**Модуль 2.2: Дослідницький аналіз даних**

**Візуалізація даних**

**Matplotlib**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#matplotlib)Пакет Matplotlib широко використовується у мові Python для графічного відображення даних. В основному він використовується через інтерфейс pyplot для створення простих візуалізацій даних. Але можна також отримати зображення високої якості для журнальних статей, книг та інших публікацій.

Для встановлення Matplotlib за допомогою менеджера пакетів pip введіть у командному рядку вашої операційної системи наступну команду:

pip install matplotlib

**Лінійні графіки**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BB%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D1%96-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B8)На найвищому рівні Matplotlib знаходиться модуль matplotlib.pyplot, який виконує побудову графіків. Основним елементом зображення є фігура (Figure), на неї накладається одне або більше полів з графіками, осі координат, текстові написи тощо. Для побудови графіка використовується функція plot.

Якщо в якості параметра функції plot передати список, то значення з цього списку будуть відкладені по осі ординат (вісь y), а по осі абсцис (вісь x) будуть відкладені індекси елементів масиву. Якщо ми хочемо задати значення відразу по двох осях (x, y), необхідно в plot передати два списки.

Наприклад, побудуємо графік денної погоди в Полтаві за перші 8 днів вересня 2021 року.

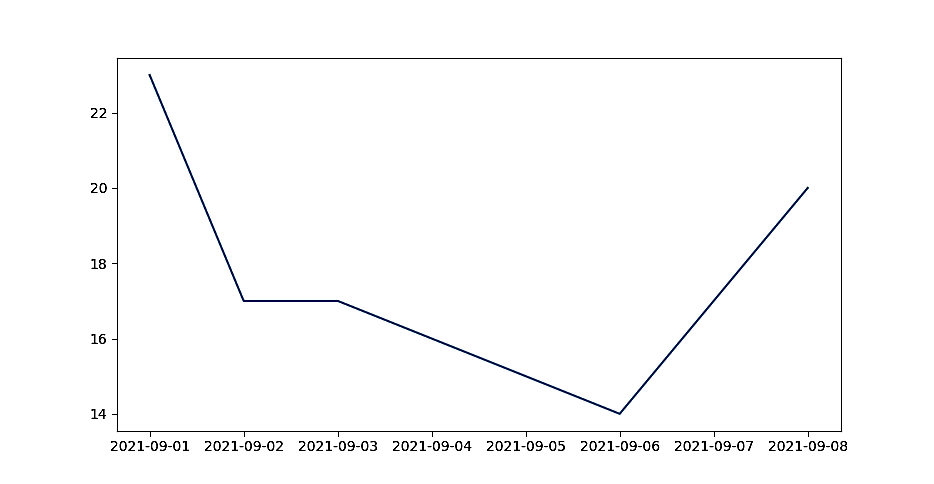
import matplotlib.pyplot as plt

import pandas as pd

date = pd.date\_range(start='2021-09-01', freq='D', periods=8)

plt.plot(date, [23, 17, 17, 16, 15, 14, 17, 20])

plt.show()



**Текст на графіках**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%82%D0%B5%D0%BA%D1%81%D1%82-%D0%BD%D0%B0-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B0%D1%85)Найчастіше на графіку нам потрібно додати назви осей та графіка, можливо текстові примітки і, якщо графіків кілька, обов'язково легенду.

Для цього використовується функція xlabel для осі x та ylabel для y осі. Розберемося з аргументами цих функцій.

Функції приймають наступні параметри:

* xlabel (або ylabel) — рядок тексту підпису.

А також параметри конструктора класу matplotlib.text.Text:

* fontsize або size — розмір шрифту, число або значення зі списку xx-small, x-small, small, medium, large, x-large, xx-large
* fontstyle — стиль шрифту значення зі списку normal, italic, oblique
* fontweight — товщина шрифту може визначатися або числом в діапазоні від 0 до 1000, або значенням зі списку ultralight, light, normal, regular, book, medium, roman, semibold, demibold, demi, bold, heavy, extra bold, black
* color — значення кольору для тексту підпису. Можливості вибору кольору занадто великі і подивитися варіанти можна за цим посиланням (<https://matplotlib.org/stable/gallery/color/named_colors.html>), де вони теж наведені не повністю.

plt.xlabel('Дата', fontsize='small', color='midnightblue')

plt.ylabel('Температура', fontsize='small', color='midnightblue')

Назву графіка можна задати методом title

plt.title('Денна погода у м. Полтава', fontsize=15)

За допомогою параметра loc можна задати вирівнювання заголовка center, left, right

Також, як і для методів xlabel та ylabel, доступні параметри конструктора класу Text.

Щоб помістити текст на полі графіка, необхідно викликати метод text. Перший та другий її аргументи — це координати позиції, третій — текст напису, приклад використання:

plt.text(date[0], 15, 'Осінь досить тепла', color="blue")

І додати легенду можна, викликавши метод legend, не забувши додати параметр label у метод plot. І остаточний результат буде наступним.

import matplotlib.pyplot as plt

import pandas as pd

date = pd.date\_range(start='2021-09-01', freq='D', periods=8)

plt.plot(date, [23, 17, 17, 16, 15, 14, 17, 20], label='day temperature')

plt.xlabel('Дата', fontsize='small', color='midnightblue')

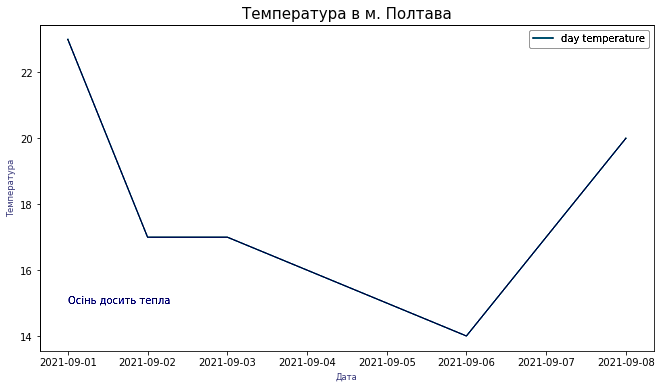
plt.ylabel('Температура', fontsize='small', color='midnightblue')

plt.title('Температура в м. Полтава', fontsize=15)

plt.text(date[0], 15, 'Осінь досить тепла', color="blue")

plt.legend()

plt.show()



**Компонування графіків**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D1%96%D0%B2)Якщо ми хочемо розмістити два графіки, хорошим способом буде використовувати метод subplots. Функція subplots повертає два об'єкти, перший - це Figure, підкладка, на якій будуть розміщені поля з графіками, другий - об'єкт (або масив об'єктів) Axes, через який можна отримати повний доступ до налаштування зовнішнього вигляду елементів, що відображаються.

import matplotlib.pyplot as plt

import pandas as pd

date = pd.date\_range(start='2021-09-01', freq='D', periods=8)

fig, axs = plt.subplots()

axs.plot(date, [23, 17, 17, 16, 15, 14, 17, 20], label='day temperature')

axs.plot(date, [19, 11, 16, 11, 10, 10, 11, 16], label='night temperature')

plt.xlabel('Дата', fontsize='small', color='midnightblue')

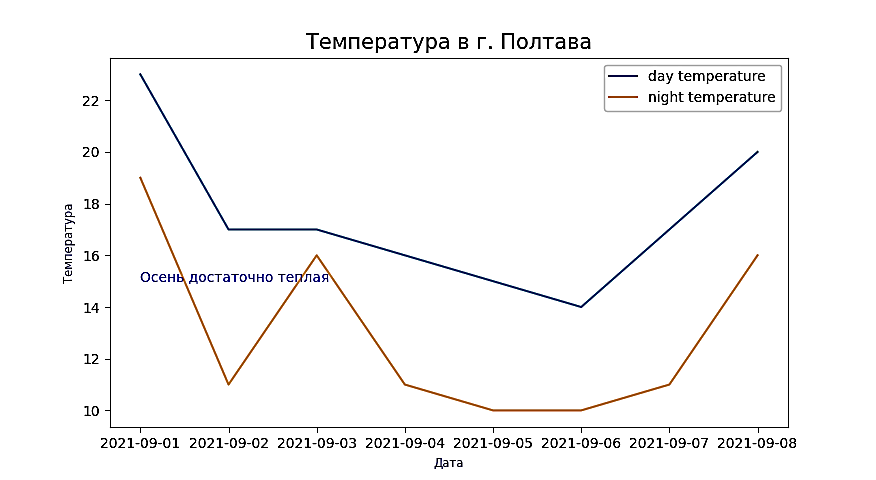
plt.ylabel('Температура', fontsize='small', color='midnightblue')

plt.title('Температура в м. Полтава', fontsize=15)

plt.text(date[0], 15, 'Осінь досить тепла', color="blue")

plt.legend()

plt.show()



