

Programowanie obliczeń komputerowych

W110PA-SI0075G

rok akademicki 2025/26
semestr zimowy

Wykład 5

Karol Tarnowski

karol.tarnowski@pwr.edu.pl

L-1 p. 221

Plan

Biblioteka matplotlib

- plot()
- hierarchia obiektów
- scatter()
- opisy osi
- legenda
- formatowanie obiektów graficznych
- wykresy 3d

Biblioteka matplotlib

Biblioteka dostarcza zestaw narzędzi do tworzenia wykresów 2D oraz 3D

Współpraca z biblioteką numpy

Funkcjonalności wzorowane na funkcjonalnościach pakietu Matlab

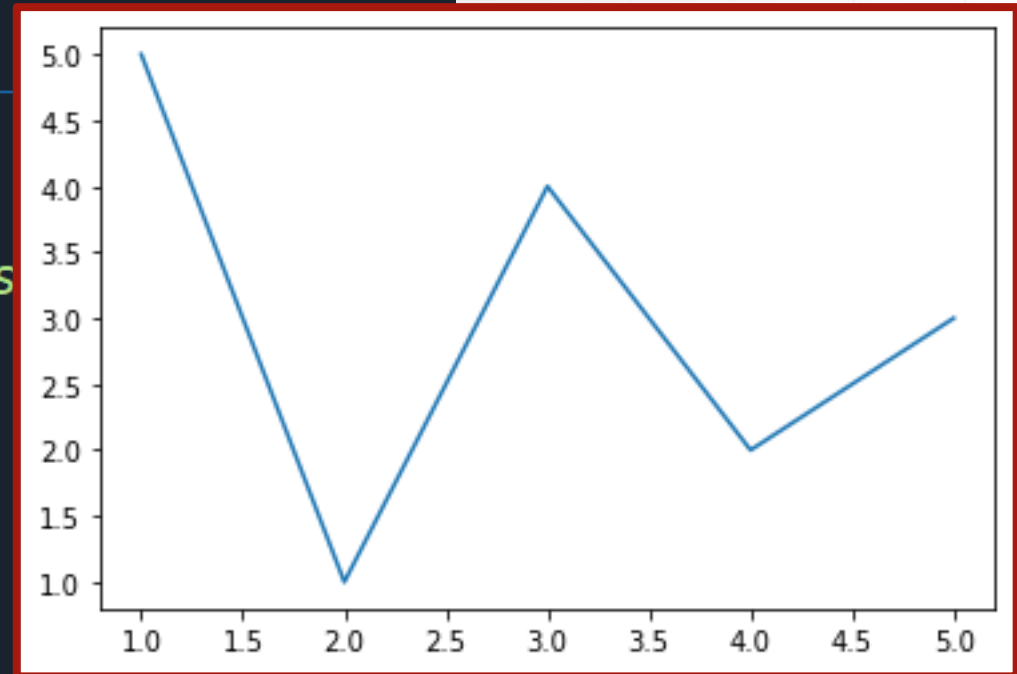
https://matplotlib.org/stable/tutorials/introductory/quick_start.html

Biblioteka matplotlib - podstawy

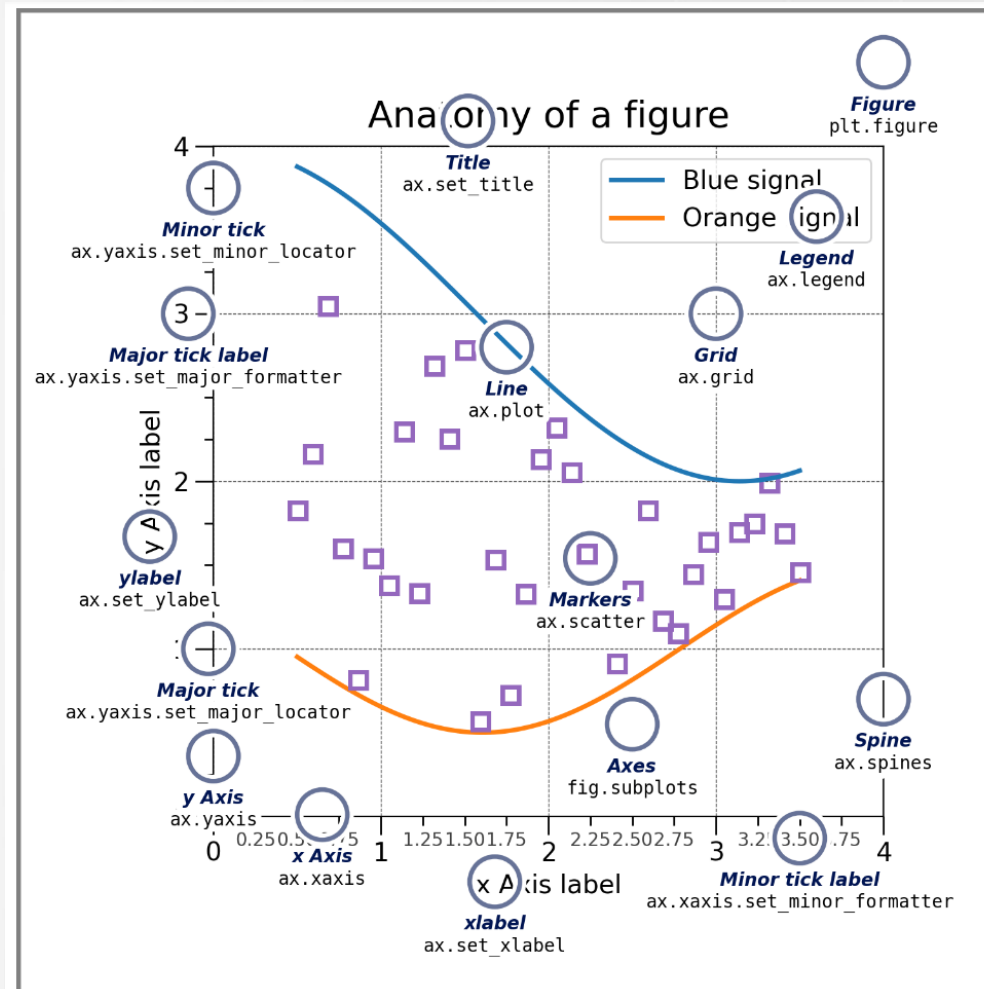
```
matplotlib_01.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący tworzenie prostego wykresu.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  #import numpy as np
9
10 #utworzenie obiektu wykresu (figure)
11 fig, ax = plt.subplots()
12 x = [1, 2, 3, 4, 5]
13 y = [5, 1, 4, 2, 3]
14 ax.plot(x, y)
```

Biblioteka matplotlib - podstawy

```
matplotlib_01.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący tworzenie pros
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  #import numpy as np
9
10 #utworzenie obiektu wykresu (figure)
11 fig, ax = plt.subplots()
12 x = [1, 2, 3, 4, 5]
13 y = [5, 1, 4, 2, 3]
14 ax.plot(x, y)
```



