

# Python Praca z Danymi



# Python Praca z Danymi

## Agenda



1

**Flow pracy z danymi**

2

Biblioteki do pracy z danymi

3

Biblioteka NumPy

4

Biblioteka Pandas

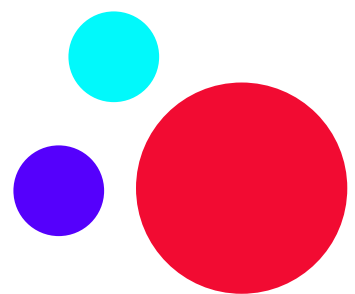
5

Podstawy wizualizacji w matplotlib i seaborn

6

Środowisko do analizy danych





# Python Praca z Danymi

## Agenda

- 1 Flow pracy z danymi
- 2 **Biblioteki do pracy z danymi**
- 3 Biblioteka NumPy
- 4 Biblioteka Pandas
- 5 Podstawy wizualizacji w matplotlib i seaborn
- 6 Środowisko do analizy danych

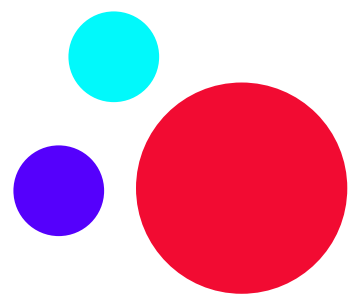


# Python Praca z Danymi

## Agenda



- 1 Flow pracy z danymi
- 2 Biblioteki do pracy z danymi
- 3 **Biblioteka NumPy**
- 4 Biblioteka Pandas
- 5 Podstawy wizualizacji w matplotlib i seaborn
- 6 Środowisko do analizy danych



# Python Praca z Danymi

## Agenda



- 1 Flow pracy z danymi
- 2 Biblioteki do pracy z danymi
- 3 Biblioteka NumPy
- 4 **Biblioteka Pandas**
- 5 Podstawy wizualizacji w matplotlib i seaborn
- 6 Środowisko do analizy danych





# Python Praca z Danymi

## Agenda

- 1 Flow pracy z danymi
- 2 Biblioteki do pracy z danymi
- 3 Biblioteka NumPy
- 4 Biblioteka Pandas
- 5 **Podstawy wizualizacji w matplotlib i seaborn**
- 6 Środowisko do analizy danych

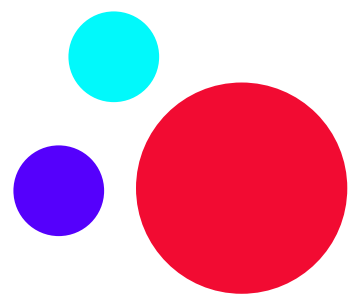




# Python Praca z Danymi

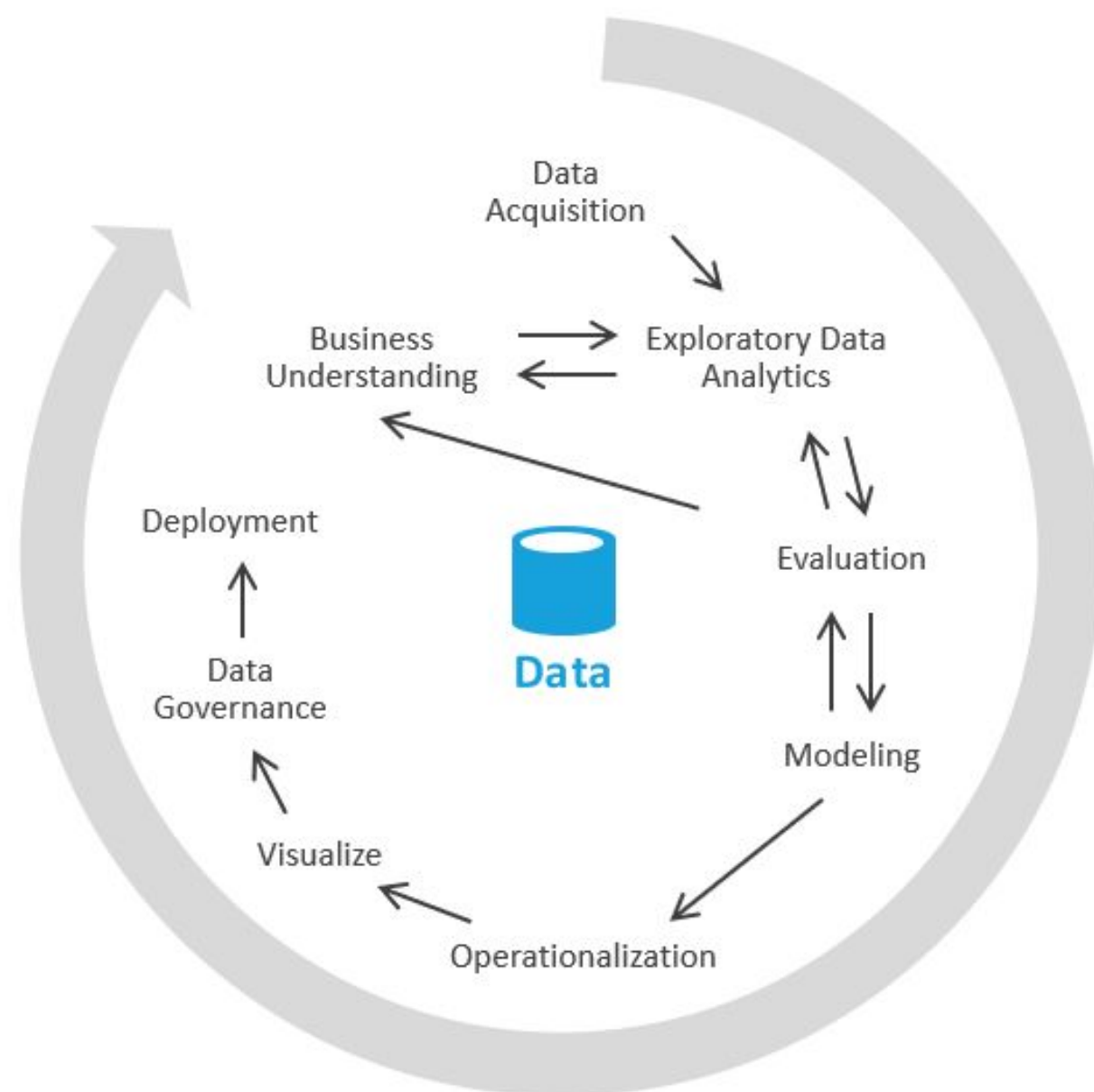
## Agenda

- 1 Flow pracy z danymi
- 2 Biblioteki do pracy z danymi
- 3 Biblioteka NumPy
- 4 Biblioteka Pandas
- 5 Podstawy wizualizacji w matplotlib i seaborn
- 6 **Środowisko do analizy danych**



