

Języki i techniki programowania

Wykład 6

Maciej Rybczyński

newaxis, wszystkie pary

```
import numpy as np
```

```
a = np.array([1,2,3,4])  
print(a, '\n')
```

```
b = np.array([10,100,1000])  
b = b[:, np.newaxis]  
print(b, '\n')
```

równoważne z: `b = b.reshape(3,1)`

```
print(a + b)
```

spróbuj: `a==b`, `a!=b`, `a<b`, etc.

```
= RESTART: D:/CERNBox/zajeci  
[1 2 3 4]  
  
[[ 10]  
 [100]  
 [1000]]  
  
[[ 11  12  13  14]  
 [101 102 103 104]  
 [1001 1002 1003 1004]]
```

Liczby całkowite w numpy

```
import numpy as np

a = np.array([125,9], dtype=np.int64)

print(a, a.dtype.name, '\n')

print(a**10)

# int16 Integer (-32768 do 32767)
# int32 Integer (-2147483648 do 2147483647)
# int64 Integer (-9223372036854775808 do 9223372036854775807)

= RESTART: D:/CERNBox/zajecia/Inzynieria_danyc
[125    9] int64

[8985370930000934825          3486784401]
```

Zobacz: <https://numpy.org/doc/stable/user/basics.types.html>

Liczby całkowite w Pythonie, long

```
a = 2**1256
```

```
print(a)
```

```
print('\n')
```

```
print(type(a))
```

```
= RESTART: D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_c.py =  
12407222026186465122610617074208361552495076951106434828423395574574003290740681  
04983835153786221049409201337570462347005770631581345017541425216985425629706731  
33486621251348802723686095018190747177117115827677383562251056999546037518531100  
56445908894219814402235938106225033503804511552651196517814301016148410108331609  
39339005760608760471109018093451230071413602199639148199936
```

```
<class 'int'>    long integer!
```

Long integer w numpy

```
import numpy as np
```

```
a = np.array([125], dtype=object)
```

```
print(a, a.dtype.name, '\n')
```

```
print(a**35)
```

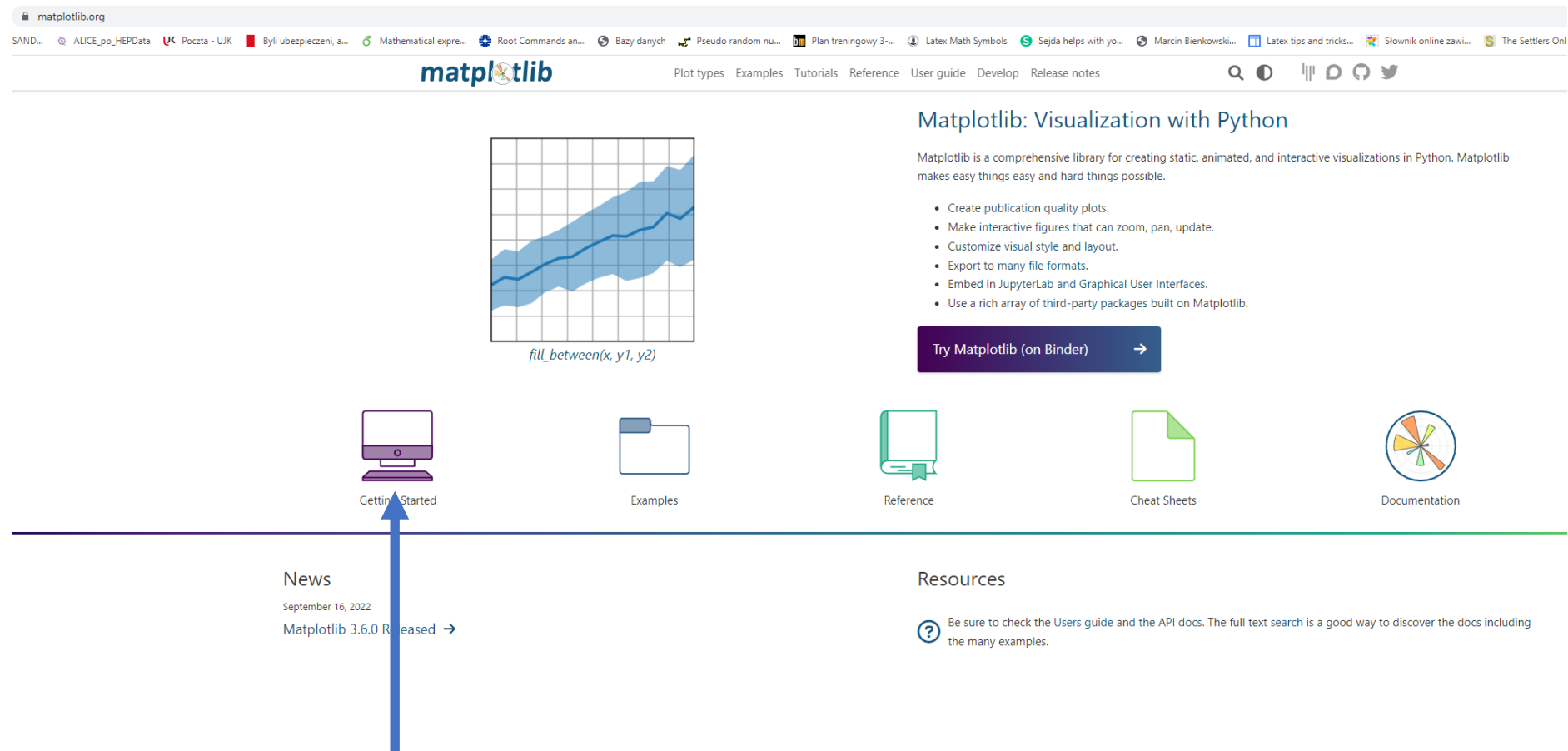
```
= RESTART: D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_d.py :  
[125] object
```

```
[24651903288156618919116517665087069677287701097156968899071216583251953125]
```

Matplotlib

Matplotlib, biblioteka do kreślenia 2D w Pythonie

<https://matplotlib.org/>



The screenshot shows the Matplotlib website homepage. At the top is a navigation bar with the Matplotlib logo and links for Plot types, Examples, Tutorials, Reference, User guide, Develop, and Release notes. Below the navigation bar is a large plot titled `fill_between(x, y1, y2)` showing a blue line with a shaded area. To the right of the plot is the heading "Matplotlib: Visualization with Python" followed by a description and a list of features. Below the features is a button that says "Try Matplotlib (on Binder)". At the bottom of the page is a horizontal bar with icons and labels for "Getting Started", "Examples", "Reference", "Cheat Sheets", and "Documentation". A blue arrow points from the "Getting Started" icon to the "News" section on the left, which contains the text "September 16, 2022" and "Matplotlib 3.6.0 Released →".

matplotlib.org

SAND... ALICE_pp_HEPData Poczta - UJK Byli ubezpieczeni, a... Mathematical expre... Root Commands an... Bazy danych Pseudo random nu... Plan treningowy 3-... Latex Math Symbols Sejda helps with yo... Marcin Bienkowski... Latex tips and tricks... Słownik online zawi... The Settlers Onl

matplotlib

Plot types Examples Tutorials Reference User guide Develop Release notes

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Try Matplotlib (on Binder) →

Getting Started Examples Reference Cheat Sheets Documentation

News

September 16, 2022

Matplotlib 3.6.0 Released →

Resources

Be sure to check the Users guide and the API docs. The full text search is a good way to discover the docs including the many examples.

kliknij tutaj

Instalowanie w Windows

matplotlib.org/stable/users/getting_started/

ALICE_pp_HEPData Poczta - UJK Byli ubezpieczeni, a... Mathematical expe... Root Commands an... Bazy danych Pseudo random nu... Plan treningowy 3-... Latex Math Symbols Sejda helps with yo...

matplotlib

Plot types Examples Tutorials Reference User guide Develop Release notes

Getting started

Installation quick-start

Install using pip:

```
pip install matplotlib
```

Install using conda:

```
conda install matplotlib
```

Further details are available in the [Installation Guide](#).

