Programowanie w Pythonie
Biblioteka Matplotlib

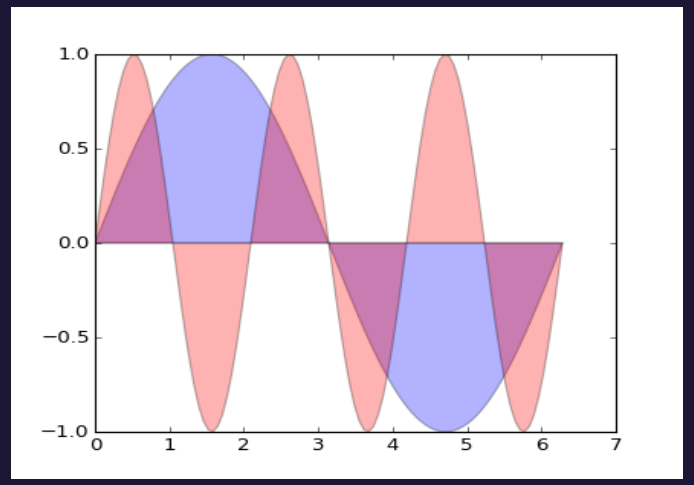
Łukasz Mioduszewski, UKSW 2022

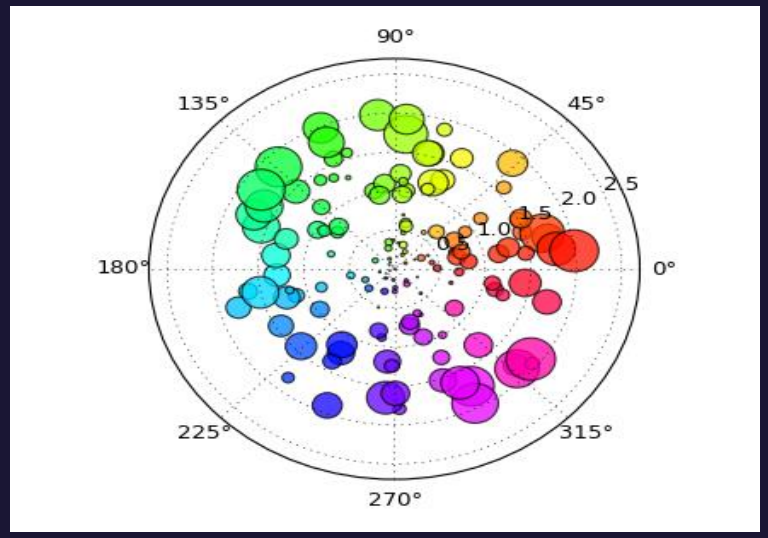
Uwaga do zadania domowego

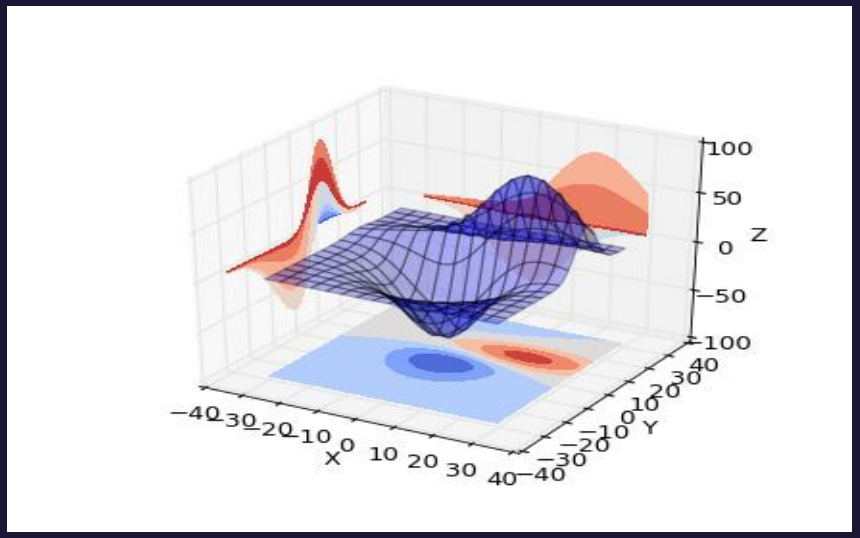
- Komunikaty o błędach w Pythonie są zwykle przydatne
- jeśli je łapiemy to zwykle i tak warto je wyświetlić
- Nie zawsze błąd będzie taki jak przewidujemy

Wizualizacja danych

- Dane można zwykle zwizualizować na wiele sposobów
  - Wykresy, histogramy, diagramy, mapy, grafy…
- Narzędzia do wizualizacji danych to umożliwiają
- Często zdarza się że danych jest bardzo dużo i trzeba na ich podstawie podejmować ważne decyzje – wykres ma w tym pomóc
  - Cel: przekształcenie skomplikowanych danych w prosty obrazek







Matplotlib

- **Matplotlib** to jedno z najpotężniejszych narzędzi do wizualizacji danych w Pythonie
- **Matplotlib** to biblioteka do tworzenia wykresów 2-D
  - Łatwo znaleźć przykłady
- Aby zaimportować **`matplotlib`** do skryptu

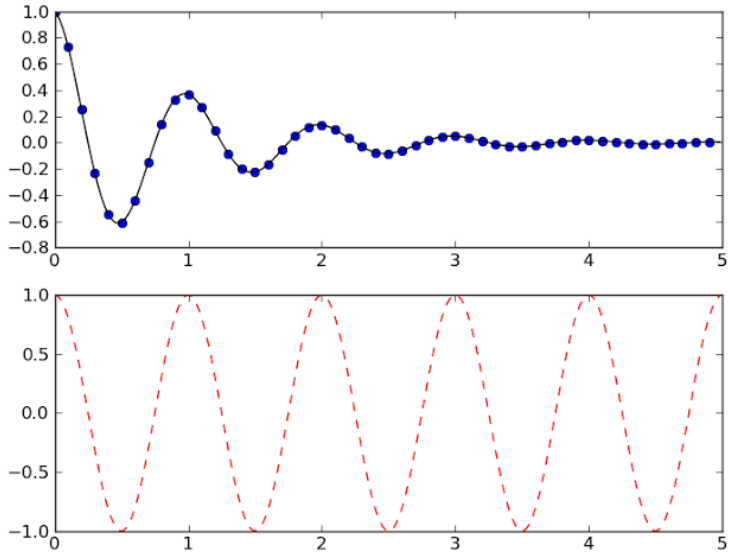
```
import matplotlib.pyplot as plt
```

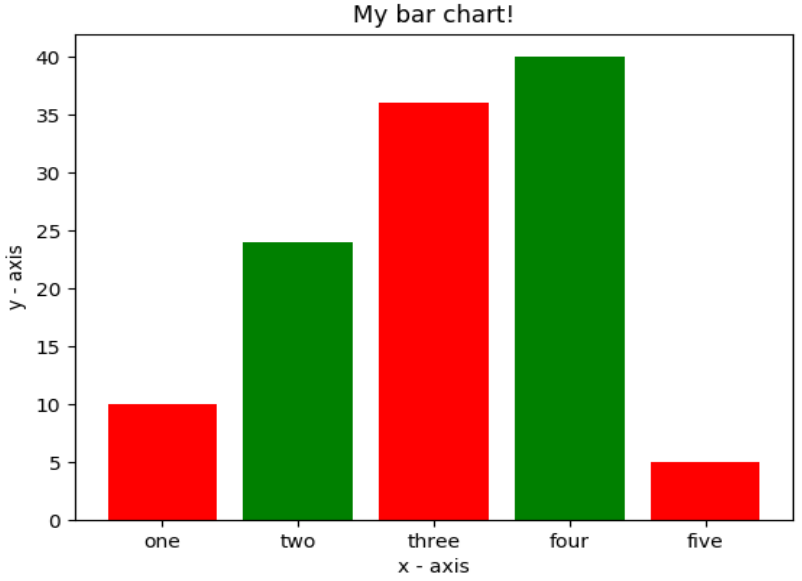
- Aby zainstalować matplotlib
  - Najprościej przy użyciu **`pip`**
  - **`pip install matplotlib`**