# Python Praca z Danyymi

## Flow pracy z danymi



infoShare  
ACADEMY





# Python Praca z Danymi

Zbieranie danych



Web Scraping

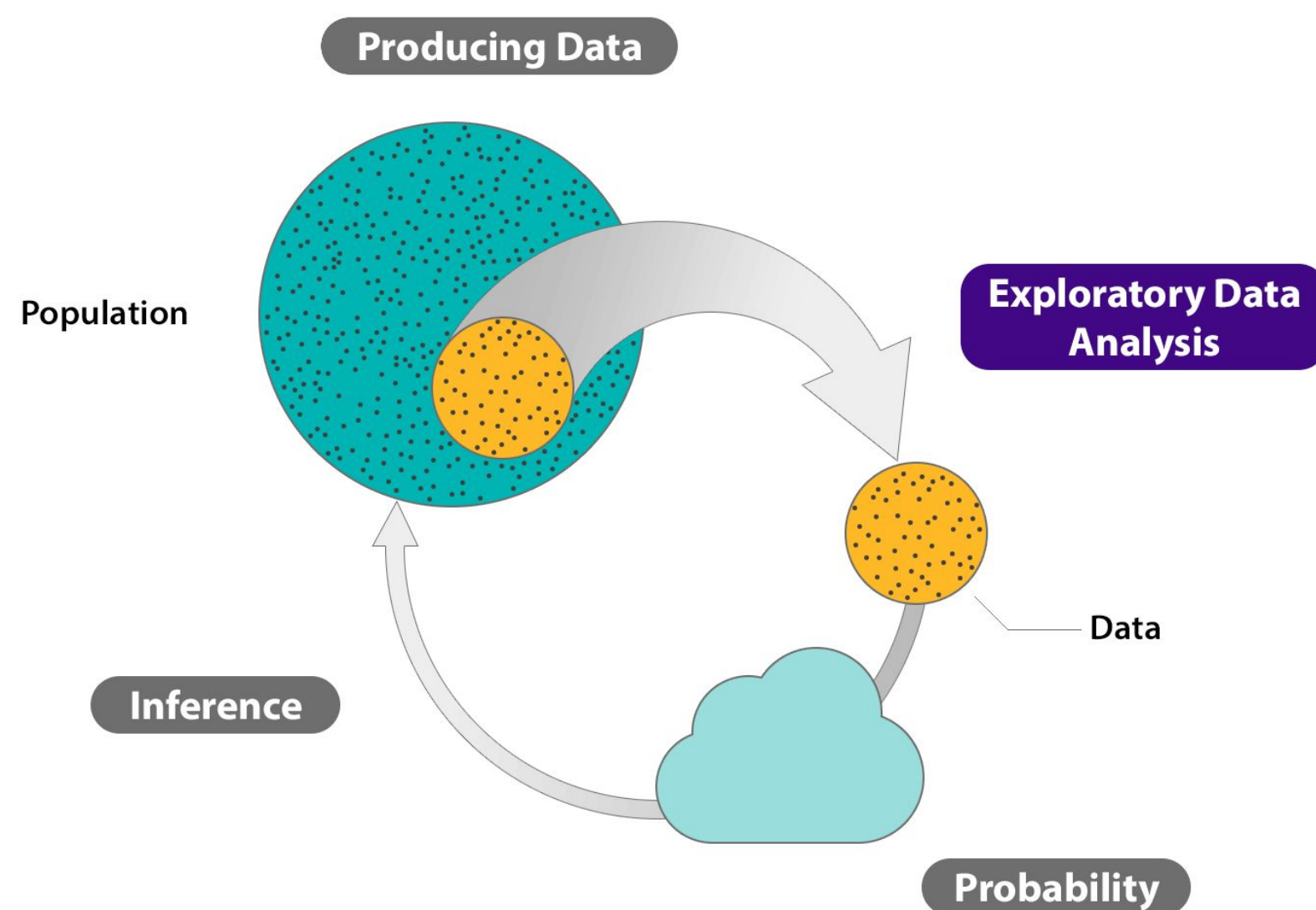
infoShare  
ACADEMY



# Python Praca z Danyymi

## Eksploracyjna Analiza Danych (EDA)

infoShare  
ACADEMY





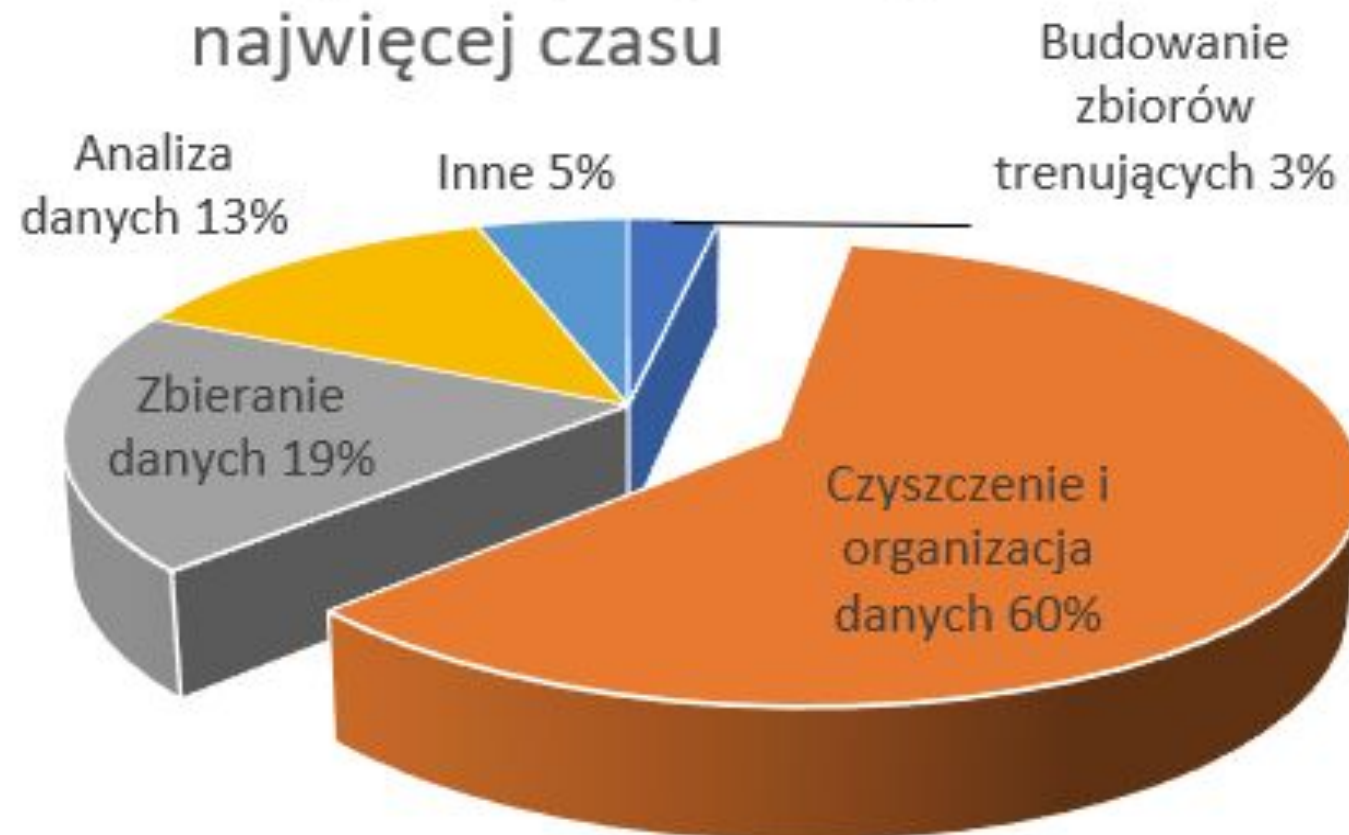


# Python Praca z Danyymi

## Oczyszczanie Danych

infoShare  
ACADEMY

Na co analityk danych poświęca  
najwięcej czasu







# Python Praca z Danyymi

## Wizualizacja Danych

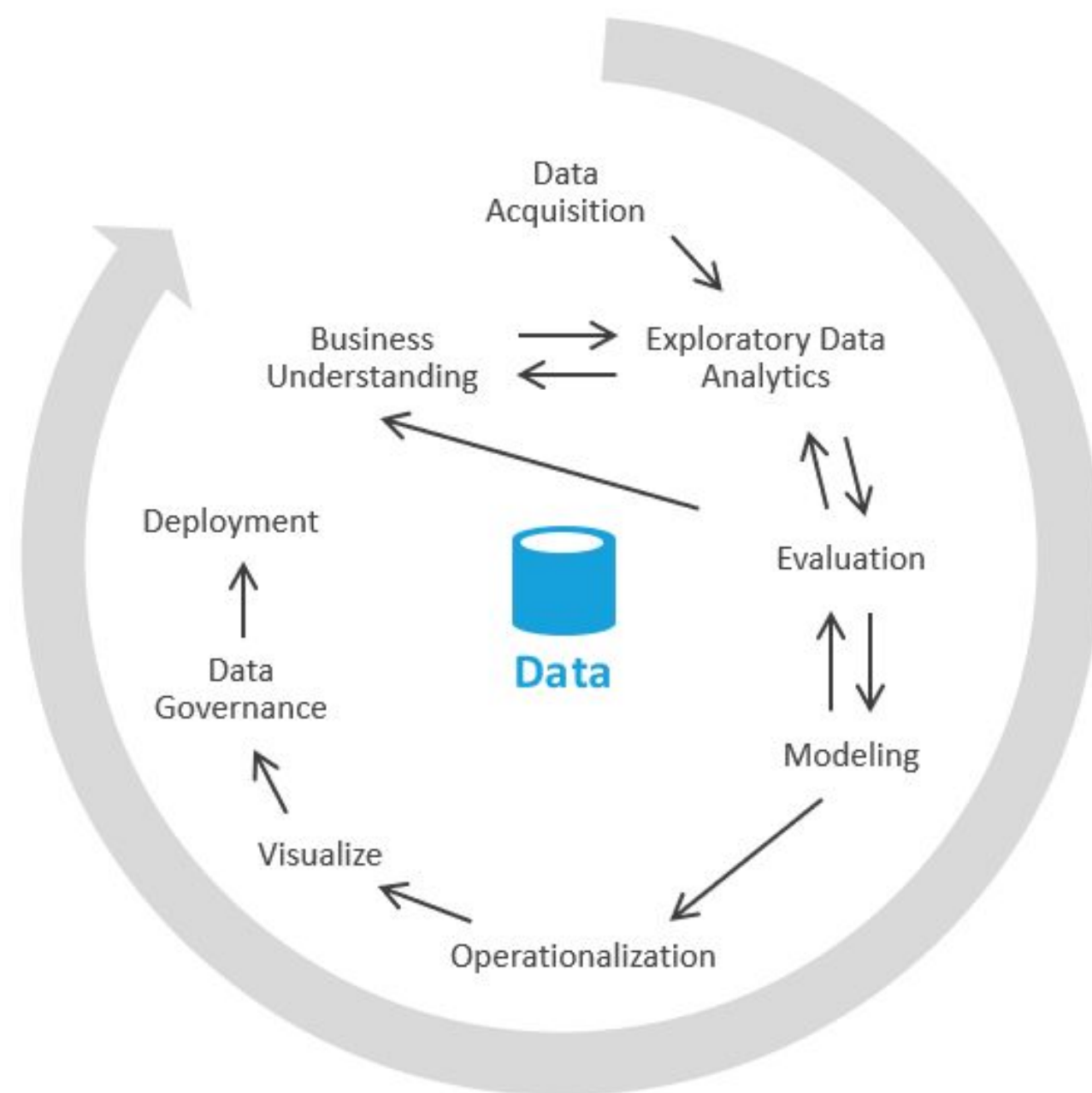
infoShare  
ACADEMY





# Python Praca z Danyymi

## Flow pracy z danymi – podsumowanie







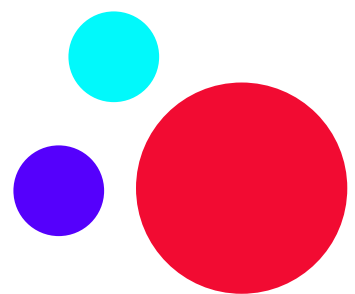
# Python Praca z Danymi

## Biblioteki

info **Share**  
ACADEMY







# Python Praca z Danymi

Biblioteki używane w Data Science

NumPy 

 pandas

 **SciPy**

matplotlib 

 seaborn

 scikit  
learn

info **Share**  
ACADEMY



# Python Praca z Danyami

## NumPy

NumPy 

# NumPy

The fundamental package for scientific computing with Python

LATEST RELEASE: [NUMPY 1.26](#). [VIEW ALL RELEASES](#)

**NumPy 1.26.0 released** 2023-09-16

### POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

### NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

### OPEN SOURCE

Distributed under a liberal [BSD license](#), NumPy is developed and maintained [publicly on GitHub](#) by a vibrant, responsive, and diverse [community](#).

### INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

### PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

### EASY TO USE

NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

**info** **Share**  
ACADEMY





# Python Praca z Danymi

W jakim języku?

NumPy 



info **Share**  
ACADEMY





# Python Praca z Danymi

array – podstawa NumPy

info **Share**  
ACADEMY

```
arr_1d = np.array([1, 2, 3, 4, 5])
```



```
array([1, 2, 3, 4, 5])
```

array jednowymiarowa



# Python Praca z Danymi

array czy lista?

infoShare  
ACADEMY

**array**



```
array([1, 2, 3, 4, 5])
```



wszystkie elementy mają  
ten sam typ danych

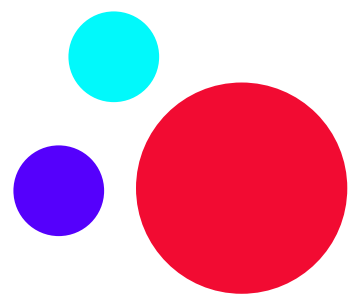
**lista**



```
[1, 2.5, 'hello', True]
```



różne elementy  
typu danych



# Python Praca z Danyymi

## NumPy – instalacja

`pip install numpy`

`conda install numpy`

`conda install -n nazwa_twojego_srodowiska numpy`

`import numpy as np.`

`print(np.__version__)`







# Python Praca z Danymi

## NumPy – podstawowe operacje



### Tworzenie tablicy:

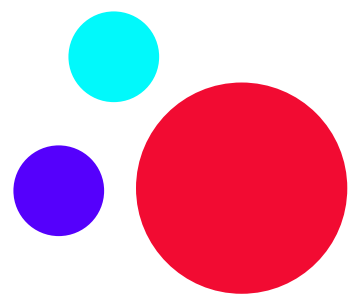
```
import numpy as np
```

```
arr_1d = np.array([1, 2, 3, 4, 5])
```

← tworzenie  
jednowymiarowej tablicy



```
array([1, 2, 3, 4, 5])
```



# Python Praca z Danymi

## NumPy – podstawowe operacje

### Operacje matematyczne:

```
import numpy as np
```

```
arr = np.array([1, 2, 3, 4, 5])
```

```
result_add = arr + 2
```



dodawanie

```
array([3, 4, 5, 6, 7])
```

```
result_multiply = arr * 3
```



mnożenie

```
array([ 3, 6, 9, 12, 15])