Draw a first plot

Here is a minimal example plot:

```
import matplotlib.pyplot as plt
import numpy as np

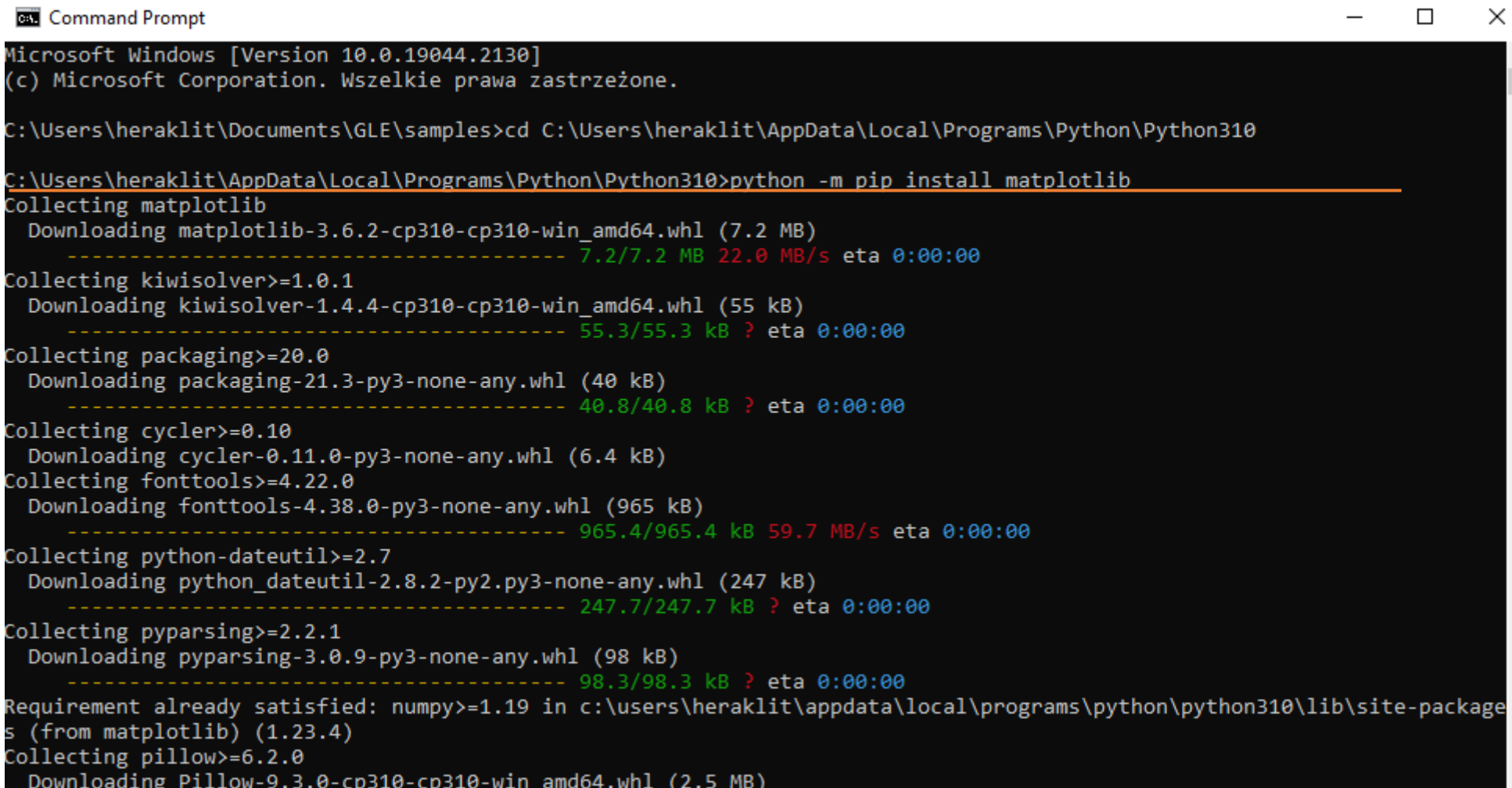
x = np.linspace(0, 2 * np.pi, 200)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y)
plt.show()
```

([Source code](#), [png](#))

Otwórz wiersz poleceń,
przejdź do katalogu gdzie jest zainstalowany python

Łatwa instalacja!



```
Command Prompt
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\heraklit\Documents\GLE\samples>cd C:\Users\heraklit\AppData\Local\Programs\Python\Python310


C:\Users\heraklit\AppData\Local\Programs\Python\Python310>python -m pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.6.2-cp310-cp310-win_amd64.whl (7.2 MB)
----- 7.2/7.2 MB 22.0 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp310-cp310-win_amd64.whl (55 kB)
----- 55.3/55.3 kB ? eta 0:00:00
Collecting packaging>=20.0
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
----- 40.8/40.8 kB ? eta 0:00:00
Collecting cycller>=0.10
  Downloading cycller-0.11.0-py3-none-any.whl (6.4 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.38.0-py3-none-any.whl (965 kB)
----- 965.4/965.4 kB 59.7 MB/s eta 0:00:00
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
----- 247.7/247.7 kB ? eta 0:00:00
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
----- 98.3/98.3 kB ? eta 0:00:00
Requirement already satisfied: numpy>=1.19 in c:\users\heraklit\AppData\Local\Programs\Python\Python310\lib\site-packages
 (from matplotlib) (1.23.4)
Collecting pillow>=6.2.0
  Downloading Pillow-9.3.0-cp310-cp310-win_amd64.whl (2.5 MB)
```

wpisz polecenie:

python -m pip install matplotlib

Wymaganie połączenie z internetem

Pierwszy wykres

 lec6_e.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/mo

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt

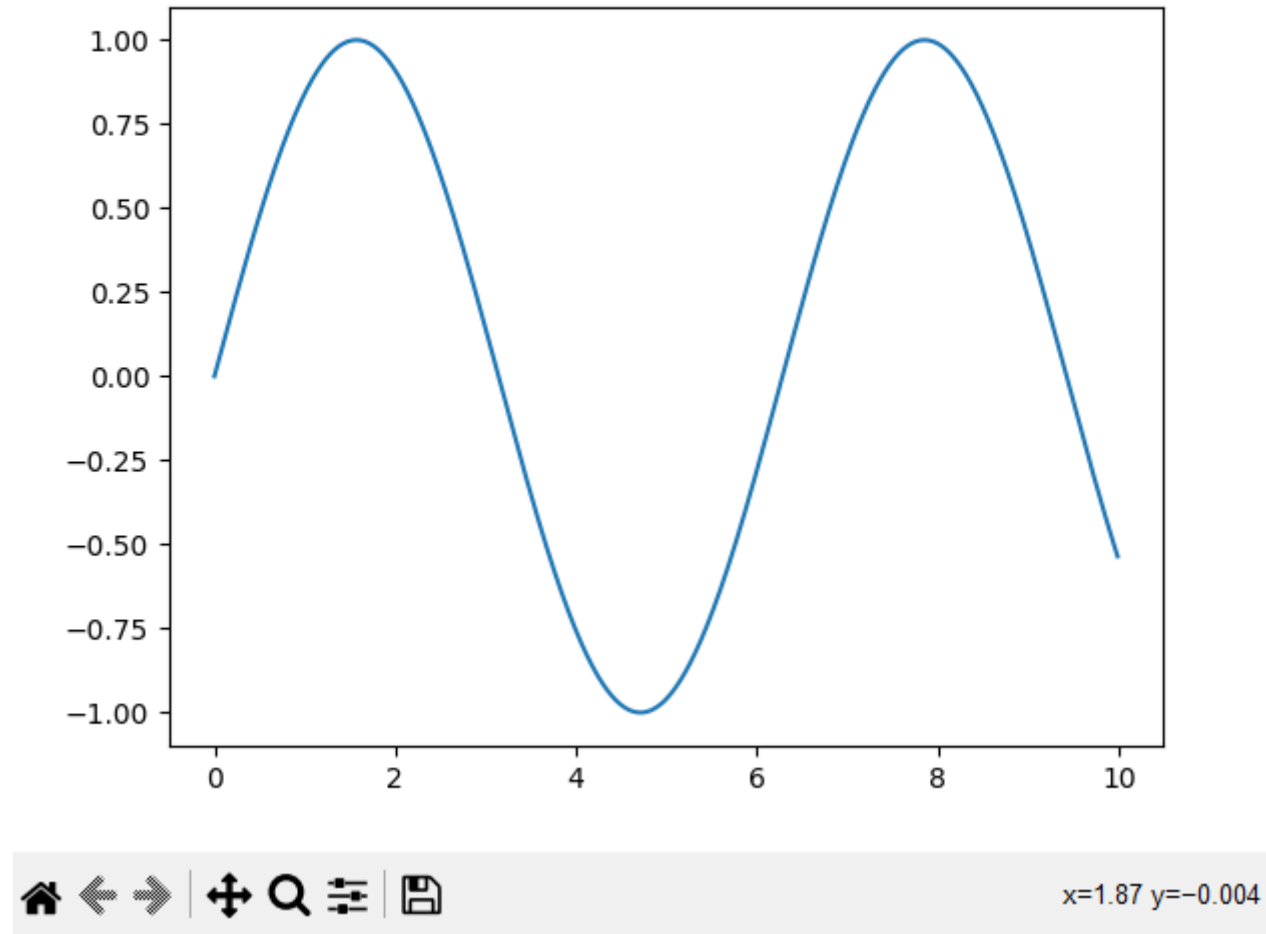
x = np.arange(0,10,0.01)

y = np.sin(x)

plt.plot(x,y)

plt.show()
```

$x = [0, 0.01, 0.02, \dots, 9.99]$ jest tablicą
 $y = \sin(x)$ jest również tablicą



Możesz zapisać wykres jako plik .pdf, .ps, .eps, etc.
Możesz przybliżyć, przesunąć, etc.

Lepszy wykres

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.size'] = 14      # rozmiar czcionki

x = np.arange(0,10,0.01)
y = np.sin(x)

plt.plot(x, y, '-', color='red', linewidth=2.8)  # '-' ciągła linia

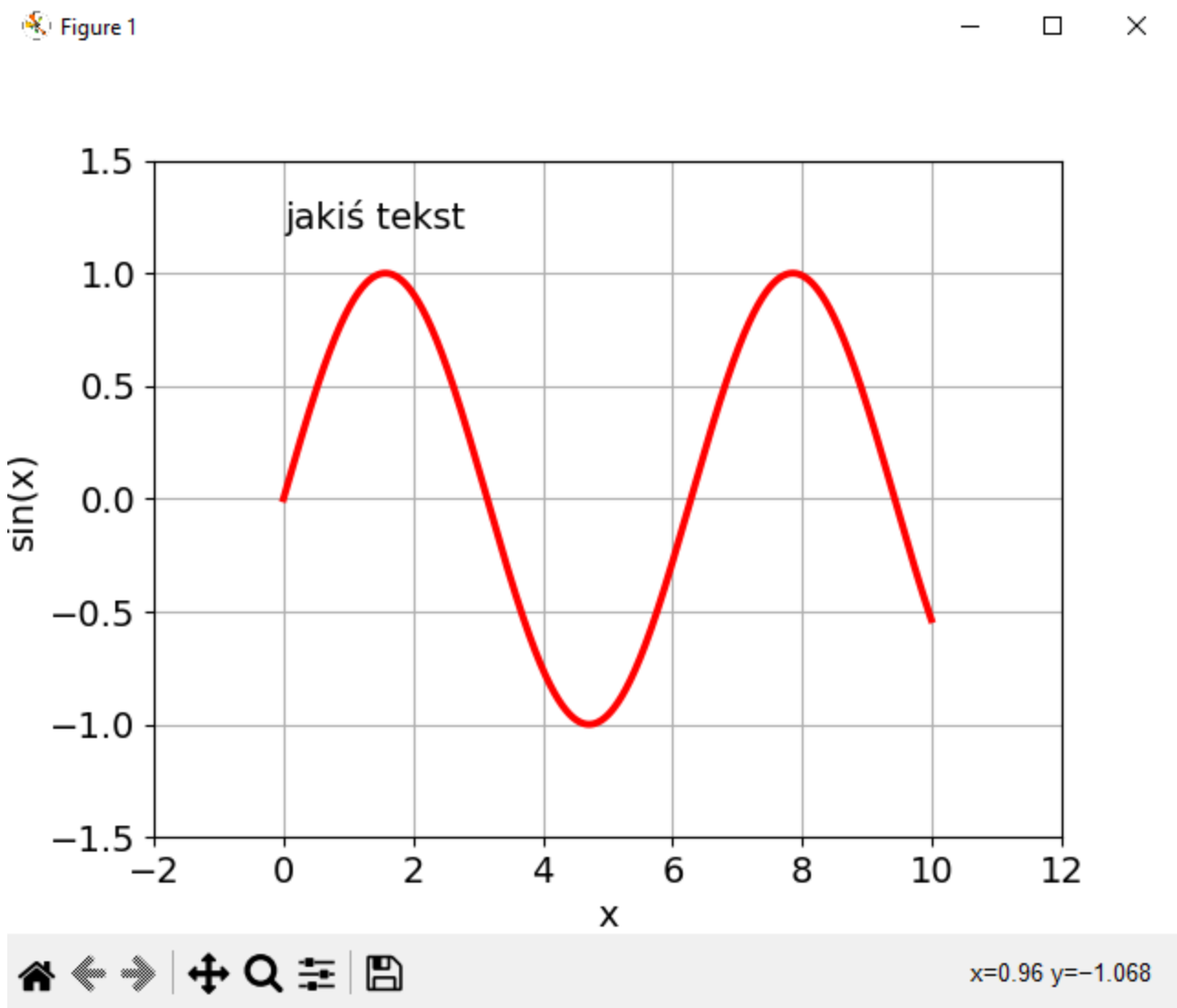
plt.xlabel('x')
plt.ylabel('sin(x)')      # podpisy osi

plt.axis([-2, 12, -1.5, 1.5])  # zakres osi

plt.grid(True)

plt.text(0,1.2, 'jakiś tekst')  # tekst zaczyna się w x=0 i y=1.2

plt.show()
```



Linie

 lec6_g.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_g.py

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.size'] = 14
plt.rcParams['lines.linewidth'] = 3    # grubość linii

x = [0,10]
y = np.array([1,1])