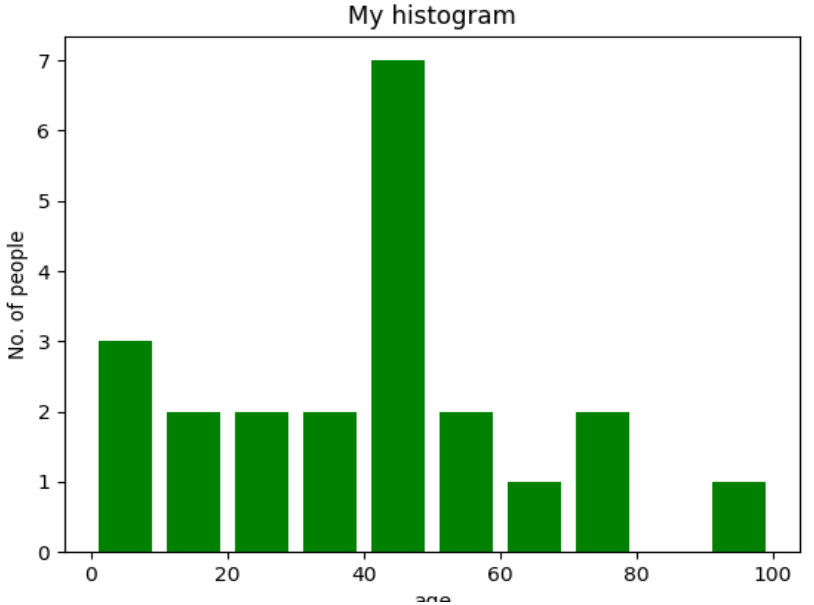
matplotlib

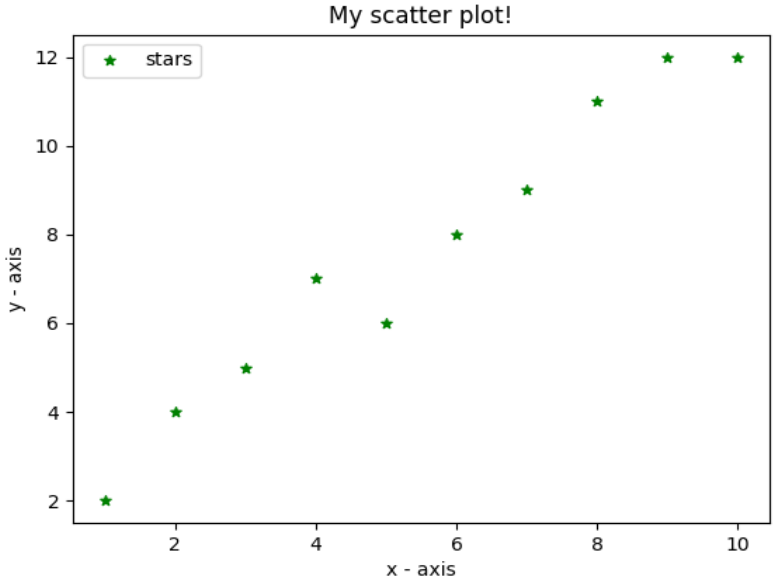
- Naśladuje MATLAB, **`matplotlib.pyplot`** to zestaw poleceń takich jak w programie MATLAB
- Każde polecenie **`pyplot`** zmienia w jakiś sposób obrazek, np.
  - tworzy obrazek (figure)
  - tworzy obszar do rysowania na obrazku
  - rysuje wykres na tym obszarze
  - wzbogaca wykres o podpisy osi itp.
- **Uwaga** stan wykresu jest zachowywany między poleceniami
- Dwie linijki występują praktycznie zawsze:
  - Rodzaj wykresu (Czy to wykres liniowy, słupkowy, kołowy itp.)
  - Wyświetlenie wykresu (w jaki sposób ma być wyświetlony/zapisany)