Biblioteka matplotlib - podstawy



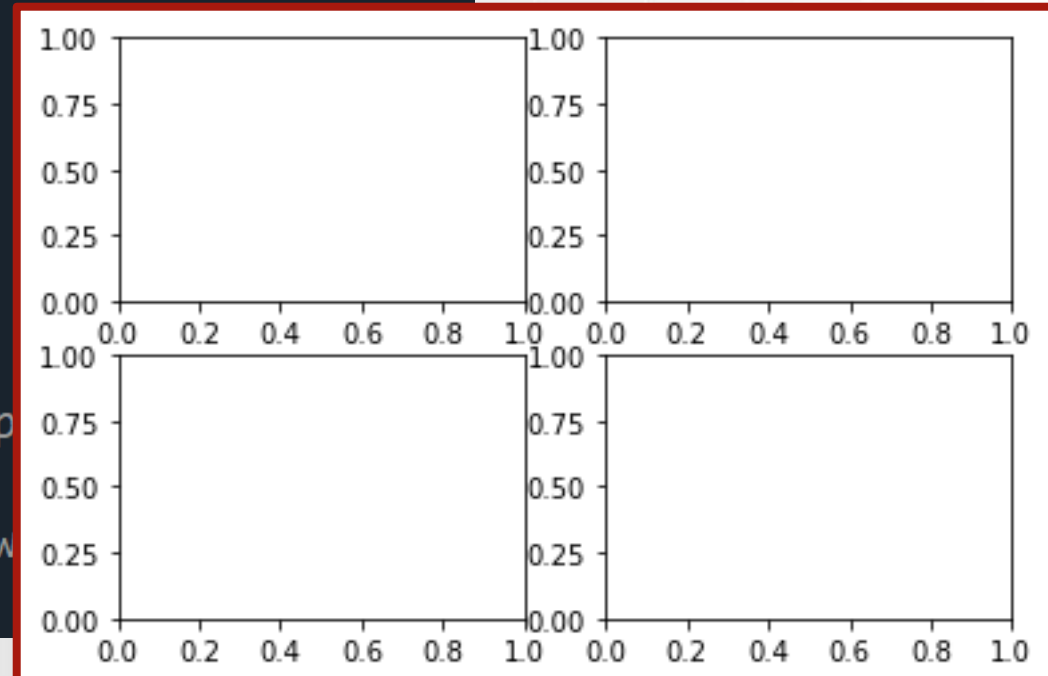
https://matplotlib.org/stable/tutorials/introductory/quick_start.html

Klasa Figure

```
matplotlib_02.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący tworzenie obiektu Figure.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  #import numpy as np
9
10 # utworzenie obiektu wykresu (Figure)
11 # tworzenie pustego wykresu bez osi
12 fig = plt.figure()
13 # tworzenie wykresu z jednym układem współrzędnych
14 fig, ax = plt.subplots()
15 # tworzenie wykresu z siatką 2x2 układów współrzędnych
16 fig, axs = plt.subplots(2,2)
```

Klasa Figure

```
matplotlib_02.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący tworzenie obiektu Figure.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  #import numpy as np
9
10 # utworzenie obiektu wykresu (Figure)
11 # tworzenie pustego wykresu bez osi
12 fig = plt.figure()
13 # tworzenie wykresu z jednym układem wsp
14 fig, ax = plt.subplots()
15 # tworzenie wykresu z siatką 2x2 układów
16 fig, axs = plt.subplots(2,2)
```

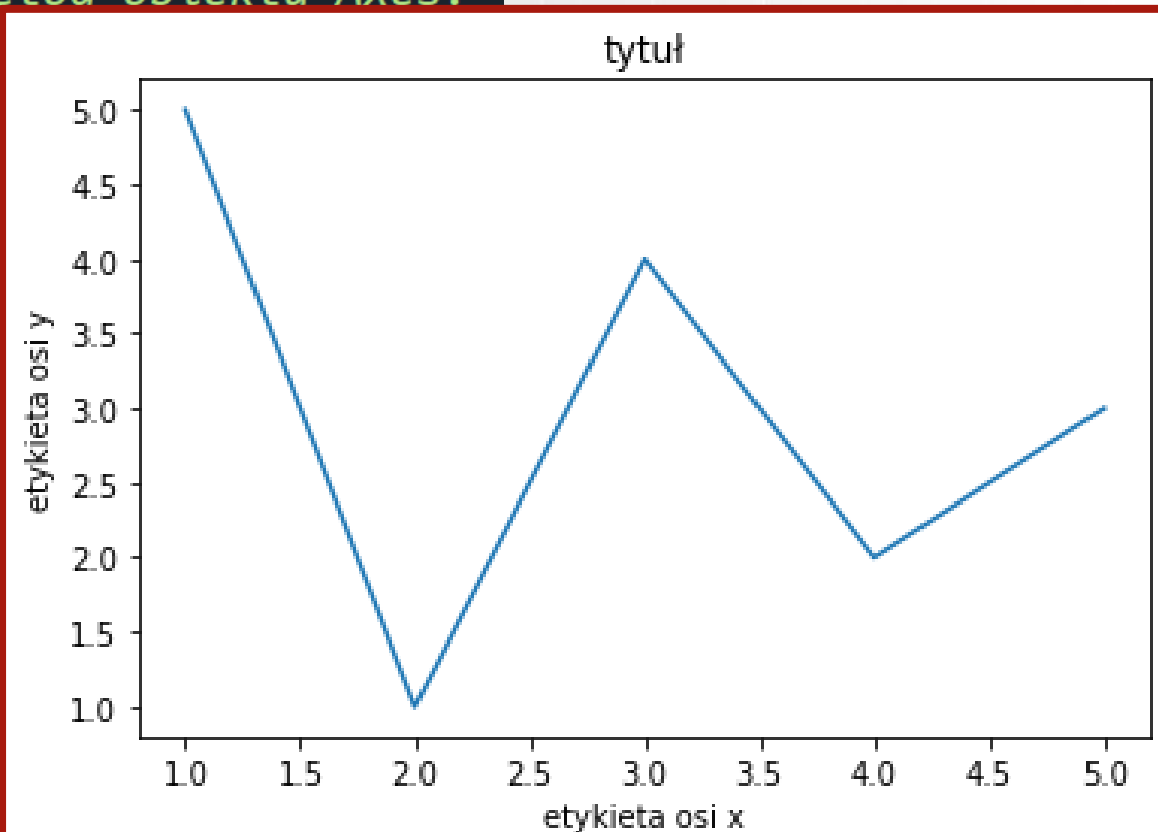


Klasa Axes

```
matplotlib_03.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykorzystanie metod obiektu Axes.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  #import numpy as np
9
10 #utworzenie obiektu wykresu (figure)
11 fig, ax = plt.subplots()
12 x = [1, 2, 3, 4, 5]
13 y = [5, 1, 4, 2, 3]
14 ax.plot(x, y)
15 ax.set_title('tytuł')
16 ax.set_xlabel('etykieta osi x')
17 ax.set_ylabel('etykieta osi y')
```

Klasa Axes

```
matplotlib_03.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykorzystanie metod obiektu Axes.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  #import numpy as np
9
10 #utworzenie obiektu wykresu (figure)
11 fig, ax = plt.subplots()
12 x = [1, 2, 3, 4, 5]
13 y = [5, 1, 4, 2, 3]
14 ax.plot(x, y)
15 ax.set_title('tytuł')
16 ax.set_xlabel('etykieta osi x')
17 ax.set_ylabel('etykieta osi y')
```



Biblioteka matplotlib

Przydatne klasy:

- Figure
- Axes
- Axis
- Locator
- Formatter
- Artist

Biblioteka matplotlib

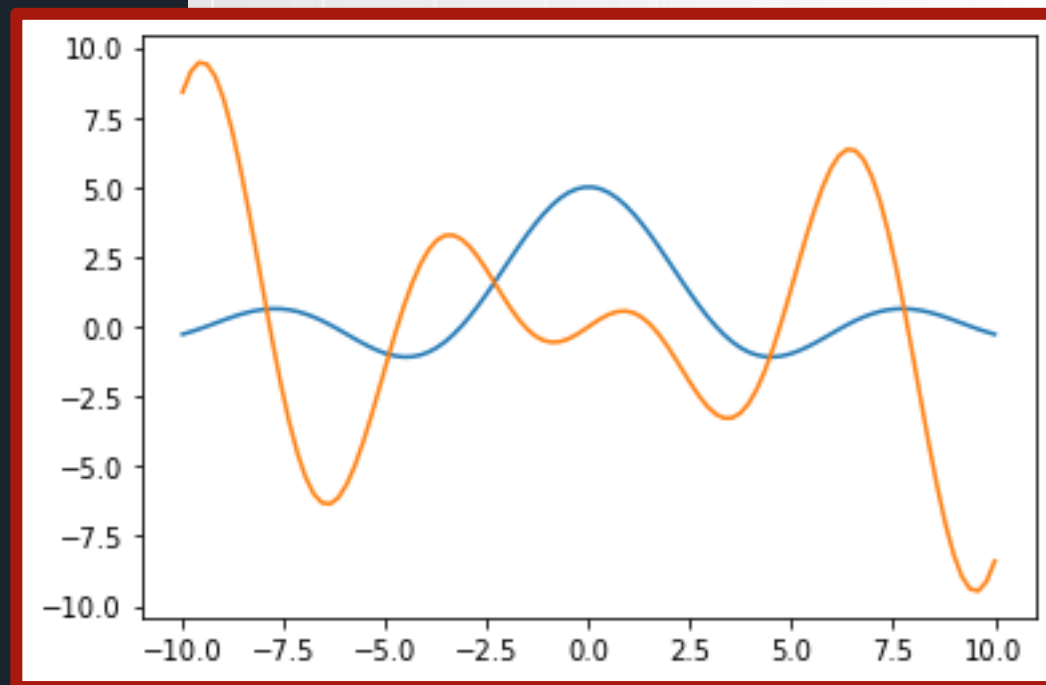
```
matplotlib_04a.py X matplotlib_04b.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący podejście obiektowe i proceduralne.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  import numpy as np
9
10 # przygotowanie danych
11 x = np.linspace(-10,10,100)
12 y1 = 5*np.sin(x)/x
13 y2 = x*np.cos(x)
14
15 # stworzenie wykresu (podejście obiektowe)
16 # fig - obiekt wykresu
17 # ax - obiekt układu współrzędnych
18 fig, ax = plt.subplots()
19 # wywołanie metody plot() obiektu ax
20 ax.plot(x, y1)
21 ax.plot(x, y2)
```

Biblioteka matplotlib

```
matplotlib_04a.py X matplotlib_04b.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący podejście obiektowe i proceduralne.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  import numpy as np
9
10 # przygotowanie danych
11 x = np.linspace(-10,10,100)
12 y1 = 5*np.sin(x)/x
13 y2 = x*np.cos(x)
14
15 # stworzenie wykresu (podejście proceduralne)
16 # wywołanie funkcji figure() tworzącej wykres
17 plt.figure()
18 # wywołanie funkcji plot() tworzącej obiekt linii
19 # w osiach wykresu
20 plt.plot(x, y1)
21 plt.plot(x, y2)
```

Biblioteka matplotlib

```
matplotlib_04a.py X matplotlib_04b.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący podejście obiektowe i proceduralne.
4  """
5
6  #import matplotlib as mpl
7  import matplotlib.pyplot as plt
8  import numpy as np
9
10 # przygotowanie danych
11 x = np.linspace(-10,10,100)
12 y1 = 5*np.sin(x)/x
13 y2 = x*np.cos(x)
14
15 # stworzenie wykresu (podejście proceduralne)
16 # wywołanie funkcji figure() tworzącej wykres
17 plt.figure()
18 # wywołanie funkcji plot() tworzącej obiekt linii
19 # w osiach wykresu
20 plt.plot(x, y1)
21 plt.plot(x, y2)
```



Biblioteka matplotlib

```
15 # stworzenie wykresu (podejście obiektowe)
16 # fig - obiekt wykresu
17 # ax - obiekt układu współrzędnych
18 fig, ax = plt.subplots()
19 # wywołanie metody plot() obiektu ax
20 ax.plot(x, y1)
21 ax.plot(x, y2)
```

```
15 # stworzenie wykresu (podejście proceduralne)
16 # wywołanie funkcji figure() tworzącej wykres
17 plt.figure()
18 # wywołanie funkcji plot() tworzącej obiekt linii
19 # w osiach wykresu
20 plt.plot(x, y1)
21 plt.plot(x, y2)
```

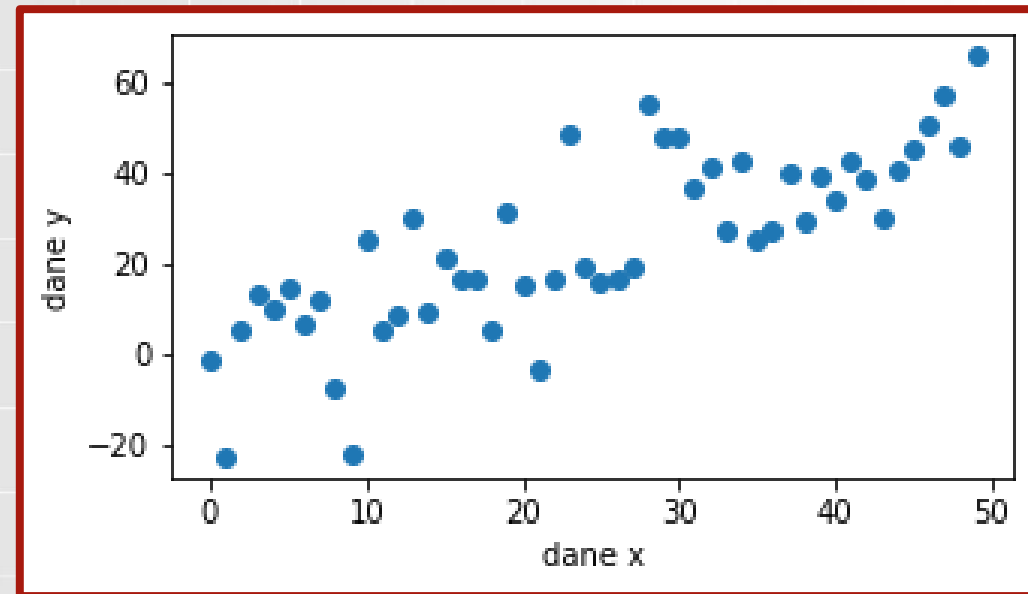
Wykres - scatter

```
matplotlib_05.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykres typu scatter.
4  Dane do wykresu przekazane jako słownik.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11 # przygotowanie danych w postaci słownika
12 # zawierającego tablice ndarray
13 data = {'x': np.arange(50) }
14 data['y'] = data['x'] + 10 * np.random.randn(50)
15
16 # utworzenie wykresu
17 fig, ax = plt.subplots(figsize=(5, 2.7))
18 # wykres typu scatter z danych w słowniku data
19 ax.scatter('x', 'y', data=data)
20 # etykiety osi
21 ax.set_xlabel('dane x')
22 ax.set_ylabel('dane y')
```

