №1

Виведення вимірювальної інформації за допомогою Matplotlib в Python

Мета: Ознайомитися з методами представлення та візуалізації вимірювальних даних у Python. Навчитися застосовувати різні засоби для аналізу та подання результатів.

1.1 Теоретичні відомості

Matplotlib — це найпопулярніша бібліотека Python для **візуалізації даних**, низькорівнева бібліотека для побудови графів на Python, яка служить утилітою візуалізації. Matplotlib був створений Джоном Д. Хантером. Matplotlib має відкритий вихідний код, і ми можемо вільно його використовувати. Matplotlib здебільшого написаний на Python, кілька сегментів написані на C, Objective-C та Javascript для сумісності з платформою. Вихідний код Matplotlib знаходиться в цьому репозиторії github за адресою https://github.com/matplotlib/matplotlib

Встановлення відбувається за допомогою команди **pip install matplotlib**. Після встановлення Matplotlib імпортуйте його у свої програми, додавши наступний оператор: **import module**. Для перевірки версії використовується:

```
import matplotlib
print(matplotlib.__version__)
```

Основні можливості:

- Побудова **лінійних графіків**, стовпчастих діаграм, гістограм, кругових діаграм.
- Підтримка **2D та 3D графіків**.
- Налаштування стилів: кольори, маркери, шрифти, сітки.
- Збереження графіків у різних форматах: PNG, PDF, SVG.
- Можливість інтерактивної роботи (наприклад, у Jupyter Notebook).

Більшість утиліт Matplotlib знаходяться в pyplot підмодулі та зазвичай імпортуються під plt псевдонімом:

```
import matplotlib.pyplot as plt
```

Функція plot() використовується для малювання точок (маркерів) на діаграмі. За замовчуванням plot() функція малює лінію від точки до точки. Функція приймає параметри для визначення точок на діаграмі.

Параметр 1 – це масив, що містить точки на **осі x** .

Параметр 2 – це масив, що містить точки на **осі y** .

Якщо ми не вкажемо точки на осі x, вони отримають значення за замовчуванням 0, 1, 2, 3 тощо, залежно від довжини точок y. Можна використовувати аргумент ключового слова marker, щоб підкреслити кожен пункт за допомогою вказаного маркера:

Таблиця 1.1 – Види маркерів

Маркер	Опис
'o'	Коло
'*'	Зірка
'.'	Точка

','	Піксель
'x'	Хрестик
'X'	Хрестик (заповнений)
'+'	Плюс
'P'	Плюс (заповнений)
's'	Квадрат
'D'	Ромб
'd'	Ромб (тонкий)
'p'	Пентагон (п’ятикутник)
'H'	Шестикутник
'h'	Шестикутник
'v'	Трикутник вниз
'^'	Трикутник вгору
'<'	Трикутник вліво
'>'	Трикутник вправо
'1'	Малий трикутник вниз
'2'	Малий трикутник вгору
'3'	Малий трикутник вліво
'4'	Малий трикутник вправо
'|'	Вертикальна лінія
'_'	Горизонтальна лінія

Також використовується параметр скороченого рядкового запису , щоб вказати маркер. Цей параметр також називається fmt і записується з таким синтаксисом: *marker|line|color*

```
import matplotlib.pyplot as plt
import numpy as np
import random
arr = [random.randint(1,50) for _ in range(50)]
plt.plot(arr, 'o-.r')
plt.show()
```

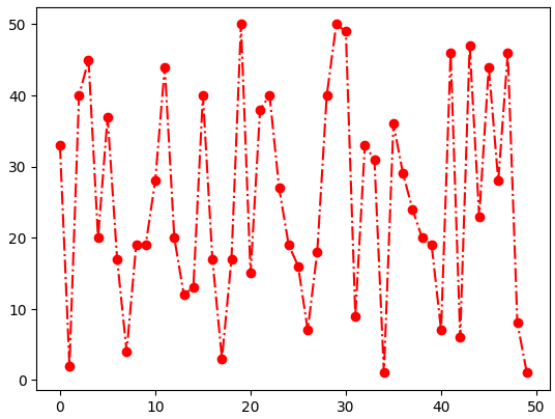


Рис.1.1 – Візуалізація графіка за допомогою fmt формату.

Таблиця 1.2 – Види ліній

Синтаксис лінії	Опис
'-'	Суцільна лінія
':'	Пунктирна лінія (крапки)
'--'	Штрихова лінія (тире)
'-.'	Штрихпунктирна лінія (тире з крапками)

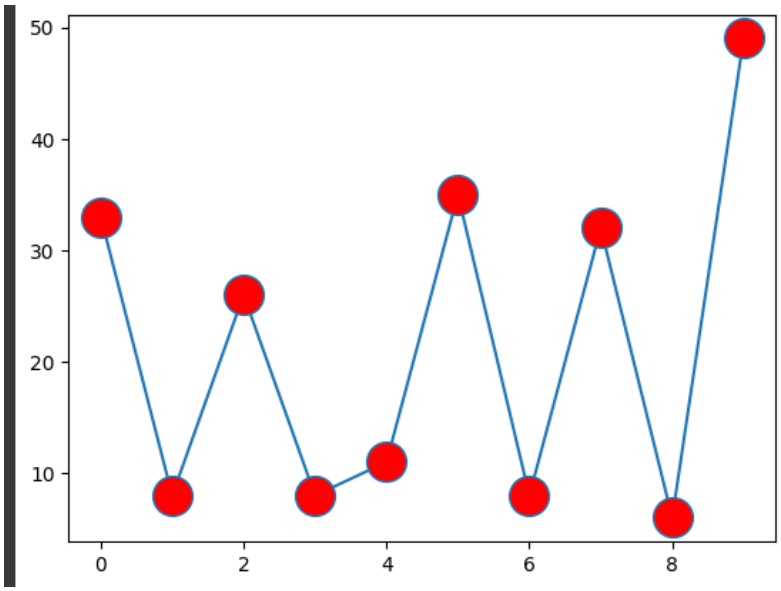
Якщо пропустити значення лінії в параметрі fmt, лінія не буде намальована.