```

```
result_sqrt = np.sqrt(arr)
```



pierwiastkowanie

```
array([1. , 1.41421356, 1.73205081, 2. , 2.23606798])
```

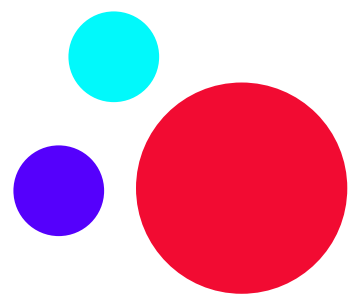




# Zadanie 9.1

## Podstawowe operacje (instrukcja)

1. Stwórz dwie jednowymiarowe tablice NumPy o takiej samej długości zawierające liczby całkowite. Następnie wykonaj poniższe operacje:
  - A. Dodaj obie tablice do siebie i wyświetl wynik.
  - B. Pomnóż jedną z tablic przez stałą wartość 3 i wyświetl wynik.
  - C. Oblicz pierwiastek kwadratowy z każdego elementu jednej z tablic i wyświetl wynik.
  - D. Oblicz sumę oraz średnią arytmetyczną elementów drugiej tablicy.



# Python Praca z Danymi

NumPy – tablice jedno- i wielowymiarowe

## Jednowymiarowa tablica (wektor)

```
array([1, 2, 3, 4, 5])
```

## Dwuwymiarowa tablica (macierz)

```
[[ 1 2 3]
 [ 4 5 6]
 [ 7 8 9]]
```

## Trójwymiarowa tablica (tensor)

```
[[[ 1 2 3]
    [ 4 5 6]]
```

```
[[[ 7 8 9]
    [10 11 12]]]
```





# Python Praca z Danymi

## NumPy – tablice jedno- i wielowymiarowe

### Dwuwymiarowa tablica (macierz)

```
matrix_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

```
array([[1, 2, 3],  
       [4, 5, 6],  
       [7, 8, 9]])
```

### Trójwymiarowa tablica (tensor)

```
tensor_3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
```

```
array([[1, 2, 3],  
       [4, 5, 6]],
```

```
array([[7, 8, 9],  
       [10, 11, 12]]],
```



# Python Praca z Danymi

## NumPy – indeksowanie



### Indeksowanie jednowymiarowe

```
import numpy as np
```

```
arr = np.array([1, 2, 3, 4, 5])
```

```
element_at_index_2 = arr[2]
```



3

**array([1, 2, 3, 4, 5])**

dostęp do trzeciego  
elementu (indeks 2)





# Python Praca z Danymi

## NumPy – wykorzystanie indeksów



### Indeksowanie dwuwymiarowe (macierz)

```
matrix = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

```
element_at_2nd_row_3rd_column = matrix[1, 2]
```

← dostęp do elementu  
w drugim wierszu  
i trzeciej kolumnie

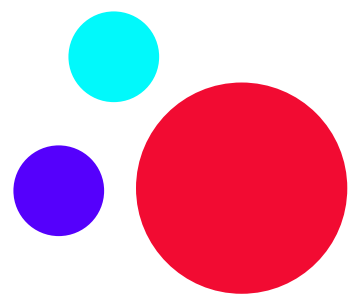
```
array([[1, 2, 3],
```

```
      [4, 5, 6],
```

← drugi wiersz

```
      [7, 8, 9]])
```

← trzecia wiersz



# Python Praca z Danymi

## NumPy – wykorzystanie indeksów



### Wycinanie (slicing)

```
arr = np.array([1, 2, 3, 4, 5])
```

```
subset_array = arr[1:4] ← dostęp do podzbioru tablicy (indeks 1 do 3)
```

```
array([1, 2, 3, 4, 5])
```



```
[2 3 4]
```

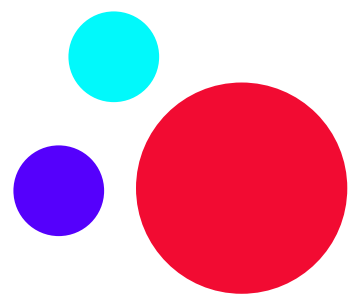




# Zadanie 9.2

## Indeksowanie (instrukcja)

1. Stwórz dwuwymiarową tablicę NumPy o wymiarach (3, 5) zawierającą liczby całkowite. Tablica ta powinna reprezentować macierz danych, gdzie każdy wiersz to zestaw informacji dla danego obiektu, a kolumny to różne cechy.
  - A. Wybierz i wyświetl trzeci wiersz tej macierzy za pomocą indeksowania.
  - B. Otrzymaj i wyświetl fragment macierzy, który zawiera dane z drugiej do czwartej kolumny.
  - C. Przeprowadź warunkowe indeksowanie, wybierając tylko te elementy, które są większe niż 5.



