**Лабораторна робота №7**

**Тема:** Побудова графіків математичних функцій у Python

**Мета:** набути навичок роботи з бібліотекою Matplotlib для візуалізації даних

**Теоретична частина**

**Matplotlib** – бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Отримувані зображення можуть бути використані як ілюстрації в публікаціях. Зображення, які генеруються в різних форматах, можуть бути використані в інтерактивній графіці, наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де потрібно будувати діаграми (англ. plotting).

Бібліотека Matplotlib побудована на принципах ООП, але має процедурний інтерфейс pylab, який надає аналоги команд MATLAB.

Пакет підтримує багато видів графіків і діаграм:

* Графіки (line plot)
* Діаграми розсіювання (scatter plot)
* Стовпчасті діаграми (bar chart) і гістограми (histogram)
* Секторні діаграми (pie chart)
* Діаграми «Стовбур-листя» (stem plot)
* Контурні графіки (contour plot)
* Поля градієнтів (quiver)
* Спектральні діаграми (spectrogram)

Набір підтримуваних форматів зображень, векторних і растрових, можна отримати з словника FigureCanvasBase.filetypes. Типові підтримувані формати: EPS, EMF, JPEG, PDF, PNG, PostScript, RGBA, SVG, SVGZ, TIFF.

Розглянемо побудову графіків на прикладах.

**Набір точок**

>>> **import** **matplotlib.pyplot** **as** **plt**

>>> plt.plot([1, 3, 2, 4])

[<matplotlib.lines.Line2D object at 0x01A00430>]

>>> plt.show()

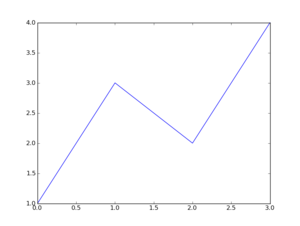
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_1.png)

Рисунок 1

Функція plot() будує графік, а функція show() його показує. Аргумент, що приймається функцією plot() – це послідовність y-значень. Інший, який ми опустили, що стоїть перед y – це послідовність x-значень. Оскільки його немає, графік генерується для чотирьох зазначених y, список з чотирьох x: [0, 1, 2, 3].

## Функція

**from** **numpy** **import** \* # для використання функційexpта linspace

**import** **matplotlib.pyplot** **as** **plt**

**def** f(t):

**return** t\*\*2\*exp(-t\*\*2)

t = linspace(0, 3, 51) *# 51 точка між 0 та 3*

y = f(t)

plt.plot(t, y)

plt.show()

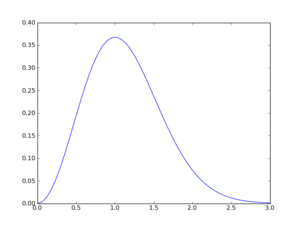
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_2.png)

Рисунок 2

Якщо функція більше ніде не використовується, то можна отримати ще більш компактний код, задавши її відразу ж після визначення масиву t:

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace(0, 3, 51)

y = t\*\*2\*exp(-t\*\*2)

plt.plot(t, y)

plt.show()

## Налаштування вигляду графіків

Крім того, щоб просто побудувати криву, було б добре її назвати, позначити осі, вивести легенду (це особливо стане в нагоді, якщо будувати кілька графіків). Крім того, інколи потрібно змінити вигляд самої кривої, межі її побудови.

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace(0, 3, 51)

y = t\*\*2\*exp(-t\*\*2)

plt.plot(t, y, 'g--', label='t^2\*exp(-t^2)')

plt.axis([0, 3, -0.05, 0.5]) *# задання [xmin, xmax, ymin, ymax]*

plt.xlabel('t') *# позначення вісі абсцис*

plt.ylabel('y') *# позначення е вісі ординат*

plt.title('My first normal plot') *# назва графіка*

plt.legend() *# вставка легенди (тексту в label)*

plt.show()

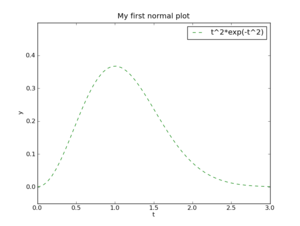
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_3.png)