Wykres - scatter

matplotlib_05.py X

```
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykres typu scatter.
4  Dane do wykresu przekazane jako słownik.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11 # przygotowanie danych w postaci słownika
12 # zawierającego tablice ndarray
13 data = {'x': np.arange(50) }
14 data['y'] = data['x'] + 10 * np.random.randn(50)
15
16 # utworzenie wykresu
17 fig, ax = plt.subplots(figsize=(5, 2.7))
18 # wykres typu scatter z danych w słowniku data
19 ax.scatter('x', 'y', data=data)
20 # etykiety osi
21 ax.set_xlabel('dane x')
22 ax.set_ylabel('dane y')
```

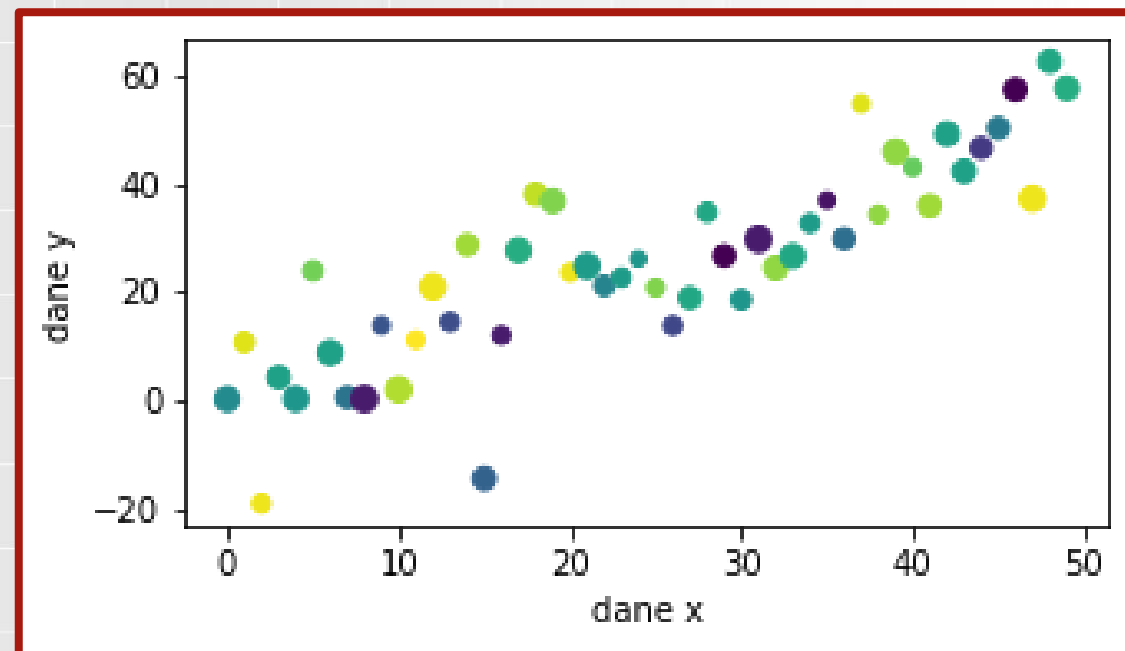


Wykres - scatter

```
matplotlib_06.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykres typu scatter.
4  Dane do wykresu przekazane jako słownik.
5  Punkty mają różne rozmiary oraz kolory.
6  """
7
8  #import matplotlib as mpl
9  import matplotlib.pyplot as plt
10 import numpy as np
11
12 # przygotowanie danych w postaci słownika
13 # zawierającego tablice ndarray
14 data = {'x': np.arange(50) }
15 data['y'] = data['x'] + 10 * np.random.randn(50)
16 data['s'] = np.random.randint(20, 60, 50)
17 data['c'] = np.random.randint(0, 50, 50)
18
19 # utworzenie wykresu
20 fig, ax = plt.subplots(figsize=(5, 2.7))
21 # argument c - pozwala przekazać kolor znacznika
22 # argument s - pozwala przekazać rozmiar znacznika
23 ax.scatter('x', 'y', c='c', s='s', data=data)
24 ax.set_xlabel('dane x')
25 ax.set_ylabel('dane y')
```

Wykres - scatter

```
matplotlib_06.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykres typu scatter.
4  Dane do wykresu przekazane jako słownik.
5  Punkty mają różne rozmiary oraz kolory.
6  """
7
8  #import matplotlib as mpl
9  import matplotlib.pyplot as plt
10 import numpy as np
11
12 # przygotowanie danych w postaci słownika
13 # zawierającego tablice ndarray
14 data = {'x': np.arange(50) }
15 data['y'] = data['x'] + 10 * np.random.randn(50)
16 data['s'] = np.random.randint(20, 60, 50)
17 data['c'] = np.random.randint(0, 50, 50)
18
19 # utworzenie wykresu
20 fig, ax = plt.subplots(figsize=(5, 2.7))
21 # argument c - pozwala przekazać kolor znacznika
22 # argument s - pozwala przekazać rozmiar znacznika
23 ax.scatter('x', 'y', c='c', s='s', data=data)
24 ax.set_xlabel('dane x')
25 ax.set_ylabel('dane y')
```