# Python Praca z Danymi

## NumPy – funkcje numeryczne

- `np.add()`
- `np.subtract()`
- `np.multiply()`
- `np.divide()`
- `np.power()`
- `np.sqrt()`
- `np.abs()`
- ... i wiele innych





# Python Praca z Danymi

## NumPy – funkcje numeryczne – sum

```
import numpy as np
```

```
arr = np.array([1, 2, 3, 4, 5])
```

```
sum_result = np.sum(arr)
```

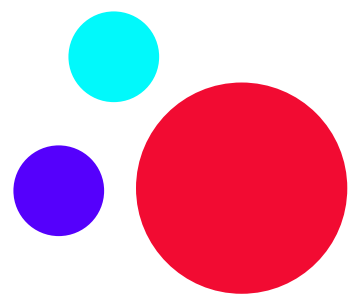


obliczanie sumy  
elementów



15





# Python Praca z Danymi

## NumPy – funkcje numeryczne – sum

```
matrix = np.array([[1, 2, 3],  
                  [4, 5, 6],  
                  [7, 8, 9]])
```

```
total_sum = np.sum(matrix)  
45
```



suma elementów w  
całej macierzy

```
column_sums = np.sum(matrix, axis=0)  
array([12, 15, 18])
```



suma elementów  
w poszczególnych  
kolumnach

```
row_sums = np.sum(matrix, axis=1)
```



suma elementów w  
poszczególnych  
wierszach

```
array([ 6, 15, 24])
```





# Python Praca z Danymi

NumPy – funkcje numeryczne – np.mean()

```
arr = np.array([1, 2, 3, 4, 5])
```

```
mean_result = np.mean(arr)
```

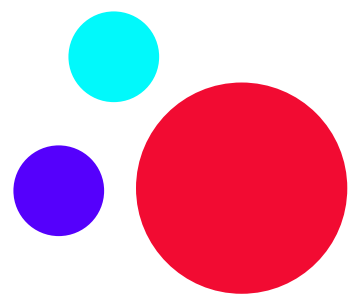


obliczanie średniej  
arytmetycznej  
elementów



3

infoShare  
ACADEMY



# Python Praca z Danymi

## NumPy – funkcje numeryczne – np.mean()

```
matrix = np.array([[1, 2, 3],  
                  [4, 5, 6],  
                  [7, 8, 9]])
```

```
total_mean = np.mean(matrix) ← dla całej macierzy  
5
```

```
column_means = np.mean(matrix, axis=0) ← dla poszczególnych  
array([4., 5., 6.]) kolumn
```

```
row_means = np.mean(matrix, axis=1) ← dla poszczególnych  
array([2., 5., 8.]) wierszy
```





# Python Praca z Danymi

NumPy – funkcje numeryczne – np.std()

```
arr = np.array([1, 2, 3, 4, 5])
```

```
std_result = np.std(arr)
```



obliczanie odchylenia  
standardowego  
elementów



**1.4142135623730951**

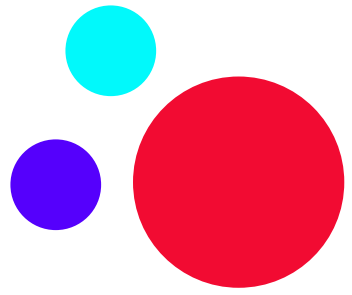
infoShare  
ACADEMY



# Zadanie 9.3

## Funkcje matematyczne (instrukcja)

1. Stwórz dwuwymiarową tablicę NumPy reprezentującą dane pomiarowe dla trzech różnych cech w czasie. Każdy wiersz tej macierzy powinien reprezentować różne pomiary dla jednej cechy w kolejnych okresach czasu. Następnie wykonaj następujące operacje statystyczne:
  - A. Oblicz sumę dla każdej cechy.
  - B. Oblicz średnią arytmetyczną dla każdej cechy.
  - C. Oblicz odchylenie standardowe dla każdej cechy.

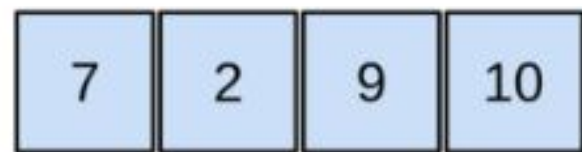


# Python Praca z Danyymi

## NumPy – podsumowanie

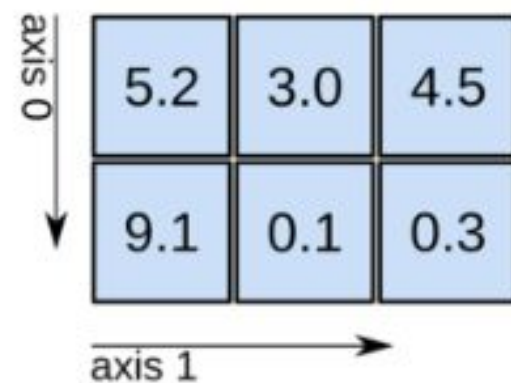
infoShare  
ACADEMY

1D array



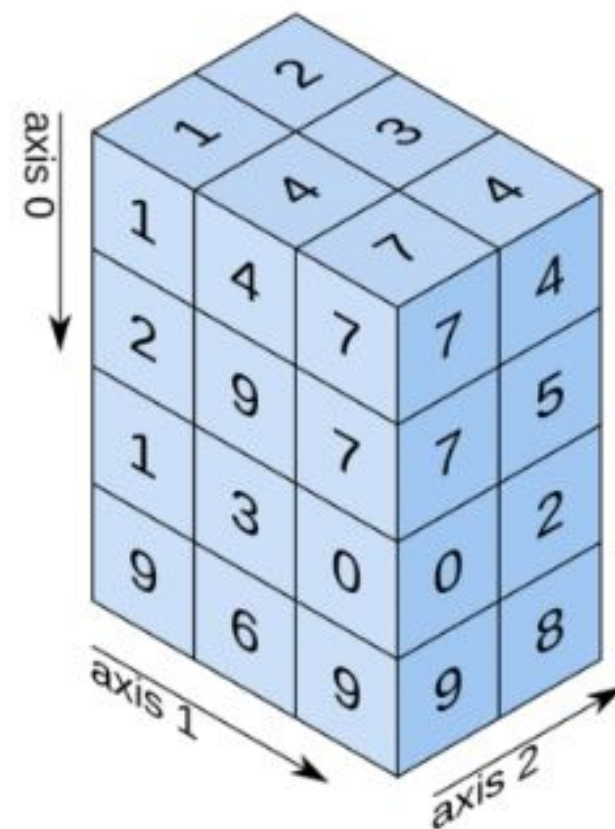
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (4, 3, 2)





# Python Praca z Danyami

## Pandas



### pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

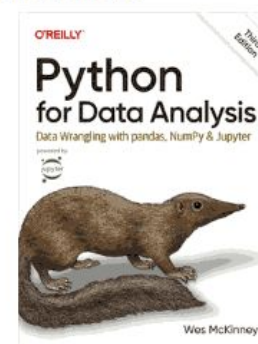
Install pandas now!

#### Latest version: 2.1.3

- What's new in 2.1.3
- Release date: Nov 10, 2023
- Documentation (web)
- Download source code

#### Follow us

#### Get the book



#### Previous versions

- 2.1.2 (Oct 26, 2023)  
changelog | docs | code
- 2.1.1 (Sep 20, 2023)  
changelog | docs | code
- 2.1.0 (Aug 30, 2023)  
changelog | docs | code
- 2.0.3 (Jun 28, 2023)  
changelog | docs | code

Show more

#### Getting started

- Install pandas
- Getting started

#### Documentation

- User guide
- API reference
- Contributing to pandas
- Release notes

#### Community

- About pandas
- Ask a question
- Ecosystem

#### With the support of:





# Python Praca z Danyymi

## Pandas – instalacja

pip install pandas

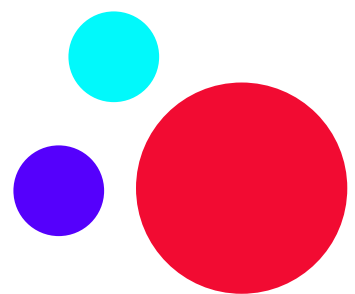
conda install pandas

import pandas as pd

print(pd.\_\_version\_\_)







# Python Praca z Danyymi

Pandas – struktury danych

infoShare  
ACADEMY



Series

	apples
0	3
1	2
2	0
3	1

+

Series

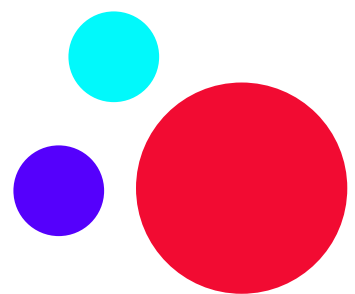
	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2





# Python Praca z Danyymi

Pandas – struktury danych

infoShare  
ACADEMY



**Series**

	apples
0	3
1	2
2	0
3	1

+

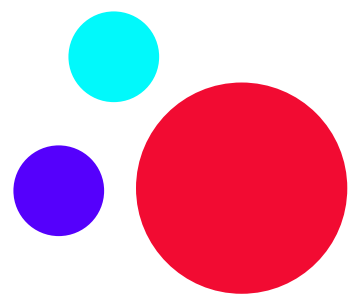
**Series**

	oranges
0	0
1	3
2	7
3	2

=

**DataFrame**

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2



# Python Praca z Danyymi

## Pandas – Series



```
import pandas as pd
```

```
data = [10, 20, 30, 40, 50]
```

```
series = pd.Series(data, name='test series')
```



tworzenie Series  
z listy

```
0    10
```

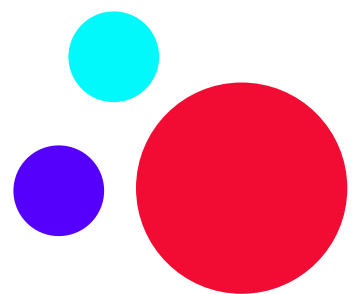
```
1    20
```

```
2    30
```

```
3    40
```

```
4    50
```

```
Name: test series, dtype: int64
```