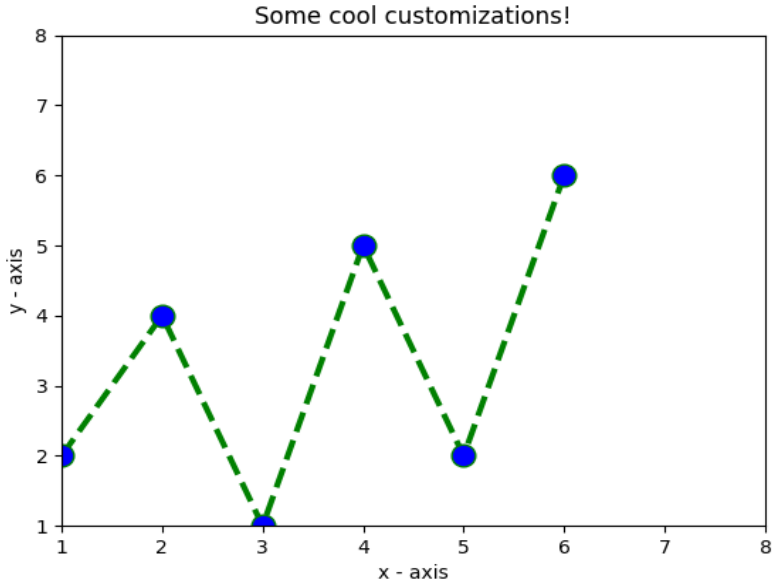
Parę przykładów

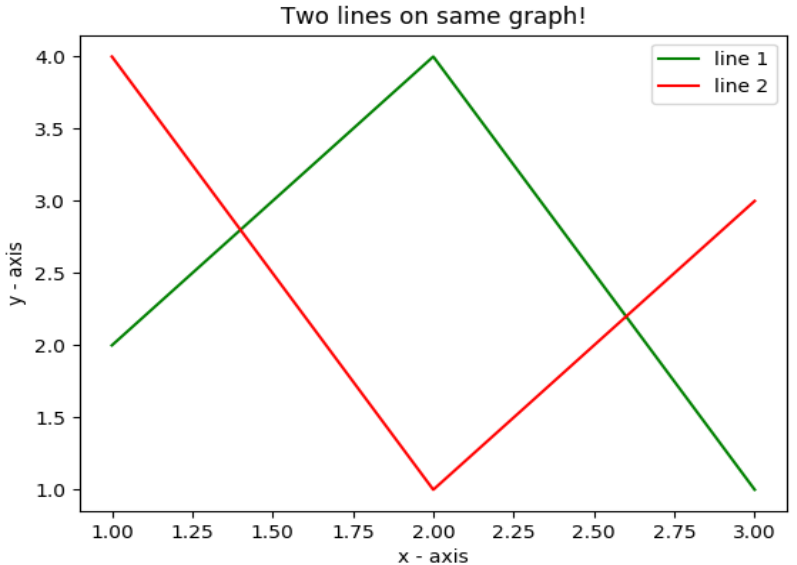






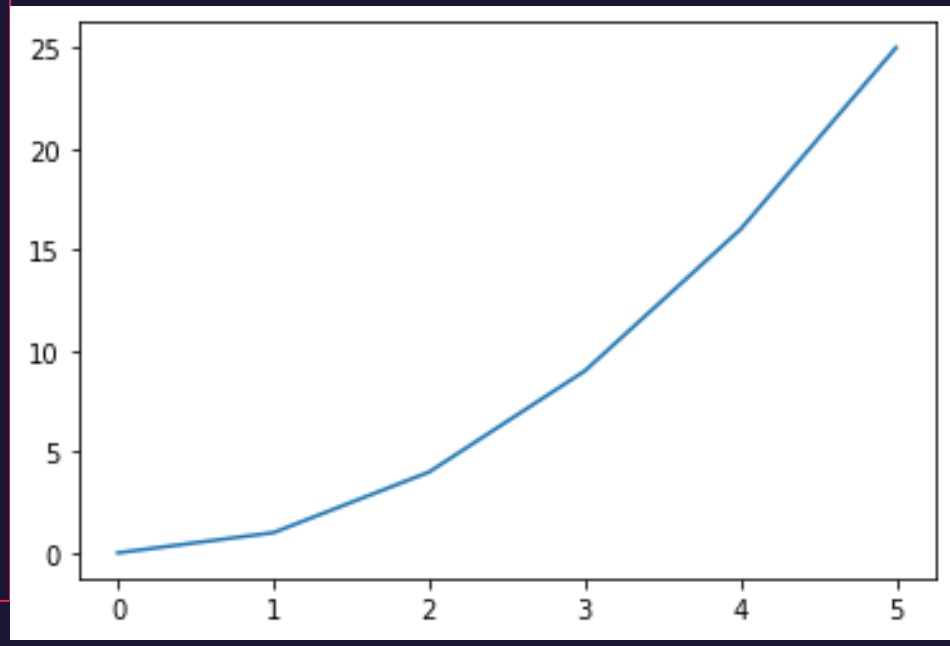






Wykres liniowy

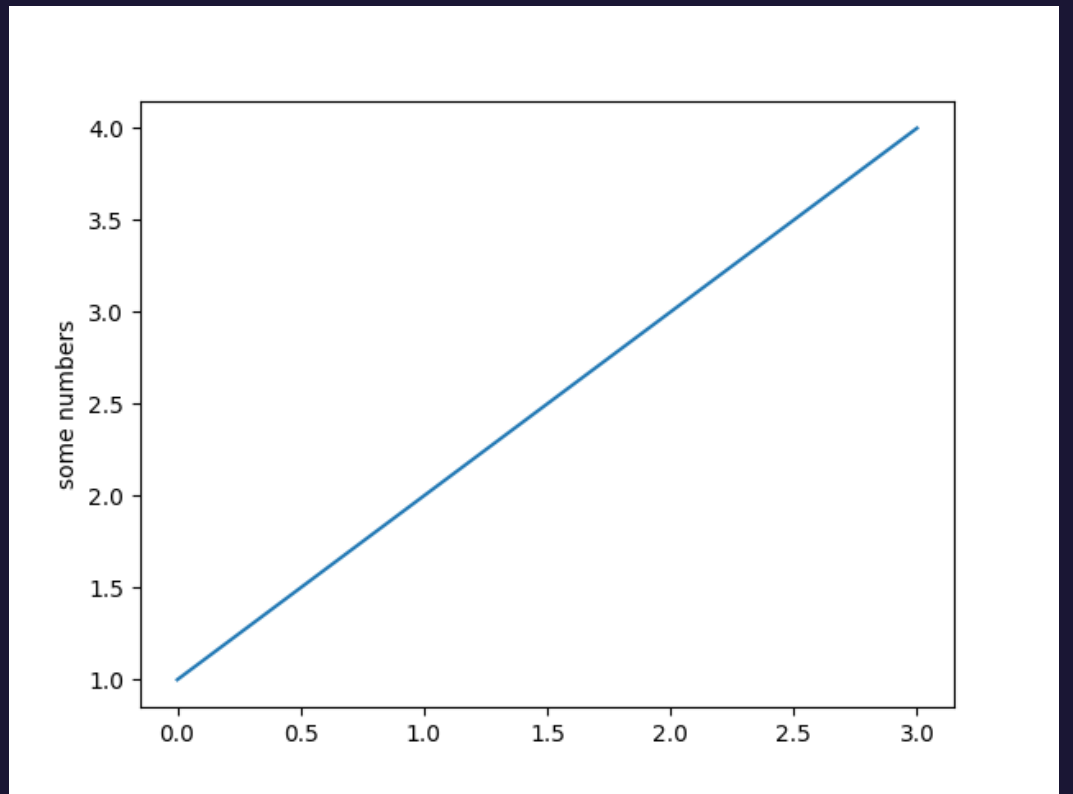
```
import matplotlib.pyplot as plt

# Dane do narysowania
x_values = [0, 1, 2, 3, 4, 5 ]
y_values = [0, 1, 4, 9, 16,25]

# domyślnie polecenie plot tworzy linie
plt.plot(x_values, y_values)

# wyświetl wykres
plt.show()
```



Wykres liniowy

- Jeśli polecenie **`plot()` dostanie tylko jedną listę**
- zostanie uznana za listę wartości y
- wartości x wygenerują się automatycznie (0, 1, 2, 3…)
- wartości x zawsze jest tyle samo co y

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()
```

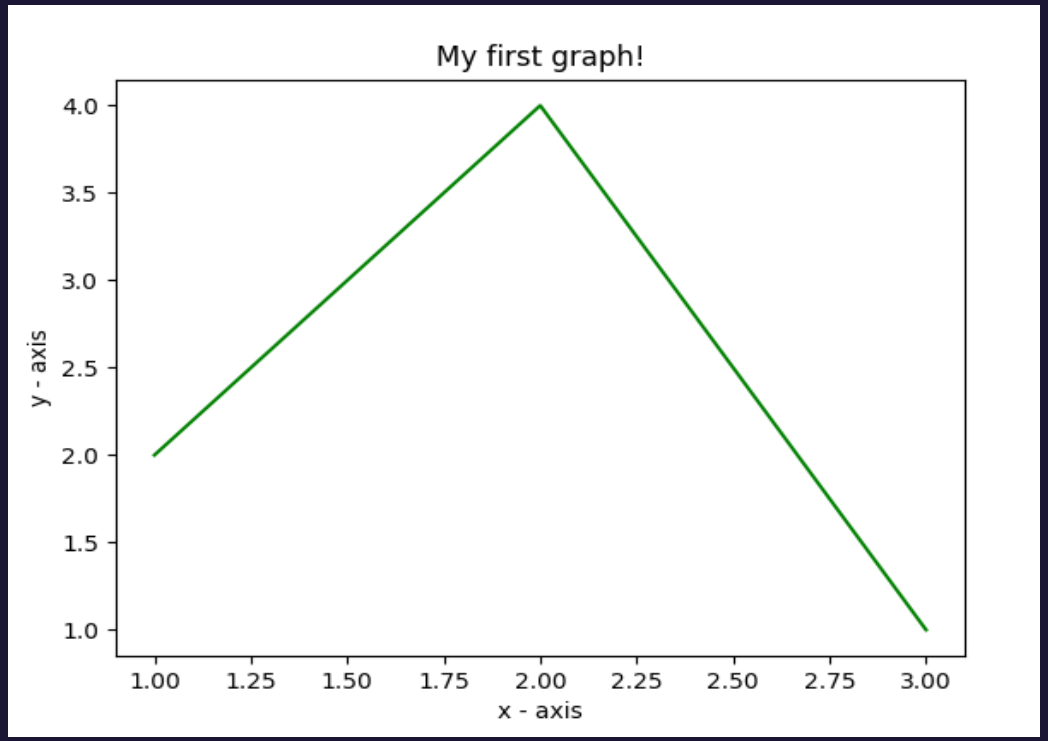


Polecenia dla pyplot lub dla axes

- `text()` : dodaj tekst w wybranym miejscu
- `xlabel()`: podpis osi x
- `ylabel()`: podpis osi y
- `title()` : tytuł wykresu
- `clear()` : wyczyść wykres
  - jeśli wywołam plot() wiele razy, plt.show() narysuje wszystko
- `savefig()`: zapisz wykres do pliku
- `legend()` : pokaż legendę wykresu

Przepis na wykres

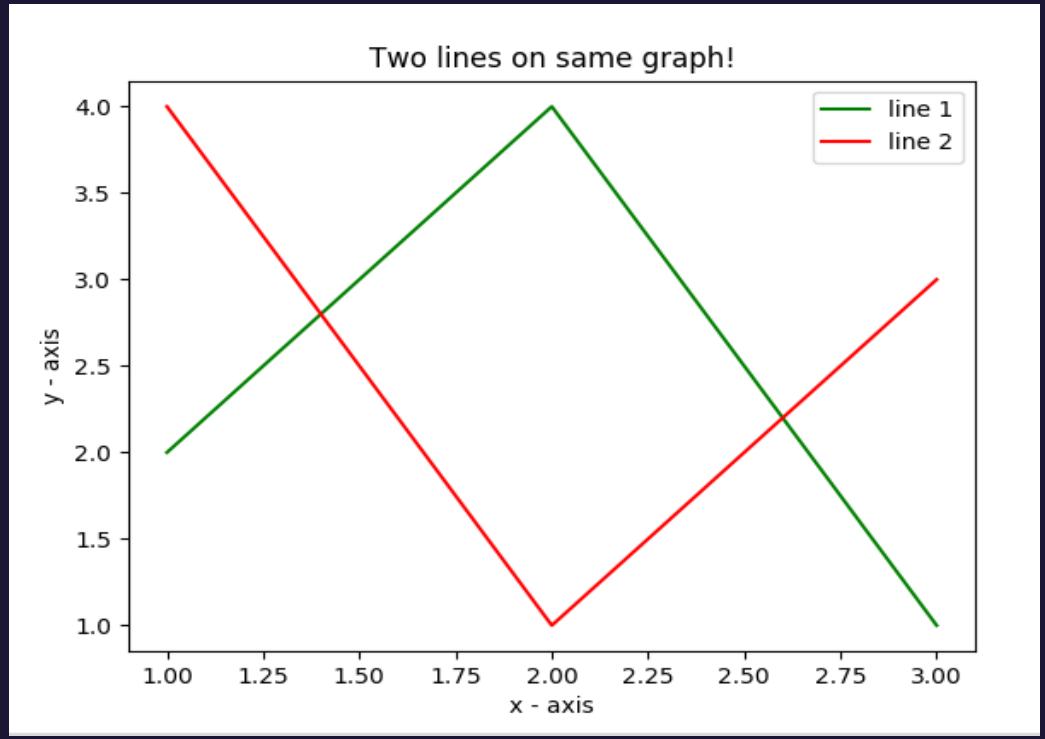
```
# importing the required module
import matplotlib.pyplot as plt