Рисунок 3

Крім зазначених нововведень, позначених в коментарях, в аргументах функції plot() ми бачимо два нових. Останній задає текст легенди графіка. Строковий аргумент g-- відповідальний за те, що змінився вигляд кривої. У порівнянні з попереднім прикладом, графік позеленів (green) і вимальовується - штриховою лінією. За замовчуванням цей аргумент b-, що означає синю (blue) суцільну лінію. Нижче наведена таблиця, яка дозволяє вибрати потрібний аргумент.

|  |  |
| --- | --- |
| b, blue | синій колір |
| c, cyan | блакитний колір |
| g, green | зелений колір |
| k, black | чорний колір |
| m, magenta | пурпурний колір |
| r, red | червоний колір |
| w, white | білий колір |
| y, yellow | жовтий колір |
| - | суцільна лінія |
| -- | штрихова лінія |
| -. | штрих-пунктирна лінія |
| : | пунктирна лінія |

## Декілька кривих на одному графіку

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace(0, 3, 51)

y1 = t\*\*2\*exp(-t\*\*2)

y2 = t\*\*4\*exp(-t\*\*2)

plt.plot(t, y1, label='t^2\*exp(-t^2)')

plt.plot(t, y2, label='t^4\*exp(-t^2)')

*# декоративна частина*

plt.xlabel('t')

plt.ylabel('y')

plt.title('Plotting two curves in the same plot')

plt.legend()

plt.show()

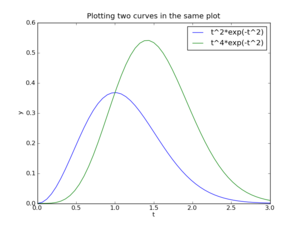
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_4.png)

Рисунок 4

## Декілька кривих на одному графіку - 2

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace(0, 3, 51)

y1 = t\*\*2\*exp(-t\*\*2)

y2 = t\*\*4\*exp(-t\*\*2)

y3 = t\*\*6\*exp(-t\*\*2)

plt.plot(t, y1, 'g^', *# маркери із зелених трикутників*

t, y2, 'b--', *# синя штриховая*

t, y3, 'ro-') *# червоні круглі маркери*

*# з'єднані суцільною лінією*

plt.xlabel('t')

plt.ylabel('y')

plt.title('Plotting with markers')

plt.legend(['t^2\*exp(-t^2)',

't^4\*exp(-t^2)',

't^6\*exp(-t^2)'], *# список легенди*

loc='upper left') *# положення легенди*

plt.show()

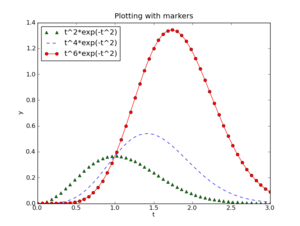
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_5.png)

Рисунок 5

Ми змінили положення легенди, інакше б вона закривала червоний графік. За замовчуванням легенда розташовується в правому верхньому куті, але можна її і перенести за рахунок аргументу loc. Цьому аргументу можна привласнювати і чисельне значення, але звичайно легше сприймається рядок. У таблиці нижче приводяться можливі варіанти.

|  |  |  |
| --- | --- | --- |
| Місце | String | Code |
| кращий варіант | best | 0 |
| вгорі справа | upper right | 1 |
| вгорі зліва | upper left | 2 |
| внизу справа | lower left | 3 |
| внизу зліва | lower right | 4 |
| справа | right | 5 |
| посередині зліва | center left | 6 |
| посередині справа | center right | 7 |
| посередині внизу | lower center | 8 |
| посередині вгорі | upper center | 9 |
| посередині | center | 10 |

**Маркери**

Тут також показано як можна об'єднувати відразу три графіка в одній інструкції. Крім того, видно, що можна не тільки використовувати маркери (y1) або лінії (y2), але і об'єднувати їх разом (y3). Найбільш часто в наукових дослідженнях і журналах призводять графіки, що відрізняються один від одного саме маркерами, тому і в matplotlib для їх позначення є безліч способів:

. точковий маркер;

, точки, розміром з піксель;

o кола;

˅ трикутники носом вниз;

˄ трикутники носом догори;

> трикутники дивляться вправо;

< трикутники дивляться вліво;

s квадрати;

p п’ятикутнки;

\* зірочки;

h шестикутники;

H повернені шестикутники;

+ плюси;

x хрестики;

D ромби;

d вузькі ромби;

| вертикальні зарубки.