Таблиця 1.3 – Види доступних кольорів.

Синтаксис кольору	Опис
'r'	Червоний
'g'	Зелений
'b'	Синій
'c'	Блакитний (Cyan)
'm'	Пурпурний (Magenta)
'y'	Жовтий
'k'	Чорний
'w'	Білий

Також для контролю розміра маркера використовується ключове слово **marksize,** або сокорочено **ms.** Щоб змінити колір лінії навколо маркера використовується ключове слово **markeredgecolor**, або скорочено **mec**, також для зміни кольору самого маркера використовується **markerfacecolor**, або скорочено **mfc.** Підтримується шістнацядковий формат запису кольору '#4CAF50'

```
import matplotlib.pyplot as plt

import numpy as np

import random

ypoints = np.array([random.randint(1,50) for _ in range(10)])

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

plt.show()
```



Для зміни вигляду ліній використовуються ключові слова:

Linestyle або ls – зміна виду лінії

Color або c – зміна кольору лінії

Linewidth або lw – зміна ширини лінії

Для побудови декількох кривих, що відображають вимірювальні дані використовують plt.plot() для кожного набору даних.

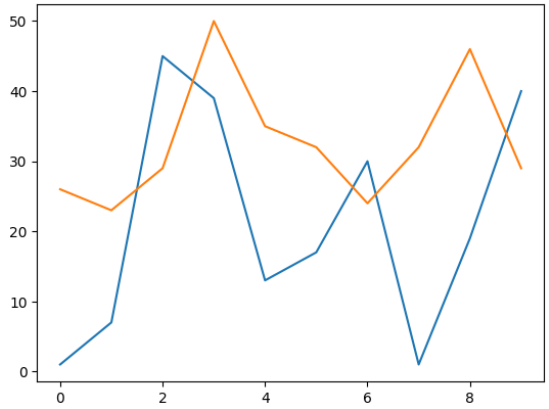


Рис.1.3 – Візуалізація для декількох наборів даних.

У Pyplot використовує функції xlabel()та ylabel(), щоб встановити мітку для осей x та y та для встановлення назви графіка використовується title()

```
import matplotlib.pyplot as plt

import numpy as np

import random

points1 = np.array([random.randint(20,30) for _ in range(10)])

point2 = np.array([random.randint(1,50) for _ in range(10)])

font1 = {'family':'serif','color':'blue','size':20}

font2 = {'family':'serif','color':'darkred','size':15}

plt.plot(points1)

plt.plot(point2)

plt.title("Wather", fontdict=font1)

plt.xlabel("Day", fontdict=font2)

plt.ylabel("Temperature", fontdict=font2)

plt.show()
```

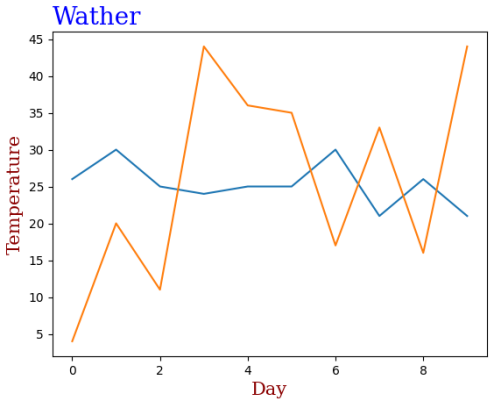


Рис.1.4 – Додавання назви графіка та осей.

Використовується fontdict параметр у xlabel(), ylabel() та title(), щоб встановити властивості шрифту для заголовка та підписів. Також параметр loc використовується для позиціонування заголовка.

Для увімкнення сітки використовується grid()функція. У випадку якщо потрібно відображати лише певні лінії сітки передаються аргументи до axis та допустимі значення «x», «y» та «both». Сама сітка має властивості які можна редагувати color = ' color ', linestyle = ' linestyle ', linewidth = number

```
import numpy as np

import matplotlib.pyplot as plt

x = np.array([random.randint(20,30) for _ in range(10)])

plt.title("Sports Watch Data")

plt.xlabel("Average Pulse")

plt.ylabel("Calorie Burnage")

plt.plot(x)

plt.grid(axis = "x", color = 'green', linestyle = '--', linewidth = 0.5)

plt.show()
```

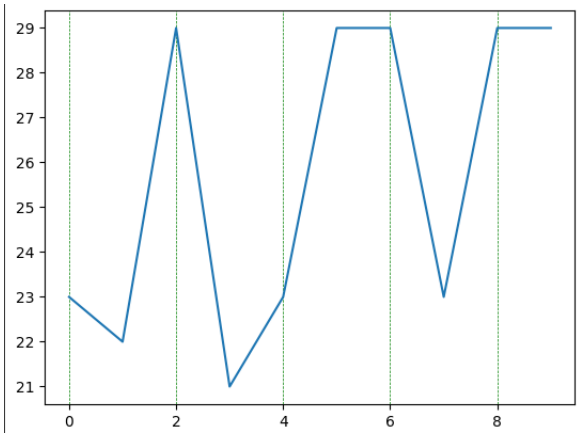


Рис.1.5 – Редагування сітки.