# x axis values
x = [1,2,3]
# corresponding y axis values
y = [2,4,1]

# plotting the points
plt.plot(x, y)

# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')

# giving a title to my graph
plt.title('My first graph!')

# function to show the plot
plt.show()
```



- Zdefiniuj wartości x oraz y jako listy
- Narysuj wykres poleceniem plot()
- Nazwij osie używając xlabel() oraz ylabel()
- Nazwij wykres używając title()
- Pokaż wykres używając show()

Wiele krzywych

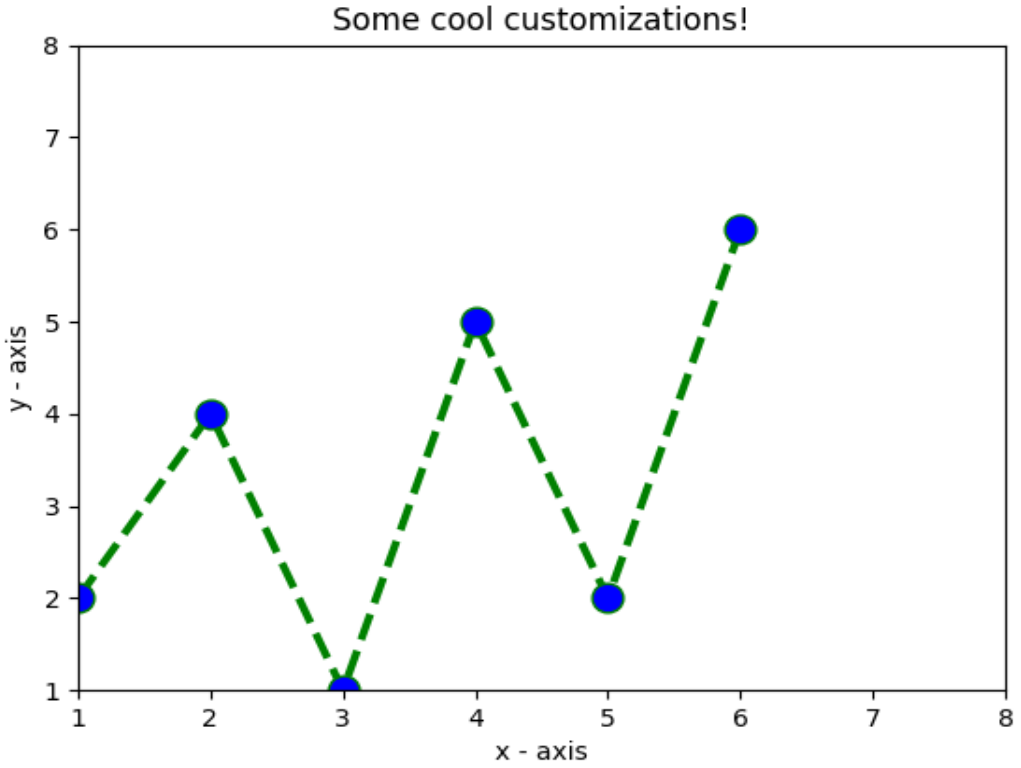
```
import matplotlib.pyplot as plt
# line 1 points
x1 = [1,2,3]
y1 = [2,4,1]
# plotting the line 1 points
plt.plot(x1, y1, label="line 1")
# line 2 points
x2 = [1,2,3]
y2 = [4,1,3]
# plotting the line 2 points
plt.plot(x2, y2, label = "line 2")
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('Two lines on same graph!')
# show a legend on the plot
plt.legend()
# function to show the plot
plt.show()
```



- Nazwa krzywej jest parametrem polecenia plot()
- Legenda wykresu wyświetla nazwy krzywych

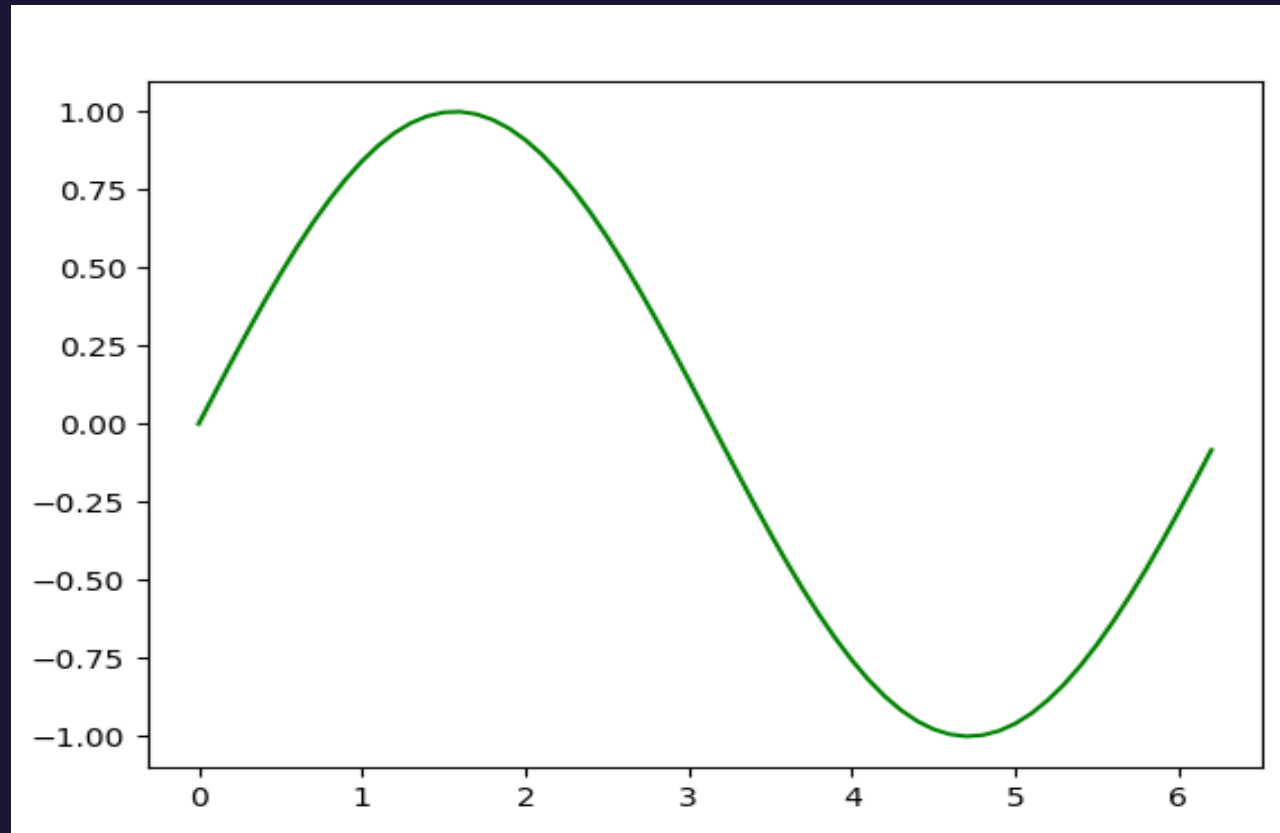
Tuning