# Python Praca z Danyymi

## Pandas – DataFrame

infoShare  
ACADEMY

Series

	apples
0	3
1	2
2	0
3	1

+

Series

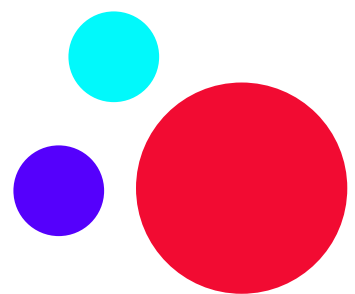
	oranges
0	0
1	3
2	7
3	2

=

DataFrame

	apples	oranges
0	3	0
1	2	3
2	0	7
3	1	2





# Python Praca z Danymi

## Pandas – DataFrame

```
import pandas as pd
```

```
data = {'Imię': ['Anna', 'Bartek', 'Celina'],  
        'Wiek': [25, 30, 22],  
        'Miasto': ['Warszawa', 'Kraków', 'Gdańsk']}
```



tworzenie DataFrame  
z listy słowników

```
df = pd.DataFrame(data)
```

	Imię	Wiek	Miasto
0	Anna	25	Warszawa
1	Bartek	30	Kraków
2	Celina	22	Gdańsk



# Zadanie 9.4

## Tworzenie Series i DataFrame (instrukcja)

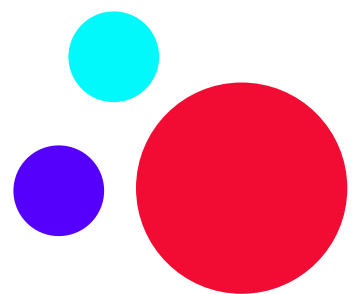
1. Twoim zadaniem jest stworzenie danych w formie Series i DataFrame, aby przeprowadzić prostą analizę danych osobowych kursantów. Poniżej znajdują się dane kilku kursantów z kursu programowania. Twoim celem jest stworzenie odpowiednich struktur danych w Pandas, aby móc łatwo analizować te informacje

**Kursant 1: Imię: Adam, Wiek: 26, Znane języki programowania: ['Python', 'JavaScript']**

**Kursant 2: Imię: Marta, Wiek: 22, Znane języki programowania: ['Java', 'C++', 'Python']**

**Kursant 3: Imię: Michał, Wiek: 24, Znane języki programowania: ['JavaScript', 'HTML', 'CSS']**

- A. Stwórz obiekt Series dla jednego z kursantów, zawierający informacje o imieniu, wieku i znanych językach programowania.
- B. Stwórz DataFrame, w którym każdy wiersz będzie reprezentował jednego kursanta, a kolumny będą odpowiadać imieniu, wiekowi i znającym językom programowania.
- C. Wyświetl informacje o wszystkich kursantach, takie jak imię, wiek i języki programowania.



# Python Praca z Danymi

Pandas – DataFrame czy array?

infoShare  
ACADEMY



```
array([[70, 90, 80],  
       [68, 80, 93],  
       [86, 72, 68]])
```



	0	1	2
0	70	90	80
1	68	80	93
2	86	72	68





# Python Praca z Danymi

## Źródła danych

```
import pandas as pd
```

```
df = pd.read_csv('nazwa_pliku.csv')
```



```
df = pd.read_excel('nazwa_pliku.xlsx')
```



```
from sqlalchemy import create_engine
```

```
engine = create_engine  
( 'nazwa_bazy_danych://uzytkownik:haslo@host:port/baza_dan  
ych' )
```

```
df = pd.read_sql_query('SELECT * FROM tabela', engine)
```



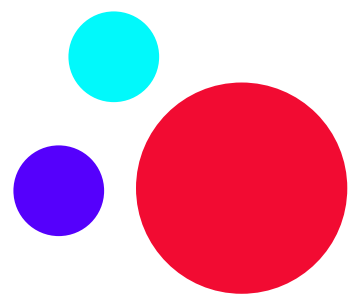
infoShare  
ACADEMY



# Zadanie 9.5

## Wczytywanie danych .csv (instrukcja)

1. Pobierz dane ze strony pasażerach Titanica:  
<https://www.kaggle.com/datasets/brendan45774/test-file>
  - A. Wczytaj dane z pliku CSV do DataFrame w sytuacji gdy plik CSV jest w tym samym folderze, co skrypt na którym pracujesz. informacje o imieniu, wieku i znanych językach programowania.
  - B. Wczytaj dane z pliku CSV do DataFrame, umieść plik CSV w innym folderze niż skrypt na którym pracujesz.



# Python Praca z Danyymi

## Pandas – eksploracyjna analiza danych

**E**xploratory

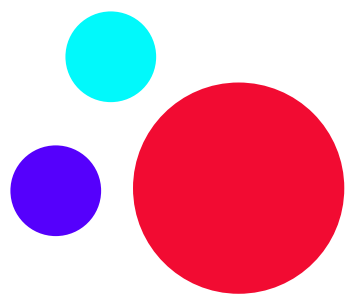
**D**ata

**A**nalysis



info **Share**  
ACADEMY





# Python Praca z Danymi

## Pandas – przeglądanie danych

```
df = pd.read_csv('Life Expectancy Data.csv')
```



Wczytanie  
danych

### Head

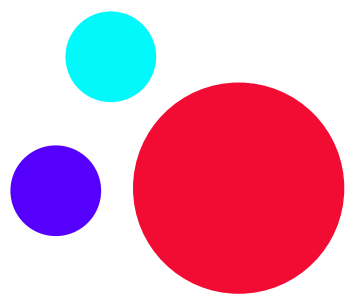
```
df.head()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1	584.259210	33736494.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1	612.696514	327582.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1	631.744976	31731688.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1	669.959000	3696958.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1	63.537231	2978599.0

### Tail

```
df.tail()
```

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.0	68.0	31	...	67.0	7.13	65.0	33.6	454.366654	12777511.0
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.0	7.0	998	...	7.0	6.52	68.0	36.7	453.351155	12633897.0
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.0	73.0	304	...	73.0	6.53	71.0	39.8	57.348340	125525.0
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.0	76.0	529	...	76.0	6.16	75.0	42.1	548.587312	12366165.0
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.0	79.0	1483	...	78.0	7.10	78.0	43.5	547.358878	12222251.0



# Python Praca z Danymi

## Pandas – przeglądanie danych

df.info()



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   Country                                   2938 non-null   object
1   Year                                       2938 non-null   int64
2   Status                                    2938 non-null   object
3   Life expectancy                           2928 non-null   float64
4   Adult Mortality                           2928 non-null   float64
5   infant deaths                             2938 non-null   int64
6   Alcohol                                   2744 non-null   float64
7   percentage expenditure                    2938 non-null   float64
8   Hepatitis B                               2385 non-null   float64
9   Measles                                   2938 non-null   int64
10  BMI                                        2904 non-null   float64
11  under-five deaths                         2938 non-null   int64
12  Polio                                     2919 non-null   float64
13  Total expenditure                         2712 non-null   float64
14  Diphtheria                               2919 non-null   float64
15  HIV/AIDS                                  2938 non-null   float64
16  GDP                                        2490 non-null   float64
17  Population                                2286 non-null   float64
18  thinness 1-19 years                       2904 non-null   float64
19  thinness 5-9 years                        2904 non-null   float64
20  Income composition of resources            2771 non-null   float64
21  Schooling                                 2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```





# Python Praca z Danymi

## Pandas – przeglądanie danych

infoShare  
ACADEMY

**df.shape**

**(2938, 22)**

l. wierszy

l. kolumn

**df.size**

**64636**



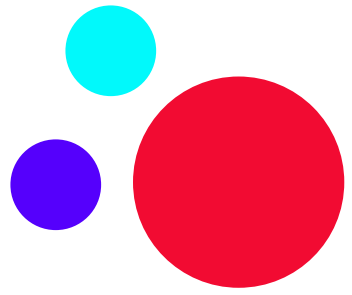
$x * y$

A	B

} y

} x





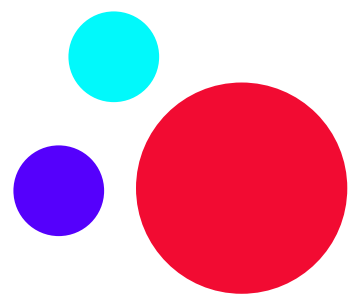
# Python Praca z Danyimi

## Pandas – przeglądanie danych

**df.describe()**

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	38.321247
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	20.044034
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000

infoShare  
ACADEMY



# Zadanie 9.6

## Analiza danych (instrukcja)

1. Na podstawie zbioru danych o pasażerach Titanica użyj poznanych funkcji z biblioteki Pandas do analizy danych(head(), tail(), info(), describe(), size, shape). Stwórz raport, w którym wyświetlisz zebrane informacje o zbiorze danych.







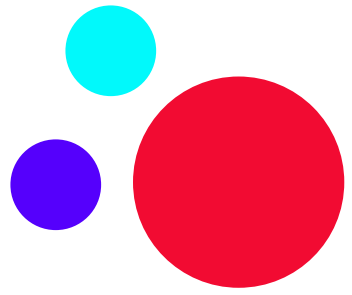
# Python Praca z Danymi

## Pandas – brakujące wartości

`df.isnull()`

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
2933	False	False	False	False	False	False	False
2934	False	False	False	False	False	False	False
2935	False	False	False	False	False	False	False
2936	False	False	False	False	False	False	False
2937	False	False	False	False	False	False	False





# Python Praca z Danymi

## Pandas – brakujące wartości

`df.isnull().sum()`

