Тема 5. Спеціалізовані графіки

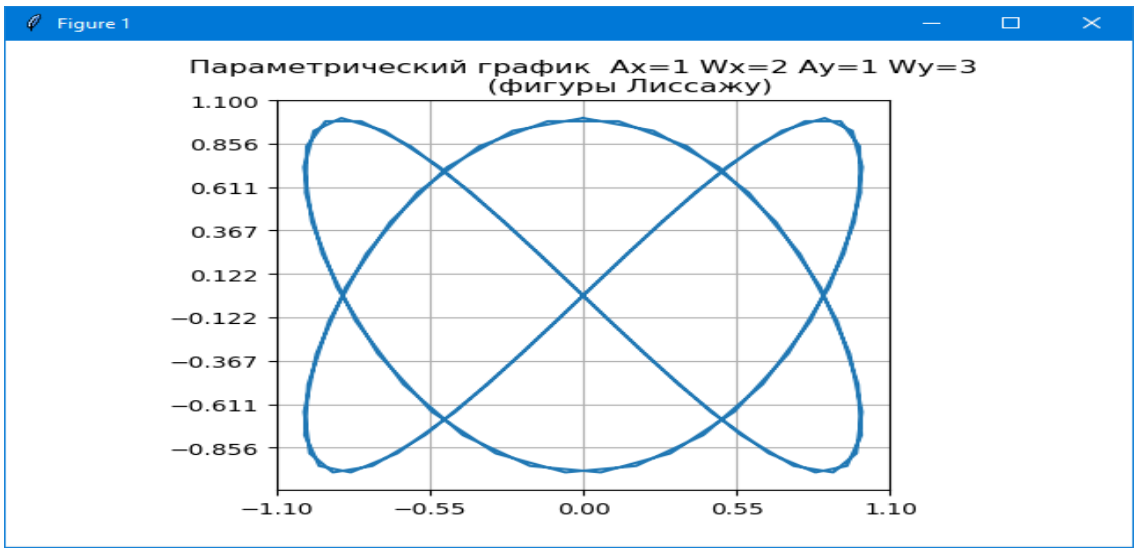
П9. Параметричний графік

Масив x повинен бути монотонно зростаючим. Можна будувати будь-яку параметричну лінію x = x (t), y = y (t).

Роздільні можливості монітора по різних осях можуть бути різними. Тому щоб кола виглядали як кола, а не як еліпси (а квадрати як квадрати, а не як прямокутники), потрібно встановити параметр - aspect ratio (зазвичай рівний 1).

Побудуємо x = Ax * cos (Wx * t), y = Ay * sin (Wy * t).

```
importmatplotlib.pyplot as plt
importnumpy as np
Ax=1; Ay=1; Wx = 2; Wy=3
# незалежна змінна
t=np.linspace(0,4*np.pi,100)
# Встановимо параметр - aspect (ratio)
axx=plt.subplot(111)
axx.set_aspect(1)
так можна змінити крок сітки на графіку
plt.xticks([i for i in \ np.linspace(-1.1*Ax, 1.1*Ax,
5)])
plt.yticks([i for i in \ np.linspace(-1.1*Ay, 1.1*Ay,
10)]))
# Задаємо заголовок діаграми
plt.title("Параметричний графік" \
          " Ax={0} Wx={1} Ay={2} Wy={3}\n" \
          "(Фігури Лісажу)" \
               .format(Ax, Wx, Ay, Wy))
plt.plot(Ax*np.sin(Wx*t),Ay*np.cos(Wy*t))
plt.grid(True)
візуалізуємо графіки
plt.show()
```

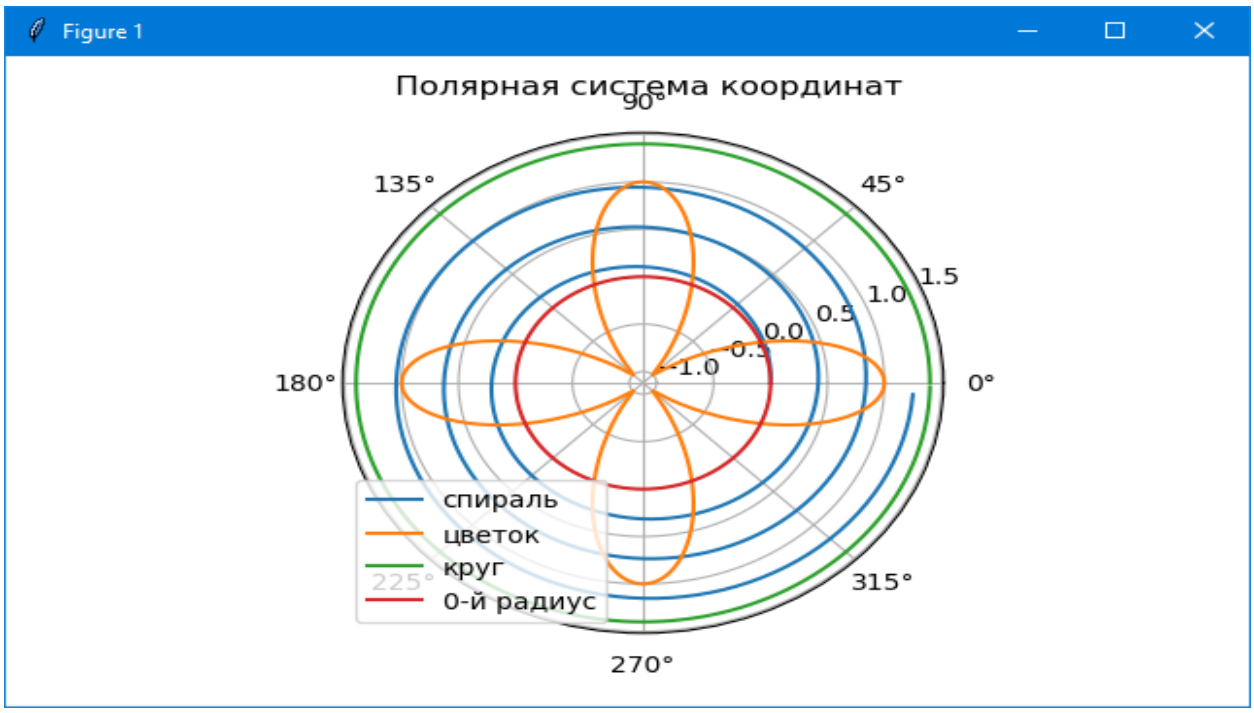


П10. Полярні координати

Крім часто використовуваної декартової системи координат, досить широко застосовується і полярна система координат, зручна в різних радіальних задачах, координати точок в ній задається за допомогою радіус-вектора ρ, що йде з початку координат і кута θ. Кут може бути заданий у радіанах або градусах, matplotlib використовує градуси.

Чотири графіки в полярних координатах:

```
importmatplotlib.pyplot as plt
importnumpy as np
theta = np.arange(0., 2., 1./180.) * np.pi
#plt.title('Поляна система координат')
plt.polar(3*theta, theta/5, label="спіраль");
plt.polar(theta, 0+np.cos(4*theta), label="квітка");
plt.polar(theta, [1.4] * len (theta), label = "коло");
plt.polar(theta, 0*theta, label="0-й радіус");
plt.title("Полярна система координат")
lt.grid(True)
plt.legend(loc='lower left')
візуалізуємо графіки
plt.show()
```



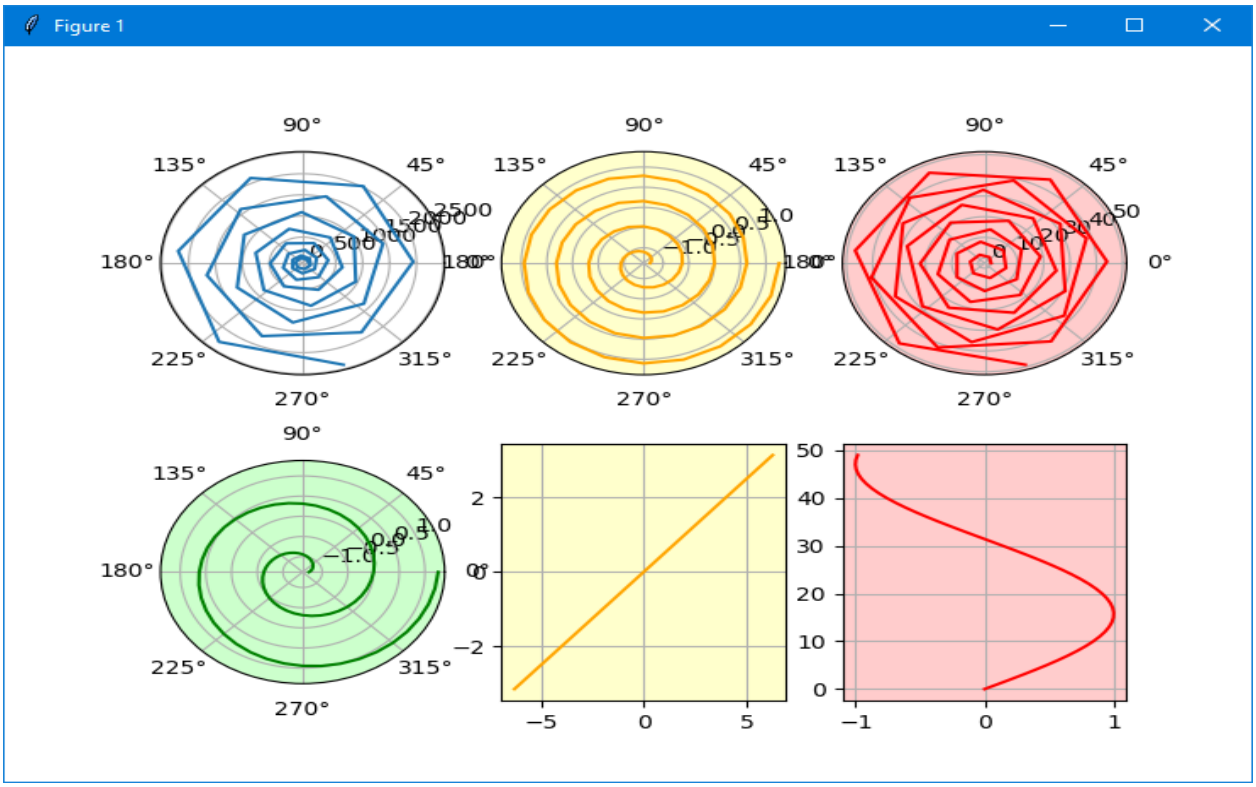
Для побудови в полярних координатах використовується функція polar(). У аргументах першими йдуть кути, потім радіуси. У прикладі використовується той самий масив для кутів і радіус-векторів і малюється чотири різні криві.