Для відображення одночасно кількох графіків одразу використовується функція subplot(). Функція subplot()приймає три аргументи, які описують розташування фігури. Макет організовано в рядки та стовпці, які представлені першим та другим аргументами. Третій аргумент представляє індекс поточного графіка. Отже, якщо нам потрібен малюнок з 2 рядками та 1 стовпцем (тобто два графіки будуть відображатися один над одним, а не поруч), ми можемо написати синтаксис таким чином: plt.subplot(2, 1, 1). Заголовки додаються що кожного графіка окремо через title(). Також можна додати заголовок до всього малюнка за допомогою suptitle()функції:

```
import matplotlib.pyplot as plt

import numpy as np

#plot 1:

x = np.array([0, 1, 2, 3])

y = np.array([3, 2, 1, 10])


plt.subplot(1, 2, 1)
```

```
plt.plot(x,y)
plt.title("SALES")

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([4, 20, 30, 40])

plt.subplot(1, 2, 2)
plt.plot(x,y)
plt.title("INCOME")

plt.suptitle("MY SHOP")
plt.show()
```

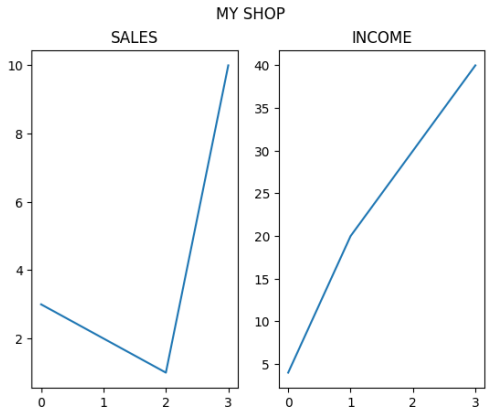


Рис.1.6 – Subplot для графіків.

Функція plt.scatter() використовується для побудови точкових діаграм (scatter plots), де кожна точка визначається координатами (x, y) і може мати власний колір, розмір та форму.

```
plt.scatter(x, y, s=None, c=None, marker=None,
            cmap=None, norm=None, vmin=None, vmax=None,
            alpha=None, linewidths=None, edgecolors=None,
            plotnonfinite=False, **kwargs)
```

Таблиця 1.4 – Параметри scatter.

Параметр	Опис
x, y	Масиви координат точок (обов’язкові).
s	Розмір точок (одне число або масив).
c	Колір точок (рядок, список, масив).
marker	Стиль маркера (наприклад 'o', '^', 's' та інші).
cmap	Карта кольорів (наприклад 'viridis', 'plasma', 'coolwarm').

norm	Нормалізація кольорових значень.
vmin, vmax	Мінімум і максимум для нормалізації кольору.
alpha	Прозорість (0 – прозорий, 1 – непрозорий).
linewidths	Товщина обведення маркерів.
edgecolors	Колір країв маркерів ('none', 'black', інші).
plotnonfinite	Якщо True, то точки з NaN чи inf теж будуть показані.
kwargs	Додаткові параметри (успадковуються від Line2D).

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

# Дані
np.random.seed(0)
x = np.linspace(0, 10, 50)
y = np.sin(x) + np.random.normal(0, 0.1, 50)

sizes = np.linspace(20, 300, 50)          # розміри
colors = np.linspace(0, 1, 50)            # значення для кольору

# Створюємо нормалізацію
norm = mcolors.Normalize(vmin=0, vmax=1)

plt.figure(figsize=(8, 6))

sc = plt.scatter(
    x, y,
    s=sizes,
    c=colors,
    marker='o',
    cmap='plasma',
    norm=norm,
    alpha=0.7,
    linewidths=1.5,
    edgecolors='black',
    plotnonfinite=True,
```

```
    label="Вимірювальні дані"
)