```
Country      0
Year         0
Status       0
Life expectancy    10
Adult Mortality    10
infant deaths     0
Alcohol       194
percentage expenditure    0
Hepatitis B     553
Measles        0
  BMI          34
under-five deaths    0
Polio           19
Total expenditure    226
Diphtheria       19
  HIV/AIDS        0
GDP             448
Population      652
  thinness 1-19 years    34
  thinness 5-9 years    34
Income composition of resources    167
Schooling       163
dtype: int64
```



# Python Praca z Danymi

## Pandas – brakujące wartości

```
df_new = df.dropna()
```



usunięcie wierszy, w których  
występują brakujące wartości

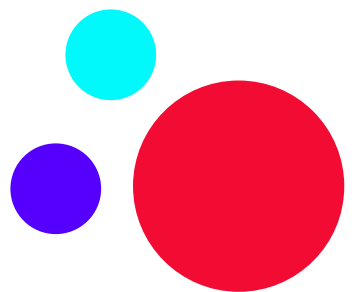
df.shape  
**(2938, 22)**



df\_new.shape  
**(1649, 22)**

usunięto 1649 wierszy





# Python Praca z Danymi

## Pandas – brakujące wartości

### pandas.DataFrame.dropna

```
DataFrame.dropna(*, axis=0, how=_NoDefault.no_default,  
thresh=_NoDefault.no_default, subset=None, inplace=False, ignore_index=False)
```

Remove missing values.

[\[source\]](#)

#### Parameters:

**axis** : {0 or 'index', 1 or 'columns'}, default 0

Determine if rows or columns which contain missing values are removed.

- 0, or 'index' : Drop rows which contain missing values.
- 1, or 'columns' : Drop columns which contain missing value.

Only a single axis is allowed.

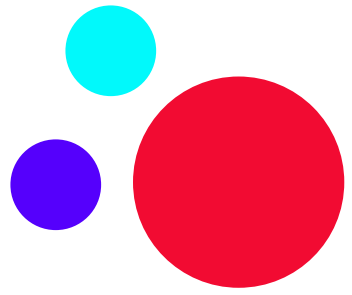
**how** : {'any', 'all'}, default 'any'

Determine if row or column is removed from DataFrame, when we have at least one NA or all NA.

- 'any' : If any NA values are present, drop that row or column.
- 'all' : If all values are NA, drop that row or column.

infoShare  
ACADEMY





# Python Praca z Danymi

## Pandas – brakujące wartości

```
df_filled = df.fillna(df.mean())
```

**df**

Country	0
Year	0
Status	0
Life expectancy	10
Adult Mortality	10
infant deaths	0
Alcohol	194
percentage expenditure	0
Hepatitis B	553
Measles	0
BMI	34
under-five deaths	0
Polio	19
Total expenditure	226
Diphtheria	19
HIV/AIDS	0
GDP	448
Population	652
thinness 1-19 years	34
thinness 5-9 years	34
Income composition of resources	167
Schooling	163
dtype:	int64



**df\_filled**

Country	0
Year	0
Status	0
Life expectancy	0
Adult Mortality	0
infant deaths	0
Alcohol	0
percentage expenditure	0
Hepatitis B	0
Measles	0
BMI	0
under-five deaths	0
Polio	0
Total expenditure	0
Diphtheria	0
HIV/AIDS	0
GDP	0
Population	0
thinness 1-19 years	0
thinness 5-9 years	0
Income composition of resources	0
Schooling	0
dtype:	int64



# Zadanie 9.7

## Imputacja braków danych (instrukcja)

1. Dla zbioru danych o pasażerach Titanica:
  - A. Sprawdź statystykę brakujących danych.
  - B. Usuń kolumnę z największą liczbą brakujących danych.
  - C. Uzupełnij brakujące dane dla pozostałych kolumn średnią arytmetyczną wartości tych kolumn.







# Python Praca z Danymi

## Pandas – średnia, mediana, moda, odchylenie

```
srednia = df['Life expectancy'].mean()
```

Średnia: 69.22493169398908

```
mediana = df['Life expectancy'].median()
```

Mediana: 72.1

```
moda = df['Life expectancy'].mode()[0]
```

Moda: 73.0

```
odchylenie_std = df['Life expectancy'].std()
```

Odchylenie standardowe: 9.523867487824301

moda może mieć  
więcej niż jedną  
wartość, dlatego  
wybieramy pierwszą





# Zadanie 9.8

## Statystyki (instrukcja)

1. Dla zbioru danych o pasażerach Titanica oblicz: średnią arytmetyczną, medianę, modę oraz odchylenie standardowe dla kolumny wiek (Age).





# Python Praca z Danymi

## Pandas – operacje na kolumnach DataFrame

```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
```

Dodawanie nowej kolumny:

1. Bezpośrednie przypisanie wartości

```
df['C'] = [7, 8, 9]      ←      nowa kolumna C
```

2. Wykorzystanie istniejącej kolumny do obliczeń

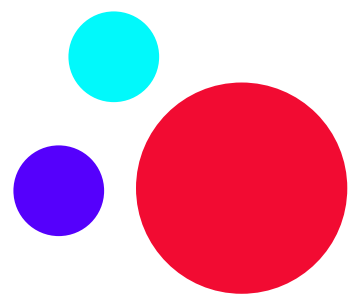
```
df['D'] = df['A'] * 2      ←      nowa kolumna D
```

3. Użycie funkcji do tworzenia nowej kolumny

```
def oblicz_nowa_wartosc(x):
```

```
    return x * 3
```

```
df['E'] = df['B'].apply(oblicz_nowa_wartosc)
```



# Python Praca z Danymi

## Pandas – operacje na kolumnach DataFrame



```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]})
```

Usuwanie kolumny:

1. Metoda drop:

```
df = df.drop('B', axis=1)
```

2. Metoda pop:

```
kolumna_b = df.pop('B')
```





# Zadanie 9.9

## Dodawanie/usuwanie kolumn (instrukcja)

1. Utwórz DataFrame o nazwie dane, zawierający dwie kolumny: "Imię" i "Wiek". Wypełnij go dowolnymi danymi dla co najmniej trzech osób.
2. Dodaj nową kolumnę o nazwie "Miejscowość", która będzie zawierała informacje o miejscu zamieszkania każdej osoby.
3. Usuń kolumnę "Wiek" z DataFrame.
4. Dodaj nową kolumnę "Zawód" z informacjami o zawodach osób.



# Python Praca z Danymi

## Pandas – zmiany nazwy kolumn

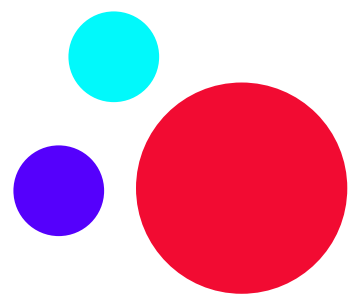
```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
```



```
df = df.rename(columns={'A': 'Nowa_A', 'B': 'Nowa_B'})
```



```
df = pd.DataFrame({'Nowa_A': [1, 2, 3], 'Nowa_B': [4, 5, 6]})
```



# Zadanie 9.10

## Zmiana nazw kolumn (instrukcja)

1. Utwórz DataFrame o nazwie oceny, zawierający trzy kolumny: "Przedmiot", "Ocena\_Pisemna" i "Ocena\_Ustna". Wypełnij go danymi dotyczącymi ocen studentów w dwóch kategoriach.
2. Przemianuj kolumnę "Ocena\_Pisemna" na "Ocena\_Sprawdzian", aby lepiej odzwierciedlić naturę ocen.
3. Dodaj nową kolumnę "Ocena\_Końcowa", która będzie średnią ważoną z ocen pisemnych i ustnych (zakładamy wagę 70% dla ocen pisemnych i 30% dla ustnych).





# Python Praca z Danymi

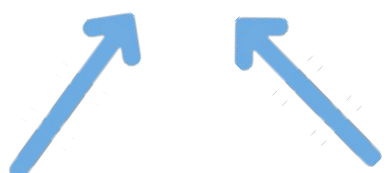
Pandas – operacje na kolumnach i wierszach