Якщо ми хочемо розділити графіки на частини, то у функцію subplots ми повинні передати кількість рядків та стовпців. Більше про метод можна прочитати [тут](https://matplotlib.org/stable/gallery/subplots_axes_and_figures/subplots_demo.html) (<https://matplotlib.org/stable/gallery/subplots_axes_and_figures/> subplots\_demo .html).

import matplotlib.pyplot as plt

import pandas as pd

date = pd.date\_range(start='2021-09-01', freq='D', periods=8)

fig, axs = plt.subplots(2, 1)

axs[0].plot(date, [23, 17, 17, 16, 15, 14, 17, 20], label='day temperature')

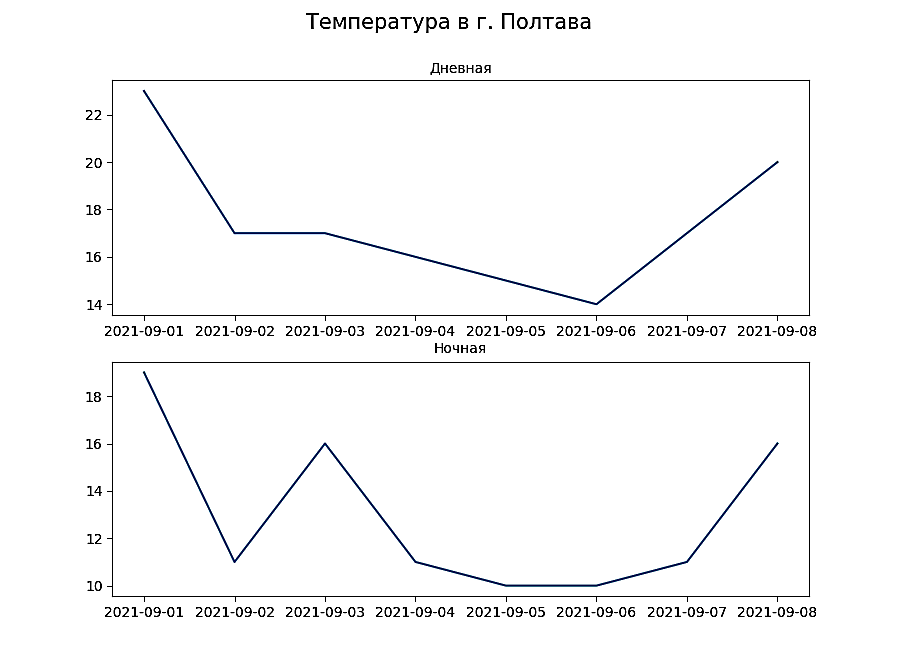
axs[1].plot(date, [19, 11, 16, 11, 10, 10, 11, 16], label='night temperature')

axs[0].set\_title('Денна', fontsize=10)

axs[1].set\_title('Нічна', fontsize=10)

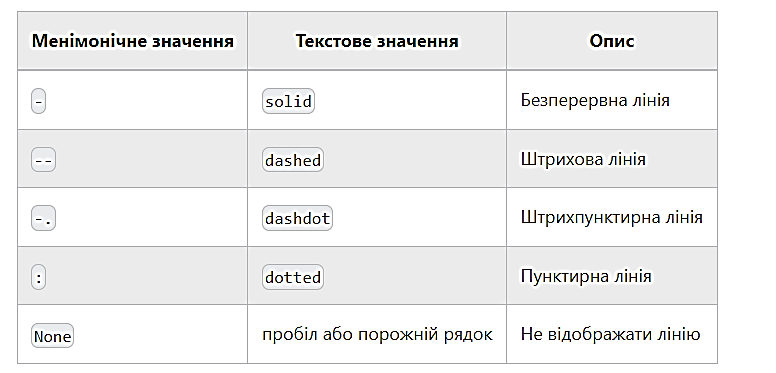
fig.suptitle('Температура в м. Полтава', fontsize=15)

plt.show()



**Налаштування графіків**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BD%D0%B0%D0%BB%D0%B0%D1%88%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D1%96%D0%B2)

Стиль лінії графіка задається через параметр linestyle, який може приймати значення з таблиці



Колір лінії графіка задається через параметр color (c — скорочений варіант). Варіантів встановлення формату, як уже говорилося, - багато. Так можна використовувати формат RGB або RGBA — кортеж значень з плаваючою крапкою в діапазоні [0, 1] (0.5, 0.6, 0.1) або значення в hex форматі #0a0a0a. Також колір можна задати за допомогою набору символів b, g, r, c, m, y, k, w

import matplotlib.pyplot as plt

import pandas as pd

date = pd.date\_range(start="2021-09-01", freq="D", periods=8)

plt.plot(

date,

[23, 17, 17, 16, 15, 14, 17, 20],

label="day temperature",

linestyle="--",

color="#FF5733",

)

plt.plot(

date,

[19, 11, 16, 11, 10, 10, 11, 16],

label="night temperature",

linestyle=":",

color="#061358",

)

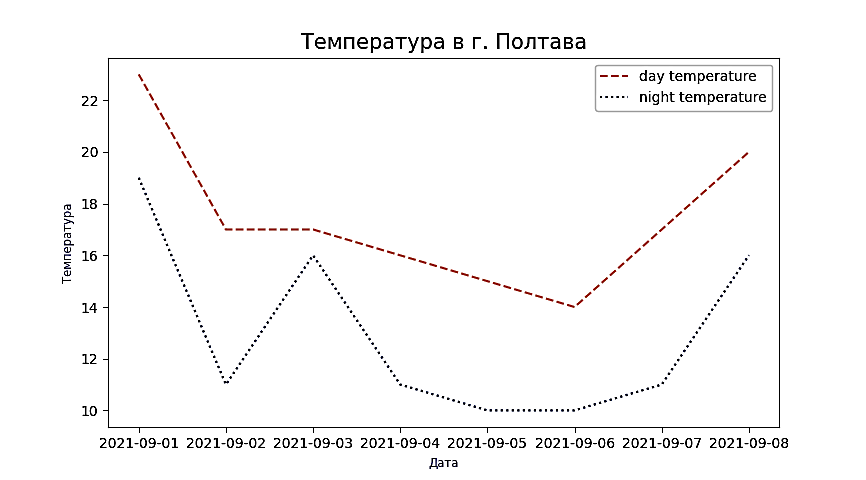
plt.xlabel("Дата", fontsize="small", color="midnightblue")

plt.ylabel("Температура", fontsize="small", color="midnightblue")

plt.title("Температура в м. Полтава", fontsize=15)

plt.legend()

plt.show()



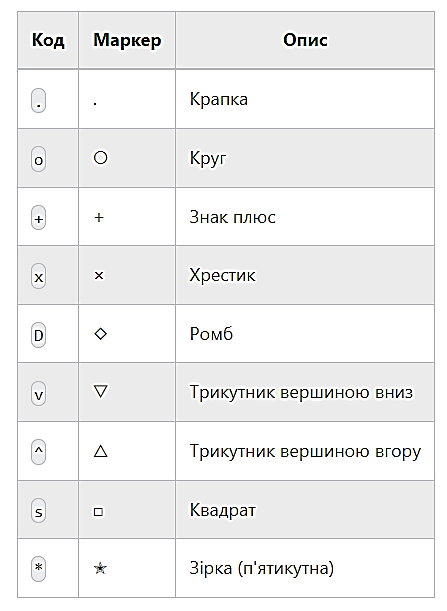
Щоб встановити для межі осі інше значення, використовуються методи xlim та ylim.

Також можна відобразити сітку за допомогою методу grid

Товщина лінії визначається значенням аргументу lineweight (або просто lw) в пунктах (pt).

Щоб під час виклику методу plot включити маркери — це символи, що виводяться в кожній точці даних графіка, потрібно визначити аргумент marker.