plt.colorbar(sc, label="Значення кольору")
plt.xlabel("Час (с)")
plt.ylabel("Амплітуда")
plt.legend()
plt.grid(True)
plt.show()
```

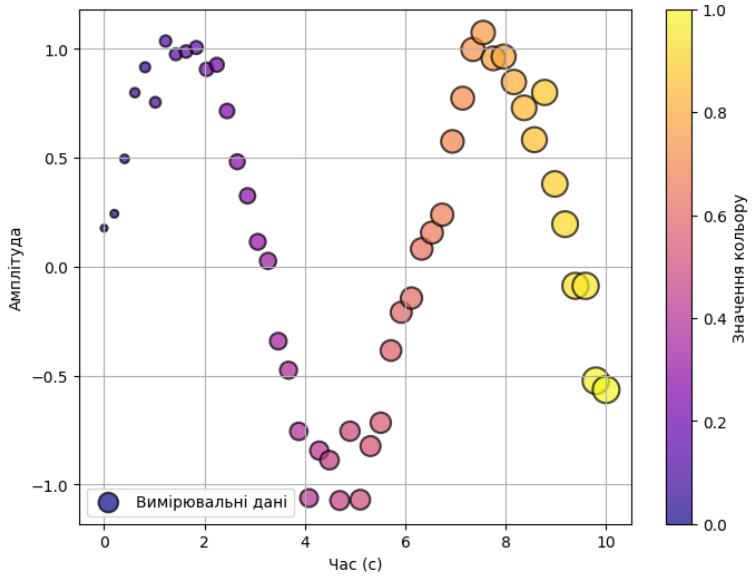


Рис.1.7 – Відображення scatter та взаємодія з основними параметрами.

Для побудови стовпчастих діаграм використовується функція bar(). Основні параметри:

Таблиця 1.5 – Параметри bar.

Параметр	Опис
x	Позиції стовпчиків (список або масив).
height	Висоти стовпчиків (список або масив).
width	Ширина стовпчиків (одне число або масив). За замовчуванням 0.8.
bottom	Зміщення стовпчиків відносно осі X (початок стовпчика).
align	Вирівнювання: 'center' (по центру x) або 'edge' (по краю).
data	Альтернативний спосіб передати дані (словник, DataFrame тощо).
color	Колір стовпчиків (рядок, список).
edgecolor	Колір обведення стовпчиків.
linewidth	Товщина ліній обведення.
tick_label	Підписи категорій на осі X.
label	Підпис для легенди.
alpha	Прозорість (0 – прозорий, 1 – непрозорий).
hatch	Штрихування (наприклад, '/', '\\', `'
kwargs	Інші параметри від Patch (наприклад zorder, capsize).

```
x = [1, 2, 3, 4, 5]
height = [5, 7, 3, 8, 6]
height2 = [2, 3, 1, 4, 2]
labels = ["A", "B", "C", "D", "E"]

plt.figure(figsize=(8,6))

plt.bar(
    x, height,
    width=0.6,                  # ширина
    bottom=0,                   # зміщення
    align="center",             # вирівнювання
    color="skyblue",            # колір
    edgecolor="black",          # колір обведення
    linewidth=1.5,              # товщина обведення
    tick_label=labels,          # підписи
    label="Основні дані",       # легенда
    alpha=0.8,                  # прозорість
    hatch="//",                 # штрихування
    zorder=3                    # порядок малювання
)

plt.bar(
    x, height2,
    width=0.6,
    bottom=height,              # розташування поверх height
    color="orange",
    edgecolor="black",
    linewidth=1.5,
    label="Додаткові дані",
    alpha=0.7
)
```

```
plt.xlabel("Категорії")
plt.ylabel("Значення")
plt.legend()
plt.grid(True, zorder=0)
plt.show()
```

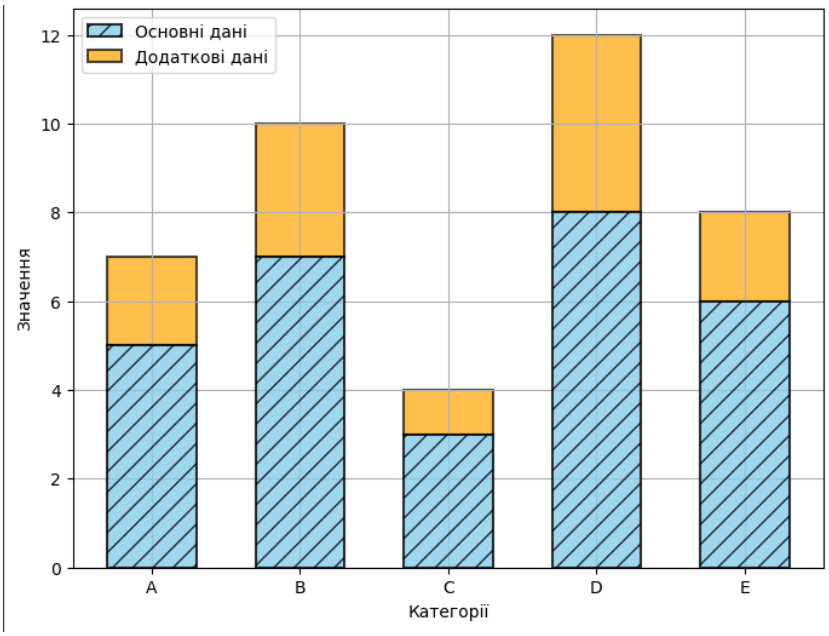


Рис.1.8 – Відображення стовпчастих діаграм та взаємодія з основними параметрами.

Гістограма — це графік, що показує розподіл частот .Це графік, що показує кількість спостережень у кожному заданому інтервалі. У Matplotlib використовується hist() функцію для створення гістограм.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
```

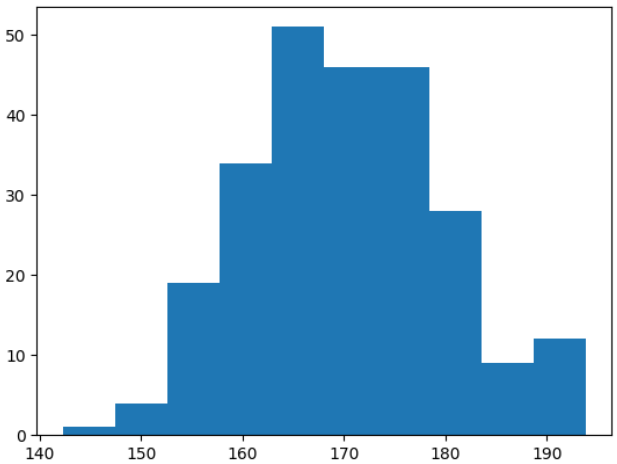


Рис.1.9 – Графік побудований за допомогою hist().