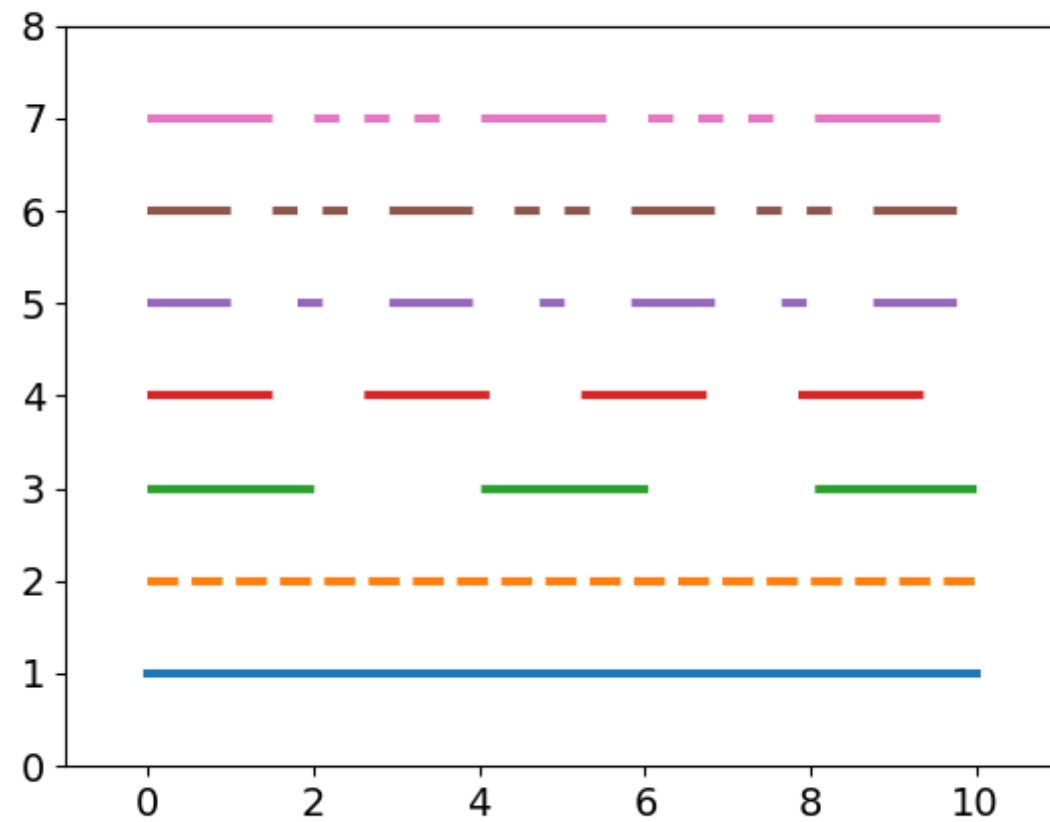
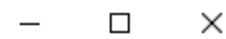
plt.plot(x, y, '-')
plt.plot(x, y*2, '--')
plt.plot(x, y*3, '-', dashes=[20,20])
plt.plot(x, y*4, '-', dashes=[15,11])
plt.plot(x, y*5, '-', dashes=[10,8,3,8])
plt.plot(x, y*6, '-', dashes=[10,5,3,3,3,5])
plt.plot(x, y*7, '-', dashes=[15,5,3,3,3,3,3,5])

plt.axis([-1, 11, 0, 8])


plt.show()
```

dashes = [linia, przerwa, linia, przerwa, ...]

Figure 1



Znaczniki

 lec6_h.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wykklad_6/lec6_h.py (3.10.7) —

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt

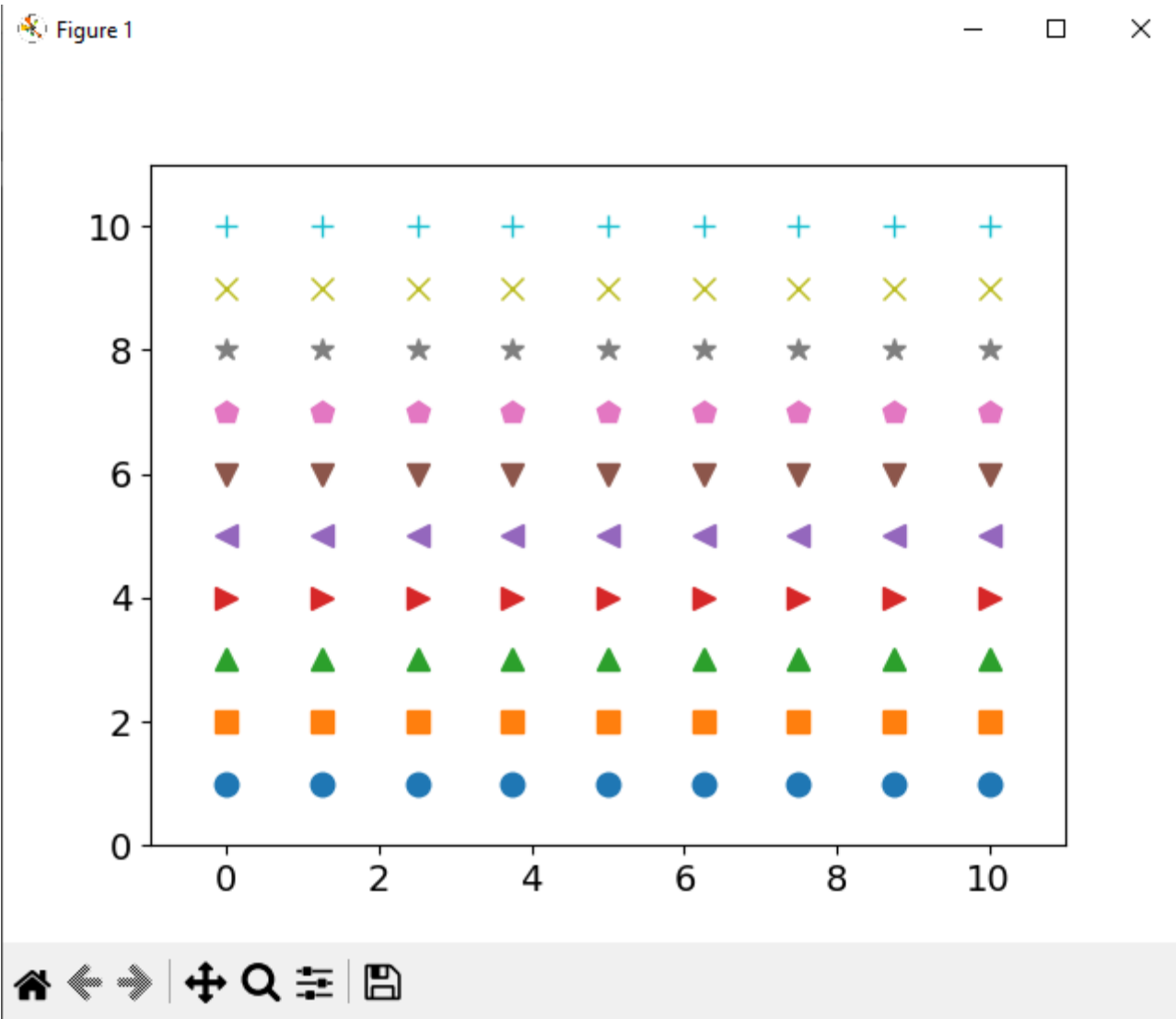
plt.rcParams['font.size'] = 14

x = np.linspace(0,10,9)    # 9 liczb od 0 do 10
y = np.ones(9)             # [1,0,1.0,...,1.0]

plt.plot(x, y, 'o', markersize=9)    # kółko
plt.plot(x, y*2, 's', ms=9)          # prostokąt, ms=markersize
plt.plot(x, y*3, '^', ms=9)          # trójkąt
plt.plot(x, y*4, '>', ms=9)          # trójkąt
plt.plot(x, y*5, '<', ms=9)          # trójkąt
plt.plot(x, y*6, 'v', ms=9)          # trójkąt
plt.plot(x, y*7, 'p', ms=9)          # pięciokąt
plt.plot(x, y*8, '*', ms=9)          # gwiazdka
plt.plot(x, y*9, 'x', ms=9)          # x
plt.plot(x, y*10, '+', ms=9)         # +

plt.axis([-1, 11, 0, 11])

plt.show()
```