Популярні стилі маркерів



І остаточно гарно оформлений графік буде виглядати так:

import matplotlib.pyplot as plt

import pandas as pd

date = pd.date\_range(start="2021-09-01", freq="D", periods=8)

plt.plot(

date,

[23, 17, 17, 16, 15, 14, 17, 20],

label="day temperature",

linestyle="--",

color="#FF5733",

linewidth=2,

marker="D",

)

plt.plot(

date,

[19, 11, 16, 11, 10, 10, 11, 16],

label="night temperature",

linestyle=":",

color="#061358",

linewidth=2,

marker="^",

)

plt.ylim(0, 25)

plt.xlabel("Дата", fontsize="small", color="midnightblue")

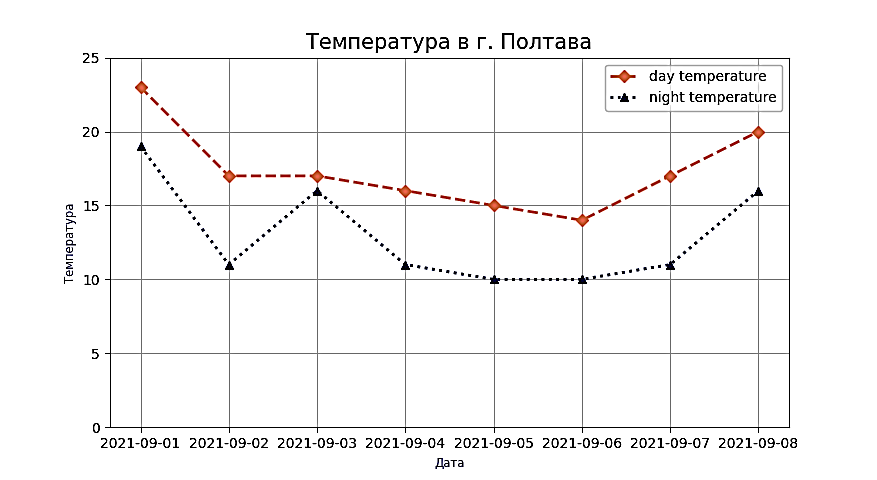
plt.ylabel("Температура", fontsize="small", color="midnightblue")

plt.title("Температура в м. Полтава", fontsize=15)

plt.legend()

plt.grid()

plt.show()



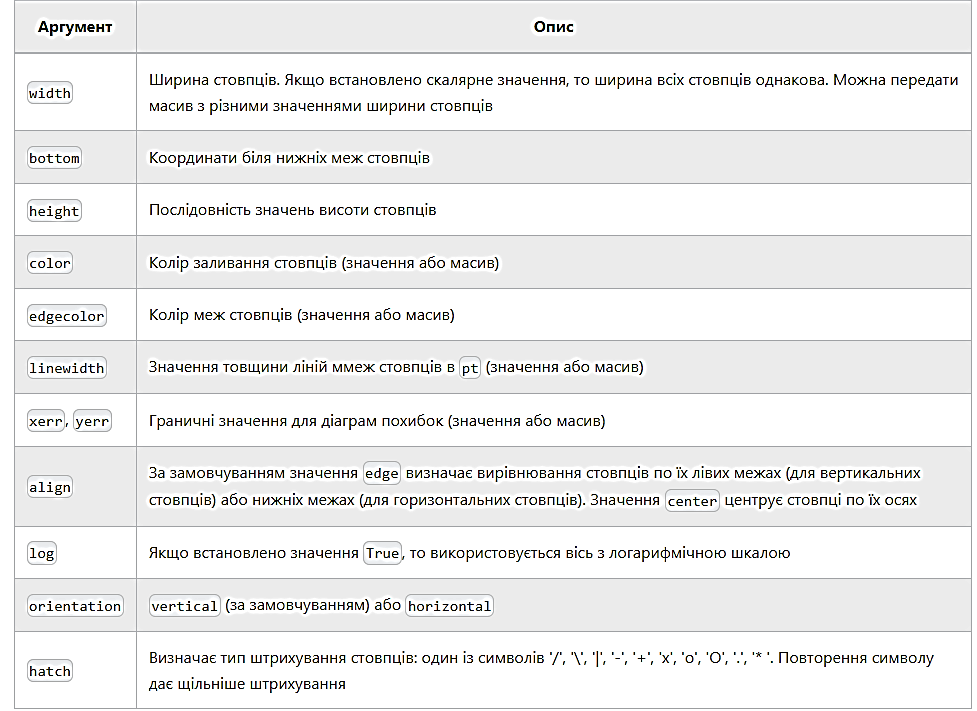
**Стовпчикові діаграми**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%81%D1%82%D0%BE%D0%B2%D0%BF%D1%87%D0%B8%D0%BA%D0%BE%D0%B2%D1%96-%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B8)Для створення стовпчикової діаграми використовується метод bar (barh) модуля pyplot, який формує графік із прямокутних смуг, що визначаються їх лівою межею та висотою.

* bar — вертикальна стовпчаста діаграма;
* barh — горизонтальна стовпчаста діаграма.

За замовчуванням значення ширини прямокутників дорівнює 0.8, але воно змінюється за допомогою аргументу width.

Параметри, що визначають стовпчикову діаграму.



Приклад:

import matplotlib.pyplot as plt

plt.bar(

["США", "Китай", "Японія", "Велика Британія"],

[39, 38, 27, 22],

color=["b", "r", "y", "g"],

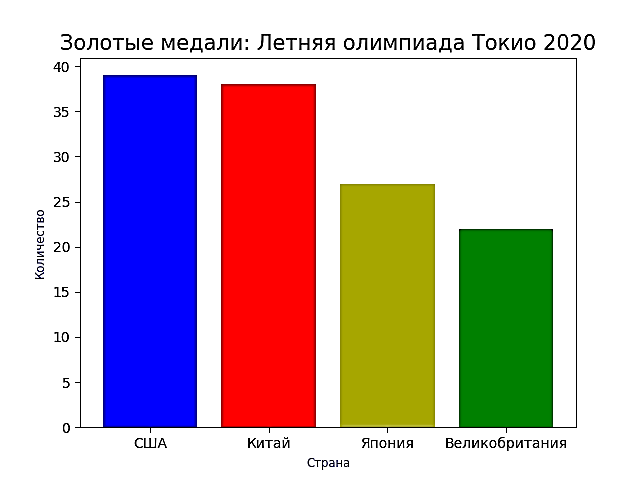
)

plt.xlabel("Країна", fontsize="small", color="midnightblue")

plt.ylabel("Кількість", fontsize="small", color="midnightblue")

plt.title("Золоті медалі: Літня олімпіада Токіо 2020", fontsize=15)

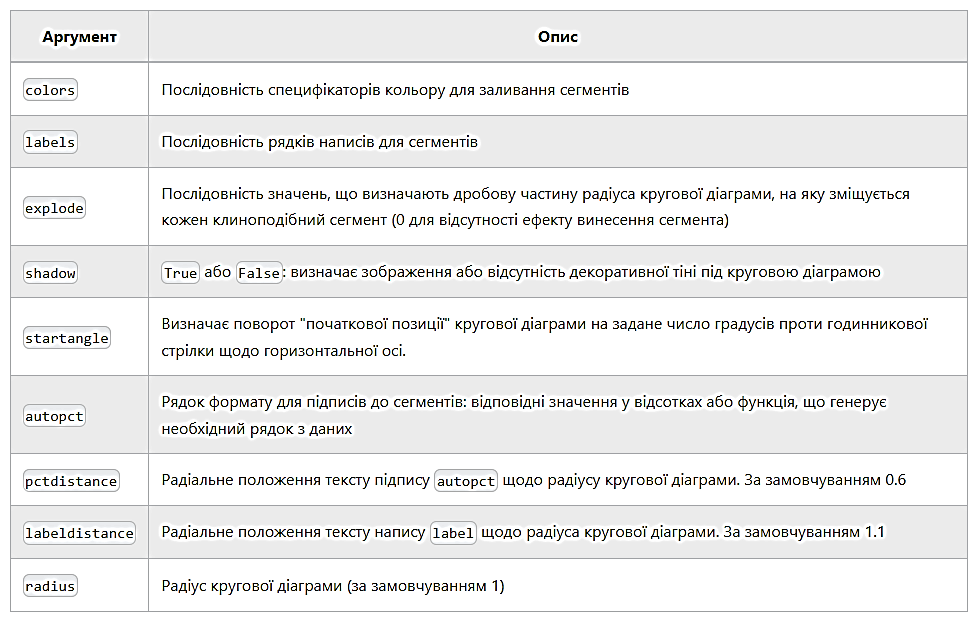
plt.show()



**Кругові діаграми**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BA%D1%80%D1%83%D0%B3%D0%BE%D0%B2%D1%96-%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B8)Кругову або секторну діаграму можна побудувати за допомогою методу pie. Значення нормалізуються за їх сумою, якщо ця сума більша за 1, інакше значення безпосередньо інтерпретуються як частки.