За допомогою pie() функції можна малювати кругові діаграми. За замовчуванням побудова першого клину починається від осі x і рухається проти годинникової стрілки. Параметр startangleвизначається кутом у градусах, кут за замовчуванням дорівнює 0.

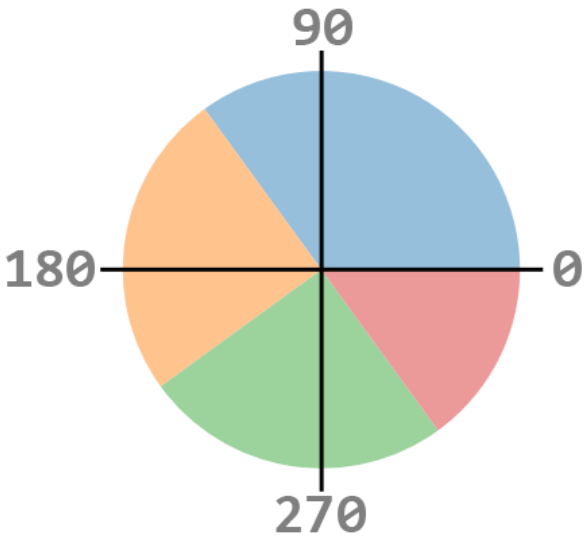


Рис.1.10 – Принцип побудови клинів.

Таблиця 1.6 – Параметри bar.

Параметр	Опис
x	Послідовність чисел (значення секторів).
explode	Зміщення секторів (наприклад, [0, 0.1, 0, 0]).
labels	Підписи секторів.
colors	Кольори секторів.
autopct	Формат відсотків (наприклад, '%1.1f%%').
pctdistance	Відстань тексту від центру (для відсотків).
shadow	Чи малювати тінь (True/False).
labeldistance	Відстань підписів від центру.
startangle	Кут початку (наприклад, 90).
radius	Радіус кола.
counterclock	Напрямок (True – проти годинникової, False – за).
wedgeprops	Словник для стилю секторів (ширина лінії, колір межі тощо).
textprops	Словник для стилю тексту (шрифт, колір).
center	Центр кола (наприклад, (0, 0)).
frame	Чи малювати рамку навколо діаграми.
rotatelabels	Чи повертати підписи секторів.
normalize	Чи нормалізувати дані до суми 1.
data	Словник або DataFrame (для зручного виклику з даними).

```
import matplotlib.pyplot as plt

values = [40, 25, 20, 15]

labels = ["Python", "C++", "Java", "JS"]

colors = ["gold", "skyblue", "lightcoral", "lightgreen"]

explode = [0.1, 0, 0, 0]

plt.pie(
    values,
```

```
    explode=explode,
    labels=labels,
    colors=colors,
    autopct='%1.1f%%',
    pctdistance=0.7,
    shadow=True,
    labeldistance=1.2,
    startangle=90,
    radius=1.1,
    counterclock=True,
    wedgeprops={'edgecolor': 'black', 'linewidth': 1, 'linestyle': '--'},
    textprops={'fontsize': 12, 'color': 'darkblue'},
    center=(0, 0),
    frame=True,
    rotatelabels=True,
    normalize=True
)

plt.title("Популярність мов програмування")
plt.show()
```

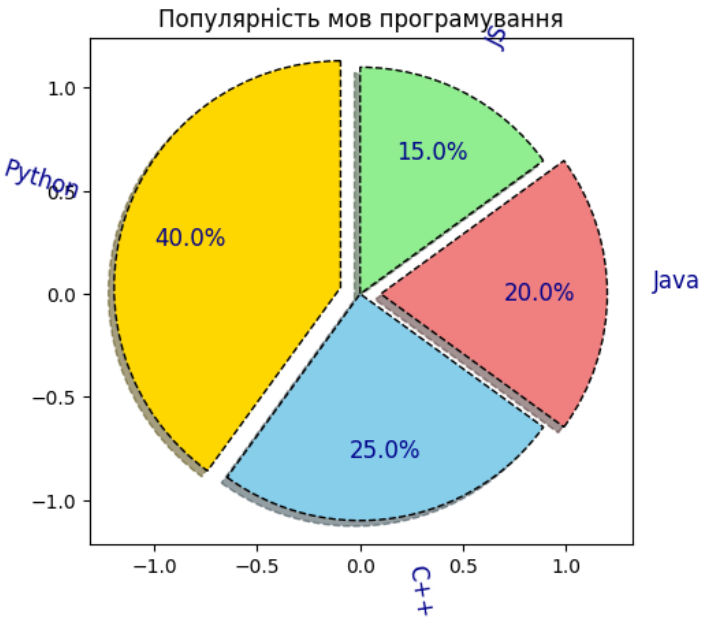


Рис.1.11 – Зображення кругової діаграми.