Więcej znaczników

 lec6_i.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_i.py (3.10.7)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.size'] = 14
plt.rcParams['lines.markersize'] = 12

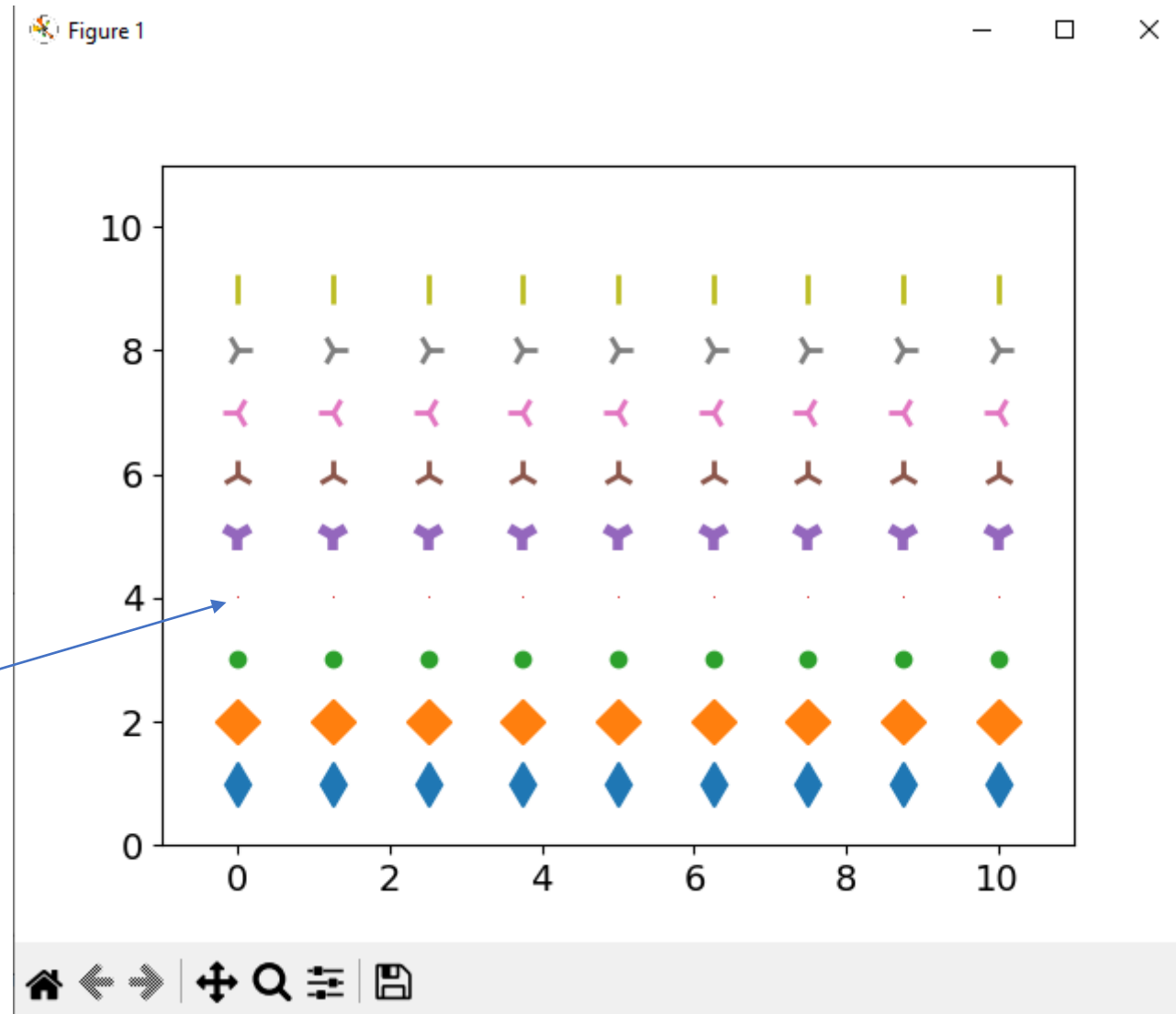
x = np.linspace(0,10,9)
y = np.ones(9)

plt.plot(x, y, 'd')           # wąski diament
plt.plot(x, y*2, 'D')         # diament
plt.plot(x, y*3, '.')         # punkt
plt.plot(x, y*4, ',')         # piksel
plt.plot(x, y*5, '1', mew=4)  # mew = marker edge width
plt.plot(x, y*6, '2', mew=2)
plt.plot(x, y*7, '3', mew=2)
plt.plot(x, y*8, '4', mew=2)
plt.plot(x, y*9, '|', mew=2)

plt.axis([-1, 11, 0, 11])

plt.show()
```

piksel



Kolory

 lec6_j.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wykklad_6/lec

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.size'] = 14
plt.rcParams['lines.linewidth'] = 3
```

```
x = np.linspace(0,10,9)
y = np.ones(9)
```

```
plt.plot(x, y, color='k')
plt.plot(x, y*2, color=(0.5,0.5,0.9)) # (R, G, B)
plt.plot(x, y*3, color='DeepSkyBlue')
plt.plot(x, y*4, color='OrangeRed')
plt.plot(x, y*5, color='#00FF00')
```

b=blue, g=green, r=red, c=cyan, m=magenta, y=yellow, k=black,
w=white

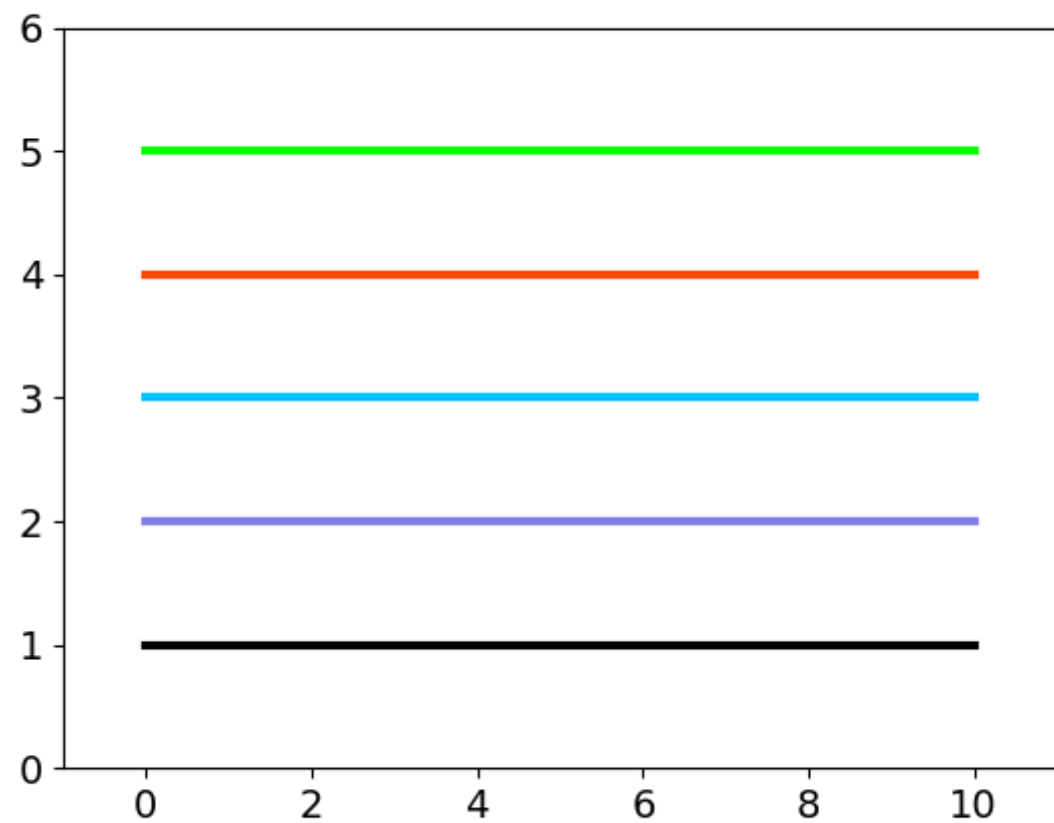
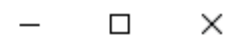
```
plt.axis([-1, 11, 0, 6])
```

```
plt.show()
```