Перша утворює спіраль, оскільки з кожним новим кутом змінюється і радіус, друга малює квітку відповідно до рівняння троянди, третя і четверта - кола через незмінність радіусу для всіх кутів (в даному випадку він дорівнює 1.4 і 0).

Ще один приклад:

```
importmatplotlib.pyplot as plt
importnumpy as np

fig = plt.figure (figsize = (8., 6.))
x = np.arange(50)
y = x**2 + 2.5

lag = 0.05 * np.pi
r = np.arange(0.0, 5.0, 0.1)
# phi = r * np.pi
phi = np.arange (-2 * np.pi, 2 * np.pi + lag, lag)
r=phi*0.2
# x та y
ax1 = fig.add_subplot(231, projection='polar')
ax1.plot(x, y)
ax1.grid(True)

ax2 = fig.add_subplot(232, projection='polar',
facecolor ='#FFFFCC')
```

```
ax2.plot(phi*2., r, 'orange') # facecolor='#FFFFCC') #
axisbg
ax2.grid(True)

ax3 = fig.add_subplot(233, projection='polar',
facecolor='#FFCCCC')
ax3.plot(x, x, 'r')
ax3.grid(True)

ax4 = fig.add_subplot(234, projection='polar',
facecolor='#CCFFCC')
ax4.plot(phi, 0.2*phi, 'g') #(-1.5*phi, r, 'g')
ax4.grid(True)

ax5 = fig.add_subplot(235, facecolor='#FFFFCC')
ax5.plot(phi, phi*0.5, 'orange')
ax5.grid(True)

ax6 = fig.add_subplot(236, facecolor='#FFCCCC')
ax6.plot(np.sin(0.1*x), x, 'r')
ax6.grid(True)
###plt.legend(loc='lower left')
plt.show()
```

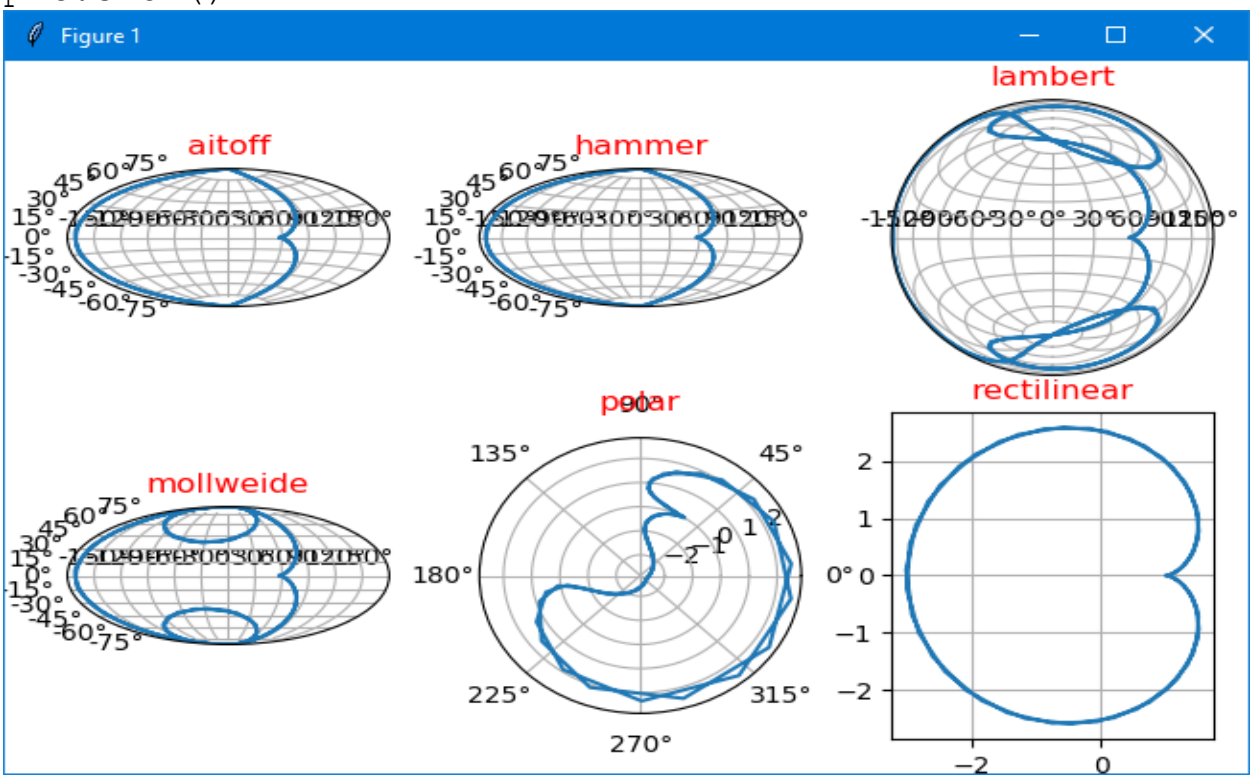


matplotlib дозволяє малювати у полярної системі координат, а й у інших. За це відповідає параметр projection, який у разі рівності значенню 'polar' аналогічний значенню параметра polar=True.

У matplotlib підтримуються такі проекції:
'aitoff', 'hammer', 'lambert', 'mollweide', 'polar', 'rectilinear'.

Найчастіше використовуються два останні варіанти, решта є екзотичними для рутинних завдань у науковій графіці.

```
importmatplotlib.pyplot as plt
importnumpy as np
a = 1.
x = np.arange(-2*np.pi, 2*np.pi, 0.2)
y = np.sin(x) * np.cos(x)
# Рівняння кардіойди
xz = a * (2 * np.cos (x) - np.cos (2 * x))
yz = a*(2*np.sin(x) - np.sin(2*x))
label = ['aitoff', 'hammer', 'lambert', 'mollweide',
'polar', 'rectilinear']
fig = plt.figure(figsize=(10,8))

for i in range(len(label)):
    ax = fig.add_subplot(231+i, projection=label[i])
    ax.plot(xz, yz)
    ax.set_title(label[i], color='r')
    ax.grid(True)
plt.tight_layout()
plt.show()
```



Полярна система координат має спеціальні функції, які дозволяють налаштовувати зовнішній вигляд малюнка.

Для радіусу R:

1. `ax.set_rlabel_position(phi)`- переміщає вісь ординат (радіус) по колу на кут phi (у ГРАДУСАХ) від положення нуля;
2. `ax.set_rmax(R)`- дозволяє обмежити область зміни радіуса R малюнку;
3. `ax.set_rmin(R)`- дозволяє обмежити область зміни радіуса R малюнку;
4. `ax.set_rlim()`- дозволяє обмежити область зміни радіуса R малюнку;
5. `ax.set_rscale()`- дозволяє зробити шкалу радіусів логарифмічної;
6. `ax.set_rgrid()`- дозволяє налаштувати для осі радіусу допоміжну сітку, положення поділів, формат підписів до них тощо.

Для кута phi (в matplotlib він називається theta):

1. `ax.set_theta_zero_location(loc)`- Переміщає положення нуля на певне положення. Loc приймає значення 'N', 'NW', 'W', 'SW', 'S', 'SE', 'E' або 'NE'. За умовчанням положення нуля знаходиться у положенні "схід" або "3 години";
2. `ax.set_theta_offset(phi)`- Переміщає положення нуля на кут phi (у радіанах). За умовчанням положення нуля знаходиться у положенні "схід" або "3 години";
3. `ax.set_theta_direction(loc)`- Визначає напрямок обходу. Loc може бути -1 або за годинниковою стрілкою і 1 або проти годинникової стрілки;
4. `ax.set_thetagrids()`- дозволяє налаштувати для осі кутів допоміжну сітку, положення поділів, формат підписів до них тощо.

П11. Графікрозсіювання

Такий тип графіків дозволяє зображати одночасно дві множини даних, які не утворюють кривої, а саме двовимірне безліч точок. Кожна точка має дві координати. Графік розсіювання часто використовується визначення зв'язку між двома величинами і дозволяє визначити більш точні межі вимірів.