Аргументи для методу pie



Як приклад побудуємо кругову діаграму розподілу програмістів Python за посадами з другого домашнього завдання pandas

import matplotlib.pyplot as plt

labels = [

"Junior Software Engineer",

"Senior Software Engineer",

"Software Engineer",

"System Architect",

"Technical Lead",

]

data = [63, 31, 100, 2, 11]

explode = [0.15, 0, 0, 0, 0]

plt.pie(

data,

labels=labels,

shadow=True,

explode=explode,

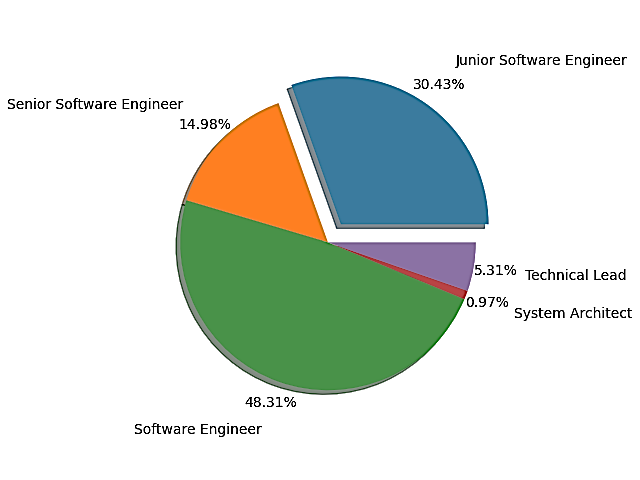
autopct="%.2f%%",

pctdistance=1.15,

labeldistance=1.35,

)

plt.show()



**Діаграми в полярних координатах**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B8-%D0%B2-%D0%BF%D0%BE%D0%BB%D1%8F%D1%80%D0%BD%D0%B8%D1%85-%D0%BA%D0%BE%D0%BE%D1%80%D0%B4%D0%B8%D0%BD%D0%B0%D1%82%D0%B0%D1%85)

Для отримання графіка в полярних координатах (r,θ) використовується метод polar, в який передаються аргументи theta (незалежна змінна) і r.

Побудуємо, наприклад, графік полярної троянди: 𝑟(𝜃)=sin⁡(6𝜃)*r*(*θ*)=sin(6*θ*)

import matplotlib.pyplot as plt

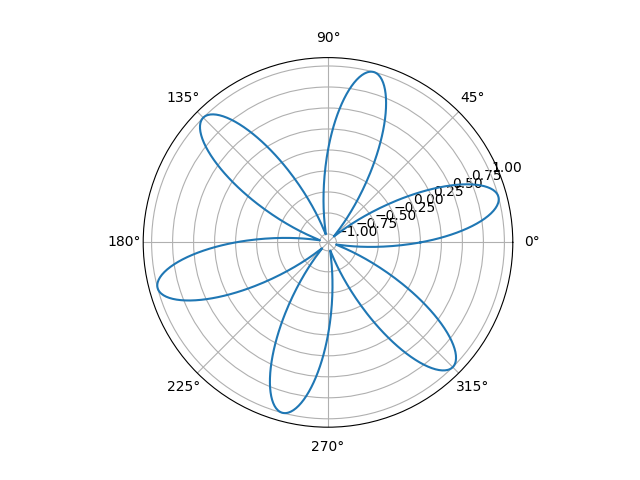
import numpy as np

theta = np.linspace(0, 2.0 \* np.pi, 1000)

r = np.sin(6 \* theta)

plt.polar(theta, r)

plt.show()



**Тривимірні графіки**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%82%D1%80%D0%B8%D0%B2%D0%B8%D0%BC%D1%96%D1%80%D0%BD%D1%96-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D0%B8)Бібліотека Matplotlib підтримує побудову тривимірних графіків. Для створення тривимірного графіка необхідно імпортувати об'єкт Axes3D з модуля mpl\_toolkits.mplot3d і визначити аргумент projection внутрішнього графіка значенням 3d:

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = fig.add\_subplot(projection='3d')

**Лінійний графік**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BB%D1%96%D0%BD%D1%96%D0%B9%D0%BD%D0%B8%D0%B9-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA)Приклад рівняння спіралі у тривимірному просторі

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = fig.add\_subplot(projection="3d")

theta\_max = 8 \* np.pi

n = 1000

theta = np.linspace(0, theta\_max, n)

x = theta

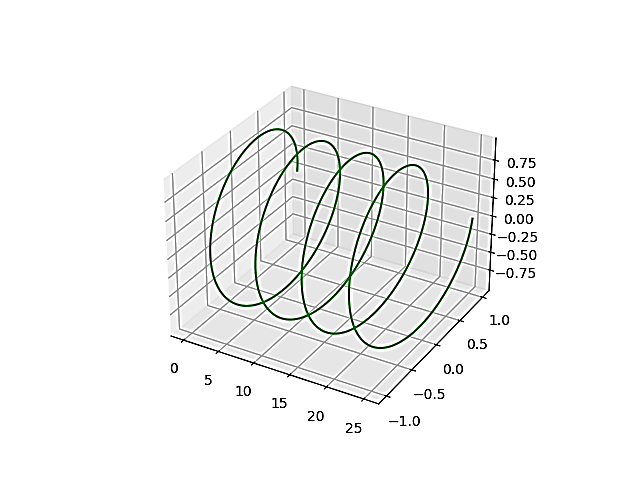
z = np.sin(theta)

y = np.cos(theta)

ax.plot(x, y, z, "g")

plt.show()

Параметри функції plot масиви з координатами точок по осях x, y та z



**Діаграма розсіювання**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B4%D1%96%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%B0-%D1%80%D0%BE%D0%B7%D1%81%D1%96%D1%8E%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)Для побудови тривимірної діаграми розсіювання використовується функція scatter з Axes3D. Крім параметрів масивів з координатами точок по осях x, y та z, вона приймає параметр s з розмірами маркера, що дорівнює за замовчуванням 20, який може бути масивом розмірів для кожного маркера.

Покажемо побудову на прикладі.

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = fig.add\_subplot(projection="3d")

x = [5, 10, 15, 20]

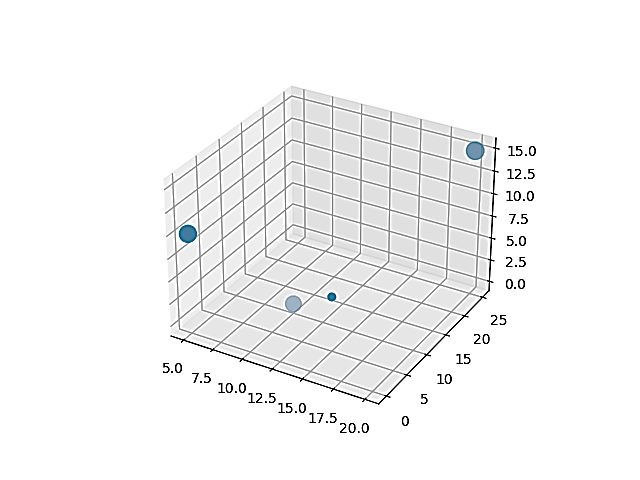
z = [10, 0, 5, 15]

y = [0, 10, 5, 25]

s = [150, 130, 30, 160]

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

plt.show()



**Каркасна поверхня**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BA%D0%B0%D1%80%D0%BA%D0%B0%D1%81%D0%BD%D0%B0-%D0%BF%D0%BE%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D1%8F)Для побудови каркасної поверхні використовується функція plot\_wireframe з Axes3D

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import axes3d

import numpy as np

fig = plt.figure()

ax = fig.add\_subplot(projection="3d")

grid = np.arange(-10, 10, 0.5)

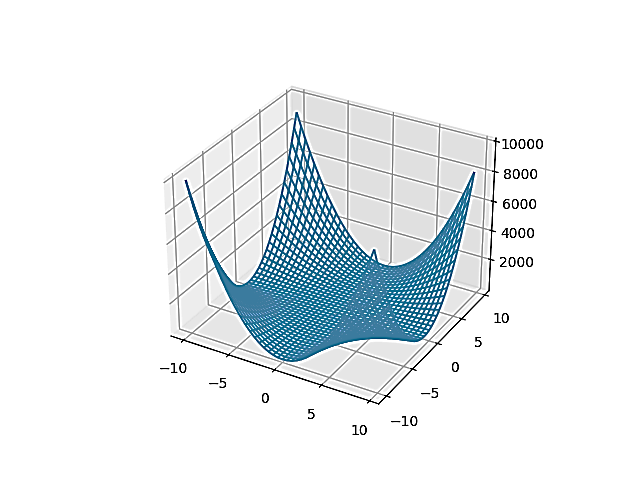
x, y = np.meshgrid(grid, grid)

z = x \*\* 2 \* y \*\* 2 + 2

ax.plot\_wireframe(x, y, z)

plt.show()

Ми будуємо поверхню для рівняння 𝑦=𝑥2𝑦2+2*y*=*x*2*y*2+2. Спочатку ми складаємо список від -10 до 10 з кроком 0.5. Потім, за допомогою методу meshgrid, ми складаємо матрицю для координат x та y. Координати для функції plot\_wireframe повинні бути у вигляді матриць.