Można stosować wszystkie kolory używane w HTML, zobacz:

https://www.w3schools.com/colors/colors_names.asp

Figure 1



Znaczniki i linie

 lec6_k.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wykklad_6/lec6_k.py (3.10.7)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.size'] = 14
plt.rcParams['lines.linewidth'] = 3

x = np.linspace(0,10,9)
y = np.ones(9)

plt.plot(x, y, 'r-')           # czerwona ciągła linia
plt.plot(x, y*2, 'o-', ms=10)  # kółko i ciągła linia
plt.plot(x, y*3, 's--', ms=10) # prostokąt i przerywana linia

plt.plot(x, y*4, 'D-', color='k',
         ms=10, mfc='w', mec='b', mew=2)

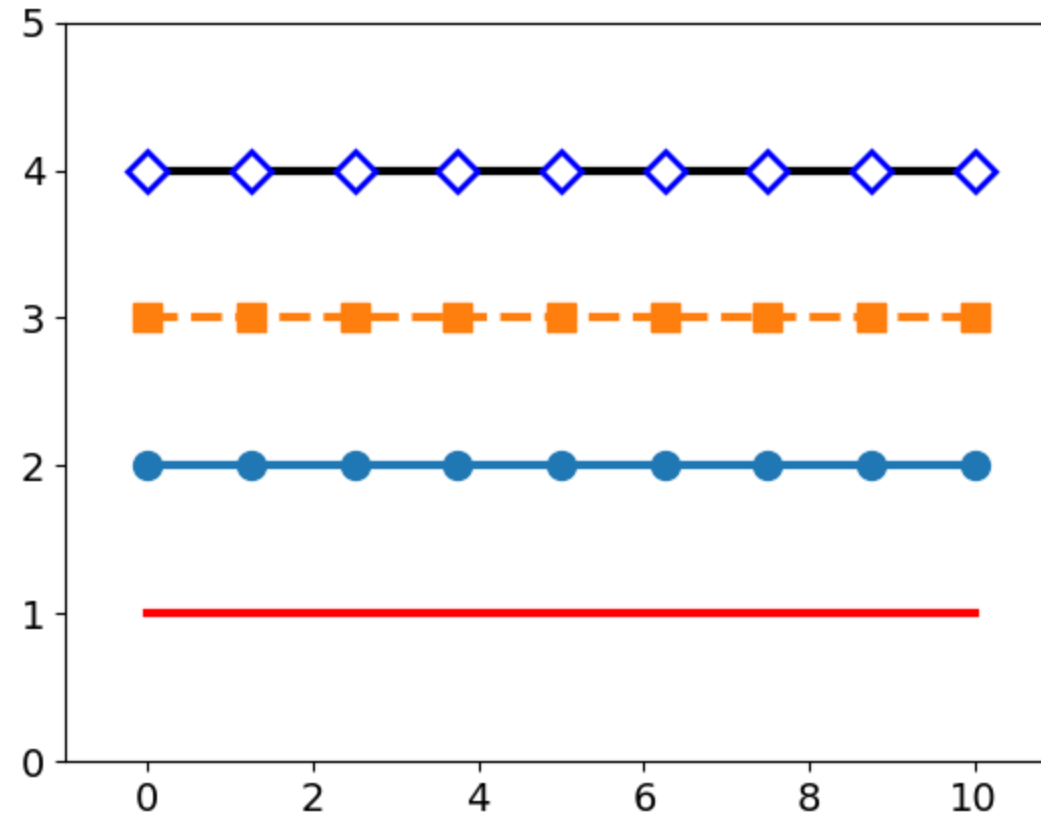
plt.axis([-1, 11, 0, 5])
plt.show()
```

ms = marker size,
mfc = marker face color
mec = marker edge color
mew = marker edge width

sprawdź: mfc='None'

Figure 1

— □ ×



Legenda

lec6_1.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_1.py (3.10.7)

— □

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['font.size'] = 14
plt.rcParams['legend.fontsize'] = 18 # rozmiar czcionki w legendzie

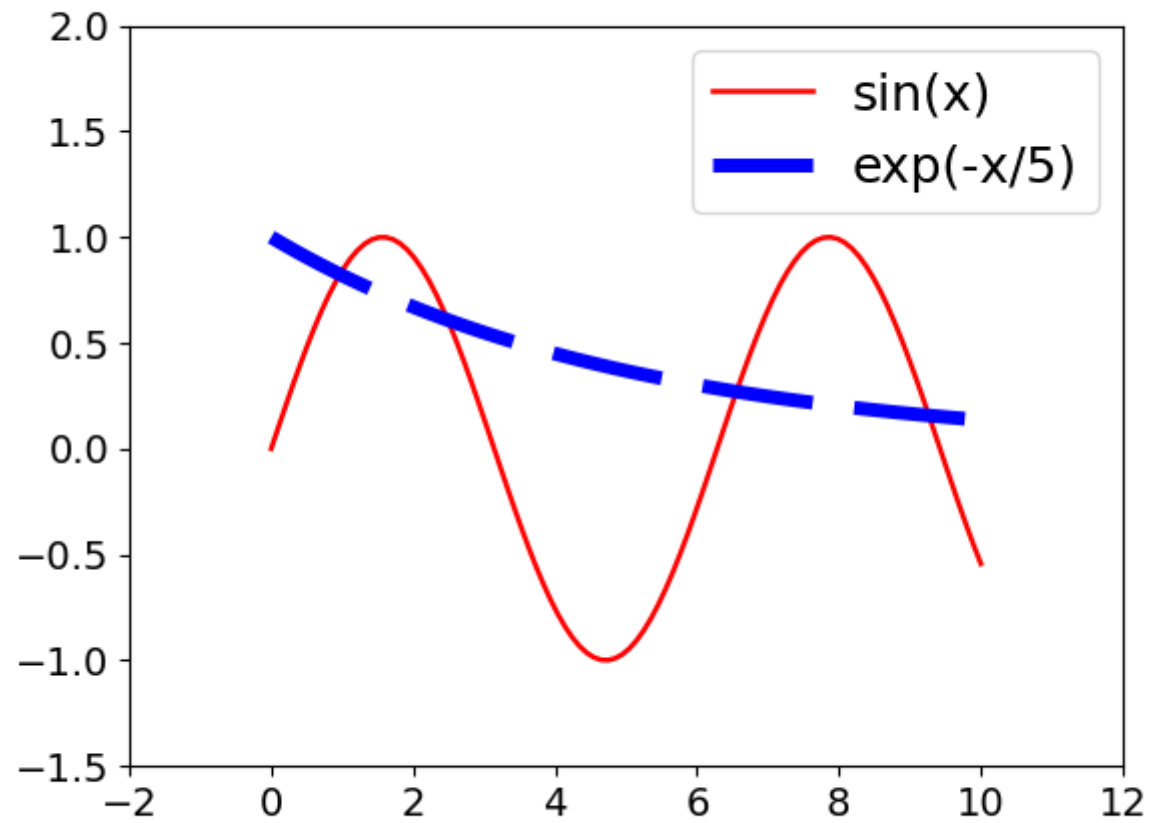
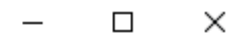
x = np.linspace(0,10,100)
y1 = np.sin(x)
y2 = np.exp(-x/5.)

plt.plot(x, y1, 'r-', lw=1.8, label='sin(x)')
plt.plot(x, y2, 'b-', lw=5, dashes=[8,3], label='exp(-x/5)')

plt.axis([-2, 12, -1.5, 2.0])
plt.legend(loc='upper right') # położenie legendy
plt.show()
```

położenie = best, upper right, upper left, lower right, lower left, center left,
center right, lower center, upper center, center