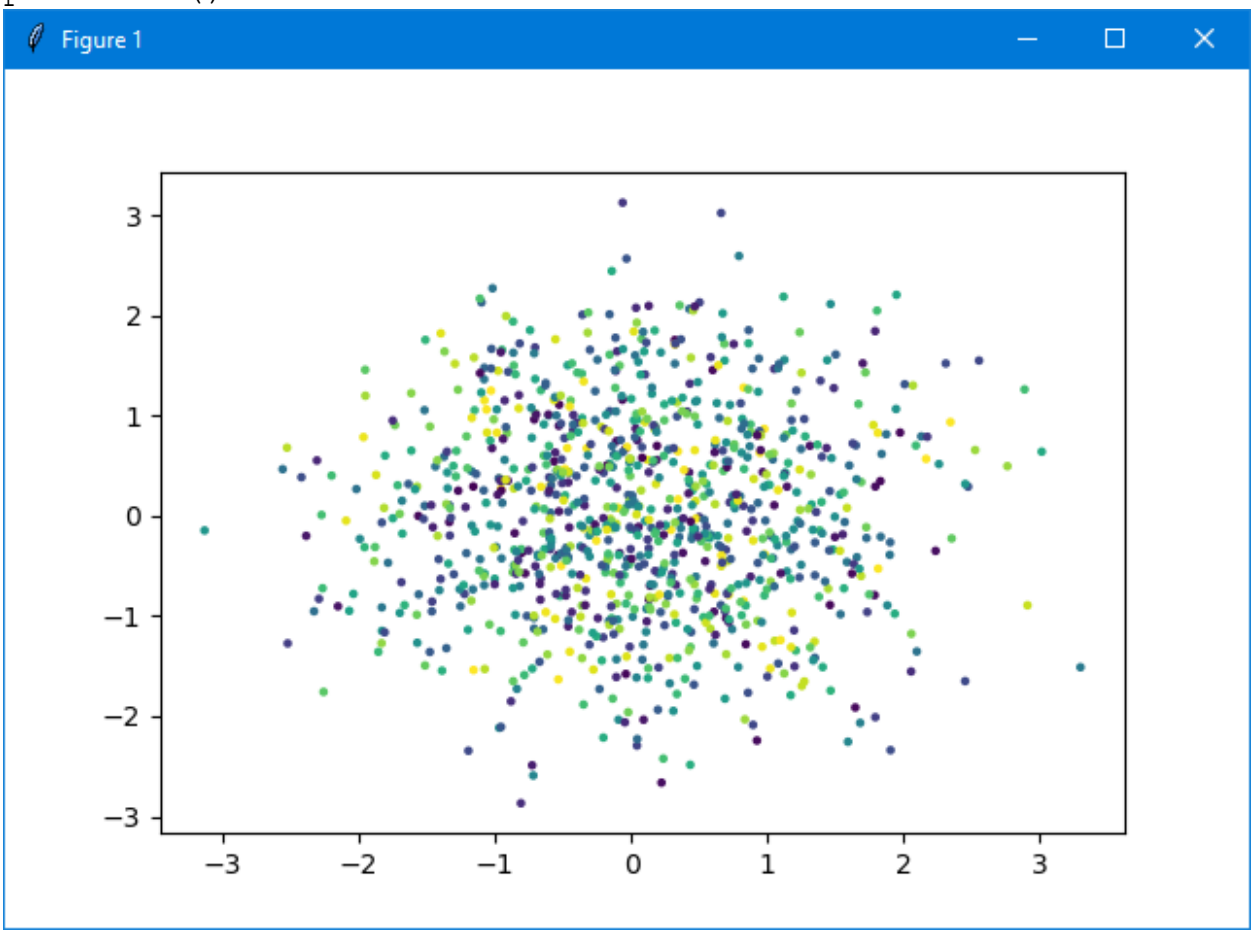
У модулі matplotlib.pyplot є своя функція, для графіка розсіювання (scatter plot) це функція scatter().

Вона приймає дві послідовності та зображує їх на площині у вигляді маркерів, за умовчанням вони круглі та сині. Але природно, з ними можна попрацювати за допомогою keywords тієї ж функції:

- **s**задає розмір маркерів і може бути одним числом для всіх, так і представляти масив значень
- **c**задає колір маркерів, або один для всіх, або безліч
- **marker**визначає тип маркеру.

```
importmatplotlib.pyplot as plt
importnumpy as np
```

```
x = np.random.randn(1000)
y = np.random.randn(1000)

size = 5
colors = np.random.rand(1000)

plt.scatter(x, y, s=size, c=colors)
plt.show()
```



П12. Налаштування в стилі LATEX

Ось приклад налаштування майже всього, що можна налаштувати.

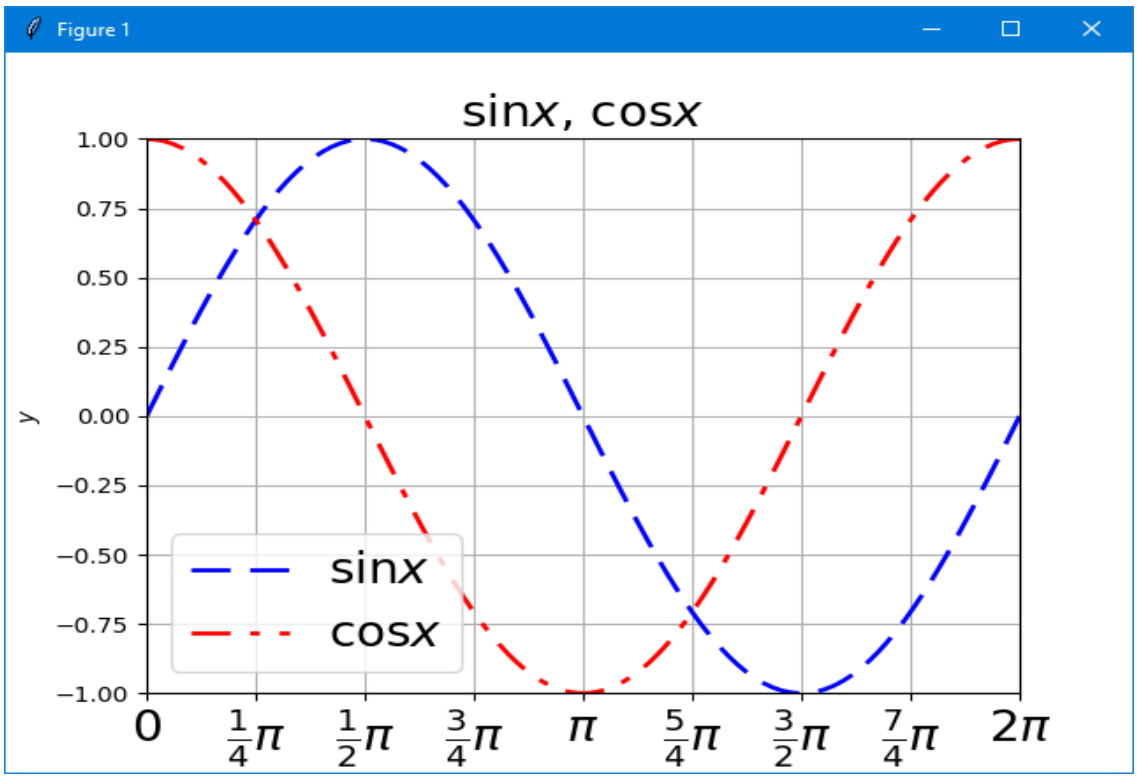
Можна задати послідовність засічок на осі x (і y) та підписи до них (у них, як і в інших текстах, можна використовувати LATEX-івські позначення).

Задати підписи осей x та y та заголовок графіка.

У всіх текстових елементах можна встановити розмір шрифту. Можна задати товщину ліній і штрихи (так, на графіці косинуса малюється штрих довжини 8, потім ділянка довжини 4 не малюється, потім ділянка довжини 2 малюється, потім ділянка довжини 4 знову не малюється, і так за циклом; оскільки товщина лінії дорівнює 2, ці короткі штрихи довжини

Можна поставити підписи до кривих (legend); де розмістити ці підписи також можна регулювати

```
importmatplotlib.pyplot as plt
importnumpy as np
plt.axis([0,2*np.pi,-1,1])

plt.xticks(np.linspace(0,2*np.pi,9),
       ('0',r'$\frac{1}{4}\pi$',r'$\frac{1}{2}\pi$',

r'$\frac{3}{4}\pi$',r'$\pi$',r'$\frac{5}{4}\pi$',

r'$\frac{3}{2}\pi$',r'$\frac{7}{4}\pi$',r'$2\pi$'),
      fontsize=20)

plt.xlabel(r'$x$')
plt.ylabel(r'$y$')
plt.title(r'$\sin x$, $\cos x$',fontsize=20)

x=np.linspace(0,2*np.pi,100)
plt.plot(x,np.sin(x),linewidth=2,
color='b',dashes=[8,4], label=r'$\sin x$')

plt.plot(x,np.cos(x),linewidth=2,
color='r',dashes=[8,4,2,4], label=r'$\cos x$')

plt.legend(fontsize=20)
plt.grid(True)
візуалізуємо графіки
plt.show()
```