Візуалізація даних, що включають три змінні, часто вимагає тривимірного графічного оформлення для кращого розуміння складних взаємозв'язків та закономірностей, які двовимірні графіки не можуть виявити. Бібліотека Matplotlib у Python, завдяки своєму інструментарію mpl_toolkits.mplot3d , забезпечує потужну підтримку для 3D-візуалізації. Щоб розпочати створення 3D-графіків , першим важливим кроком є налаштування середовища 3D-графічного оформлення, увімкнувши 3D-проекцію на осях графіка. Наприклад:

```
import matplotlib.pyplot as plt

fig = plt.figure()

ax = plt.axes(projection='3d')

plt.show()
```

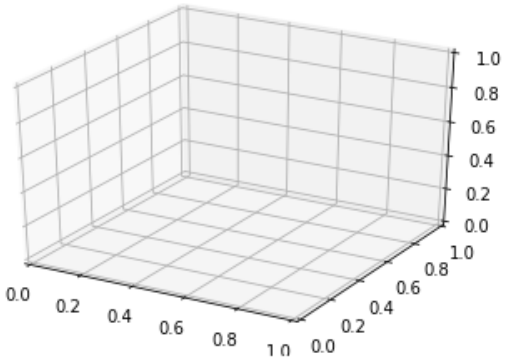


Рис.1.12 – Відображення 3д графіку.

Лінійна тривимірна діаграма з'єднує точки у тривимірному просторі для візуалізації безперервного шляху. Вона корисна для демонстрації того, як змінна змінюється з часом або простором у 3D. У цьому прикладі використовуються функції синуса та косинуса для малювання спірального шляху.

```
from mpl_toolkits import mplot3d

import numpy as np

import matplotlib.pyplot as plt


fig = plt.figure()

ax = plt.axes(projection='3d')

z = np.linspace(0, 1, 100)

x = z * np.sin(25 * z)

y = z * np.cos(25 * z)

ax.plot3D(x, y, z, 'green')

plt.show()
```

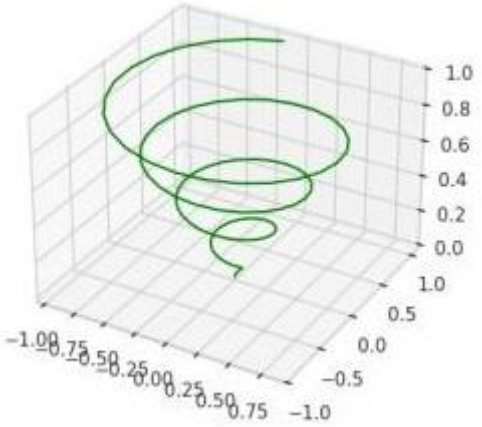


Рис.1.11 – Лінійна тривимірна діаграма.

3D- діаграма розсіювання відображає окремі точки даних у трьох вимірах, що корисно для виявлення тенденцій або кластерів. Кожна точка представляє точку зі значеннями (x, y, z), а колір можна використовувати для додавання четвертого виміру.

```
fig = plt.figure()

ax = plt.axes(projection='3d')


z = np.linspace(0, 1, 100)

x = z * np.sin(25 * z)

y = z * np.cos(25 * z)

c = x + y  # Color array based on x and y


ax.scatter(x, y, z, c=c)

ax.set_title('3D Scatter Plot')

plt.show()
```

Поверхневі діаграми показують гладку поверхню, що охоплює сітку значень (x, y) та формується значеннями z. Вони чудово підходять для візуалізації функцій з двома змінними, забезпечуючи чітку топографію даних.

```
x = np.outer(np.linspace(-2, 2, 10), np.ones(10))

y = x.copy().T

z = np.cos(x**2 + y**3)


fig = plt.figure()

ax = plt.axes(projection='3d')

ax.plot_surface(x, y, z, cmap='viridis', edgecolor='green')

plt.show()
```

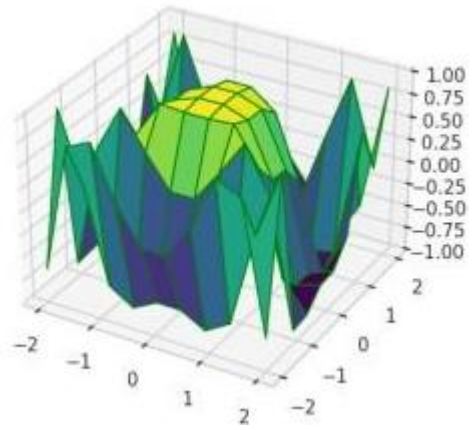


Рис.1.14 – Відображення поверхневої діаграми.

Каркасний графік подібний до графіка поверхні, але показує лише краї або «скелет» поверхні. Він корисний для розуміння структури 3D-поверхні без відволікання на колірну заливку.

```
def f(x, y):
    return np.sin(np.sqrt(x**2 + y**2))


x = np.linspace(-1, 5, 10)
y = np.linspace(-1, 5, 10)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)
fig = plt.figure()
ax = plt.axes(projection='3d')
ax.plot_wireframe(X, Y, Z, color='green')
plt.show()
```

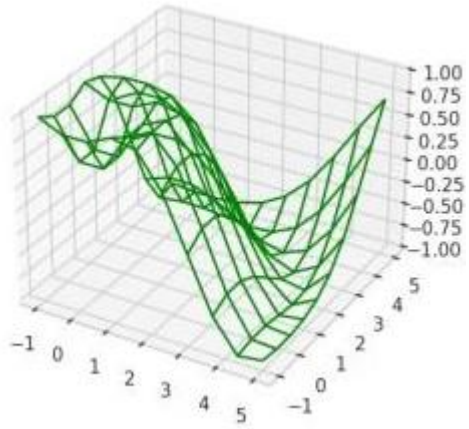


Рис.1.15 – Відображення каркасного графіку.