**Додаткові аргументи plot()**

Отже, в один аргумент ми можемо поставити відразу три параметра: першим вказуємо колір, другим – стиль лінії, третім – тип маркера. Однак вже така нотація може у людини незнайомої з нею, викликати подив. Крім того, вона не дозволяє розділяти параметри лінії і маркера, тому існує варіант з використанням keywords – також це дозволяє щедра функція plot():

|  |  |
| --- | --- |
| **Keyword argument** | **Що міняє** |
| color або c | колір лінії |
| linestyle | стиль лінії, використовуються позначення, показані вище |
| linewidth | товщина лінії у вигляді float-числа |
| marker | вид маркера |
| markeredgecolor | колір краю (edge) маркера |
| markeredgewidth | товщина краю маркера |
| markerfacecolor | колір самого маркера |
| markersize | розмір маркера |

Можна також вносити зміни у відмітки на осях координат. Робиться це за допомогою функцій xticks() і yticks(), в які передаються один або два списки значень: або просто список згаданих значень, або їх же, але спочатку ті місця, на які вони встають:

x = [5, 3, 7, 2, 4, 1]

plt.xticks(range(len(x)), ['a', 'b', 'c', 'd', 'e', 'f'])

plt.yticks(range(1, 8, 2))

Для нанесення сітки існує команда:

plt.grid(True)

Для того, щоб одну або декілька осей виставити в логарифмічному масштабі застосовуються команди plt.semilogx() и plt.semilogy().

**Збереження файлу**

**from** **numpy** **import** \*

**import** **matplotlib.pyplot** **as** **plt**

t = linspace(0, 3, 51)

y = t\*\*2\*exp(-t\*\*2)

plt.plot(t, y)

plt.savefig('name\_of\_plot.png', dpi=200)

Файл зберігається в тій же директорії з ім'ям і розширенням, зазначеним в першому аргументі. Другий необов'язковий аргумент дозволяє «на льоту» змінювати роздільну здатність картинки, що зберігається у файл.

Буває так, що дивитися на картинки в уже налаштованій програмі не потрібно і потрібно їх саме зберігати на майбутнє, щоб переглянути і порівняти їх всі разом. Тоді нам не потрібно запускати вікно перегляду результатів. Для цього до колишніх інструкцій посилаємо головному модулю повідомлення:

**import** **matplotlib**

matplotlib.use('Agg')

## Гістограми

## Для побудови гістограм (діаграм у вигляді набору стовпчиків) в Matplotlib використовуються функція bar і barh, які будують вертикальні або горизонтальні гістограми відповідно. Ці функції, як і інші функції малювання, імпортуються з модуля pylab. Функції bar і barh мають безліч необов'язкових параметрів з додатковими настройками, ми розглянемо тільки найбільш часто використовувані можливості для налаштування зовнішнього вигляду гістограм.

## Функції bar і barh мають два обов'язкових параметра:

## • Список координат розташування стовпчиків по осі X для bar або по осі Y для barh.

## • Значення, що задають висоту (довжину) стовпчиків.

## Довжини цих двох списків повинні бути рівні.

## Найпростіший приклад може виглядати так:

**import** pylab

xdata = [0, 1, 2, 4, 5, 8]

ydata = [0.1, 0.2, 0.4, 0.8, 0.6, 0.1]

pylab.bar (xdata, ydata)

pylab.show()

## 

Рисунок 6

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

y = np.random.randn(1000)

plt.hist(y, 25)

plt.show()

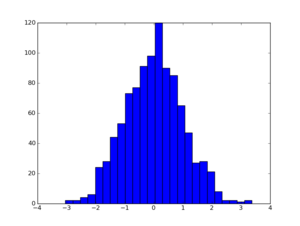
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_Histogram.png)

Рисунок 7

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

data1=10\*np.random.rand(5)

data2=10\*np.random.rand(5)

data3=10\*np.random.rand(5)

locs = np.arange(1, len(data1)+1)

width = 0.27

plt.bar(locs, data1, width=width)

plt.bar(locs+width, data2, width=width, color='red')

plt.bar(locs+2\*width, data3, width=width, color='green')

plt.xticks(locs + width\*1.5, locs)

plt.show()

