

# Kurs rozszerzony języka Python

## Wykład 8.

Marcin Młotkowski

7 grudnia 2022

# Plan wykładu

- 1 Wprowadzenie
- 2 Wizualizacja danych: matplotlib
  - Porównanie implementacji
  - Funkcje parametryczne

# Plan wykładu

- 1 Wprowadzenie
- 2 Wizualizacja danych: matplotlib
  - Porównanie implementacji
  - Funkcje parametryczne

# Wstęp

Analiza, przetwarzanie i wizualizacja danych

# Pakiety

- NumPy
- SciPy
- matplotlib
- Pandas

# Narzędzia

IPython

Jupyter Notebook

# Plan wykładu

- 1 Wprowadzenie
- 2 Wizualizacja danych: matplotlib
  - Porównanie implementacji
  - Funkcje parametryczne

# matplotlib

Biblioteka w Pythonie do rysowania wykresów. Bardzo różnych.



# Prosty przykład: funkcja

```
import matplotlib.pyplot as plt
```

```
import math
```

```
def frange(fr, to, st):
```

```
    while fr < to:
```

```
        yield fr
```

```
        fr += st
```

```
xs = [ wart for wart in frange(0.1, 10*math.pi, 0.1) ]
```

```
ys = [ math.sin(t)*(t*t - 30*t) for t in xs ]
```

```
plt.plot(xs, ys)
```

```
plt.show()
```

# Dwa wykresy na jednej osi

Z poprzedniego wykładu:

```
plt.plot(time, temp, color='blue')  
plt.legend(['temperatura'])
```

```
plt.twinx()
```

```
plt.plot(time, cism, color='red')  
plt.legend(['ciśnienie'])
```

```
plt.plot.show()
```

# Uzupełnienie wykresu

Dokładamy wykres wiatru:

```
plt.bar(time, wiatr)
```

```
plt.show()
```

# Wyliczanie $n$ -tego wyrazu ciągu Fibonacciego

Popularne implementacje:

- implementacja rekurencyjna  $O(2^n)$ ;
- implementacja iteracyjna  $O(n)$ ;
- implementacja potęgowanie macierzy  $\Theta(\log n)$ .

# Wyliczanie $n$ -tego wyrazu ciągu Fibonacciego

Popularne implementacje:

- implementacja rekurencyjna  $O(2^n)$ ;
- implementacja iteracyjna  $O(n)$ ;
- implementacja potęgowanie macierzy  $\Theta(\log n)$ .

Na różnych wykresach

# Porównanie implementacji: ciąg Fibonacciego

```
import timeit

def fib1(n):
    ....

def fib2(n):
    ....

def fib3(n):
    ....

xs = [ n for n in range(1, 30, 5) ]

ys1 = [ timeit.timeit(lambda : fib1(i), number=100) for i in xs ]
ys2 = [ timeit.timeit(lambda : fib2(i), number=100) for i in xs ]
ys3 = [ timeit.timeit(lambda : fib3(i), number=100) for i in xs ]
```

# Na jednym wykresie

```
plt.plot(xs, ys1, marker='x')  
plt.plot(xs, ys2, marker='o')  
plt.plot(xs, ys3, marker='*')  
  
plt.legend(['rekurencja', 'iteracja', "macierzowy"])  
  
plt.show()
```

# Na odrębnych wykresach

```
fig, (ax1, ax2, ax3) = plt.subplots(3,1)
```

```
ax1.plot(xs, ys1, marker='x')  
ax1.set_title('rekurencyjna')  
ax1.set_ylabel("sec")
```

```
ax2.plot(xs, ys2, marker='o')  
ax2.set_title('iteracyjna')
```

```
ax3.plot(xs, ys3, marker='*')  
ax3.set_title('macierzowa')
```

```
plt.show()
```



# Inny przykład: funkcja parametryczna

Krzywe Lissajoux: złożenie dwóch prostopadłych ruchów harmoniczných

$$x(t) = \sin(a * t + \pi/2)$$

$$y(t) = \sin(b * t)$$

gdzie  $t \in [-\pi, \pi]$ . Gdy  $\frac{a}{b}$  jest wymierna, to krzywa jest zamknięta.

# Implementacja

```
import matplotlib.pyplot as plt  
import math
```

```
a, b = 9, 8
```

```
ts = list(frange(-math.pi, math.pi, 0.01))
```

```
xs = [ math.sin(a * t + math.pi/2) for t in ts ]  
ys = [ math.sin(b*t) for t in ts ]
```

```
plt.plot(xs, ys)  
plt.show()
```

# Jak animować wykresy

- wykres początkowy dla pewnych danych (wektory `xdata` i `ydata`) początkowych;
- aktualizacja: zmodyfikować `xdata` i `ydata`, narysować;
- wykorzystać obiekt klasy `matplotlib.animation.FuncAnim`

# Początek

```
import matplotlib.pyplot as plt
import matplotlib.animation as animation
import math

fig = plt.figure()
ax = plt.axes(xlim=(-2, 2), ylim=(-2, 2))

xdata, ydata = [], []
line, = ax.plot([], [])

def init():
    line.set_data([], [])
    return line,
```

# Aktualizacja wykresu

```
def animate(i):  
    t = 0.01*i  
    x = math.sin(a * t + math.pi / 2.0)  
    y = math.sin(b*t)  
    xdata.append(x)  
    ydata.append(y)  
    line.set_data(xdata, ydata)  
    return line,
```

# I na koniec:

```
ani = animation.FuncAnimation(fig, animate,  
                               init_func=init, frames=500,  
                               interval=50, blit=True)  
plt.show()
```

# Trochę inna animacja

```
def animate(i):  
    t = 0.01*i  
    x = math.sin(a * t + np.pi / 2.0)  
    y = math.sin(b*t)  
    xdata.append(x)  
    xdata = xdata[-50:]  
    ydata.append(y)  
    ydata = ydata[-50:]  
    line.set_data(xdata, ydata)  
    return line,
```

# Inne możliwości i rozszerzenia matplotlib

- różne rodzaje wykresów;
- wykresy trójwymiarowe;
- mapy;