```
import matplotlib.pyplot as plt
# x axis values
x = [1,2,3,4,5,6]
# corresponding y axis values
y = [2,4,1,5,2,6]
# plotting the points
plt.plot(x, y, color='green', linestyle='dashed',
linewidth = 3,
         marker='o', markerfacecolor='blue',
markersize=12)
# setting x and y axis range
plt.ylim(1,8)
plt.xlim(1,8)
# naming the x axis
plt.xlabel('x - axis')
# naming the y axis
plt.ylabel('y - axis')
# giving a title to my graph
plt.title('Some cool customizations!')
# function to show the plot
plt.show()
```



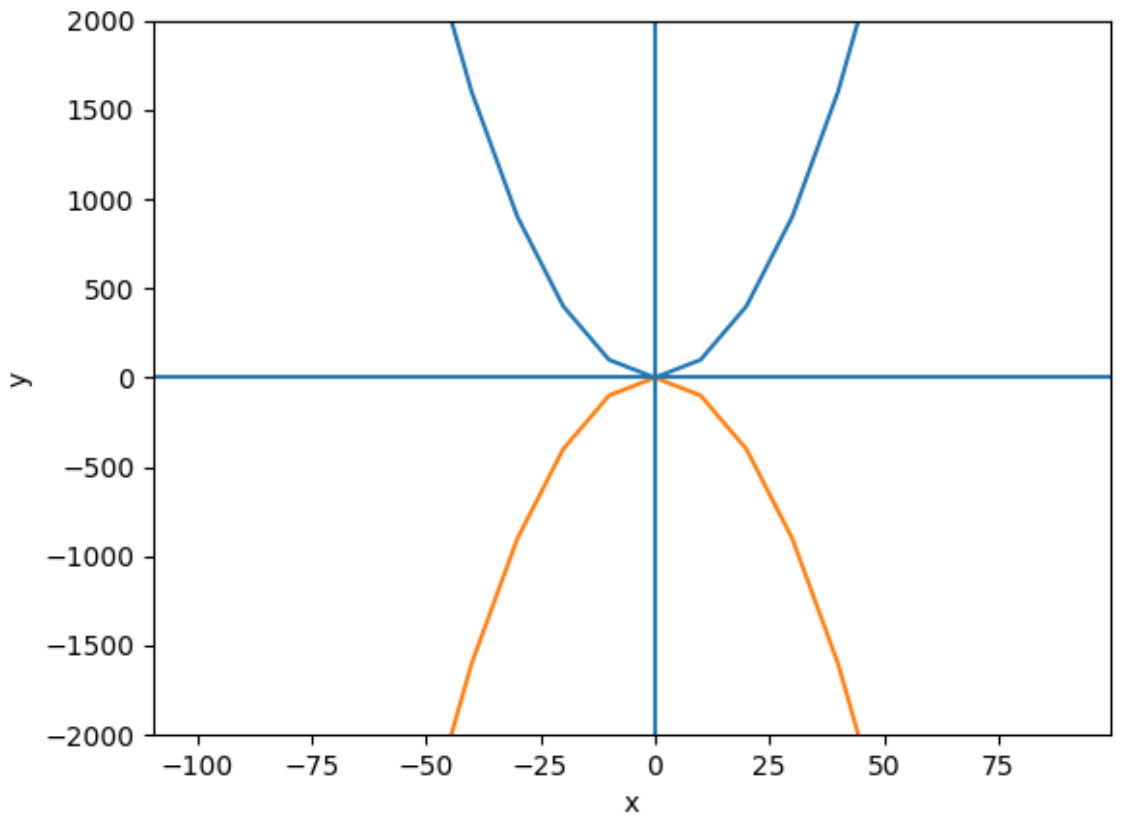
Rysowanie równań

```
# importing the required modules
import matplotlib.pyplot as plt
import numpy as np

# setting the x - coordinates
x = np.arange(0, 2*(np.pi), 0.1)
# setting the y - coordinates
y = np.sin(x)

# potting the points
plt.plot(x, y)

# function to show the plot
plt.show()
```



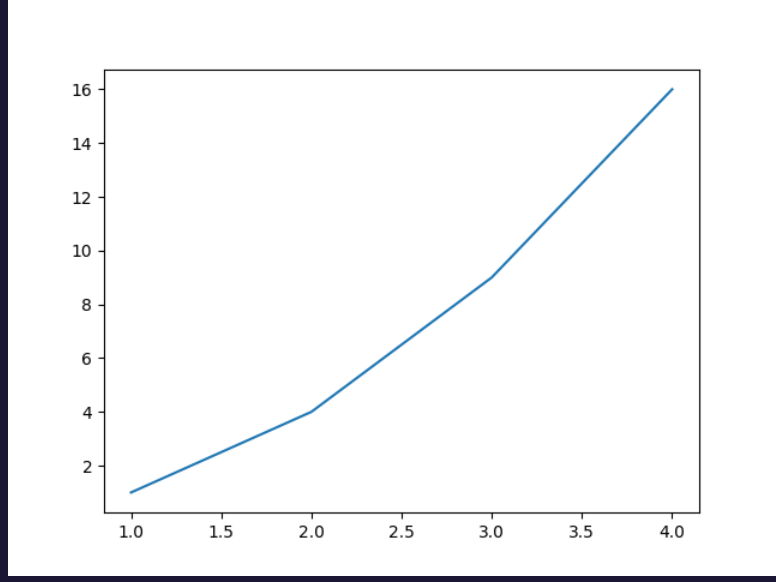
Przykład z Graph Plotting in Python | Set 1

```
import matplotlib.pyplot as plt
y1 =[]
y2 =[] # drugi zbiór wartości y
x = range(-100,100,10)
for i in x: y1.append(i**2)
for i in x: y2.append(-i**2)

plt.plot(x, y1)
plt.plot(x, y2)
plt.xlabel("x") # podpis osi x
plt.ylabel("y") # podpis osi y
plt.ylim(-2000, 2000) # granice wykresu
plt.axhline(0) # pozioma linia w zerze
plt.axvline(0) # pionowa linia w zerze

plt.savefig("kwadratowe.png") # zapisz obrazek do pliku

plt.show() # pokaż obrazek
```



Czy wykres jest zmienną?

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [1, 4, 9, 16]

plt.plot(x, y)
```



to polecenie zwraca obiekt, ale zwykle go nie potrzebujemy

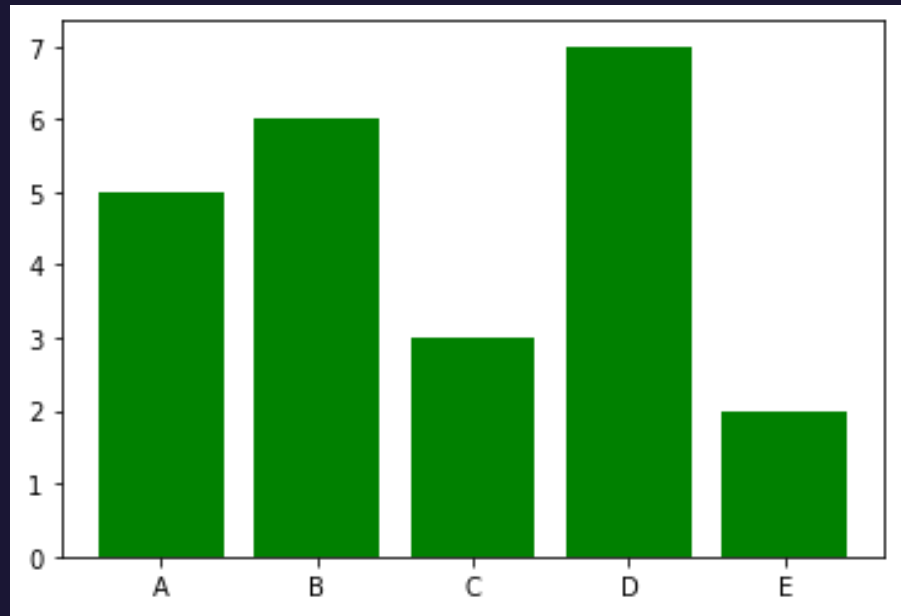
- Zmienna obrazka jest ukryta, można ją dostać przez gcf(), a osie przez gca()
- Jedyna zaleta tego rozwiązania to zgodność z poleceniami MATLAB
- żeby coś narysować i tak potrzebujemy gca(), patrz przykład na następnym slajdzie

```
import matplotlib.pyplot as plt
import numpy as np
t = np.arange(0.0, 2.0, 0.01) # wartości x
s1 = np.sin(2*np.pi*t)        # wartości y
s2 = np.sin(4*np.pi*t)        # drugie wartości y
plt.figure(1)          # pierwszy obrazek
plt.subplot(211)       # pierwszy wykres z dwóch, pozycja 1,1
plt.plot(t, s1)
plt.subplot(212)       # drugi wykres z dwóch, pozycja 1,2
plt.plot(t, 2*s1)
plt.figure(2)          # drugi obrazek
plt.plot(t, s2)
fig2=plt.gcf()         # get current figure zapisuje drugi obrazek do zmiennej
plt.figure(1)          # wracamy do pierwszego obrazka
plt.subplot(211)       # wybieramy pierwszy wykres
plt.plot(t, s2, 's') # rysujemy drugą krzywą na 1. wykresie 1. obrazka
ax = plt.gca()                 # biorę osie z 1. wykresu
ax.set_xticklabels([])         # żeby nie było podpisów wartości osi x
fig2.gca().plot(t,np.sin(t))  # fig2.plot() by nie działało
plt.show()             # ten kod jest do ściągnięcia na Moodle
```