Figure 1



x=-0.39 y=-0.722

Rozmiar rysunków i zapisywanie pliku

 lec6_m.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_m.py (3.10.7)

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.family'] = 'Times New Roman'    # krój czcionki
plt.rcParams['font.size'] = 26
```

```
plt.figure(figsize=(8, 6.9))    # rozmiar rysunku
```

```
x = np.linspace(0,10,30)
y = np.sin(x)
```

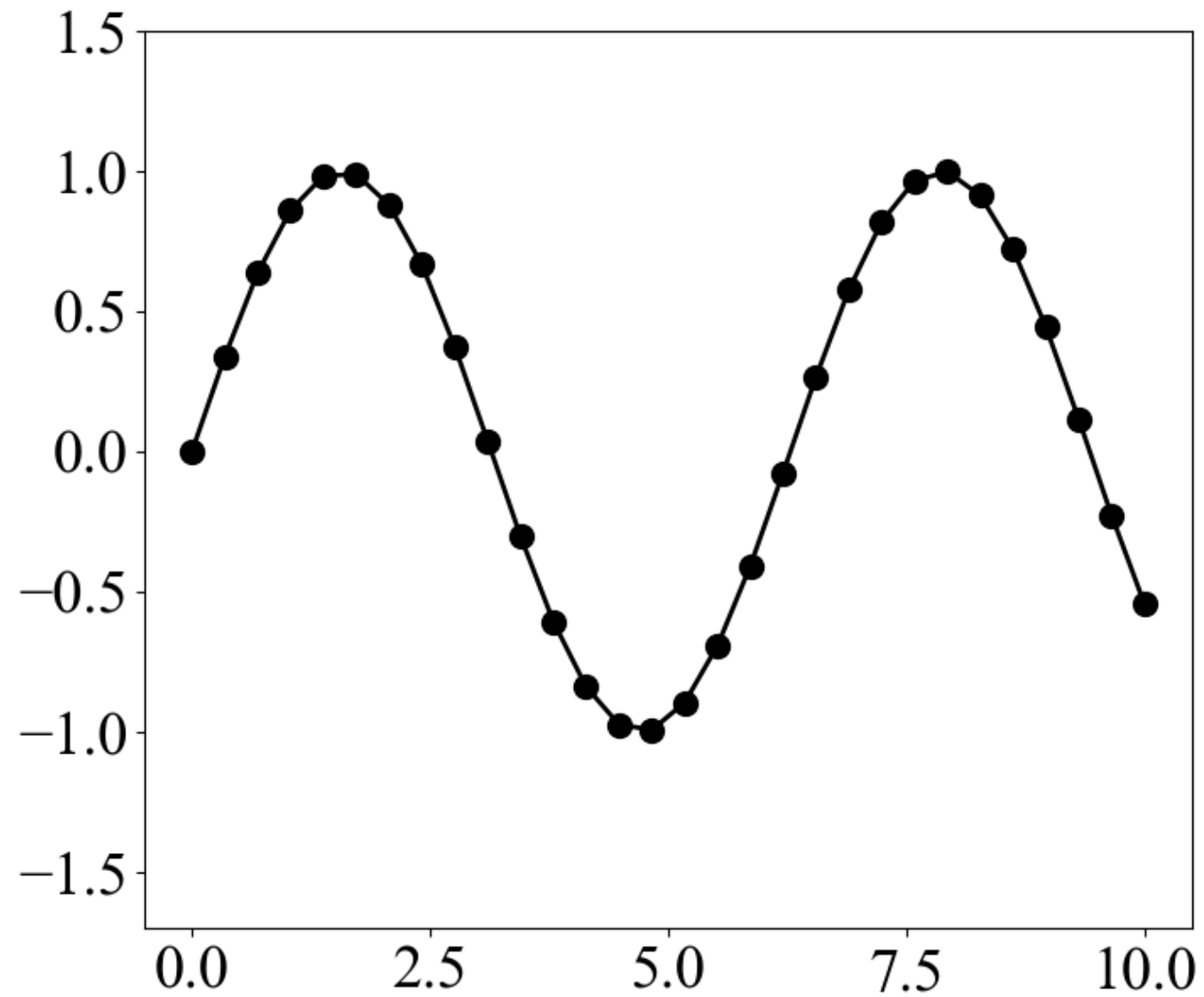
```
plt.plot(x, y, 'o-', color='k', lw=2, ms=10)
```

```
plt.axis([-0.5, 10.5, -1.7, 1.5])
```

```
plt.savefig('myfig.pdf', format='pdf', pad_inches=0.05)
plt.show()
```

lw = line width

ms = marker size



**More Stuff
(if you are interested)**

Algebra w numpy

<https://numpy.org/doc/stable/reference/routines.linalg.html>

numpy.org/doc/stable/reference/routines.linalg.html

ALICE_pp_HEPData Poczta - UJK Byli ubezpieczeni, a... Mathematical expe... Root Commands an... Bazy danych Pseudo random nu... Plan treningowy 3... LaTeX Math Symbols Sejda helps with yo... Marcin Bienkowski... Latex tips and tricks... Słownik online za

NumPy

User Guide API reference Development Release notes Learn

1.23 (stable)

numpy.ctypeslib)

Datetime Support Functions

Data type routines

Optionally SciPy-accelerated routines (numpy.dual)

Mathematical functions with automatic domain

Floating point error handling

Discrete Fourier Transform (numpy.fft)

Functional programming

NumPy-specific help functions

Input and output

Linear algebra (numpy.linalg)

numpy.dot

numpy.linalg.multi_dot

numpy.vdot

numpy.inner

numpy.outer

numpy.matmul

numpy.tensordot

numpy.einsum

numpy.einsum_path

numpy.linalg.matrix_power

numpy.kron

numpy.linalg.cholesky

numpy.linalg.qr

numpy.linalg.svd

numpy.linalg.eig

numpy.linalg.eigh

numpy.linalg.eigvals

numpy.linalg.eigvalsh

numpy.linalg.norm

numpy.linalg.cond

numpy.linalg.det

numpy.linalg.matrix_rank

numpy.linalg.slogdet

numpy.trace

numpy.linalg.solve

numpy.linalg.tensorsolve

numpy.linalg.lstsq

numpy.linalg.inv

Linear algebra (numpy.linalg)

The NumPy linear algebra functions rely on BLAS and LAPACK to provide efficient low level implementations of standard linear algebra algorithms. Those libraries may be provided by NumPy itself using C versions of a subset of their reference implementations but, when possible, highly optimized libraries that take advantage of specialized processor functionality are preferred. Examples of such libraries are OpenBLAS, MKL (TM), and ATLAS. Because those libraries are multithreaded and processor dependent, environmental variables and external packages such as `threadpoolctl` may be needed to control the number of threads or specify the processor architecture.

The SciPy library also contains a `linalg` submodule, and there is overlap in the functionality provided by the SciPy and NumPy submodules. SciPy contains functions not found in `numpy.linalg`, such as functions related to LU decomposition and the Schur decomposition, multiple ways of calculating the pseudoinverse, and matrix transcendental functions such as the matrix logarithm. Some functions that exist in both have augmented functionality in `scipy.linalg`. For example, `scipy.linalg.eig` can take a second matrix argument for solving generalized eigenvalue problems. Some functions in NumPy, however, have more flexible broadcasting options. For example, `numpy.linalg.solve` can handle "stacked" arrays, while `scipy.linalg.solve` accepts only a single square array as its first argument.

Note

The term *matrix* as it is used on this page indicates a 2d `numpy.array` object, and *not* a `numpy.matrix` object. The latter is no longer recommended, even for linear algebra. See the [matrix object documentation](#) for more information.

The @ operator

Introduced in NumPy 1.10.0, the `@` operator is preferable to other methods when computing the matrix product between 2d arrays. The `numpy.matmul` function implements the `@` operator.

Matrix and vector products

<code>dot(a, b[, out])</code>	Dot product of two arrays.
<code>linalg.multi_dot(arrays, *[, out])</code>	Compute the dot product of two or more arrays in a single function call, while automatically selecting the fastest evaluation order.
<code>vdot(a, b, /)</code>	Return the dot product of two vectors.
<code>inner(a, b, /)</code>	Inner product of two arrays.
<code>outer(a, b[, out])</code>	Compute the outer product of two vectors.

Rozwiązywanie równań, wartości własne, wektory własne, etc.

29

na przykład:

 lec6_n.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/

File Edit Format Run Options Window Help