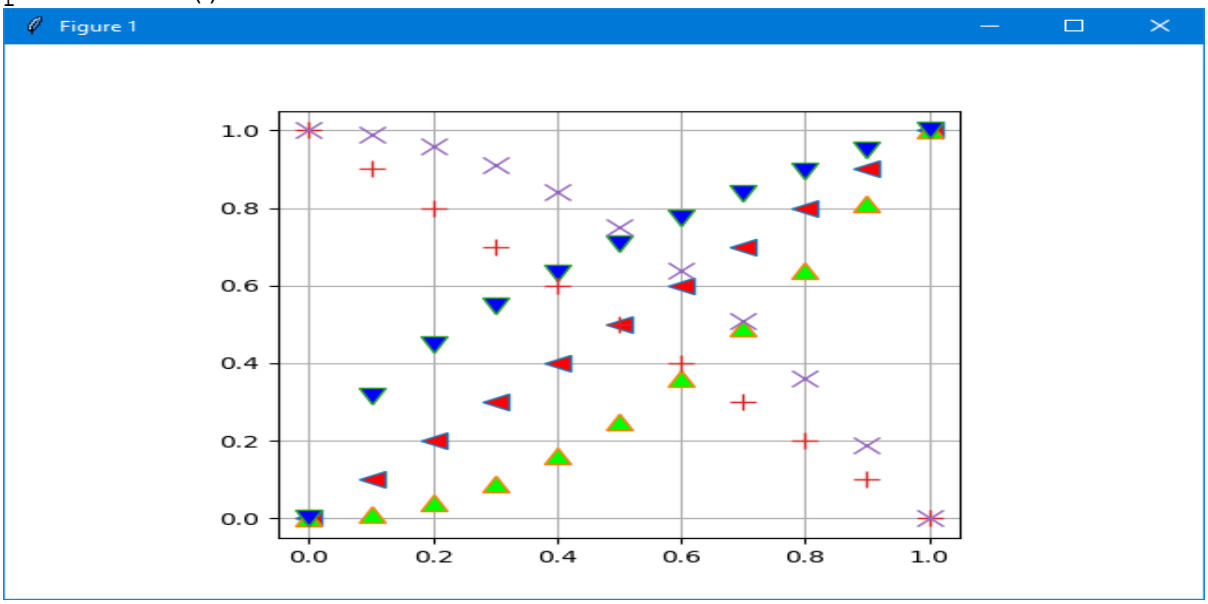


П13. Модифіковані маркери

Якщо linestyle='' то точки не з'єднуються лініями. Самі крапки малюються маркерами різних типів. Тип визначається рядком з одного символу, який чимось нагадує потрібний маркер. На додаток до стандартних маркерів, можна визначити саморобні.

```
importmatplotlib.pyplot as plt
importnumpy as np
x=np.linspace(0,1,11)
# Встановимо параметр - aspect (ratio)
axx=plt.subplot(111)
axx.set_aspect(1)
plt.axis([-0.05,1.05,-0.05,1.05])
plt.plot(x,x,linestyle='',marker='<',markersize=10,
        markerfacecolor='#FF0000')
plt.plot(x,x**2,linestyle='',marker='^',markersize=10,
        markerfacecolor='#00FF00')
plt.plot(x,x**(1/2),linestyle='',marker='v',markersize=
10,
        markerfacecolor='#0000FF')
plt.plot(x,1-x,linestyle='',marker='+',markersize=10,
        markerfacecolor='#0F0F00')
plt.plot(x,1-
x**2,linestyle='',marker='x',markersize=10,
     markerfacecolor='#0F000F')
plt.grid(True)
```

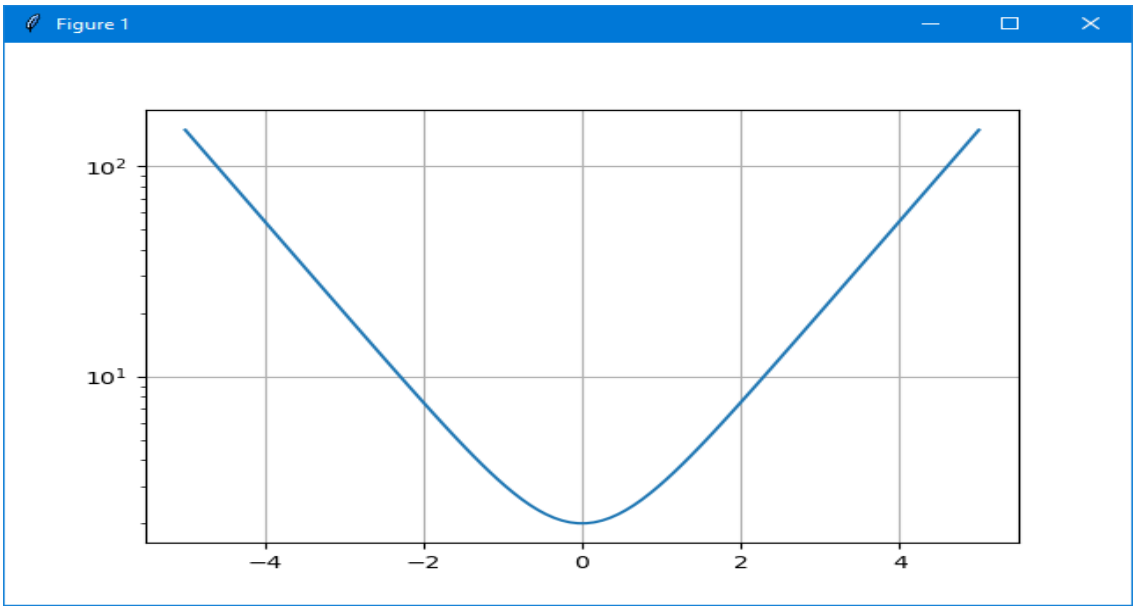
```
візуалізуємо графіки
plt.show()
```



П14. Логарифмічний масштаб

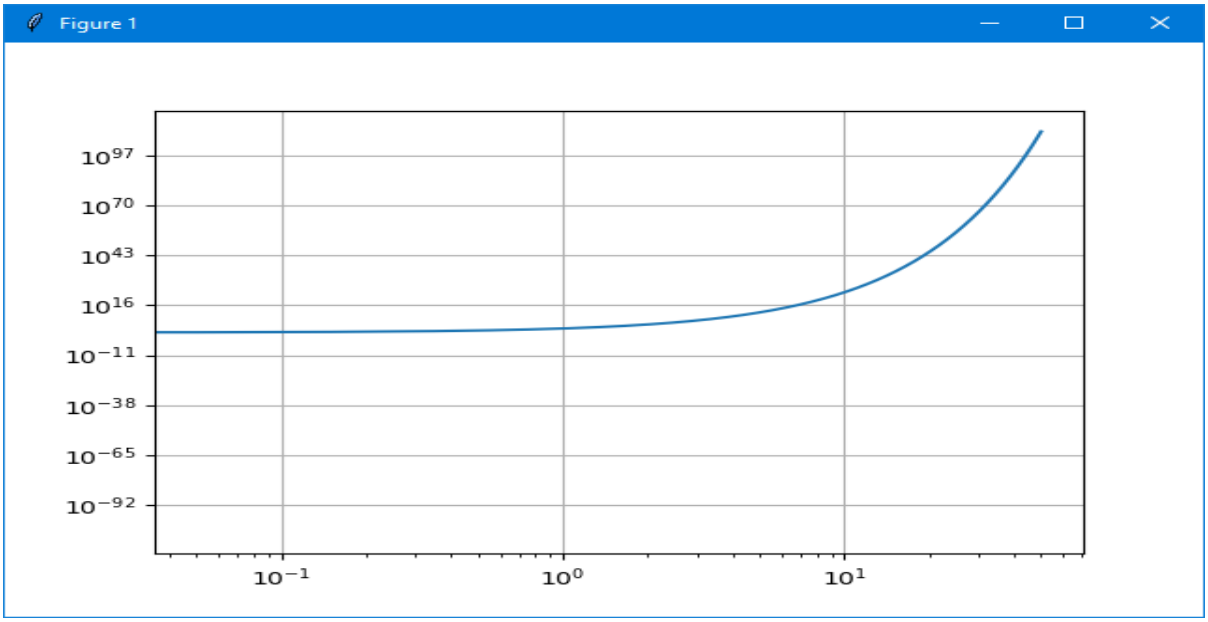
Якщо y змінюється на багато порядків, то зручно використовувати логарифмічний масштаб по y

```
importmatplotlib.pyplot as plt
importnumpy as np
x=np.linspace(-5,5,100)
plt.yscale('log')
plt.plot(x,np.exp(x)+np.exp(-x))
plt.grid(True)
візуалізуємо графіки
plt.show()
```



Можна встановити логарифмічний масштаб по обох осях.

```
importmatplotlib.pyplot as plt
importnumpy as np
x=np.linspace(-50,50,1000)
plt.xscale('log')
plt.yscale('log')
plt.plot(x,np.exp(5*x+3))
plt.grid(True)
візуалізуємо графіки
plt.show()
```



П13. Екпериментальні дані

Допустимо, є теоретична крива (резонанс без фону).

```
xt=np.linspace(-4,4,101)
yt=1/(xt**2+1)
```

Оскільки реальних експериментальних даних немає, ми їх згенеруємо. Нехай вони узгоджуються з теорією і всі статистичні помилки дорівнюють 0.1.

```
xe=np.linspace(-3,3,21)
yerr=0.1*np.ones(21)
ye=1/(xe**2+1)+yerr*np.random.normal(size=21)
```