**Поверхня**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BF%D0%BE%D0%B2%D0%B5%D1%80%D1%85%D0%BD%D1%8F)Для побудови поверхні використовується функція plot\_surface

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import axes3d

import numpy as np

fig = plt.figure()

ax = fig.add\_subplot(projection="3d")

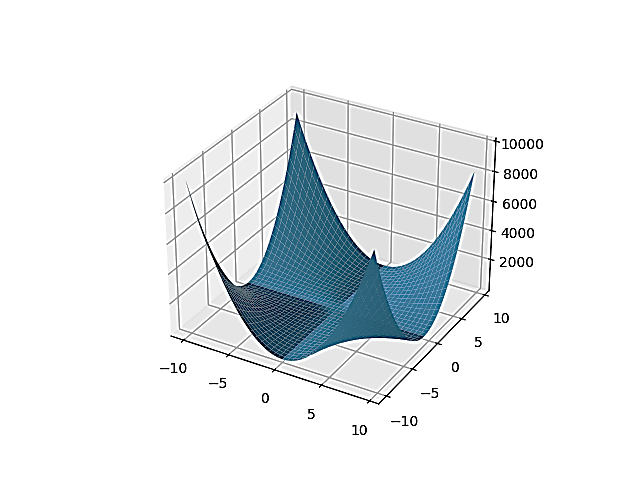
grid = np.arange(-10, 10, 0.5)

x, y = np.meshgrid(grid, grid)

z = x \*\* 2 \* y \*\* 2 + 2

ax.plot\_surface(x, y, z)

plt.show()



**Seaborn**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#seaborn)[Seaborn](https://seaborn.pydata.org/index.html) (https://seaborn.pydata.org/index.html)— це бібліотека для вирішення задач візуалізації з орієнтацією на роботу в галузі статистики. Базою seaborn є бібліотека Matplotlib.

Для встановлення бібліотеки можна скористатися пакетним менеджером pip:

pip install seaborn

Під час імпорту seaborn їй задається псевдонім sns, він є загальноприйнятим серед користувачів цієї бібліотеки, рекомендуємо вам дотримуватися цього варіанту.

import seaborn as sns

Для експериментів можна використовувати дані, що завантажуються за допомогою функції load\_dataset. Як аргумент вона приймає ім'я необхідного набору даних.

Всі ці набори даних знаходяться у вигляді csv файлів у наступному [репозиторії](https://github.com/mwaskom/seaborn-data" \t "_blank) (https://github.com/mwaskom/seaborn-data).

Якщо ми хочемо використовувати свої власні дані, нам доведеться використовувати pandas. Нехай наші власні дані знаходяться у файлі csv з ім'ям, наприклад my\_data.csv, і зберегли ми їх у тому самому місці, що і наш скрипт, тоді:

import pandas as pd

my\_data = pd.read\_csv('my\_data.csv')

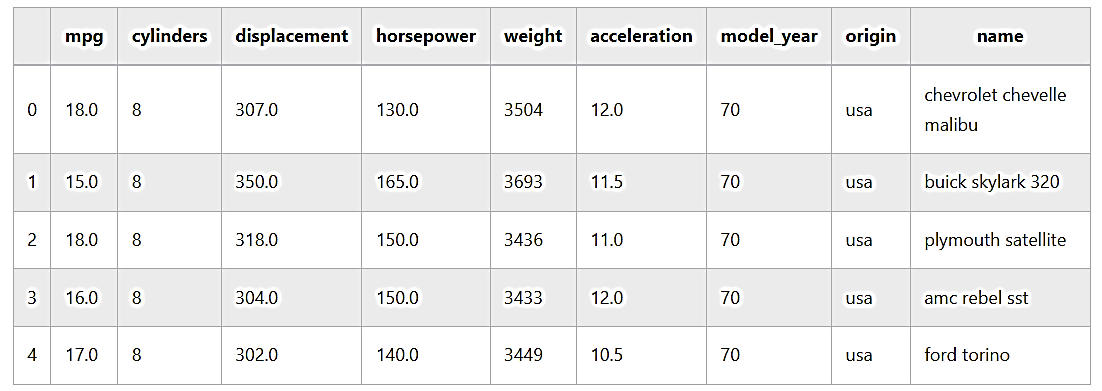
Як ілюстрацію роботи функцій seaborn ми будемо використовувати набір даних mpg:

data = sns.load\_dataset("mpg")

Функція load\_dataset сформує набір даних та поверне його у вигляді знайомого нам об'єкту pandas.DataFrame.

Набір mpg містить інформацію про характеристики ряду автомобілів. Це відкритий набір в [Kaggle](https://www.kaggle.com/uciml/autompg-dataset) (https://www.kaggle.com/datasets/uciml/autompg-dataset)

Для візуальної оцінки вмісту набору даних скористаємося методом head



**Візуалізація відносин в даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D0%B2%D1%96%D0%B4%D0%BD%D0%BE%D1%81%D0%B8%D0%BD-%D0%B2-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)Seaborn надає функції для візуалізації відносин в даних у вигляді лінійних графіків та діаграм розсіювання.

* relplot Загальний інтерфейс для візуалізації відносин з параметрами для налаштування компонування.
* lineplot Лінійний графік із семантичним групуванням.
* scatterplot Діаграма розсіювання із семантичним групуванням.

Загальні параметри функцій:

* x, y — пов'язують вісь x та y з конкретними ознаками з набору даних, переданого через параметр data. Дані повинні мати числовий тип. Фактично це назви стовпців із набору даних. Якщо вказати значення None, то побудова буде для всіх наборів відразу.
* data — набір даних у форматі pandas.DataFrame, у якому стовпці - це імена змінних, рядки - їх значення.
* ci — чисельно визначає розмір довірчого інтервалу, що відображається. Якщо присвоїти `sd', то використовує стандартне відхилення. Щоб не відображати, присвоїти None.
* hue — ім'я стовпця з набору data, який задає ознаку в наборі даних для колірного поділу даних.
* palette — палітра, яка буде використана для колірного поділу набору даних за значеннями ознаки, зазначеної в hue. Ім'я палітри можна вибрати відповідно до документації.
* hue\_order — порядок застосування кольорів для даних з набору, переданого через параметр hue.
* size — ім'я стовпця з набору data, який буде використаний для поділу даних за розміром.
* size\_order — задає порядок розподілу товщини лінії між елементами з набору даних, заданого через параметр size.
* style — ім'я стовпця з набору data, щоб задати ознаку в наборі даних, яка буде використана для поділу даних за стилем.
* markers — тип маркеру. Якщо параметр дорівнює False, то маркери не використовуватимуться.
* style\_order — порядок застосування стилів.

**Функція lineplot**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-lineplot)

Такий тип графіка найчастіше використовується для візуалізації часових рядів та залежностей між змінними, що мають безперервний характер.