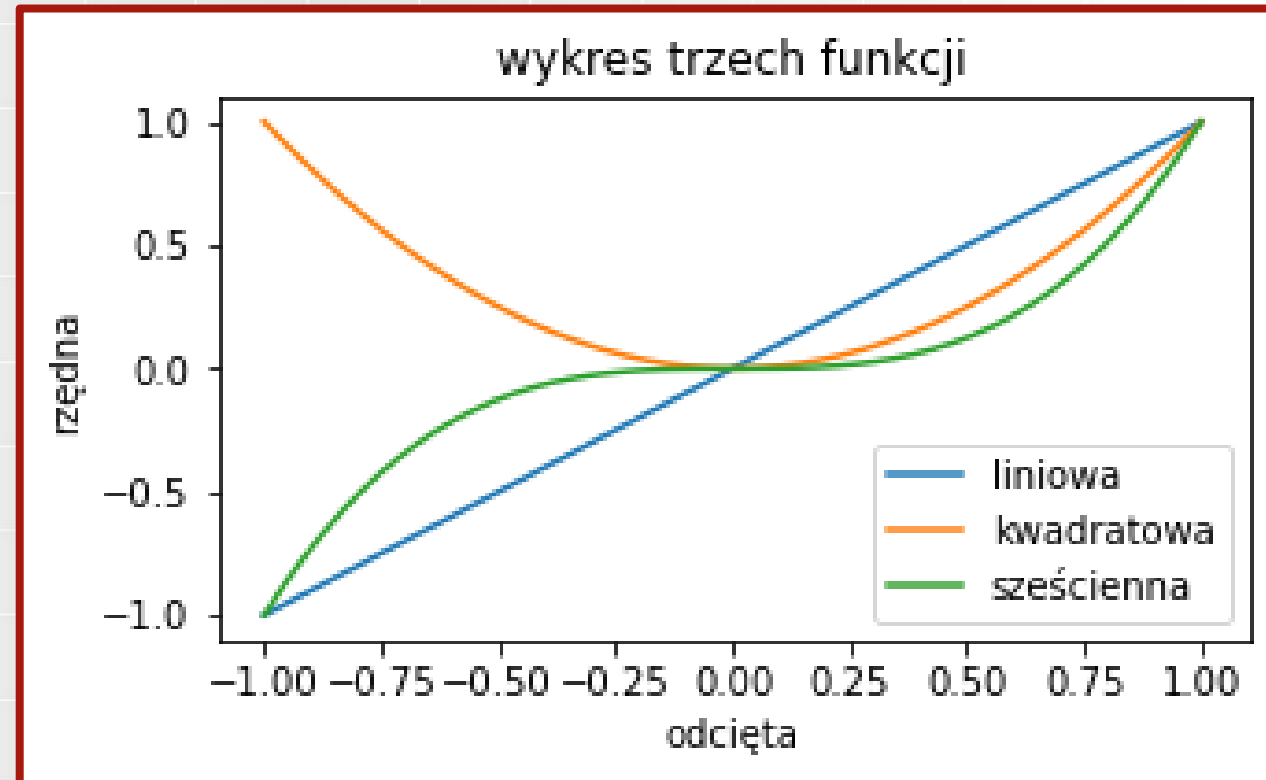


Wykres - plot

```
matplotlib_07.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykres zawierający trzy
4  serie danych oraz legendę.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11 # przygotowanie wektora argumentów (odciętych)
12 x = np.linspace(-1,1)
13
14 # utworzenie wykresu
15 fig, ax = plt.subplots(figsize=(5, 2.7))
16 # dodanie funkcji liniowej do wykresu
17 ax.plot(x, x, label="liniowa")
18 # dodanie funkcji kwadratowej
19 ax.plot(x, x**2, label="kwadratowa")
20 # dodanie funkcji sześciennnej
21 ax.plot(x, x**3, label="sześcienna")
22
23 ax.set_title('wykres trzech funkcji')
24 ax.set_xlabel('odcięta')
25 ax.set_ylabel('rzędna')
26 # umieszczenie legendy
27 ax.legend()
```

Wykres - plot

```
matplotlib_07.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykres zawierający trzy
4  serie danych oraz legendę.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11  # przygotowanie wektora argumentów (odciętych)
12  x = np.linspace(-1,1)
13
14  # utworzenie wykresu
15  fig, ax = plt.subplots(figsize=(5, 2.7))
16  # dodanie funkcji liniowej do wykresu
17  ax.plot(x, x, label="liniowa")
18  # dodanie funkcji kwadratowej
19  ax.plot(x, x**2, label="kwadratowa")
20  # dodanie funkcji sześcienniej
21  ax.plot(x, x**3, label="sześcienna")
22
23  ax.set_title('wykres trzech funkcji')
24  ax.set_xlabel('odcięta')
25  ax.set_ylabel('rzędna')
26  # umieszczenie legendy
27  ax.legend()
```



Tworzenie wykresu w funkcji

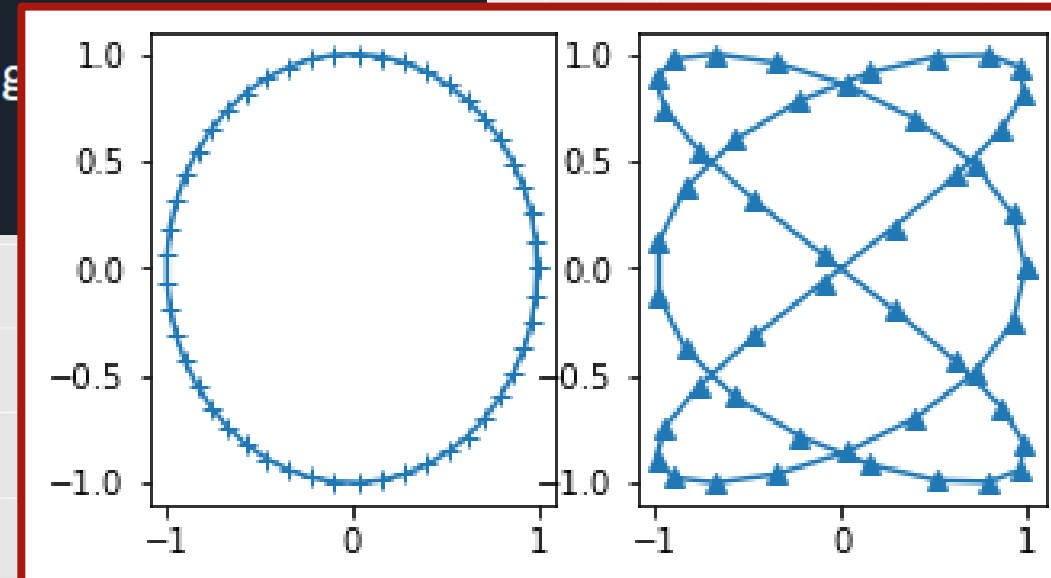
```
matplotlib_08.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykorzystanie funkcji pomocniczej
4  do tworzenia wykresu.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11  def plotter(ax, data_x, data_y, params):
12      """
13      Funkcja pomocnicza do tworzenia wykresów.
14      """
15      out = ax.plot(data_x, data_y, **params)
16      return out
```

Tworzenie wykresu w funkcji

```
matplotlib_08.py X
18 # przygotowanie danych wejściowych
19 phi = np.linspace(0, 2*np.pi)
20 x1 = np.cos(phi)
21 y1 = np.sin(phi)
22 x2 = np.cos(3*phi)
23 y2 = np.sin(2*phi)
24
25 # utworzenie wykresu
26 fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(5, 2.7))
27 plotter(ax1, x1, y1, {"marker": "+"})
28 plotter(ax2, x2, y2, {"marker": "^"})
```

Tworzenie wykresu w funkcji

```
matplotlib_08.py X
18 # przygotowanie danych wejściowych
19 phi = np.linspace(0, 2*np.pi)
20 x1 = np.cos(phi)
21 y1 = np.sin(phi)
22 x2 = np.cos(3*phi)
23 y2 = np.sin(2*phi)
24
25 # utworzenie wykresu
26 fig, (ax1, ax2) = plt.subplots(1, 2, fig
27 plotter(ax1, x1, y1, {"marker": "+"})
28 plotter(ax2, x2, y2, {"marker": "^"})
```