Wykresy słupkowe

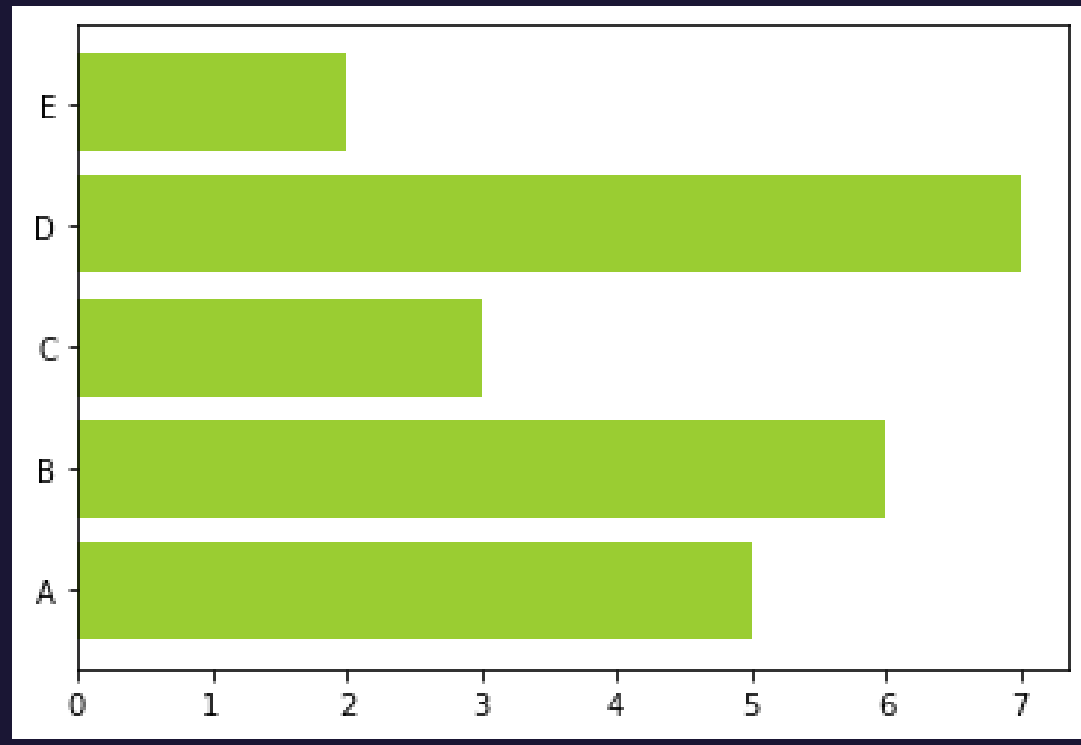
```
import matplotlib.pyplot as plt
values = [5, 6, 3, 7, 2]
names  = ["A", "B", "C", "D", "E"]

plt.bar(names, values, color="green")
plt.show()
```

- Aby narysować wykres słupkowy, zamiast plt.plot() używamy plt.bar()

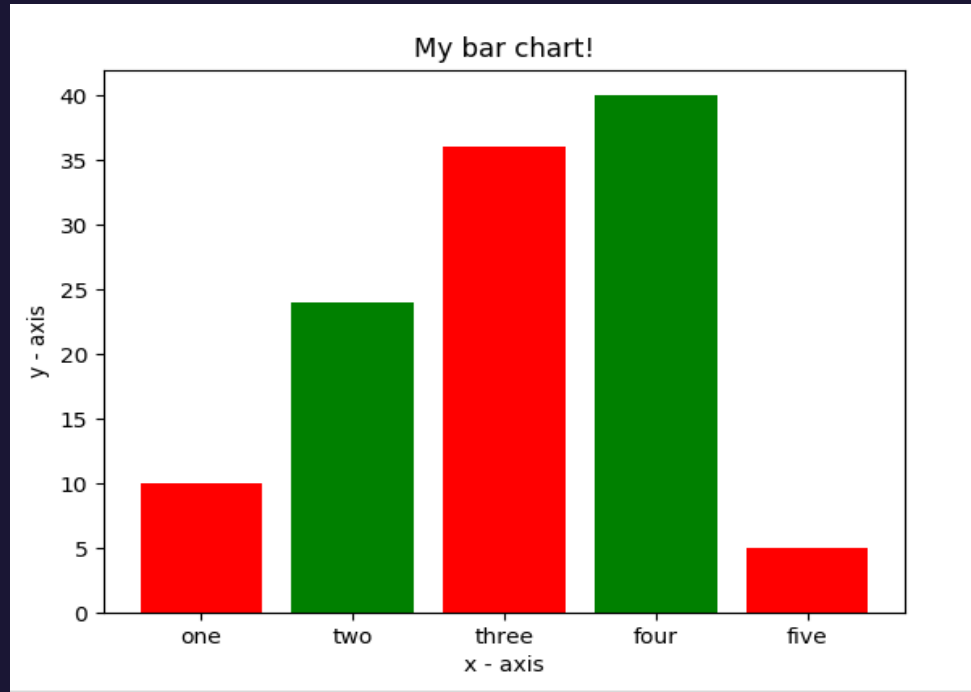


Wykresy słupkowe – wersja pozioma

```
import matplotlib.pyplot as plt
values = [5,6,3,7,2]
names  = ["A", "B", "C", "D", "E"]

# Adding an "h" after bar will flip the graph
plt.barh(names, values,
color="yellowgreen")
plt.show()
```



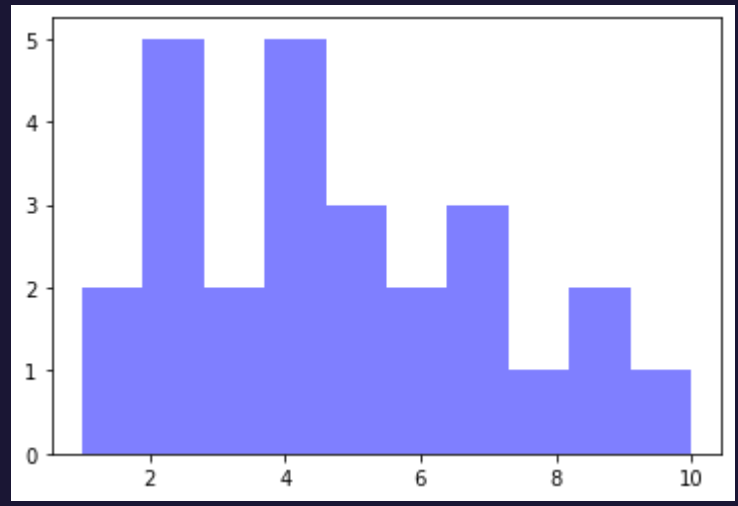
```
import matplotlib.pyplot as plt
# heights of bars
height = [10, 24, 36, 40, 5]
# labels for bars
names = ['one','two','three','four','five']
# plotting a bar chart
c1 =['red', 'green']
c2 =['b', 'g'] # we can use this for color
plt.bar(names, height, width=0.8, color=c1)
# naming the x-axis
plt.xlabel('x - axis')
# naming the y-axis
plt.ylabel('y - axis')
# plot title
plt.title('My bar chart!')
# function to show the plot
plt.show()
```



