Візуалізація даних

Лабораторна робота №3

Засоби побудови рисунків Matplotlib

Існує декілька способів написати код Matplotlib. Код, що приводиться буде заснований на інтерфейсі Pyplot. Малювання в Matplotlib зводиться до до додавання фігури на холст рисунку. У лістингу 1 наведено типовий приклад зображення кола.

Лістинг 1 – приклад малювання

```
import matplotlib.pyplot as plt

plt.axes()

circle = plt.Circle((0, 0), radius=0.75, fc='y')
plt.gca().add_patch(circle)

plt.axis('scaled')
plt.show()
```

Метод gca() повертає поточний екземпляр Axis. Налаштування осі на "'scaled'" забезпечує правильну видимість доданої форми. В результаті отримаємо зображення, що наведено на рис 1.

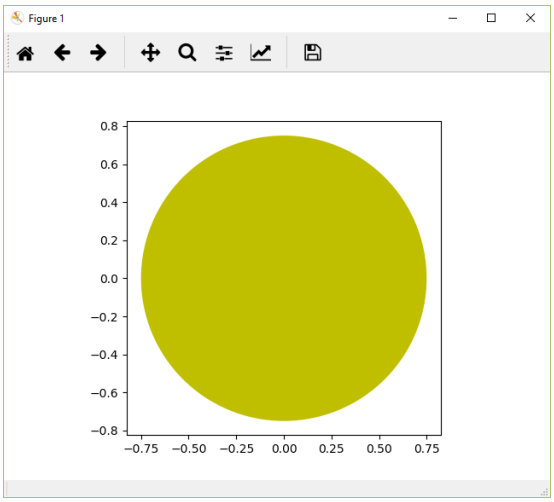

Рисунок 1 – Результат роботи лістингу 1

Більш детальний приклад з малювання наведено у лістингу 2. Результат виконання коду представлено на рисунку 2.

Лістинг 2 – Код для малювання прямокутника

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.path as mpath
import matplotlib.lines as mlines
import matplotlib.patches as mpatches
from matplotlib.collections import PatchCollection

plt.rcdefaults()

def label(xy, text):
    y = xy[1] - 0.15  # shift y-value for label so that it's
below the artist
    plt.text(xy[0], y, text, ha="center", family='sans-serif',
size=14)

fig, ax = plt.subplots()
# сітка 3x3 для малювання об’єктів
grid = np.mgrid[0.2:0.8:3j, 0.2:0.8:3j].reshape(2, -1).T

patches = []

# Коло
circle = mpatches.Circle(grid[0], 0.1, ec="none")
patches.append(circle)
label(grid[0], "Circle")

# прямокутник
rect = mpatches.Rectangle(grid[1] - [0.025, 0.05], 0.05, 0.1,
ec="none")
patches.append(rect)
label(grid[1], "Rectangle")

# сектор круга
wedge = mpatches.Wedge(grid[2], 0.1, 30, 270, ec="none")
patches.append(wedge)
label(grid[2], "Wedge")

# полігон
polygon = mpatches.RegularPolygon(grid[3], 5, 0.1)
patches.append(polygon)
label(grid[3], "Polygon")

# еліпс
ellipse = mpatches.Ellipse(grid[4], 0.2, 0.1)
patches.append(ellipse)
label(grid[4], "Ellipse")

# стрілка
arrow = mpatches.Arrow(grid[5, 0] - 0.05, grid[5, 1] - 0.05,
0.1, 0.1,
                       width=0.1)
patches.append(arrow)
```

```
label(grid[5], "Arrow")

# полілінія
Path = mpath.Path
path_data = [
    (Path.MOVETO, [0.018, -0.11]),
    (Path.CURVE4, [-0.031, -0.051]),
    (Path.CURVE4, [-0.115, 0.073]),
    (Path.CURVE4, [-0.03, 0.073]),
    (Path.LINETO, [-0.011, 0.039]),
    (Path.CURVE4, [0.043, 0.121]),
    (Path.CURVE4, [0.075, -0.005]),
    (Path.CURVE4, [0.035, -0.027]),
    (Path.CLOSEPOLY, [0.018, -0.11])]
codes, verts = zip(*path_data)
path = mpath.Path(verts + grid[6], codes)
patch = mpatches.PathPatch(path)
patches.append(patch)
label(grid[6], "PathPatch")