Встановимо стиль оформлення графіків:

sns.set\_style("whitegrid")

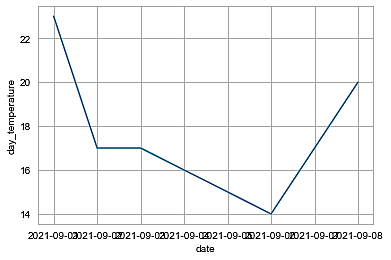
Для побудови та розбору цього типу графіка добре підійдуть наші дані про температуру.

date = pd.date\_range(start="2021-09-01", freq="D", periods=8)

day = [23, 17, 17, 16, 15, 14, 17, 20]

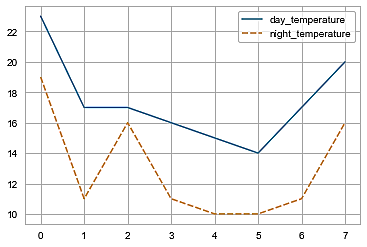
night = [19, 11, 16, 11, 10, 10, 11, 16]

df = pd.DataFrame({'date':date, 'day\_temperature': day, 'night\_temperature': night})



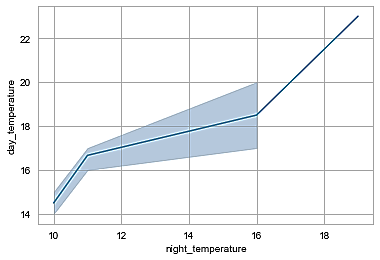
Якщо необхідно відобразити всі елементи з DataFrame набору, то не потрібно задавати параметри x та y

sns.lineplot(data=df)



Якщо ми спробуємо побудувати графік залежності нічної температури від денної, то отримаємо наступний графік:

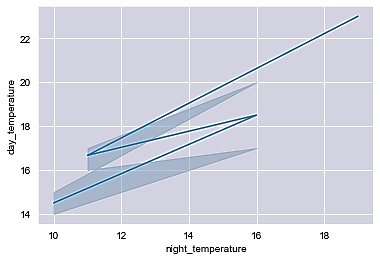
sns.lineplot(x="night\_temperature", y="day\_temperature", data=df)



У цьому разі по осі ординат відкладатимуться не значення параметра y, а його математичне очікування. Світліша область навколо лінії – це 95% довірчий інтервал. Це відбувається через те, що для нічної температури 11 є значення 16, 17, для 16 - це 17, 20, а для 10 - це 14 і 15.

Необхідно знати, що за замовчуванням seaborn здійснює сортування набору даних, цю опцію можна відключити, задаючи параметру sort значення False

sns.lineplot(x="night\_temperature", y="day\_temperature", data=df, sort=False)



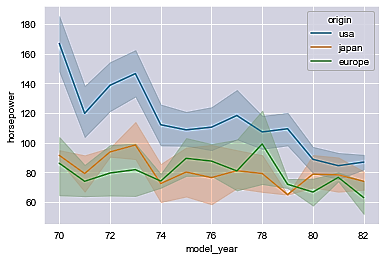
Після простого прикладу на температурах повернемося тепер до набору mpg.

Побудуємо графік залежності потужності автомобіля від року випуску окремо для кожного регіону.

sns.set\_style("darkgrid")

data = sns.load\_dataset("mpg")

sns.lineplot(x='model\_year', y='horsepower', hue="origin", data=data)

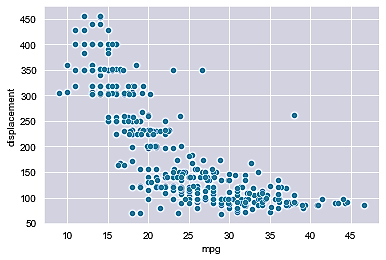


**Функція scatterplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-scatterplot)

Побудуємо точковий графік залежності дальності пробігу (displacement) автомобіля від витрат палива (mpg):

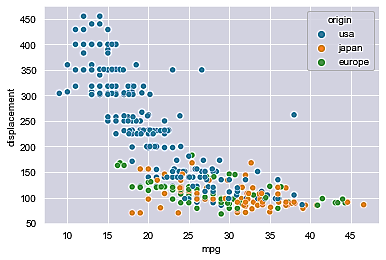
data = sns.load\_dataset("mpg")

sns.scatterplot(x='mpg', y='displacement', data=data)



Ми можемо виділити кольором регіон виробника

sns.scatterplot(x='mpg', y='displacement', hue='origin', data=data)



**Функція relplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-relplot)

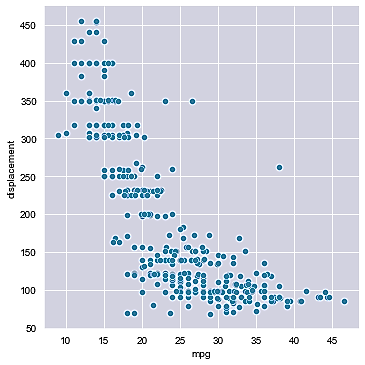
На відміну від попередніх функцій, ця функція надає можливість налаштовувати не лише зовнішній вигляд графіка, але й фігуру - підкладку, на якій розміщені всі графічні компоненти.

Параметри для налаштування фігури:

* row, col — назва стовпців, за якими буде проводитися поділ фігури на рядки та стовпці. Можна використовувати лише категоріальні ознаки.
* col\_wrap — кількість стовпців для об'єднання.
* row\_order, col\_order — список, що визначає порядок рядків та/або стовпців відповідно до перерахованих значень ознаки.
* kind — визначає тип графіка, що відображається, лінійний line і точковий scatter.
* height — висота поля графіка в дюймах.
* aspect — параметр, що визначає співвідношення сторін поля з графіком, ширина графіка розраховується за формулою 𝑤𝑖𝑑𝑡ℎ=𝑎𝑠𝑝𝑒𝑐𝑡∗ℎ𝑒𝑖𝑔ℎ𝑡*width*=*aspect*∗*height*

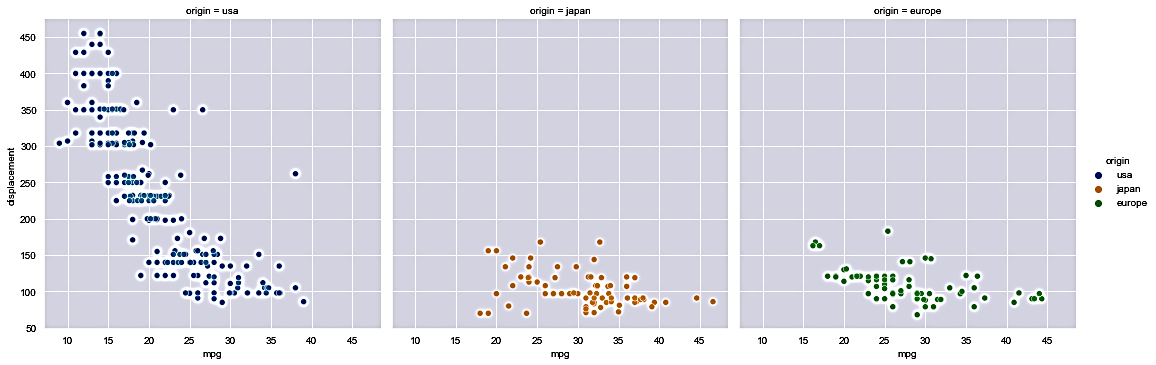
sns.relplot(x='mpg', y='displacement', kind='scatter', data=data)

В принципі ми отримаємо той самий графік, що для scatterplot



Зручність полягає в тому, що діаграму можна розмістити на трьох окремих полях, як роздільний параметр будемо використовувати регіон (origin)

sns.relplot(x='mpg', y='displacement', kind='scatter', hue='origin', col='origin', data=data)



**Візуалізація категоріальних даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D0%BA%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)

Особливість категоріальних ознак полягає в тому, що вони приймають значення з невпорядкованої множини. Це ознаки кольору, форми тощо. І до них не можна застосувати операцію порівняння. Щоправда є окремий випадок порядкових ознак, коли можна провести порівняння, наприклад кафедра – деканат – університет.

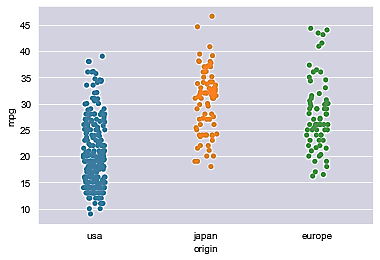
Seaborn надає набір функцій для категоріальних даних. Загальні параметри цих функцій наступні:

* x, y — імена змінних із набору data
* data — набір даних типу pandas.DataFrame, у якому стовпці - це імена ознак, рядки - значення.
* hue — ім'я змінної з набору data, яка визначає ознаку в наборі даних, який буде використаний для колірного поділу даних. Візуально групи будуть представлені у вигляді окремих елементів, що відрізняються кольором.
* order, hue\_order — список, що визначає порядок відображення елементів.
* orient — орієнтація графіка, символ v вертикальна, символ h горизонтальна.
* color — колір для всіх елементів, що відображаються.
* palette — ім'я палітри.