Цей графік поєднує 3D-поверхню з контурними лініями для виділення висоти або глибини. Це допомагає чіткіше візуалізувати форму функції та зміни градієнта у 3D-просторі.

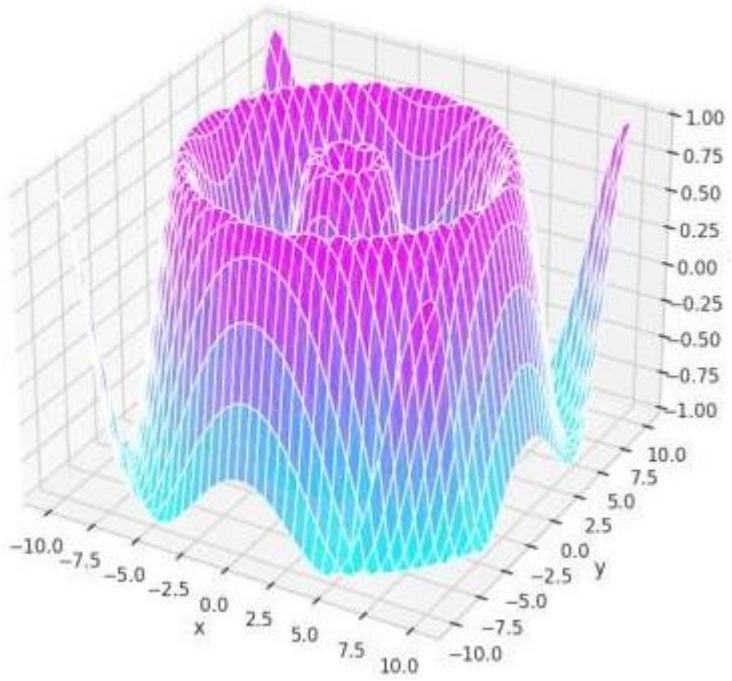
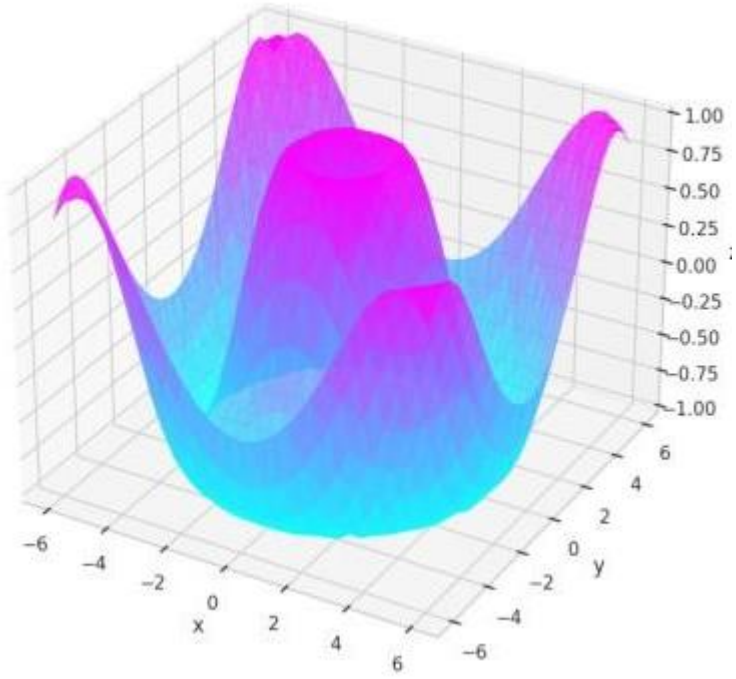


Рис.1.16 – Відображення 3D-поверхні з контурними лініями для виділення висоти або глибини

Цей графік використовує трикутні сітки для побудови 3D-поверхні з розсіяних або сітчастих даних. Він ідеально підходить, коли поверхня нерівна або коли використовуються непрямокутні сітки.

```
from matplotlib.tri import Triangulation


def f(x, y):
    return np.sin(np.sqrt(x**2 + y**2))


x = np.linspace(-6, 6, 30)
y = np.linspace(-6, 6, 30)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)


tri = Triangulation(X.ravel(), Y.ravel())


fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
ax.plot_trisurf(tri, Z.ravel(), cmap='cool', edgecolor='none', alpha=0.8)


ax.set_title('Surface Triangulation Plot')
```

```
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
plt.show()
```



1.2 Порядок виконання лабораторної роботи

1. Ознайомитись з теоретичними відомостями.
2. Побудувати графіки температури та вологості з дотриманням вимог:

Дані:

- Температура (°C): [18, 17, 16, 15, 15, 16, 18, 20, 22, 23, 24, 25, 26, 26, 25, 24, 23, 21, 20, 19, 18, 17, 17, 16]
- Вологість (%): [85, 88, 90, 92, 91, 89, 85, 80, 75, 70, 68, 65, 62, 63, 67, 72, 78, 82, 86, 89, 91, 93, 94, 95]

Вимоги до графіка:

- Окремі лінії для температури та вологості.
- Підписи для кожної лінії (label).
- Різні маркери (marker) та стилі ліній (linestyle) для кожного набору даних.
- Задайте колір ліній (color).
- Встановіть товщину ліній (linewidth).
- Підписи осей (часу та значень) та заголовок графіка.
- Відобразіть легенду (legend).
- Додайте сітку (grid).