```
import numpy as np

M = np.identity(3)*5
print(M)
print()

print(np.linalg.inv(M))    # macierz odwrotna
print()
print(np.linalg.det(M))    # wyznacznik macierzy
```

```
= RESTART: D:/CERNBox/zajecia,
```

```
[[5. 0. 0.]
 [0. 5. 0.]
 [0. 0. 5.]]
```

```
[[0.2 0.  0. ]
 [0.  0.2 0. ]
 [0.  0.  0.2]]
```

```
124.999999999999994
```

Zapisywanie i odczytywanie w numpy

<https://numpy.org/doc/stable/reference/routines.io.html>

numpy.org/doc/stable/reference/routines.io.html

...

ALICE_pp_HEPData

Poczta - UJK

Byli ubezpieczeni, a...

Mathematical expe...

Root Commands an...

Bazy danych

Pseudo random nu...

Plan treningowy 3-...

Latex Math Symbols

Sejda helps with yo...

Marcin Bienkowski...

Latex tips and tricks...

NumPy

User GuideAPI referenceDevelopmentRelease notesLearn

1.23 (stable)

C-Types Foreign Function Interface (`numpy.ctypeslib`)

Datetime Support Functions

Data type routines

Optionally SciPy-accelerated routines (`numpy.dual`)

Mathematical functions with automatic domain

Floating point error handling

Discrete Fourier Transform (`numpy.fft`)

Functional programming

NumPy-specific help functions

Input and output

`numpy.load`

`numpy.save`

`numpy.savez`

`numpy.savez_compressed`

`numpy.loadtxt`

`numpy.savetxt`

`numpy.genfromtxt`

`numpy.fromregex`

`numpy.fromstring`

`numpy.ndarray.tofile`

`numpy.ndarray.tolist`

`numpy.array2string`

`numpy.array_repr`

`numpy.array_str`

`numpy.format_float_positional`

`numpy.format_float_scientific`

`numpy.memmap`

`numpy.lib.format.open_memmap`

`numpy.set_printoptions`

`numpy.get_printoptions`

`numpy.set_string_function`

`numpy.printoptions`

`numpy.binary_repr`

`numpy.base_repr`

`numpy.DataSource`

`numpy.lib.format`

Linear algebra (`numpy.linalg`)

Logic functions

Input and output

NumPy binary files (NPY, NPZ)

<code>load</code> (file[, mmap_mode, allow_pickle, ...])	Load arrays or pickled objects from <code>.npy</code> , <code>.npz</code> or pickled files.
<code>save</code> (file, arr[, allow_pickle, fix_imports])	Save an array to a binary file in NumPy <code>.npy</code> format.
<code>savez</code> (file, *args, **kwargs)	Save several arrays into a single file in uncompressed <code>.npz</code> format.
<code>savez_compressed</code> (file, *args, **kwargs)	Save several arrays into a single file in compressed <code>.npz</code> format.

The format of these binary file types is documented in `numpy.lib.format`

Text files

<code>loadtxt</code> (fname[, dtype, comments, delimiter, ...])	Load data from a text file.
<code>savetxt</code> (fname, X[, fmt, delimiter, newline, ...])	Save an array to a text file.
<code>genfromtxt</code> (fname[, dtype, comments, ...])	Load data from a text file, with missing values handled as specified.
<code>fromregex</code> (file, regexp, dtype[, encoding])	Construct an array from a text file, using regular expression parsing.
<code>fromstring</code> (string[, dtype, count, like])	A new 1-D array initialized from text data in a string.
<code>ndarray.tofile</code> (fid[, sep, format])	Write array to a file as text or binary (default).
<code>ndarray.tolist</code> ()	Return the array as an <code>a.ndim</code> -levels deep nested list of Python scalars.

Raw binary files

<code>fromfile</code> (file[, dtype, count, sep, offset, like])	Construct an array from data in a text or binary file.
<code>ndarray.tofile</code> (fid[, sep, format])	Write array to a file as text or binary (default).

On this page

NumPy binary files

Text files

Raw binary files

String formatting

Memory mapping

Text formatting op

Base-n representa

Data sources

Binary Format Des

31

na przykład:

```
import numpy as np
```

```
L = np.random.uniform(0,100, (5,2))  
np.savetxt('data.txt', L)
```

```
G = np.loadtxt('data.txt')  
print(G)
```

```
==== RESTART: D:/CERNBox/zajecia,  
[[29.95346542  89.62627773]  
 [73.02800627  89.69336579]  
 [43.88002532  16.60657207]  
 [36.5991981   10.90497709]  
 [ 5.65016692  76.68034952]]
```

 Lister - [d:\CERNBox\zajecia\Inzynieria_danych\Python\moje\wyklad_6\data.txt]

Plik Edytuj Opcje Kodowanie Pomoc

```
2.995346542297998482e+01  8.962627772664065162e+01  
7.302800626784514293e+01  8.969336578880766808e+01  
4.388002532349967311e+01  1.660657206785697326e+01  
3.659919810230033477e+01  1.090497708988283954e+01  
5.650166919055510384e+00  7.668034951565616097e+01
```

piksele

 lec6_p.py - D:/CERNBox/zajecia/Inzynieria_danych/Python/moje/wyklad_6/lec6_p.py (3.10.7) — ☐

File Edit Format Run Options Window Help

```
import numpy as np
import matplotlib.pyplot as plt
```

```
plt.rcParams['font.size'] = 14
```

```
x = np.random.uniform(-10, 10, 100000)
```

```
y = np.random.uniform(-10, 10, 100000)
```

```
plt.plot(x, y, 'r,') # przecinek oznacza kolorowanie pojedynczych pikseli
```

```
plt.axis([-11,11,-11,11])
```

```
plt.show()
```

Figure 1