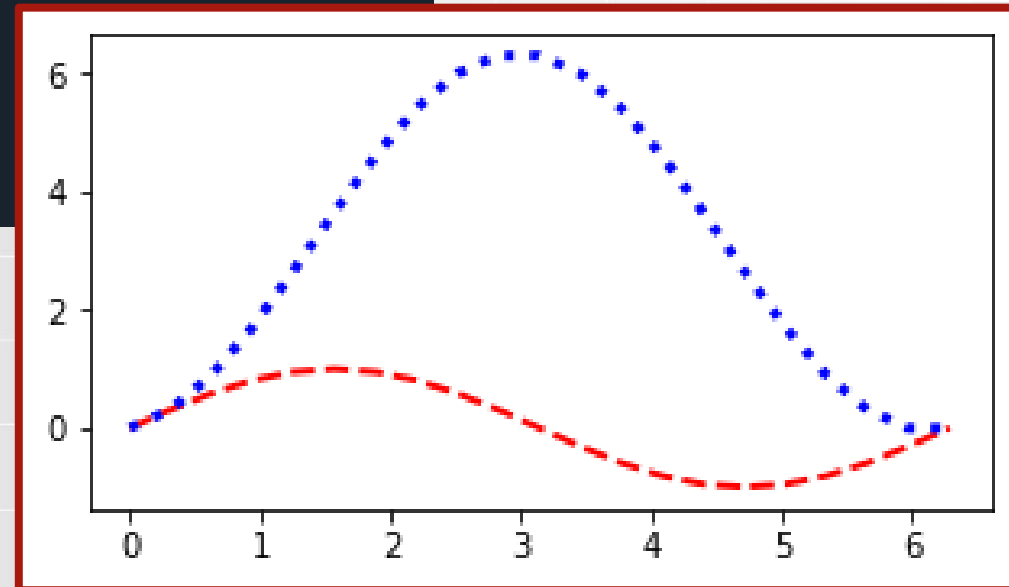


Właściwości wykresów

```
matplotlib_09.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący różne sposoby
4  modyfikowania właściwości serii danych.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11  # przygotowanie danych wejściowych
12  phi = np.linspace(0, 2*np.pi, 21)
13  x = np.sin(phi)
14  y = np.cumsum(x)
```

Właściwości wykresów

```
matplotlib_09.py X
16 # utworzenie wykresu
17 fig, ax = plt.subplots(figsize=(5, 2.7))
18 # wykres pierwszej serii danych
19 # ustalone kolor, styl i szerokość linii
20 ax.plot(phi, x, color="red", linestyle="--", linewidth=2)
21 # wykres drugiej serii danych
22 l2, = ax.plot(phi, y)
23 # ustalone kolor, styl i szerokość linii
24 l2.set_color("blue")
25 l2.set_linestyle(":")
26 l2.set_linewidth(3)
```

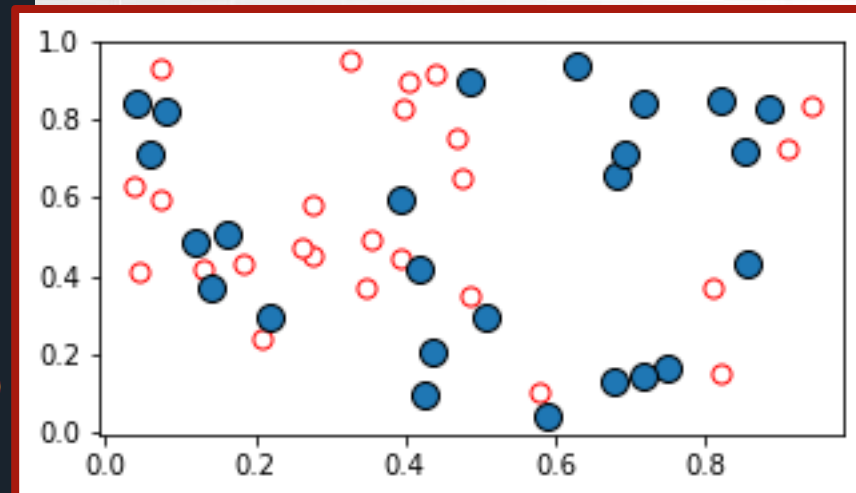


Kolory w bibliotece matplotlib

```
matplotlib_10.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący różne sposoby
4  definiowania kolorów.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11 # przygotowanie danych wejściowych
12 x1, y1, x2, y2 = np.random.rand(4, 25)
13
14 # utworzenie wykresu
15 fig, ax = plt.subplots(figsize=(5, 2.7))
16 # kolor wypełnienia znaczników - biały
17 # kolor krawędzi znaczników - czerwony
18 ax.scatter(x1, y1, s=50, facecolor="white", edgecolor="r")
19 # kolor wypełnienia znaczników - niebieski
20 # kolor krawędzi znaczników - czarny
21 ax.scatter(x2, y2, s=100, facecolor="C0", edgecolor=(0., 0., 0.))
22 # więcej o nazywaniu kolorów
23 # https://matplotlib.org/stable/tutorials/colors/colors.html
```

Kolory w bibliotece matplotlib

```
matplotlib_10.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący różne sposoby
4  definiowania kolorów.
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11  # przygotowanie danych wejściowych
12  x1, y1, x2, y2 = np.random.rand(4, 25)
13
14  # utworzenie wykresu
15  fig, ax = plt.subplots(figsize=(5, 2.7))
16  # kolor wypełnienia znaczników - biały
17  # kolor krawędzi znaczników - czerwony
18  ax.scatter(x1, y1, s=50, facecolor="white", edgecolor="r")
19  # kolor wypełnienia znaczników - niebieski
20  # kolor krawędzi znaczników - czarny
21  ax.scatter(x2, y2, s=100, facecolor="C0", edgecolor=(0., 0., 0.))
22  # więcej o nazywaniu kolorów
23  # https://matplotlib.org/stable/tutorials/colors/colors.html
```

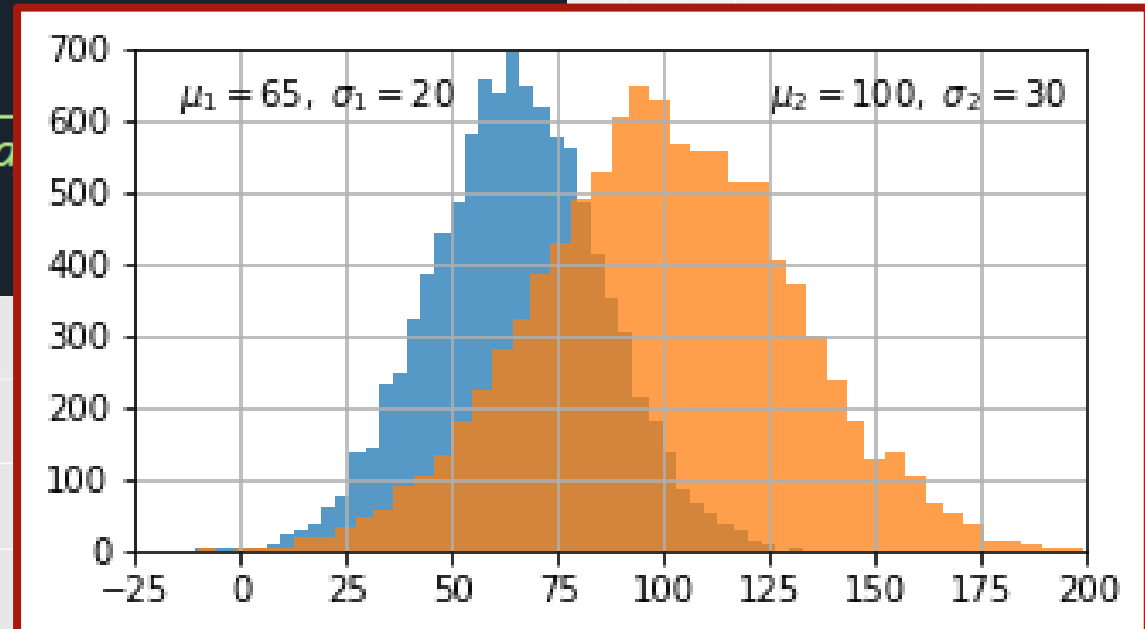


Opisy na wykresach

```
matplotlib_11.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący możliwość wprowadzania
4  oznaczeń tekstowych na wykresie.
5
6  """
7
8  #import matplotlib as mpl
9  import matplotlib.pyplot as plt
10 import numpy as np
11
12 # przygotowanie dwóch zestawów danych losowych
13 mu1, sigma1 = 65, 20
14 x1 = mu1 + sigma1 * np.random.randn(10000)
15 mu2, sigma2 = 100, 30
16 x2 = mu2 + sigma2 * np.random.randn(10000)
```