Histogram

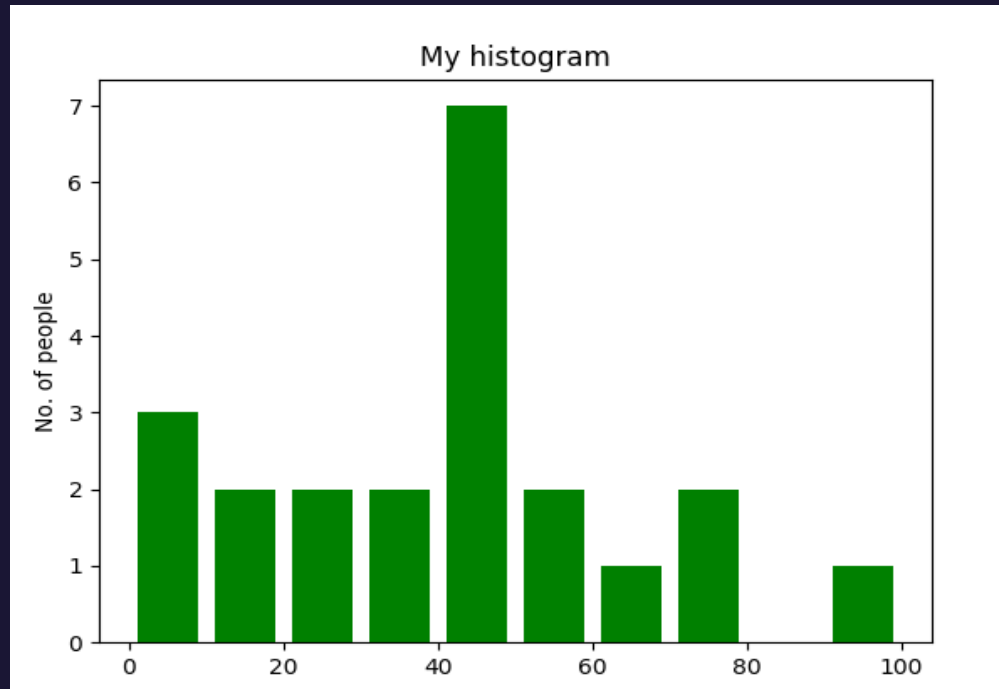
```
import matplotlib.pyplot as plt
#generate fake data
x = [2,1,6,4,2,4,8,9,4,2,4,10,6,4,5,7,7,3,2,7,5,3,5,9,2,1]
#plot for a histogram
plt.hist(x, bins = 10, color='blue', alpha=0.5)
plt.show()
```

- Polecenie hist różni się od bar tym że ma dwa argumenty więcej:
  - **Bins** — ile ma być słupków
  - **Alpha** — słupki mogą być półprzezroczyste (to działa także w bar() i w plot())

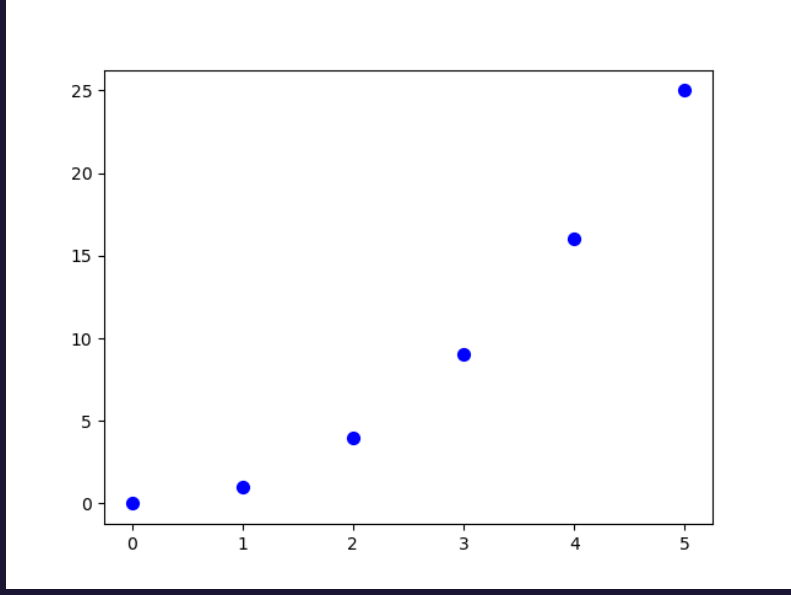


Histogram

```
import matplotlib.pyplot as plt
# frequencies
ages=[2,5,70,40,30,45,50,45,43,40,44,60,7,13,57,18,90,77,32,21,20,40]
# setting the ranges and no. of intervals
range = (0, 100)
bins = 10
# plotting a histogram
plt.hist(ages, bins, range,
color='green',histtype='bar',rwidth=0.8)
# x-axis label
plt.xlabel('age')
# frequency label
plt.ylabel('No. of people')
# plot title
plt.title('My histogram')
plt.show()
```