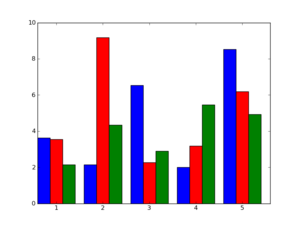
[](https://ru.wikiversity.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:PythonScience_-_Matplotlib_-_Bar_charts.png)

Рисунок 8

**Текст, примітки**

Крім тексту в назвах, підписах до осей, легенди, можна безпосередньо вставляти його в графік за допомогою простої функції text(x, y, text), де x і y координати, а text текстовий рядок. Ця функція вставляє текст відповідно до координат даних. Існує можливість вводити і в координатах графіка, в яких за (0, 0) приймається нижній лівий кут, а за (1, 1) правий верхній. Це робиться за допомогою функції figtext(x, y, text).

Текстові функції, звичайно вставляють текст в графік, але часто буває потрібно саме вказати, виділити якийсь екстремум, незвичайну точку. Це легко зробити за допомогою приміток – функції annotate( 'annotation', xy = (x1, y1), xytext = (x2, y2)). Тут замість annotation ми пишемо текст примітки, замість (x1, y1) координати цікавої точки, замість (x2, y2) координати, де ми хочемо вставити текст.

**Завдання 1:** зобразити 2d графік функції відповідно своєму варіанту та зберегти у .png файл.

**Варіанти:**

|  |  |
| --- | --- |
| **№** | **Функція** |
| 1. | Y(x)=x\*sin(5\*x), x=[-2...5] |
| 2. | Y(x)=1/x\*sin(5\*x), x=[-5...5] |
| 3. | Y(x)=2^x\*sin(10x), x=[-3...3] |
| 4. | Y(x)=x^(1/2)\*sin(10\*x), x=[0...5] |
| 5. | Y(x)=15\*sin(10\*x)\*cos(3\*x), x=[-3...3] |
| 6. | Y(x)=5\*sin(10\*x)\*sin(3\*x), x=[0...4] |
| 7. | Y(x)=sin(10\*x)\*sin(3\*x)/(x^2), x=[0...4] |
| 8. | Y(x)=5\*sin(10\*x)\*sin(3\*x)/(x^(1/2)), x=[1...7] |
| 9. | Y(x)=5\*cos(10\*x)\*sin(3\*x)/(x^(1/2)), x=[0...5] |
| 10. | Y(x)=-5\*cos(10\*x)\*sin(3\*x)/(x^(1/2)), x=[0...10] |
| 11. | Y(x)=-5\*cos(10\*x)\*sin(3\*x)/(x^x), x=[0...5] |
| 12. | Y(x)=5\*sin(10\*x)\*sin(3\*x)/(x^x), x=[0...8] |
| 13. | Y(x)=x^sin(10\*x), x=[1...10] |
| 14. | Y(x)=-x^cos(5\*x), x=[0...10] |
| 15. | Y(x)=x^cos(x^2), x=[0...10] |
| 16. | Y(x)=cos(x^2)/x, x=[0...5] |
| 17. | Y(x)=10\*cos(x^2)/x^2, x=[0...4] |
| 18. | Y(x)=(1/x)\*cos(x^2+1/x), x=[1...10] |
| 19. | Y(x)=sin(x)\*(1/x)\*cos(x^2+1/x), x=[-2...2] |
| 20. | Y(x)=5\*sin(x)\*cos(x^2+1/x)^2, x=[1...10] |
| 21. | Y(x)=5\*sin(1/x)\*cos(x^2+1/x)^2, x=[1...4] |
| 22. | Y(x)=5\*sin(1/x)\*cos(x^2)^3, x=[-4...4] |
| 23. | Y(x)=(x^3)\*cos(x^2), x=[-2...2] |
| 24. | Y(x)=(x^3)+cos(15\*x), x=[-2...2] |
| 25. | Y(x)=(3^x)+cos(15\*x), x=[-1...2] |

**Завдання 2:** Зобразити гістограму частоти появи літер у певному тексті та зберегти у .png файл.

**Завдання 3:** Зобразити гістограму частоти появи у певному тексті звичайних, питальних та окличних речень, а також речень, що завершуються трикрапкою та зберегти у .png файл.