# прямокутник з округленими краями
fancybox = mpatches.FancyBboxPatch(
    grid[7] - [0.025, 0.05], 0.05, 0.1,
    boxstyle=mpatches.BoxStyle("Round", pad=0.02))
patches.append(fancybox)
label(grid[7], "FancyBboxPatch")

# лінія
x, y = np.array([[-0.06, 0.0, 0.1], [0.05, -0.05, 0.05]])
line = mlines.Line2D(x + grid[8, 0], y + grid[8, 1], lw=5.,
alpha=0.3)
label(grid[8], "Line2D")

colors = np.linspace(0, 1, len(patches))
collection = PatchCollection(patches, cmap=plt.cm.hsv,
alpha=0.3)
collection.set_array(np.array(colors))
ax.add_collection(collection)
ax.add_line(line)

plt.axis('equal')
plt.axis('off')
plt.tight_layout()

plt.show()
```

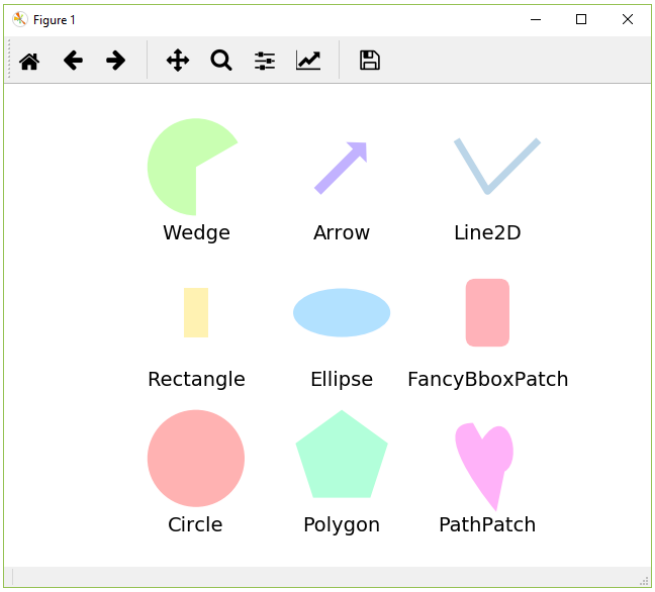

Рисунок 2 – Результат роботи лістингу 2

Анімація

При створенні анімації найбільший інтерес полягає у переміщенні деяких фігур по заданій траєкторії. У лістингу 3 наведений код, що виконує анімацію кола по колу:

Лістинг 3 – Анімація руху фігури

```
import numpy as np
from matplotlib import pyplot as plt
from matplotlib import animation

fig = plt.figure()
fig.set_dpi(100)
fig.set_size_inches(7, 6.5)

ax = plt.axes(xlim=(0, 10), ylim=(0, 10))
patch = plt.Circle((5, -5), 0.75, fc='y')

def init():
    patch.center = (5, 5)
    ax.add_patch(patch)
    return patch,

def animate(i):
    x, y = patch.center
    x = 5 + 3 * np.sin(np.radians(i))
    y = 5 + 3 * np.cos(np.radians(i))
    patch.center = (x, y)
    return patch,

anim = animation.FuncAnimation(fig, animate,
                               init_func=init,
                               frames=360,
                               interval=20,
```