**Функція stripplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-stripplot)

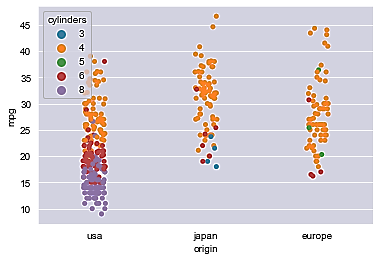
Функція будує точкову діаграму і за функціоналом схожа на функцію scatterplot

sns.stripplot(x='origin', y='mpg', data=data)



Тепер виділимо кольором кількість циліндрів у двигуна, для цього присвоїмо параметру hue значення cylinders:

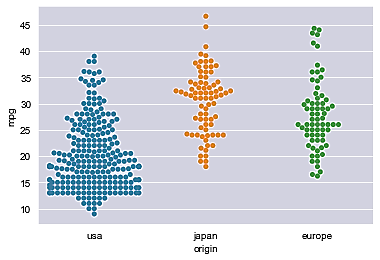
sns.stripplot(x='origin', y='mpg', hue='cylinders', data=data)



**Функція swarmplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-swarmplot)

Функція аналогічна функції stripplot, але точки, що відображаються на діаграмі, не перекриваються, це дозволяє робити висновки про переважання тих або інших значень в наборах даних з їх візуального розподілу.

sns.swarmplot(x='origin', y='mpg', data=data)



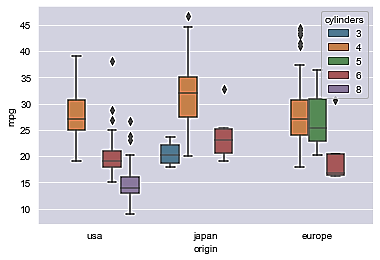
**Візуалізації розподілу категоріальних даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%97-%D1%80%D0%BE%D0%B7%D0%BF%D0%BE%D0%B4%D1%96%D0%BB%D1%83-%D0%BA%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)

Seaborn надає три функції для візуалізації розподілу категоріальних даних:

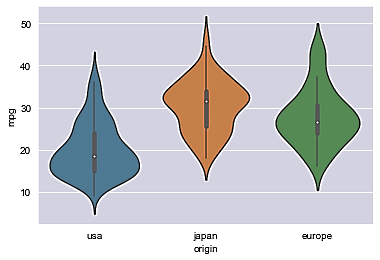
boxplot — будує діаграму типу "скринька з вусами", на ній відображаються медіанне значення, квартілі та викиди

sns.boxplot(x="origin", y="mpg", hue="cylinders", data=data)



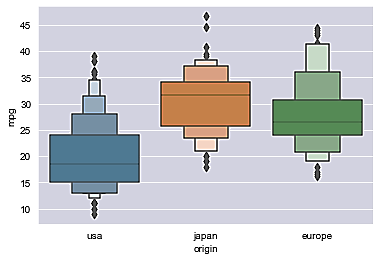
violinplot — будує діаграму, схожу на "скриньку з вусами" з оцінкою щільності ядра

sns.violinplot(x="origin", y="mpg", data=data)



boxenplot — будує діаграму з прямокутників, добре підходить для візуалізації великих наборів даних.

sns.boxenplot(x="origin", y="mpg", data=data)



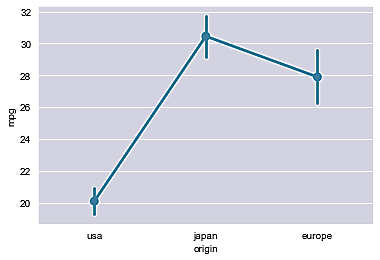
**Візуалізація оцінок категоріальних даних**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F-%D0%BE%D1%86%D1%96%D0%BD%D0%BE%D0%BA-%D0%BA%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D1%85-%D0%B4%D0%B0%D0%BD%D0%B8%D1%85)

**Функція pointplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-pointplot)

Відображає оцінку будь-якого набору даних як точку на полі графіка та довірчий інтервал у вигляді лінії, центр якої лежить на зазначеній точці.

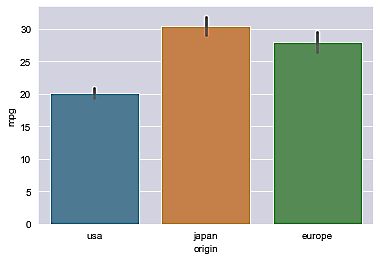
sns.pointplot(x="origin", y="mpg", data=data)



**Функція barplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-barplot)

Функція будує стовпчасту діаграму: висота бару (стовпця) визначає чисельне значення оцінки ознаки (математичне очікування), лінія, що перетинає верхню межу бару - довірчий інтервал.

sns.barplot(x="origin", y="mpg", data=data)



**Функція countplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-countplot)

Функція визначає кількість елементів з набору даних, які належать до тієї або іншої категорії, і відображає отримане значення у вигляді стовпчастої діаграми.

sns.countplot(x="cylinders", data=data)



**Візуалізація моделі лінійної регресії**

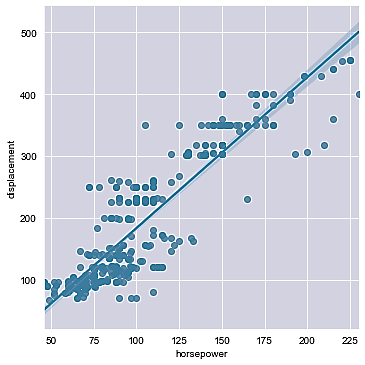
Seaborn надає набір функцій для побудови моделі лінійної регресії за переданими даними та відображають її разом із вихідним набором даних.

**Функція regplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-regplot)

Функція regplot відображає набір даних та лінію регресії, побудовану за ними.

Побудуємо лінію регресії залежності відстані, яку може проїхати автомобіль, від його потужності.

sns.lmplot(x="horsepower", y="displacement", data=data)

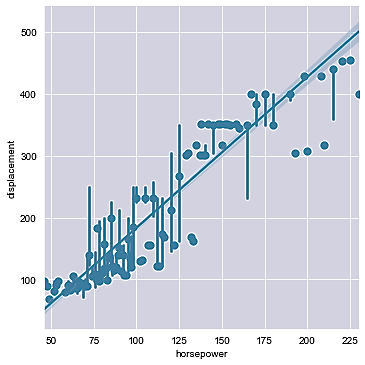


Можна побудувати діаграму з медіанною оцінкою та необхідні параметри функції при цьому:

* x\_estimator — функція для обчислення значення оцінки.
* x\_bins — число для визначення кількості груп, на які буде розбито вихідна множина значень.
* x\_ci — Число в діапазоні [0, 100] або None, розмір довірчого інтервалу.

from numpy import median

sns.lmplot(x="horsepower", y="displacement", data=data, x\_estimator=median)

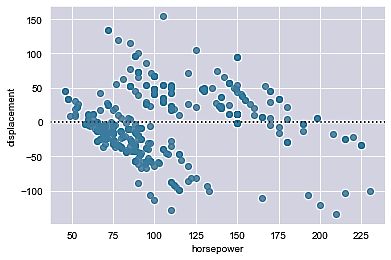


**Функція residplot**[**​**](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-residplot)

Функція відображає відхилення елементів вихідного набору даних від регресійної моделі, побудованої за ними, у вигляді діаграми розсіювання.

Кожна точка такої діаграми – це різниця між значенням елемента вихідного набору та значенням, яке видасть модель регресії у цій точці.

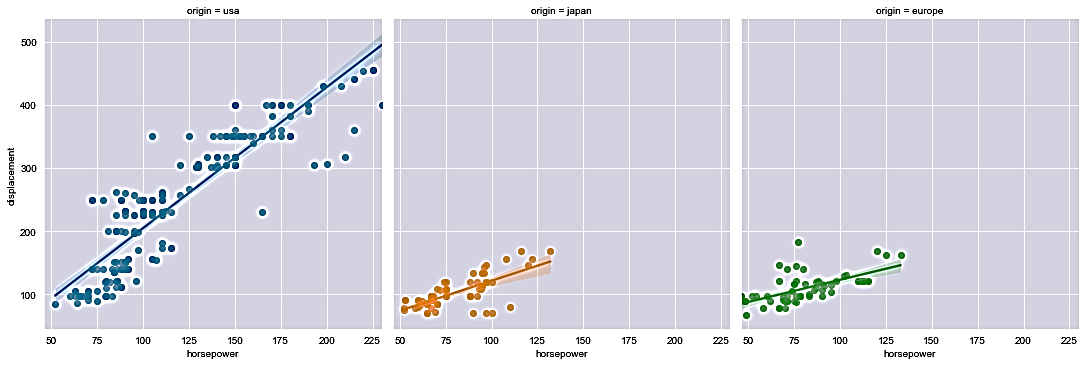
sns.residplot(x="horsepower", y="displacement", data=data)



**Функція lmplot[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1" \l "%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F-lmplot" \t "_blank)**

Функція повний аналог функції regplot з можливістю управляти компонуванням полів з графіками на підкладці як для функції relplot.

sns.lmplot(x="horsepower", y="displacement", hue="origin", col="origin", data=data)



**Візуальне оформлення графіків**[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B2%D1%96%D0%B7%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5-%D0%BE%D1%84%D0%BE%D1%80%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F-%D0%B3%D1%80%D0%B0%D1%84%D1%96%D0%BA%D1%96%D0%B2)

**Стилі**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%81%D1%82%D0%B8%D0%BB%D1%96)Найпростіший і найшвидший спосіб задати оформлення для графіка seaborn - це використовувати один із заздалегідь підготовлених стилів за допомогою функції set\_style або використовувати свої стилі з індивідуальним оформленням.

set\_style(style=None, rc=None)

* style — це або свій словник з параметрами, або ім'я стилю із заданого набору: darkgrid, whitegrid, dark, white, ticks.
* rc — словник, який перевизначає параметри аргументу style.