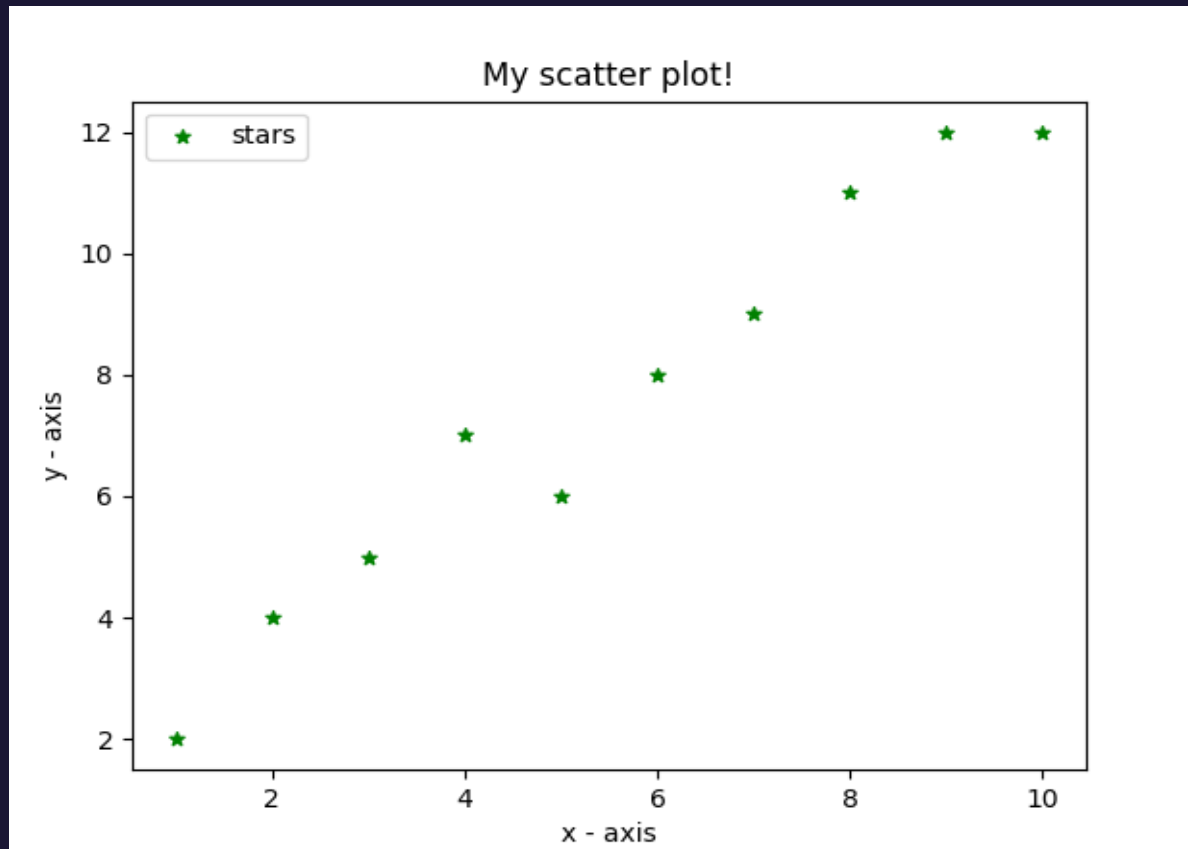


Wykresy punktowe



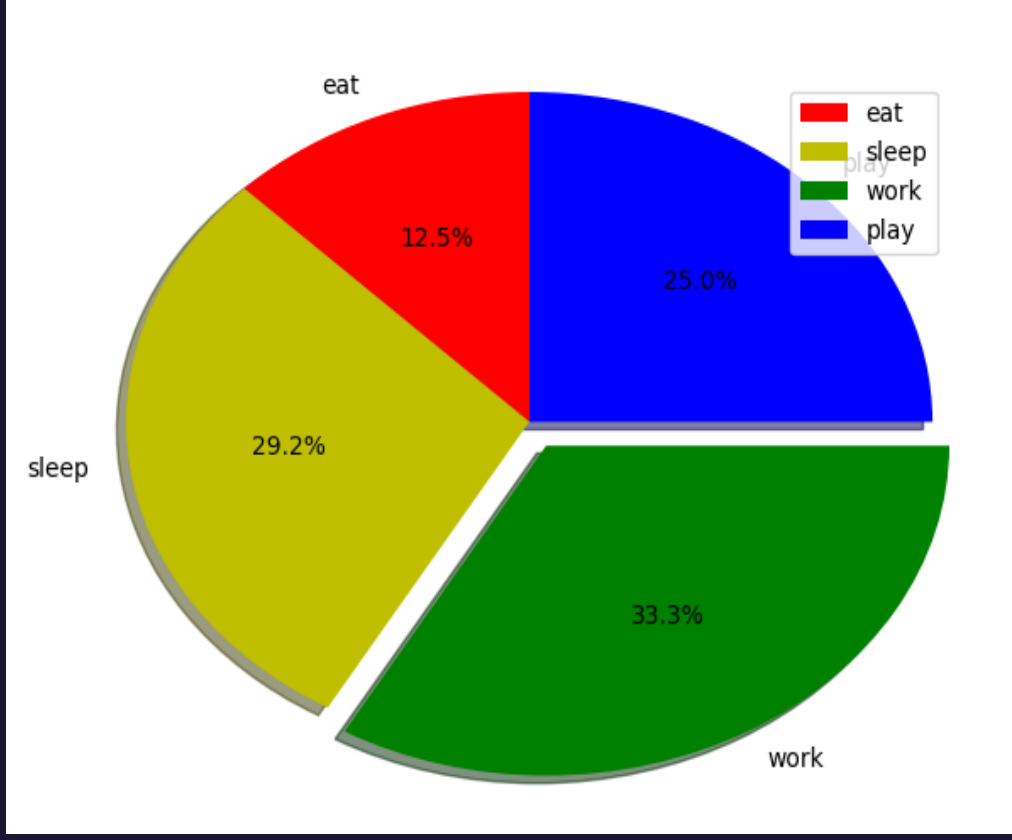

```
import matplotlib.pyplot as plt

#create data for plotting

x_values = [0,1,2,3,4,5]
y_values = [0,1,4,9,16,25]

plt.scatter(x_values, y_values, s=30, color="blue")
plt.show()
```

```python
import matplotlib.pyplot as plt
# x-axis values
x = [1,2,3,4,5,6,7,8,9,10]
# y-axis values
y = [2,4,5,7,6,8,9,11,12,12]
# plotting points as a scatter plot
plt.scatter(x, y, label= "stars", color="green", marker="*", s=30)
# x-axis label
plt.xlabel('x - axis')
# frequency label
plt.ylabel('y - axis')
# plot title
plt.title('My scatter plot!')
# showing legend
plt.legend()
# function to show the plot
plt.show()
```



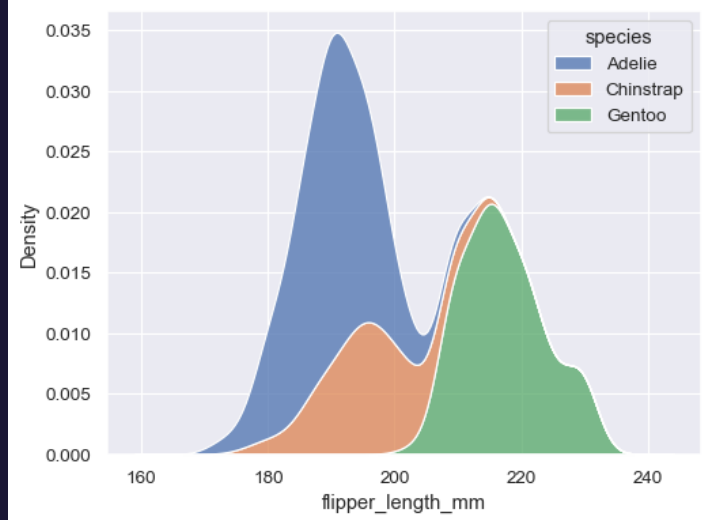
Wykres kołowy

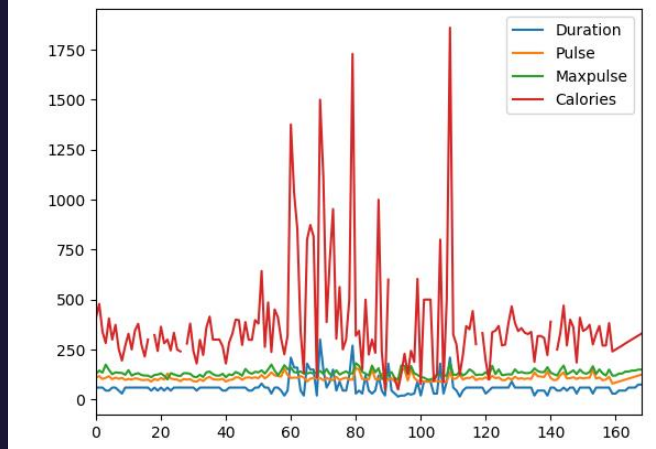
```
import matplotlib.pyplot as plt
# defining labels
activities = ['eat', 'sleep',
'work', 'play']
# portion covered by each label
slices = [3, 7, 8, 6]
# color for each label
colors = ['r', 'y', 'g', 'b']
# plotting the pie chart
plt.pie(slices, labels = activities, colors=colors,
        startangle=90, shadow = True, explode = (0, 0, 0.1, 0),
        radius = 1.2, autopct = '%1.1f%%')
# plotting legend
plt.legend()
# showing the plot
plt.show()
```

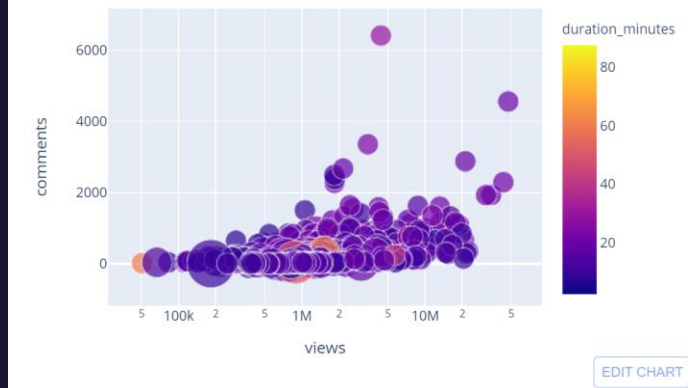


Alternatywy

- seaborn
  - https://seaborn.pydata.org/
- pandas plot (pandas.DataFrame.plot)
  - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.plot.html
- plotly (Plotly Python Open Source Graphing Library)
  - https://plotly.com/python/







Bibliografia

- Matplotlib https://matplotlib.org/index.html
- matplotlib.pyplot https://matplotlib.org/3.2.1/api/pyplot_summary.html
- Tutorial https://matplotlib.org/tutorials/index.html
- Przykłady https://matplotlib.org/gallery/index.html
- Książka https://www.packtpub.com/big-data-and-business-intelligence/mastering-matplotlib
- Slajdy https://www.just.edu.jo/~zasharif/Web/SE412/Slides/matplotlib.pptx
- Kurs widzenia komputerowego https://www.cs.cornell.edu/courses/cs4670/2018sp/