3. Створіть два графіки для візуалізації даних про успішність та відвідуваність групи студентів.(scatter та bar):
   Дані:
   - Студенти: [Бен, 'Богдан', 'Вікторія', 'Оксана', 'Максим']
   - Кількість відвідувань (з 15): [14, 12, 15, 11, 13]
   - Середній бал (з 100): [88, 75, 95, 60, 82]

Вимоги до графіків:

- scatter(): Побудуйте точковий графік, де по осі X буде кількість відвідувань, а по осі Y — середній бал. Це допоможе визначити кореляцію між цими показниками. Додайте підписи до точок.
- bar(): Побудуйте стовпчикову діаграму, що відображає середні бали кожного студента. Підпишіть стовпці і додайте заголовок графіка.

4. Візуалізуйте розподіл віку населення та співвідношення між різними віковими групами.

   Дані:

   - Вік 20 осіб: [22, 25, 25, 28, 30, 31, 31, 35, 38, 40, 42, 45, 45, 48, 50, 55, 60, 62, 65, 70]
   - Вікові групи: ['18-30', '31-45', '46-60', '61+']
   - Кількість людей у групах: [7, 8, 3, 2]

   Вимоги до графіків:

   - Гістограма (hist()):
     - Побудуйте гістограму для даних про вік 20 осіб.
     - Використайте 5-6 рівномірних інтервалів (bins).
     - Встановіть підписи для осі X (Вік) та осі Y (Кількість людей).
     - Додайте заголовок до гістограми, наприклад, "Розподіл віку населення".
     - Змініть колір стовпців та прозорість за допомогою параметрів color та alpha.
     - Додайте позначки (tick marks) на осі X, що відповідають межам інтервалів.

   - Кругова діаграма (pie()):
     - Побудуйте кругову діаграму, що відображає співвідношення кількості людей у вікових групах.
     - Використайте дані про вікові групи та кількість людей.
     - Додайте підписи (labels) для кожної частини діаграми.
     - Відобразіть відсотки (autopct) для кожної частини з одним знаком після коми.
     - Зробіть "вибух" (explode) для однієї з частин діаграми (наприклад, для найбільшої або найменшої групи), щоб виділити її.
     - Додайте тінь (shadow=True) до діаграми.
     - Встановіть заголовок для діаграми, наприклад, "Співвідношення вікових груп".

5. Створіть спільну фігуру з трьома підграфіками, що відображають динаміку продажів у трьох різних регіонах за 12 місяців.

   Дані:

   - Регіон A: [150, 160, 155, 170, 185, 190, 200, 195, 210, 220, 215, 230]
   - Регіон B: [120, 125, 130, 140, 135, 145, 150, 160, 165, 170, 175, 180]
   - Регіон C: [100, 110, 105, 115, 120, 125, 130, 140, 150, 145, 155, 160]

Вимоги до графіка:

- Використайте plt.subplots() для створення сітки 3x1.
- Кожен підграфік має відображати динаміку продажів одного регіону.
- Обов'язково додайте підписи осей та заголовок для кожного підграфіка.
- Створіть спільний заголовок для всієї фігури.

6. Побудуйте 3D-графік поверхні, що демонструє розподіл тиску в залежності від координат (X, Y).

   Дані: Використайте функцію, яка імітує вимірювання, наприклад, Z = np.sin(np.sqrt(X**2 + Y**2)), де X та Y — це координати.

Вимоги до графіка:

- Створіть сітку координат **X** та **Y** за допомогою np.meshgrid().
- Обчисліть значення **Z** (тиск) для кожної точки сітки.
- Використайте mplot3d для побудови 3D-поверхні.
- Додайте підписи для осей X, Y, Z та заголовок графіка.
- Налаштуйте кольорову палітру для кращої візуалізації.
-

1.3 Зміст звіту

1. Найменування і мета роботи;
2. Код по кожному пункту порядку виконання роботи;
3. Результати роботи по кожному пункту виконання роботи;
4. Висновки.

1.4 Контрольні запитання

1. Яке основне призначення бібліотеки Matplotlib у Python? Для чого вона використовується у наукових та інженерних розрахунках?
2. Назвіть та опишіть три основні типи графіків, які можна побудувати за допомогою Matplotlib, і поясніть, в яких випадках кожен із них є найбільш ефективним.
3. Чим відрізняються графіки, побудовані за допомогою plt.plot(), від графіків plt.scatter()? Наведіть приклад даних, для яких краще використовувати scatter().
4. Яка функція Matplotlib використовується для створення стовпчикової діаграми? Які дані (одновимірні чи двовимірні) потрібні для її побудови?
5. Поясніть, що таке гістограма (hist()). Чим вона відрізняється від стовпчикової діаграми? Для візуалізації якого типу даних вона підходить найкраще?
6. Що таке subplot у Matplotlib? Навіщо його використовують і яка функція потрібна для його створення?