Отримати список параметрів, що відповідають за оформлення графіка, можна за допомогою функції axes\_style.

Для набору mpg ми всі графіки будували у стилі darkgrid.

sns.set\_style("darkgrid")

data = sns.load\_dataset("mpg")

sns.lineplot(x='model\_year', y='horsepower', hue="origin", data=data)

sns.axes\_style()

{'axes.facecolor': '#EAEAF2',

'axes.edgecolor': 'white',

'axes.grid': True,

'axes.axisbelow': True,

'axes.labelcolor': '.15',

'figure.facecolor': 'white',

'grid.color': 'white',

'grid.linestyle': '-',

'text.color': '.15',

'xtick.color': '.15',

'ytick.color': '.15',

'xtick.direction': 'out',

'ytick.direction': 'out',

'lines.solid\_capstyle': 'round',

'patch.edgecolor': 'w',

'patch.force\_edgecolor': True,

'image.cmap': 'rocket',

'font.family': ['sans-serif'],

'font.sans-serif': ['Arial',

'DejaVu Sans',

'Liberation Sans',

'Bitstream Vera Sans',

'sans-serif'],

'xtick.bottom': False,

'xtick.top': False,

'ytick.left': False,

'ytick.right': False,

'axes.spines.left': True,

'axes.spines.bottom': True,

'axes.spines.right': True,

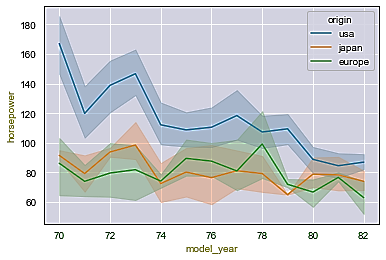
'axes.spines.top': True}

Як мовилося раніше, ми можемо перевизначати параметри оформлення.

sns.set\_style("darkgrid", {'axes.labelcolor':"(0.5,0.5,0)", 'axes.edgecolor':'#061358',

'xtick.color':'#0A5806'})

sns.lineplot(x='model\_year', y='horsepower', hue="origin", data=data)



**Контекст**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%BA%D1%81%D1%82)Також в seaborn використовуються контексти для управління масштабом зображення.

Для встановлення контексту використовується функція set\_context:

set\_context(context=None, font\_scale=1, rc=None)

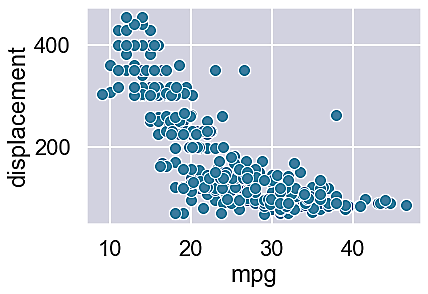
Параметри:

* context — параметр з набору: paper, notebook, talk, poster.
* font\_scale — коефіцієнт для зміни розміру шрифту.
* rc — словник із параметрами для перевизначення властивостей, заданого через аргумент context.

Для отримання списку параметрів контексту використовується функція plotting\_context

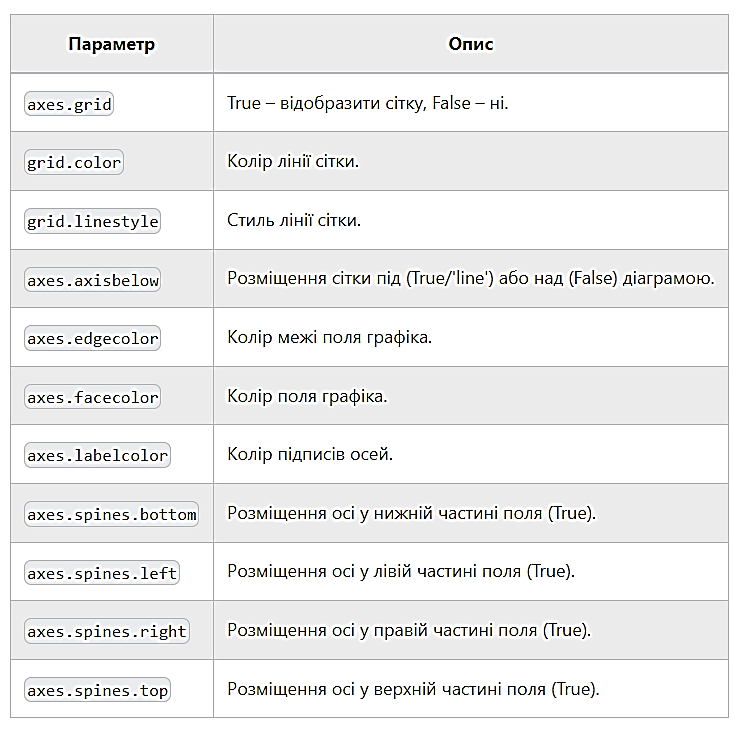
sns.set\_context("poster")

sns.scatterplot(x='mpg', y='displacement', data=data)

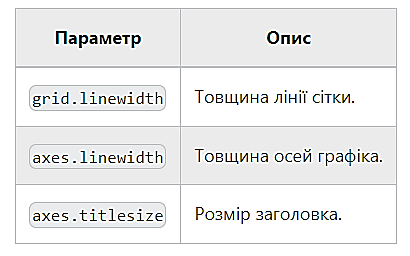


**Сітка**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%81%D1%96%D1%82%D0%BA%D0%B0)Параметри стилю для налаштування сітки



Параметри контексту для налаштування сітки



**Легенда**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BB%D0%B5%D0%B3%D0%B5%D0%BD%D0%B4%D0%B0)

Легенда на графіку відображається автоматично, якщо ви використовуєте якийсь додатковий параметр для групування даних за тими або іншими ознаками.

Безпосередньо сама бібліотека seaborn не надає інструментів для налаштування візуального оформлення легенди. Єдиний параметр - це legend.fontsize для управління розміром шрифту легенди.

**Шрифт**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D1%88%D1%80%D0%B8%D1%84%D1%82)За налаштування шрифту відповідають параметри font.family (із стилю) та font\_scale (з контексту).

**Колір**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%BA%D0%BE%D0%BB%D1%96%D1%80)Функція color\_palette повертає список кольорів.

color\_palette(palette=None, n\_colors=None, desat=None)

Параметри функції:

* palette — назва палітри або набір кольорів. Якщо значення дорівнює None, то буде повернута поточна палітра.
* n\_colors — кількість кольорів в палітрі.
* desat — коефіцієнт насиченості, початкове значення 1.

Функція set\_palette встановлює колірну палітру як поточну. Призначення параметрів збігається із зазначеними для функції color\_palette.

Щоб побачити поточну схему кольорів:

sns.palplot(sns.color\_palette())



**Додаткові джерела**

[​](https://textbook.edu.goit.global/python/data-science-remaster/v1/docs/module-02/main_1#%D0%B4%D0%BE%D0%B4%D0%B0%D1%82%D0%BA%D0%BE%D0%B2%D1%96-%D0%B4%D0%B6%D0%B5%D1%80%D0%B5%D0%BB%D0%B0)

* за потреби звертайтеся до [документації бібліотеки matplotlib](https://matplotlib.org/stable/contents.html)

(https://matplotlib.org/stable/users/index.html)

* відповідно також [документація бібліотеки seaborn](https://seaborn.pydata.org/)

(https://seaborn.pydata.org/)

* додатково [записничок з прикладами візуалізації від Google Colaboratory](https://colab.research.google.com/notebooks/charts.ipynb)

(https://colab.research.google.com/notebooks/charts.ipynb)