Opisy na wykresach

```
matplotlib_11.py X
18 # utworzenie wykresu
19 fig, ax = plt.subplots(figsize=(5, 2.7))
20 # zawierającego dwa histogramy
21 ax.hist(x1, 50, alpha=0.75) # kanał alpha - przezroczystość
22 ax.hist(x2, 50, alpha=0.75)
23 # dobranie zakresów osi
24 ax.axis([-25, 200, 0, 700])
25 # umieszczenie opisów histogramów
26 ax.text(-15, 620, r"$\mu_1=65, \sigma_1=20$")
27 ax.text(125, 620, r"$\mu_2=100, \sigma_2=30$")
28 # dodanie siatki na wykresie
29 ax.grid(True)
```



Skale wykresów

```
matplotlib_12.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący wykorzystanie skali logarytmicznej.
4
5  """
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11 # przygotowanie danych losowych
12 mu1, sigma1 = 1, 1
13 y = mu1 + sigma1 * np.random.randn(50)
14 z = 10**y
15 x = np.arange(len(y))
```

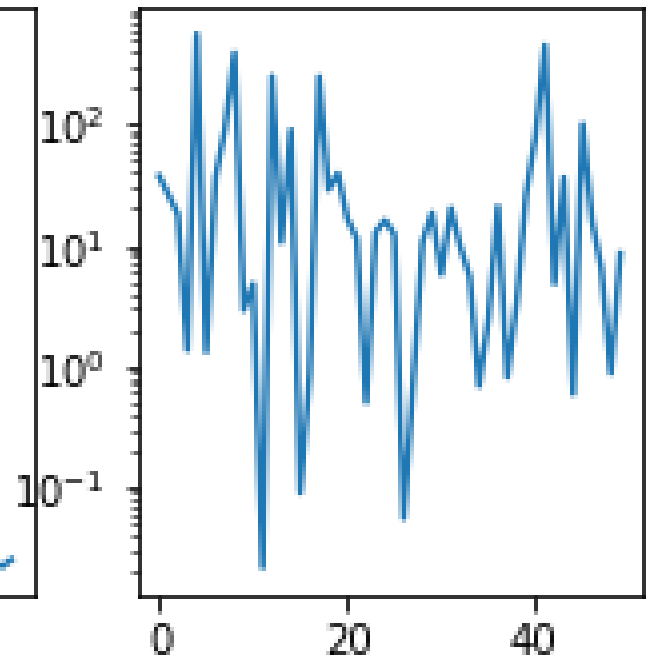
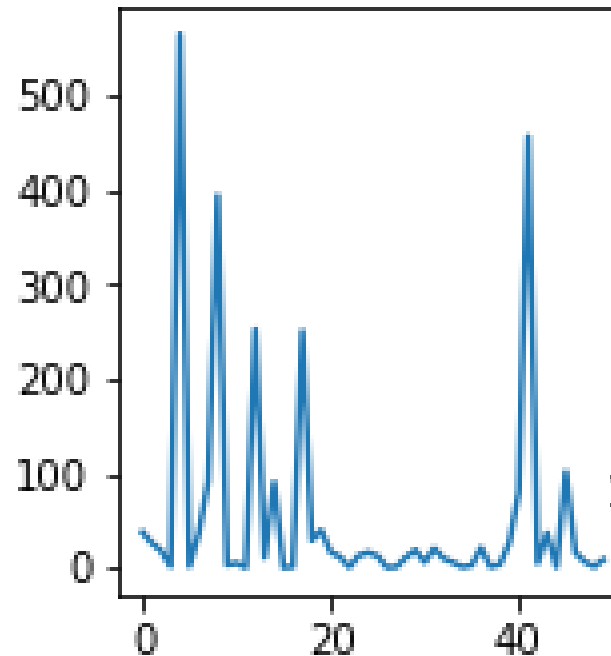
Skale wykresów

```
matplotlib_12.py X
17 # utworzenie wykresu o dwóch układach współrzędnych
18 fig, axs = plt.subplots(1, 2, figsize=(5, 2.7))
19 # wykres w skali liniowej
20 axs[0].plot(x,z)
21
22 # wykres w skali logarytmicznej
23 axs[1].plot(x,z)
24 axs[1].set_yscale('log')
```

Skale wykresów

matplotlib_12.py ✕

```
17 # utworzenie wykresu o dwóch układach współrzędnych
18 fig, axs = plt.subplots(1, 2, figsize=(5, 2.7))
19 # wykres w skali liniowej
20 axs[0].plot(x,z)
21
22 # wykres w skali logarytmicznej
23 axs[1].plot(x,z)
24 axs[1].set_yscale('log')
```



Podwójne osie

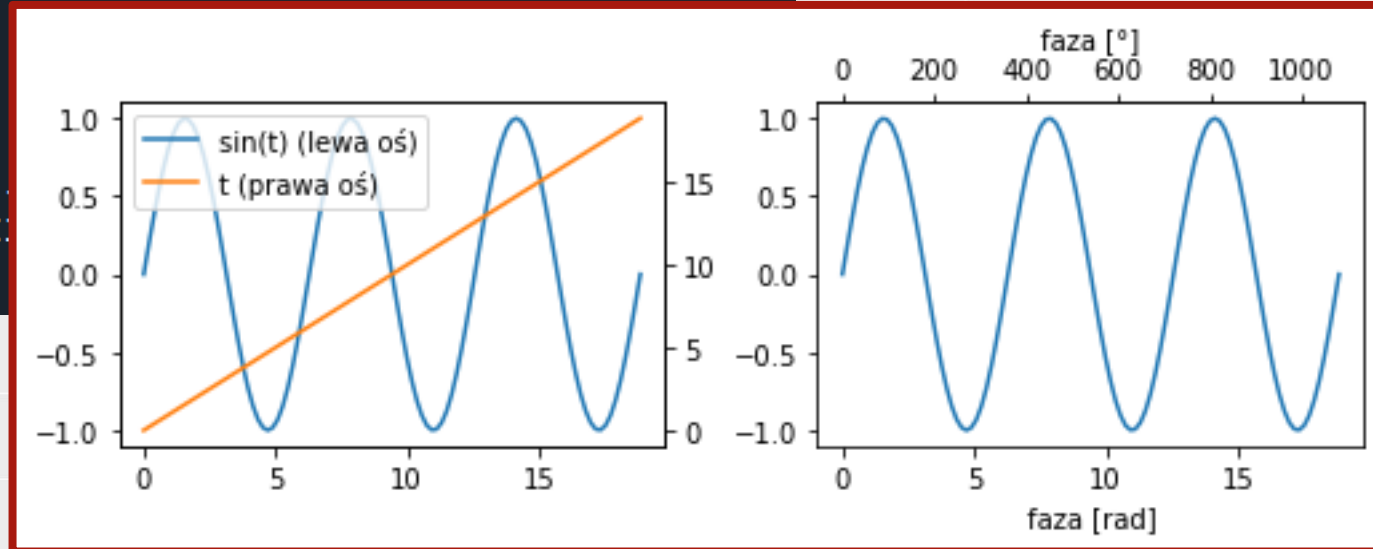
```
matplotlib_13.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący umieszczanie dodatkowych osi
4  w układzie współrzędnych.
5
6  """
7
8  #import matplotlib as mpl
9  import matplotlib.pyplot as plt
10 import numpy as np
11
12 t = np.linspace(0, 6*np.pi, 101)
13 s = np.sin(t)
```