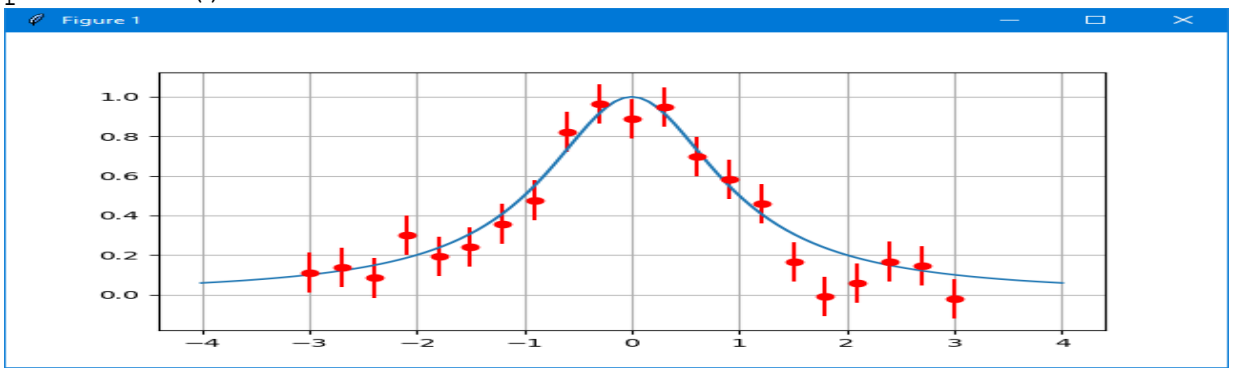
Експериментальні точки з вусами і теоретична крива однією графіку.

```
plt.plot(xt,yt)
plt.errorbar(xe,ye,fmt='ro',yerr=yerr)
```

Весь скрипт:

```
importmath
importnumpy as np
importmatplotlib.pyplot as plt
```

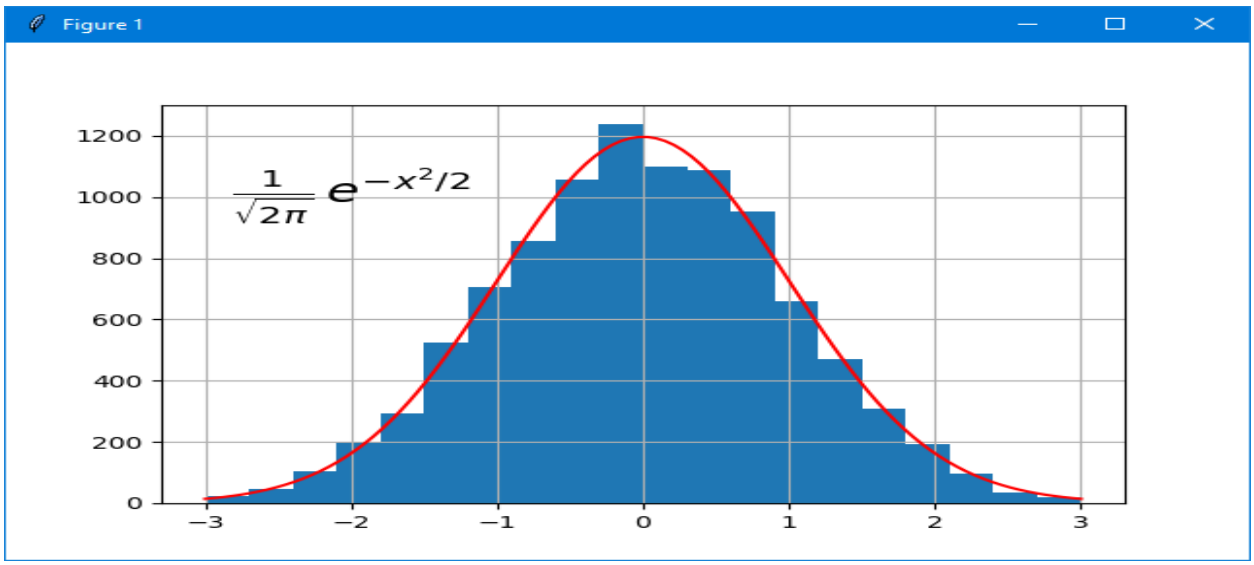
```
xt=np.linspace(-4,4,101)
yt=1/(xt**2+1)
xe=np.linspace(-3,3,21)
yerr=0.1*np.ones(21)
ye=1/(xe**2+1)+yerr*np.random.normal(size=21)
plt.plot(xt,yt)
plt.errorbar(xe,ye,fmt='ro',yerr=yerr)
plt.grid(True)
візуалізуємо графіки
plt.show()
```



П15. Гістограма

Згенеруємо N випадкових чисел з нормальним (гаусовим) розподілом (середнє 0, середньоквадратичне відхилення 1), і розкидаємо їх по 20 бін від -3 до 3 (точки за межами цього інтервалу відкидаються). Для порівняння, разом із гістограмою намалюємо Гауссову криву у тому самому масштабі. І навіть напишемо формулу Гауса.

```
importnumpy as np
importmatplotlib.pyplot as plt
importmatplotlib.mlab as ml
N=10000
r=np.random.normal(size=N)
n, bins, patches = plt.hist (r, range = (-3,3), bins =
20)
x=np.linspace(-3,3,100)
plt.plot(x,N/np.sqrt(2*np.pi)*0.3* np.exp(-
0.5*x**2),'r')

plt.text(-2,1000,r'$\frac{1}{\sqrt{2\pi}} \,e^{-
x^2/2}$',

     fontsize=20,horizontalalignment='центр',
     verticalalignment='center')
plt.grid(True)
візуалізуємо графіки
plt.show()
```

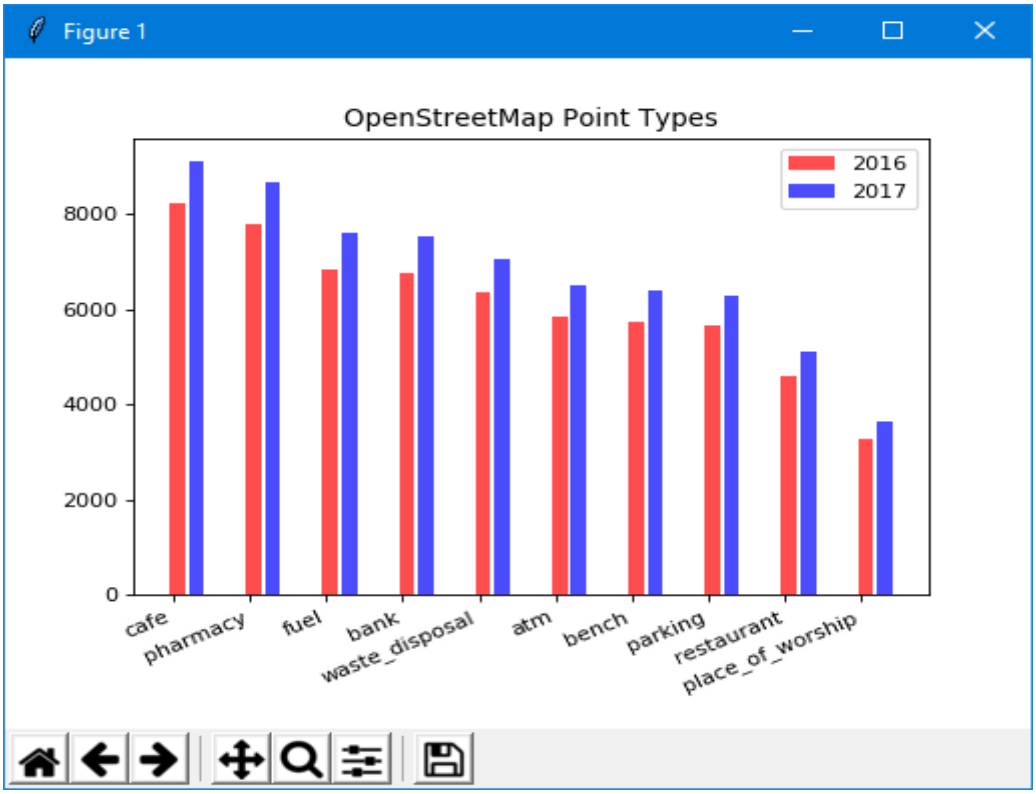


Розглянемо кілька прикладів (https://eax.me/python-matplotlib/) побудови спеціальних діаграм.

Як приклад побудуємо діаграму, що відображає, скільки точок на карті якого типу (заправка, кафе тощо) ставляться. Щоб було трохи цікавіше, вдамо, що торік точок кожного виду було на 10% менше, і спробуємо відобразити цю зміну:

```
importmatplotlib as mpl
importmatplotlib.pyplot as plt
importmatplotlib.dates як mdates
importdatetime as dt
importcsv
data_names = ['cafe', 'pharmacy', 'fuel', 'bank',
'waste_disposal',
              'atm', 'bench', 'parking', 'restaurant',
              'place_of_worship']
data_values = [9124, 8652, 7592, 7515, 7041, 6487,
6374, 6277,
               5092, 3629]
dpi = 80
fig = plt.figure(dpi = dpi, figsize = (512 / dpi, 384 /
dpi) )
mpl.rcParams.update({'font.size': 10})
plt.title('OpenStreetMap Point Types')
#ax = plt.axes()
#ax.yaxis.grid(True, zorder = 1)
xs = range(len(data_names))
plt.bar([x + 0.05 for x in xs], [d * 0.9 for d in
                                        data_values],
        width = 0.2, color = 'red', alpha = 0.7, label
= '2016',
        zorder = 2)
```