```
                                    blit=True)

plt.show()
```

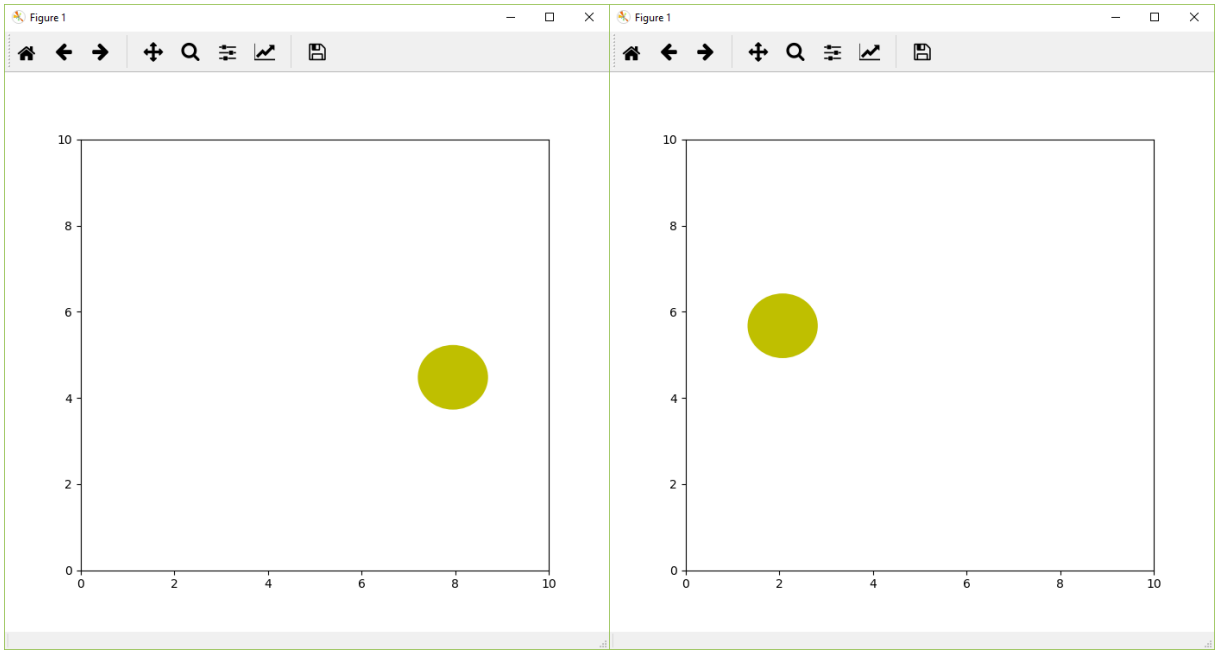

Рисунок 3 – Результати побудови анімації

Щоб зберегти анімацію на диск можна використовувати наступний код

Лістинг 4 – Збереження анімації

```
anim.save('animation.mp4', fps=30,
          extra_args=['-vcodec', 'h264',
                      '-pix_fmt', 'yuv420p'])
```

На жаль у matplotlib не має можливості збереження у форматі gif, тому для перетворення набору картинок у gif-анімацію можна використати ImageMagic:

Лістинг 5 – застосування ImageMagic при створенні анімації

```
ffmpeg -i animation.mp4 -r 10 output%05d.png
convert output*.png output.gif
```

Варіанти завдань

Розробити програму, що будує анімацію згідно варіанту, зберегти її у форматі gif та викласти на власній сторінці у соціальній мережі.

Варіант 1

Сектор круга, що рухається вздовж прямокутника та змінює кут сектора;

Варіант 2

Стрілку, один кінець якої рухається уздовж прямокутника, а другий постійно вказує у центр вікна;

Варіант 3

Полілінію у формі галочки (V), що рухається вздовж рівностороннього трикутника;

Варіант 4

Еліпс, що обертається навколо власного центру;

Варіант 5

Прямокутник, центр якого рухається по еліптичній траєкторії;

Варіант 6

Прямокутник зі округленими краями, що рухається по синусоїді;

Варіант 7

Коло, що рухається по спіралі;

Варіант 8

Полігон у формі п’ятикутника, що рухається по колу;

Варіант 9

Серце, що збільшується/зменшується