Podwójne osie

```
matplotlib_13.py ✕  
15 fig, (ax1, ax3) = plt.subplots(1,  
16                               2,  
17                               figsize=(7, 2.7),  
18                               constrained_layout=True)  
19 l1, = ax1.plot(t, s)  
20 ax2 = ax1.twinx()  
21 l2, = ax2.plot(t, range(len(t)), 'C1')  
22 ax2.legend([l1, l2], ['sin(t) (lewa oś)', 't (prawa oś)'])  
23  
24 ax3.plot(t, s)  
25 ax3.set_xlabel('faza [rad]')  
26 ax4 = ax3.secondary_xaxis('top',  
27                           functions=(np.rad2deg, np.deg2rad))  
28 ax4.set_xlabel('faza [°]')
```

Podwójne osie

```
matplotlib_13.py ✕  
15 fig, (ax1, ax3) = plt.subplots(1,  
16                               2,  
17                               figsize=(7, 2.7),  
18                               constrained_layout=True)  
19 l1, = ax1.plot(t, s)  
20 ax2 = ax1.twinx()  
21 l2, = ax2.plot(t, range(len(t)), 'C1')  
22 ax2.legend([l1, l2], ['sin(t) (lewa oś)', 't (prawa oś)'])  
23  
24 ax3.plot(t, s)  
25 ax3.set_xlabel('faza [rad]')  
26 ax4 = ax3.secondary_xaxis('top', func=  
27                               lambda x: x * 180 / np.pi)  
28 ax4.set_xlabel('faza [°]')
```



Wykresy trójwymiarowe

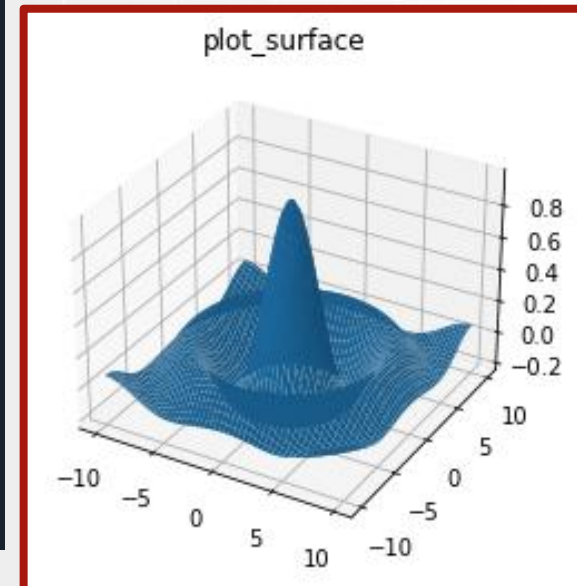
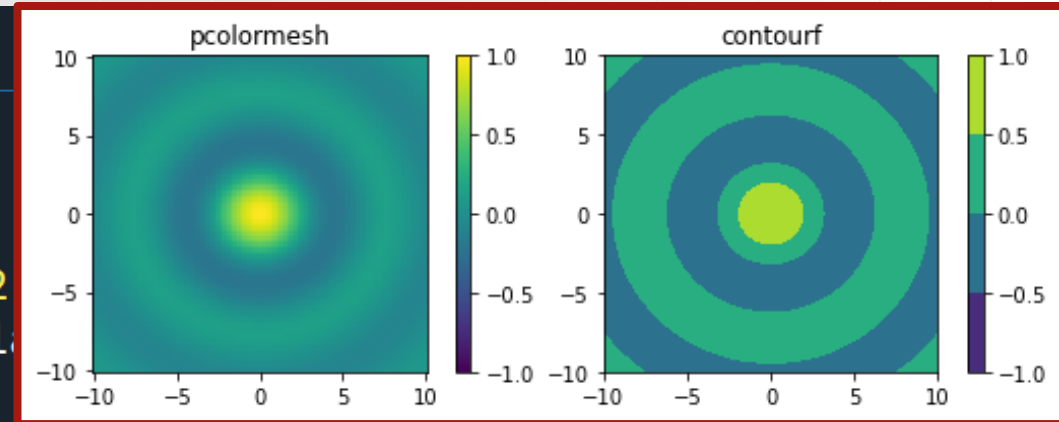
```
matplotlib_14.py X
1  # -*- coding: utf-8 -*-
2  """
3  Przykład pokazujący tworzenie wykresów trójwymiarowych.
4  """
5
6
7  #import matplotlib as mpl
8  import matplotlib.pyplot as plt
9  import numpy as np
10
11  # dane do wykresów
12  x = np.linspace(-10, 10, 100) # współrzędne x
13  y = np.linspace(-10, 10, 100) # współrzędne y
14  X, Y = np.meshgrid(x, y) # siatka współrzędnych X, Y
15  R = (X**2 + Y**2)**0.5 # obliczone R
16  Z = np.sin(R)/R # sinc(r)
```

Wykresy trójwymiarowe

```
matplotlib_14.py X
18 # przykładowe wykresy 2d
19 fig, (ax1, ax2) = plt.subplots(1,
20                                2,
21                                figsize=(7, 2.7),
22                                constrained_layout=True)
23
24 pc = ax1.pcolormesh(X, Y, Z, vmin=-1, vmax=1, shading="auto")
25 fig.colorbar(pc, ax=ax1)
26 ax1.set_title('pcolormesh')
27
28 co = ax2.contourf(X, Y, Z, levels=np.linspace(-1,1,5))
29 fig.colorbar(co, ax=ax2)
30 ax2.set_title('contourf')
31
32 # przykładowy wykres 3d
33 fig, ax3 = plt.subplots(1, subplot_kw = {'projection': '3d'})
34 ax3.plot_surface(X,Y,Z)
35 ax3.set_title('plot_surface')
```

Wykresy trójwymiarowe

```
matplotlib_14.py X
18 # przykładowe wykresy 2d
19 fig, (ax1, ax2) = plt.subplots(1,
20                               2,
21                               figsize=(7, 2),
22                               constrained_layout=True)
23
24 pc = ax1.pcolormesh(X, Y, Z, vmin=-1, vmax=1, shading="auto")
25 fig.colorbar(pc, ax=ax1)
26 ax1.set_title('pcolormesh')
27
28 co = ax2.contourf(X, Y, Z, levels=np.linspace(-1,1,5))
29 fig.colorbar(co, ax=ax2)
30 ax2.set_title('contourf')
31
32 # przykładowy wykres 3d
33 fig, ax3 = plt.subplots(1, subplot_kw = {'projection': '3d'})
34 ax3.plot_surface(X,Y,Z)
35 ax3.set_title('plot_surface')
```



Podsumowanie

Wprowadzenie w możliwości biblioteki matplotlib