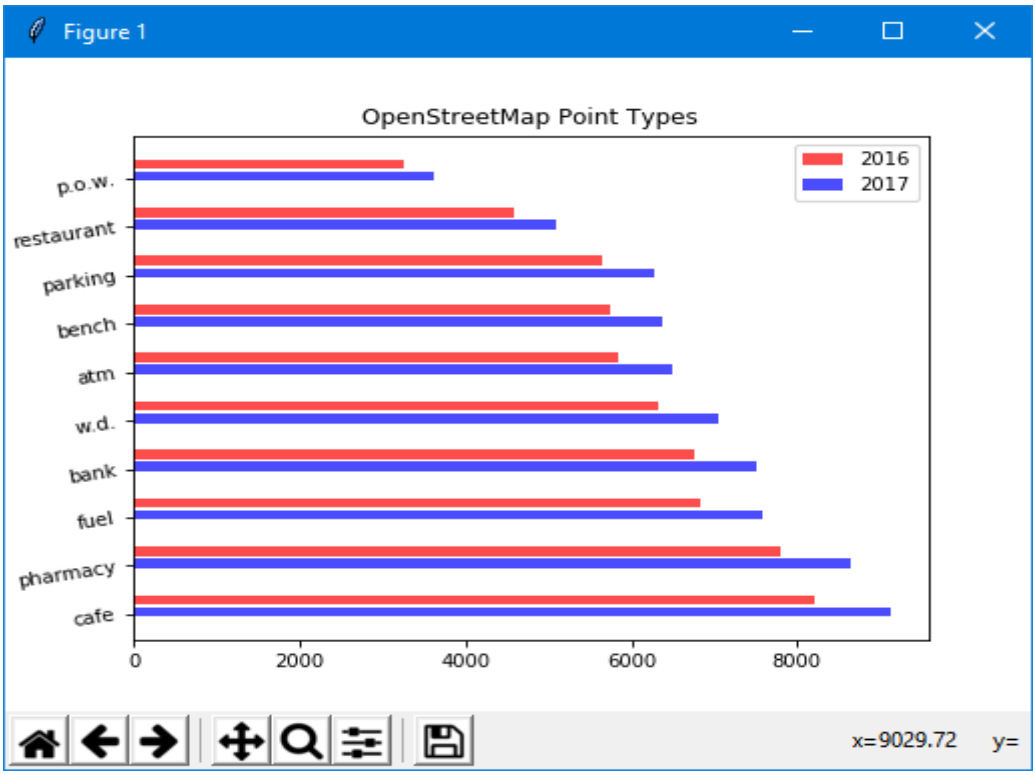
```
plt.bar([x + 0.3 for x in xs], data_values,
        width = 0.2, color = 'blue', alpha = 0.7, label
= '2017',
        zorder = 2)
plt.xticks(xs, data_names)
fig.autofmt_xdate(rotation = 25)
plt.legend(loc='upper right')
#fig.savefig('bars.png')
plt.show()
```



Ті ж дані можна відобразити, розташувавши стовпчики горизонтально:

```
importmatplotlib as mpl
importmatplotlib.pyplot as plt
importmatplotlib.dates як mdates
importdatetime as dt
importcsv

data_names = ['cafe', 'farmaceu', 'fuel', 'bank', 'wd',
'atm', 'bench', 'parking', 'restaurant', 'pow']
data_values = [9124, 8652, 7592, 7515, 7041, 6487,
6374, 6277, 5092, 3629]

dpi = 80
fig = plt.figure(dpi = dpi, figsize =
                            (512 / dpi, 384 / dpi))
mpl.rcParams.update({'font.size': 9})
```

```
plt.title('OpenStreetMap Point Types')
#ax = plt.axes()
#ax.xaxis.grid(True, zorder = 1)
xs = range(len(data_names))
plt.barh([x + 0.3 for x in xs], [ d * 0.9 for d in
data_values],
         height = 0.2, color = 'red', alpha = 0.7,
label = '2016',
         zorder = 2)
plt.barh([x + 0.05 for x in xs], data_values,
         height = 0.2, color = 'blue', alpha = 0.7,
label = '2017',
         zorder = 2)
plt.yticks(xs, data_names, rotation = 10)

plt.legend(loc='upper right')
plt.show()
```



П16. Кругова діаграма

Наприклад візуалізуємо розподіл кафе по різних містах (приклад взятий з https://eax.me/python-matplotlib/):

```
importmatplotlib as mpl
importmatplotlib.pyplot as plt
importmatplotlib.dates як mdates
importdatetime as dt
```

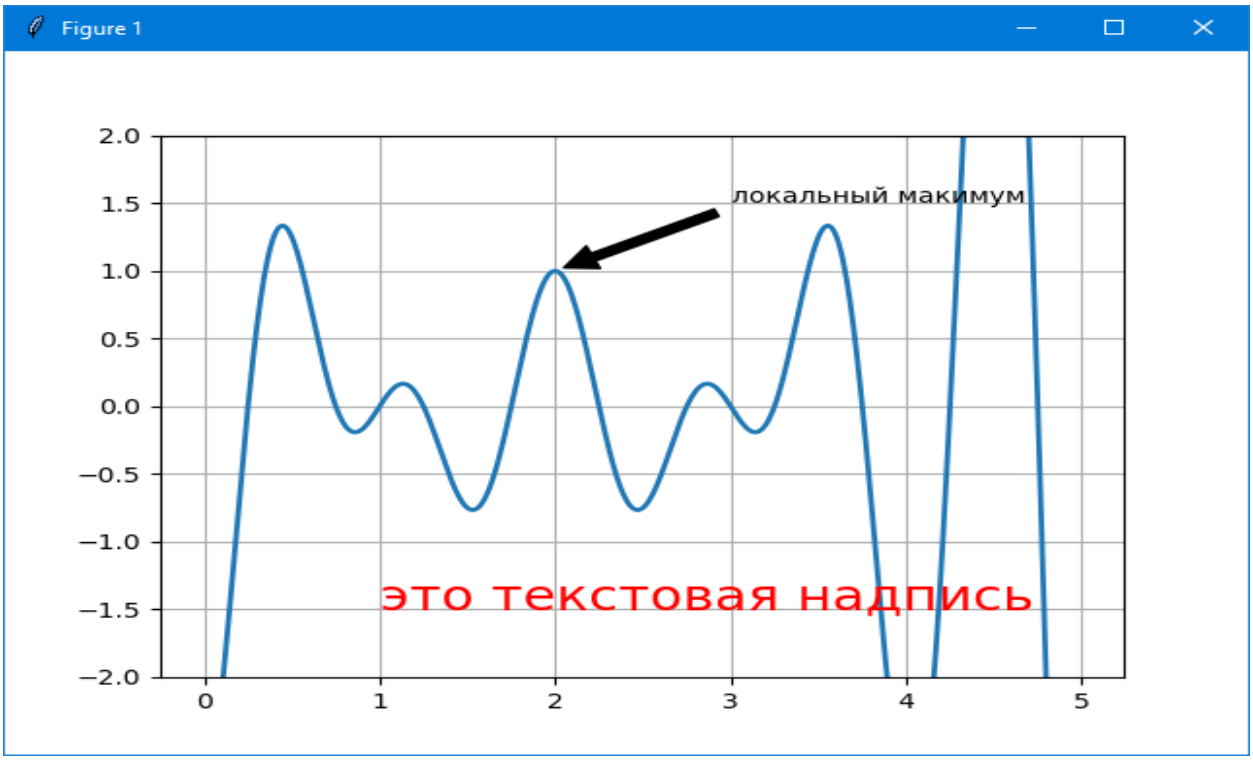
```
importcsv

data_names = ['Москва', 'Санкт-Петербург', 'Сочі',
'Архангельськ',
              'Володимир', 'Краснодар', 'Курськ',
'Вороніж',
              'Ставрополь', 'Мурманськ']
data_values = [1076, 979, 222, 189, 137, 134, 124, 124,
91, 79]

dpi = 80
fig = plt.figure(dpi = dpi, figsize = (512 / dpi, 384 /
dpi) )
mpl.rcParams.update({'font.size': 9})

plt.title('Розподіл кафе по містах Росії (%)')

xs = range(len(data_names))

plt.pie(
    data_values, autopct='%.1f', radius = 1.1,
    explode = [0.15] + [0 for _ in
range(len(data_names) - 1)] )
plt.legend(
    bbox_to_anchor = (-0.16, 0.45, 0.25, 0.25),
    loc = 'low left', labels = data_names )
plt.show()
```

Велику кількість прикладів можна знайти на https://matplotlib.org/gallery.html

П17. Текст та написи

Текст та додаткові написи (аннатації) розміщуються на діаграмі:

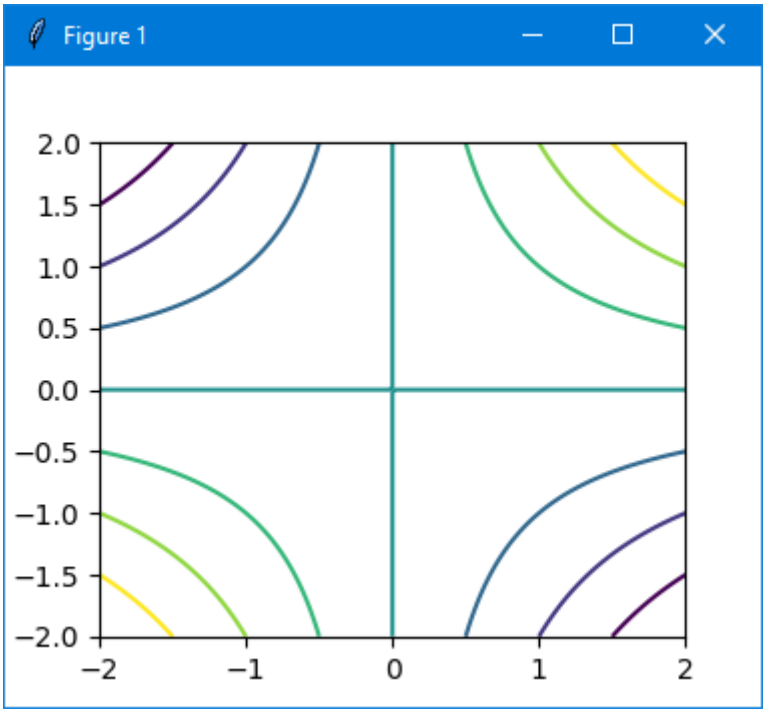
```
importnumpy as np
importmatplotlib as mpl
importmatplotlib.pyplot as plt
ax = plt.subplot(111)

t = np.arange(0.0, 5.0, 0.01)
s = (1-(t-2)*(t-2)) * np.cos(2*np.pi*t)
line, = plt.plot (t, s, lw = 2)
plt.text(1,-1.5,'це текстовий напис', size=20,
color='r')
plt.grid(True)
plt.annotate('локальний макимум', xy=(2, 1), xytext=(3,
1.5), arrowprops= dict(facecolor='black',
shrink=0.05), )
plt.ylim(-2,2)
plt.show()
```



П18. Контурні графіки

Нехай потрібно вивчити поверхню z = xy. Побудуємо її «горизонталі» (взяти з http://www.inp.nsk.su/~grozin/python/python6.html):

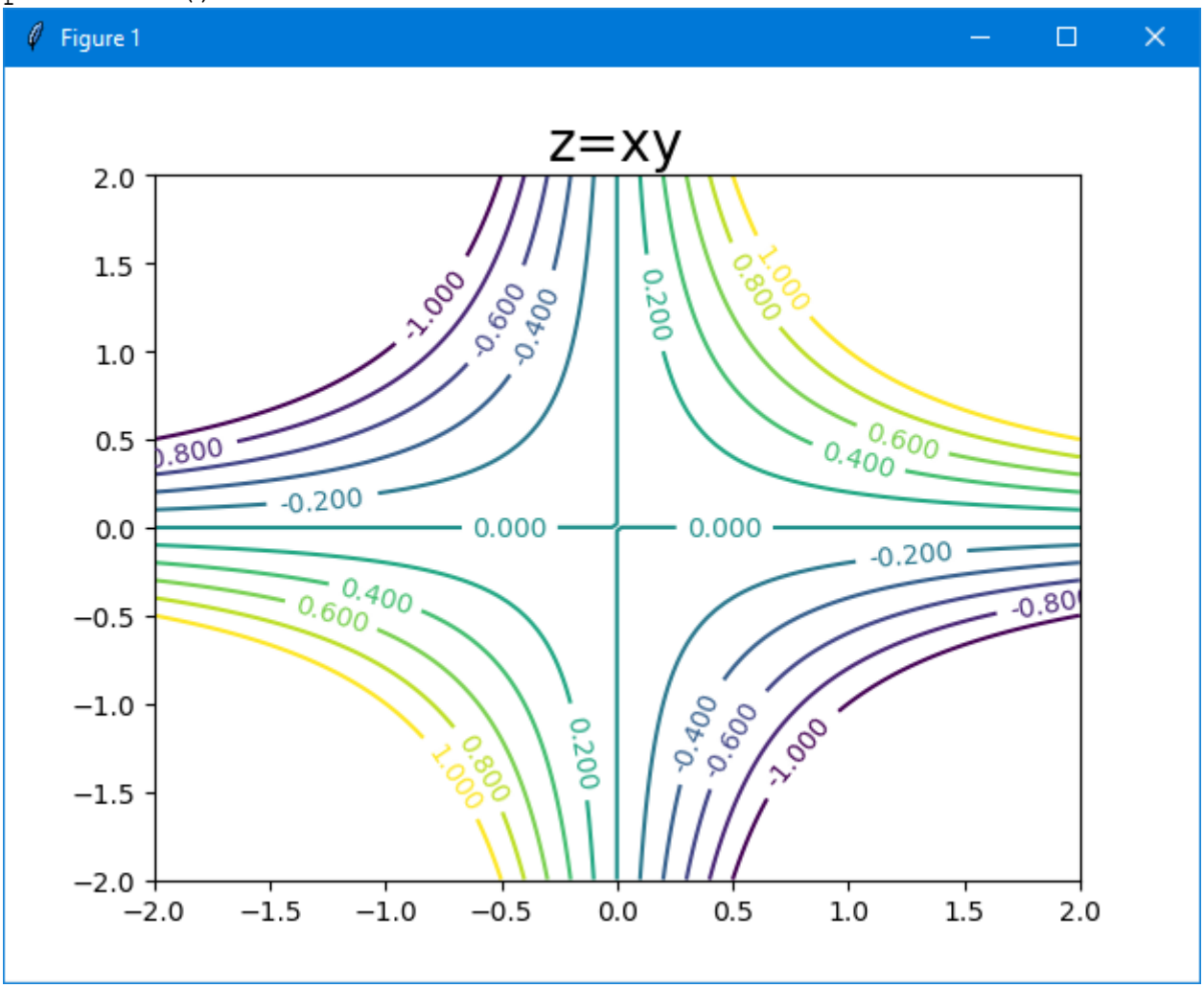
```
importnumpy as np
importmatplotlib as mpl
importmatplotlib.pyplot as plt
x=np.linspace(-1,1,50)
y=x
z = np.outer (x, y)
plt.contour(x,y,z)
plt.axes().set_aspect(1)
plt.show()
```




```
Роль функції np.outer(x,y) пакета numpy очевидна з
прикладу:
>>> x=np.array([1,2,3])
>>> y=np.array([10,20,30])
>>> x*y
array([10, 40, 90])
>>> np.outer(x,y)
array([[10, 20, 30],
       [20, 40, 60],
       [30, 60, 90]])
```

Горизонталі можна розфарбувати і підписати, а також збільшити їх кількість:

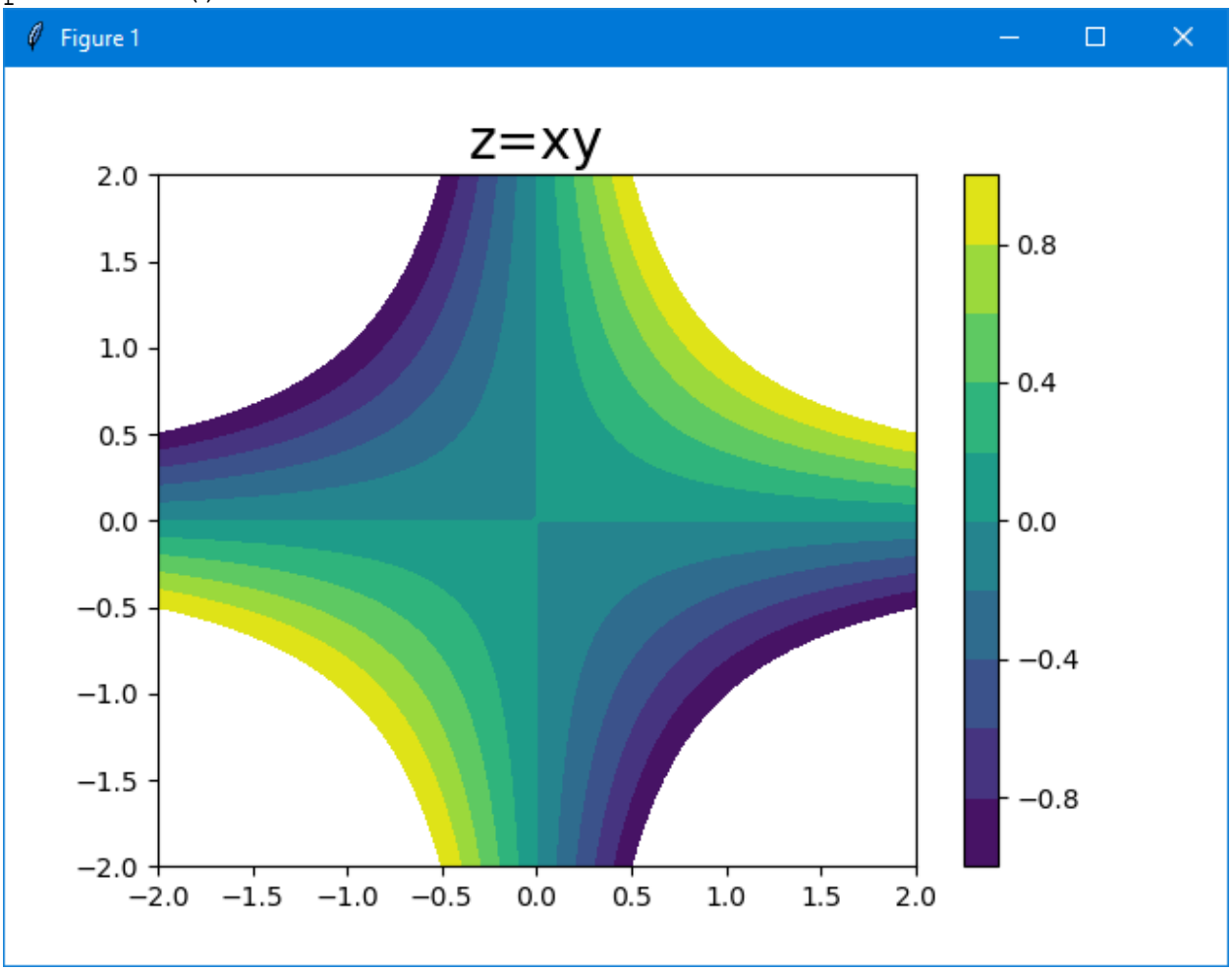
```
importnumpy as np
importmatplotlib as mpl
importmatplotlib.pyplot as plt
x=np.linspace(-2,2,100)
y=x
z = np.outer (x, y)
plt.title('z=xy',fontsize=20)
curves=plt.contour(x,y,z,np.linspace(-1,1,11))
plt.clabel(curves)
plt.show()
```



Висоту (значення Z) можна встановити кольором, як на фізичних географічних картах. colorbar показує відповідність кольорів та значень z.

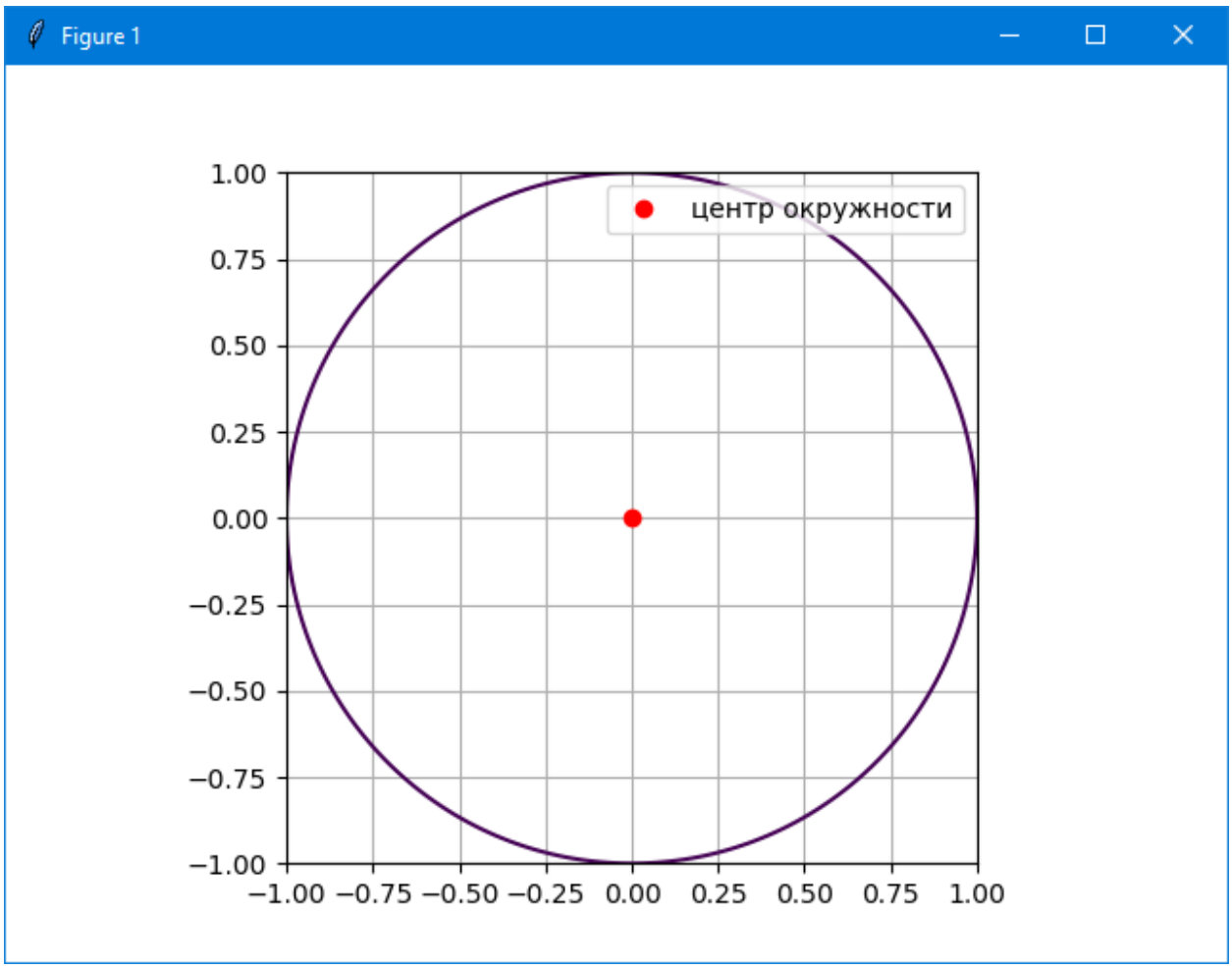
```
importnumpy as np
importmatplotlib as mpl
importmatplotlib.pyplot as plt
x=np.linspace(-2,2,100)
y=x
z = np.outer (x, y)
plt.title('z=xy',fontsize=20)
plt.contourf(x,y,z,np.linspace(-1,1,11))
```

```
plt.colorbar()
#plt.axes().set_aspect(1)
plt.show()
```



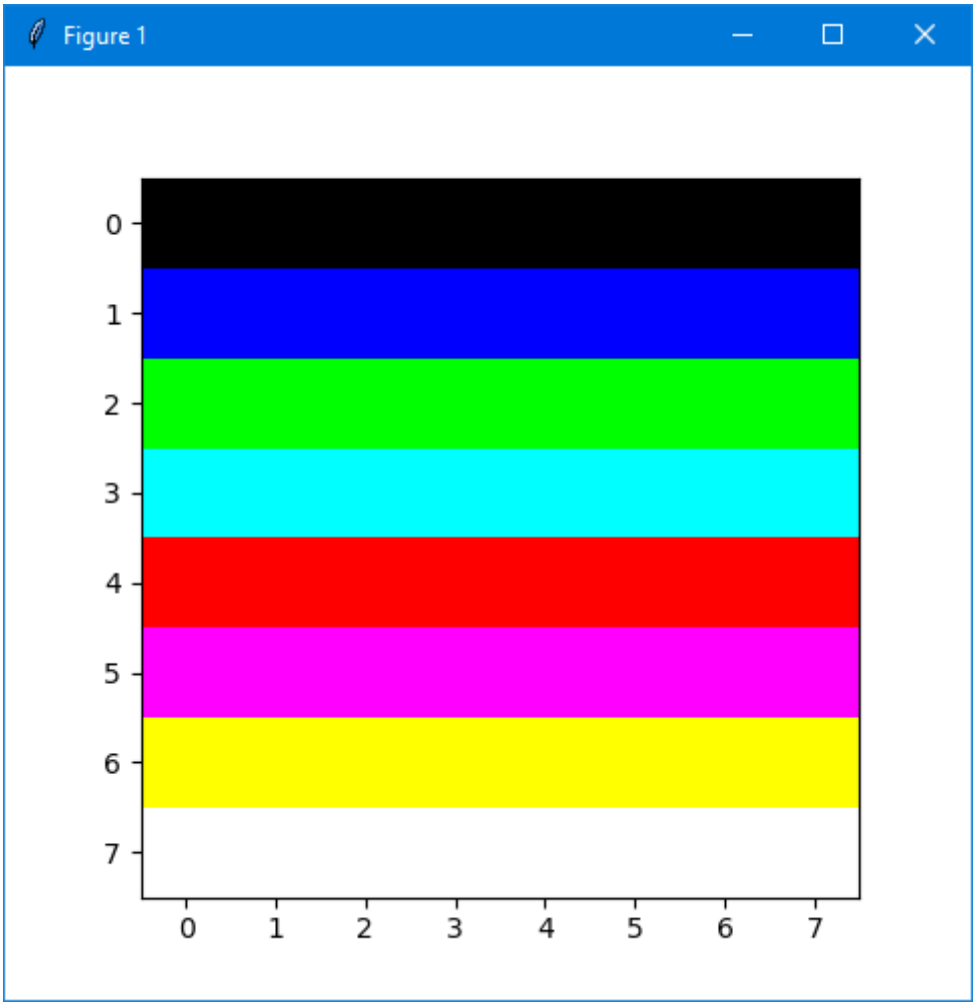
Так можна «малювати» графіки функцій на площині, задані «неявно» - F(x, y)=0:

```
importnumpy as np
importmatplotlib.pyplot as plt

x = np.linspace(-1.0, 1.0, 100)
y = np.linspace(-1.0, 1.0, 100)
X, Y = np.meshgrid(x,y)
F = X**2 + Y**2 - 1  #0.6
plt.contour(X,Y,F,[0])
plt.plot([0],[0],'ro', label="центр кола")
plt.gca().set_aspect('equal') #, щоб малюнок виглядав кругом

# Включаємо сітку
plt.grid(True)
plt.legend(loc='best')
plt.show()
```



П19. Images (піксельні картинки)

Картинка задається масивом z: z[i,j] - це колір пікселя i,j, масив із 3 елементів (rgb, числа від 0 до 1). Для наочності у прикладі формуються «натуральні» кольори до списку col:

```
importnumpy as np
importmatplotlib as mpl
importmatplotlib.pyplot as plt

n=8
u=np.linspace(0,1,n)
x,y=np.meshgrid(u,u)
z=np.zeros((n,n,3))

col=[]
forr in range(0,2):
    forg in range(0,2):
        forb in range(0,2):
            print(r, g, b)
            col.append([r, g, b])
#print (col)
```

```
forі in range(n):
    forj in range(n): # RGB
        z[i,j]=col[i]

plt.imshow(z)
plt.show()
```