```
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]}, index=['X', 'Y', 'Z'])
```

	A	B
X	1	4
Y	2	5
Z	3	6

**loc()** – indeksowanie za pomocą etykiet

```
wybrane_dane = df.loc['Y', 'B'] #wynik 5
```

  
wiersz      kolumna



# Python Praca z Danymi

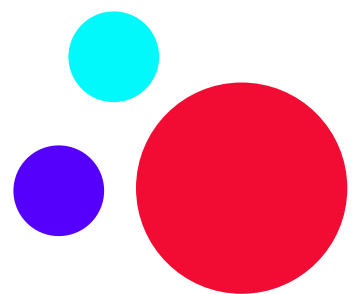
Pandas – operacje na kolumnach i wierszach

	A	B
X	1	4
Y	2	5
Z	3	6

**iloc()** – indeksowanie za pomocą indeksu numerycznego

wybrane\_dane = df.iloc[1, 0]    # wynik 2

↑    ↑  
wiersz    kolumna



# Zadanie 9.11

## loc/iloc (instrukcja)

1. Utwórz DataFrame o nazwie dane, zawierający trzy kolumny: "Imię", "Wiek" i "Wzrost". Wypełnij go danymi dla co najmniej pięciu osób.
2. Za pomocą funkcji loc, wybierz dane dla osoby o imieniu "Anna".
3. Za pomocą funkcji.iloc, wybierz dane dla drugiej osoby w DataFrame.
4. Utwórz nowy DataFrame o nazwie podsumowanie zawierający tylko kolumny "Imię" i "Wiek" z oryginalnego DataFrame.





# Python Praca z Danymi

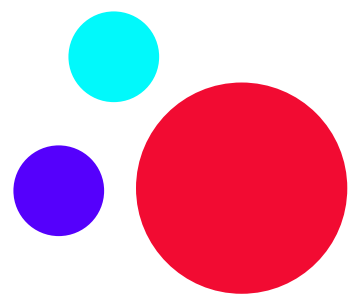
## Pandas – filtrowanie

Filtrowanie na podstawie warunków logicznych:

	Imię	Wiek	Wzrost
0	Anna	25	165
1	Jan	30	175
2	Katarzyna	22	160
3	Michał	28	180
4	Ewa	35	160

```
wyniki_filtracji = df[df['Wiek'] > 30]
```

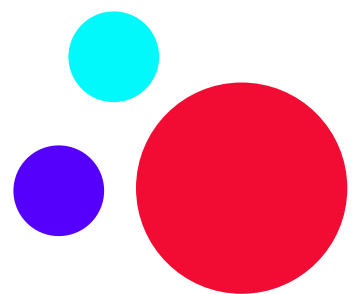
	Imię	Wiek	Wzrost
4	Ewa	35	160



# Zadanie 9.12

## Filtrowanie (instrukcja)

1. Utwórz DataFrame o nazwie dane, zawierający trzy kolumny: "Produkt", "Ilość" i "Cena". Wypełnij go danymi dotyczącymi zakupów w sklepie.
2. Za pomocą filtrowania logicznego (`df[df['warunek']]`), wybierz wszystkie zakupy, gdzie ilość zakupionego produktu jest większa niż 2 jednostki.
3. Stwórz nowy DataFrame o nazwie tanie\_produkty, który zawiera zakupy, gdzie cena jednostkowa produktu jest mniejsza lub równa 5, a ilość zakupionego produktu jest większa niż 1.



# Python Praca z Danymi

## Pandas – sortowanie



df → df\_sortowane = df.sort\_values(by='Wiek') → df\_sortowane

	Imię	Wiek	Wzrost
0	Anna	25	165
1	Jan	30	175
2	Katarzyna	22	160
3	Michał	28	180



	Imię	Wiek	Wzrost
2	Katarzyna	22	160
0	Anna	25	165
3	Michał	28	180
1	Jan	30	175



Sortowanie malejąco:

```
df_sortowane_malejaco = df.sort_values(by='Wiek', ascending=False)
```





# Python Praca z Danymi

## Pandas – sortowanie wielu kolumn



df → df\_sortowane = df.sort\_values(by=['Wiek', 'Wzrost']) → df\_sortowane

	Imię	Wiek	Wzrost
0	Anna	25	165
1	Jan	30	175
2	Katarzyna	22	160
3	Michał	28	180



	Imię	Wiek	Wzrost
2	Katarzyna	22	160
0	Anna	25	165
3	Michał	28	180
1	Jan	30	175





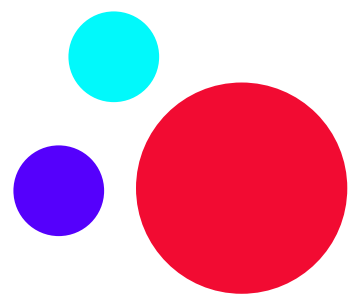
# Zadanie 9.13

## Pandas (instrukcja)

1. Oblicz dla poniższego dataframe:

```
data = {'Tytuł': ['Pan Tadeusz', 'Lalka', 'Krzyżacy', 'Ogniem i Mieczem', 'Quo Vadis'],  
        'Autor': ['Adam Mickiewicz', 'Bolesław Prus', 'Henryk Sienkiewicz', 'Henryk Sienkiewicz', 'Henryk  
Sienkiewicz'],  
        'Rok_wydania': [1834, 1890, 1900, None, 1896],  
        'Liczba_stron': [400, None, 600, 850, None],  
        'Ocena_czytelników': [4.5, 4.2, None, 4.8, 4.0]}
```

- A. Uzupełnij brakujące dane w kolumnach 'Rok\_wydania', 'Liczba\_stron' i 'Ocena\_czytelników'.
- B. Określ, która książka jest najstarsza w bibliotece.
- C. Oblicz średnią liczbę stron we wszystkich książkach.
- D. Stwórz nową ramkę danych, która zawiera tylko książki z oceną czytelników powyżej 4.5.



# Python Praca z Danyymi

Pandas – podsumowanie



infoShare  
ACADEMY

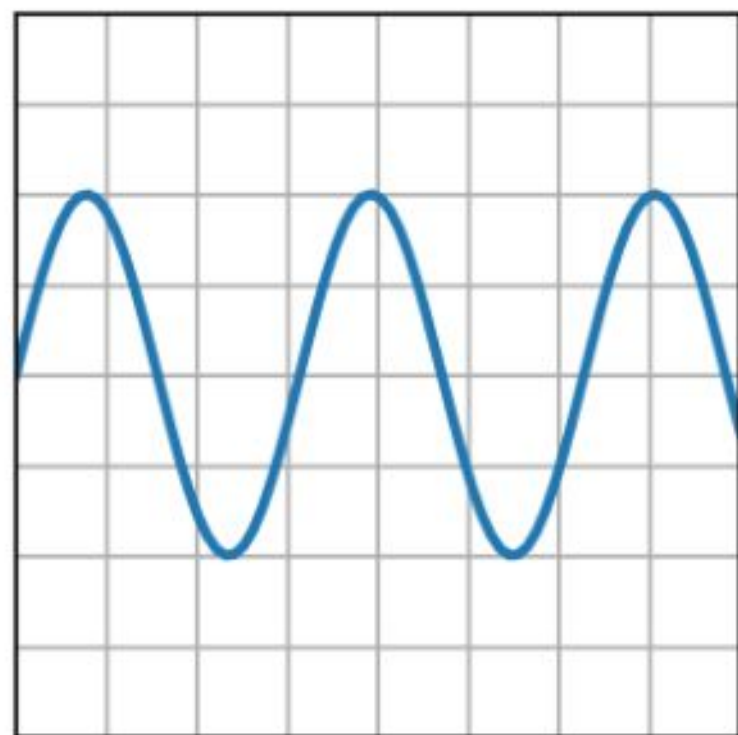




# Python Praca z Danyymi

## matplotlib

**matplotlib**



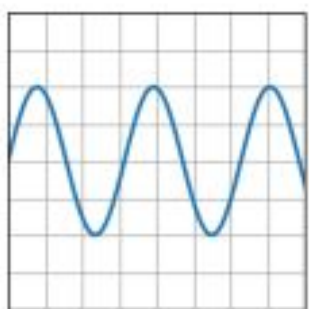
*plot(x, y)*



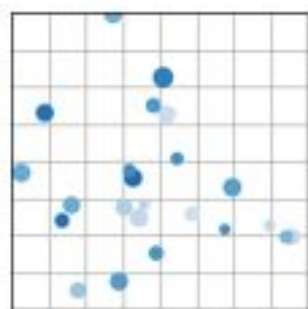
# Python Praca z Danymi

## matplotlib – typy wykresów

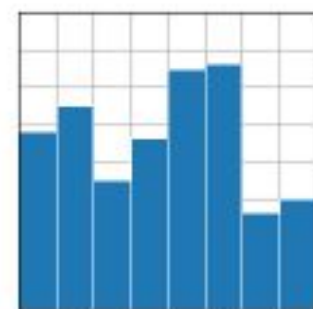
info **Share**  
ACADEMY



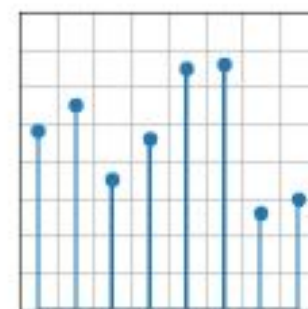
`plot(x, y)`



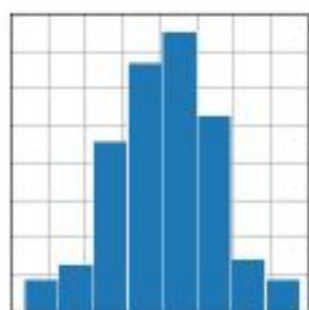
`scatter(x, y)`



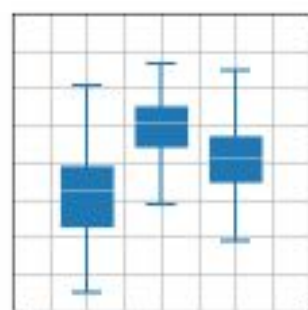
`bar(x height)`



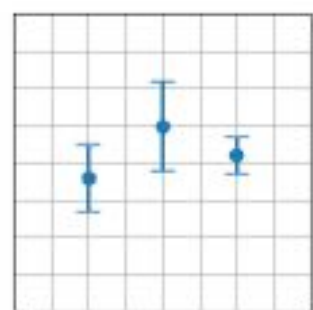
`stem(x, y)`



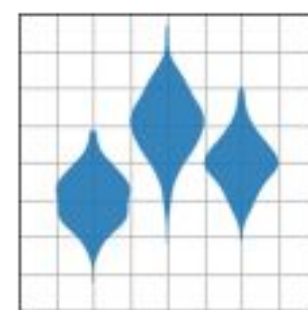
`hist(x)`



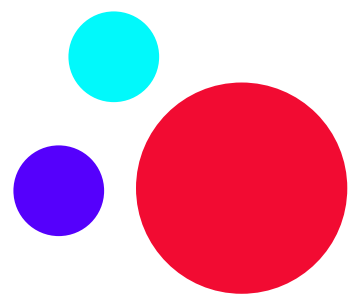
`boxplot(X)`



`errorbar(x, y, yerr, xerr)`



`violinplot(D)`



# Python Praca z Danyymi

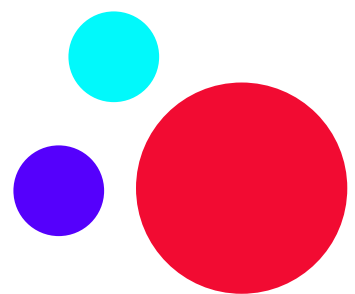
## matplotlib – instalacja

`pip install matplotlib`

`conda install matplotlib`

`import matplotlib.pyplot as plt`

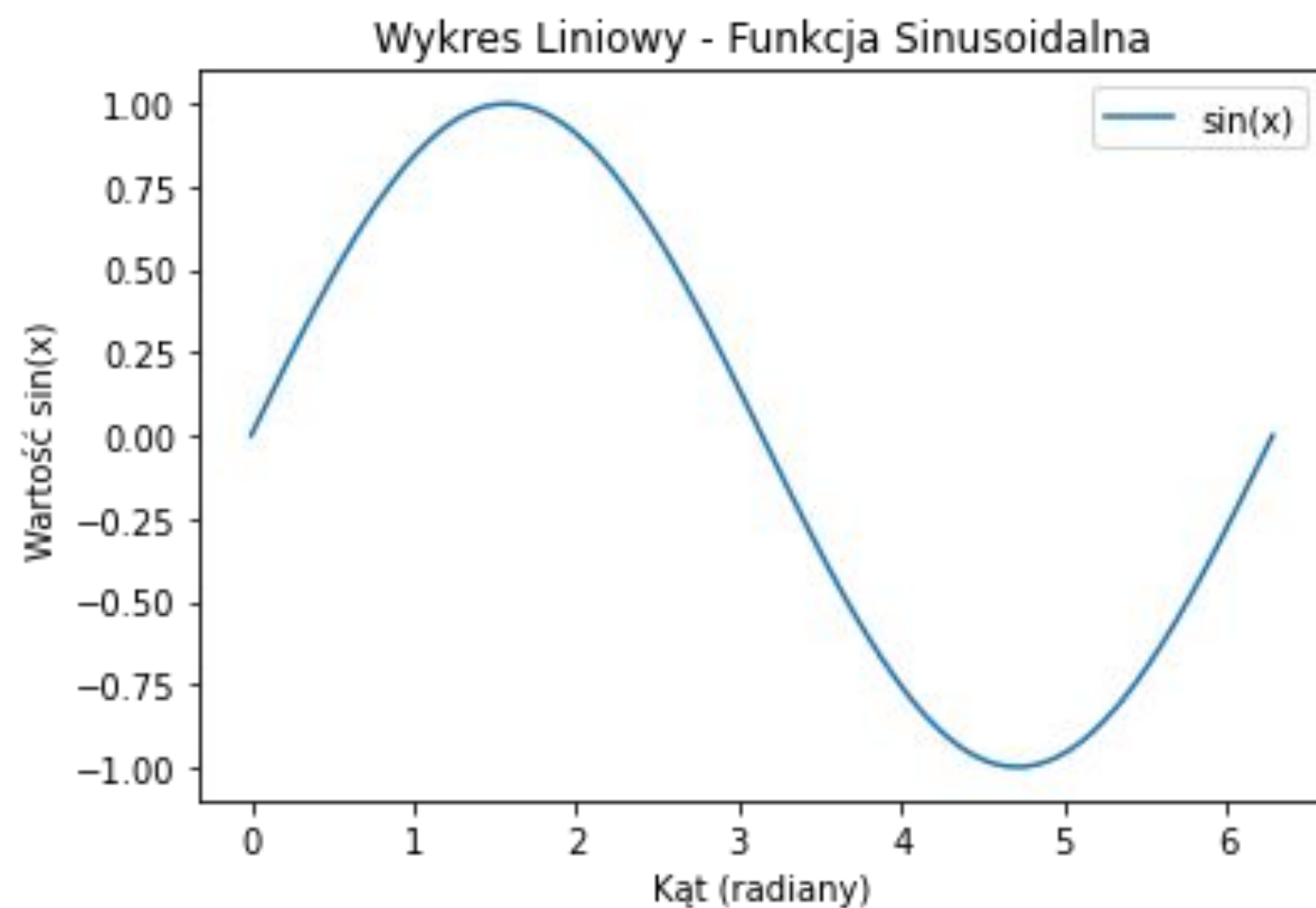




# Python Praca z Danymi

## matplotlib – wykres liniowy

infoShare  
ACADEMY





# Python Praca z Danymi

## matplotlib – wykres liniowy

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

```
import numpy as np
```

```
angles = np.linspace(0, 2*np.pi, 100)  
sin_values = np.sin(angles)
```

} generowanie danych

```
plt.plot(angles, sin_values, label='sin(x)')
```

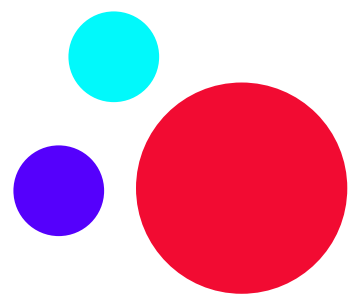
← tworzenie wykresu liniowego

```
plt.xlabel('Kąt (radiany)')  
plt.ylabel('Wartość sin(x)')  
plt.title('Wykres Liniowy – Funkcja Sinusoidalna')  
plt.legend() # Dodanie legendy
```

} dodanie etykiet i tytułu

```
plt.show()
```

← wyświetlenie wykresu



# Zadanie 9.14

## Wykres liniowy (instrukcja)

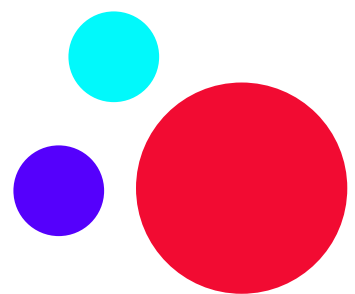
1. Przyjęto, że firma A-B Corporation prowadzi analizę wzrostu swojej sprzedaży w ciągu ostatnich 12 miesięcy. Dane o miesięcznych dochodach ze sprzedaży zostały zebrane i przedstawione poniżej. Twoim zadaniem jest stworzenie wykresu liniowego, który zobrazuje zmiany w sprzedaży w kolejnych miesiącach.

Dane miesięczne sprzedaży:

Styczeń: 50 000	Kwiecień: 60 000	Lipiec: 70 000	Październik: 68 000
Luty: 52 000	Maj: 58 000	Sierpień: 75 000	Listopad: 75 000
Marzec: 55 000	Czerwiec: 62 000	Wrzesień: 72 000	Grudzień: 80 000

1. Zaimportuj bibliotekę Matplotlib.
2. Użyj funkcji `plt.plot()` do stworzenia wykresu liniowego.
3. Oznacz osie x i y etykietami odpowiednio, dodaj tytuł wykresu.
4. Upewnij się, że wykres jest czytelny i estetyczny.
5. Dodaj legendę, aby oznaczyć dane.

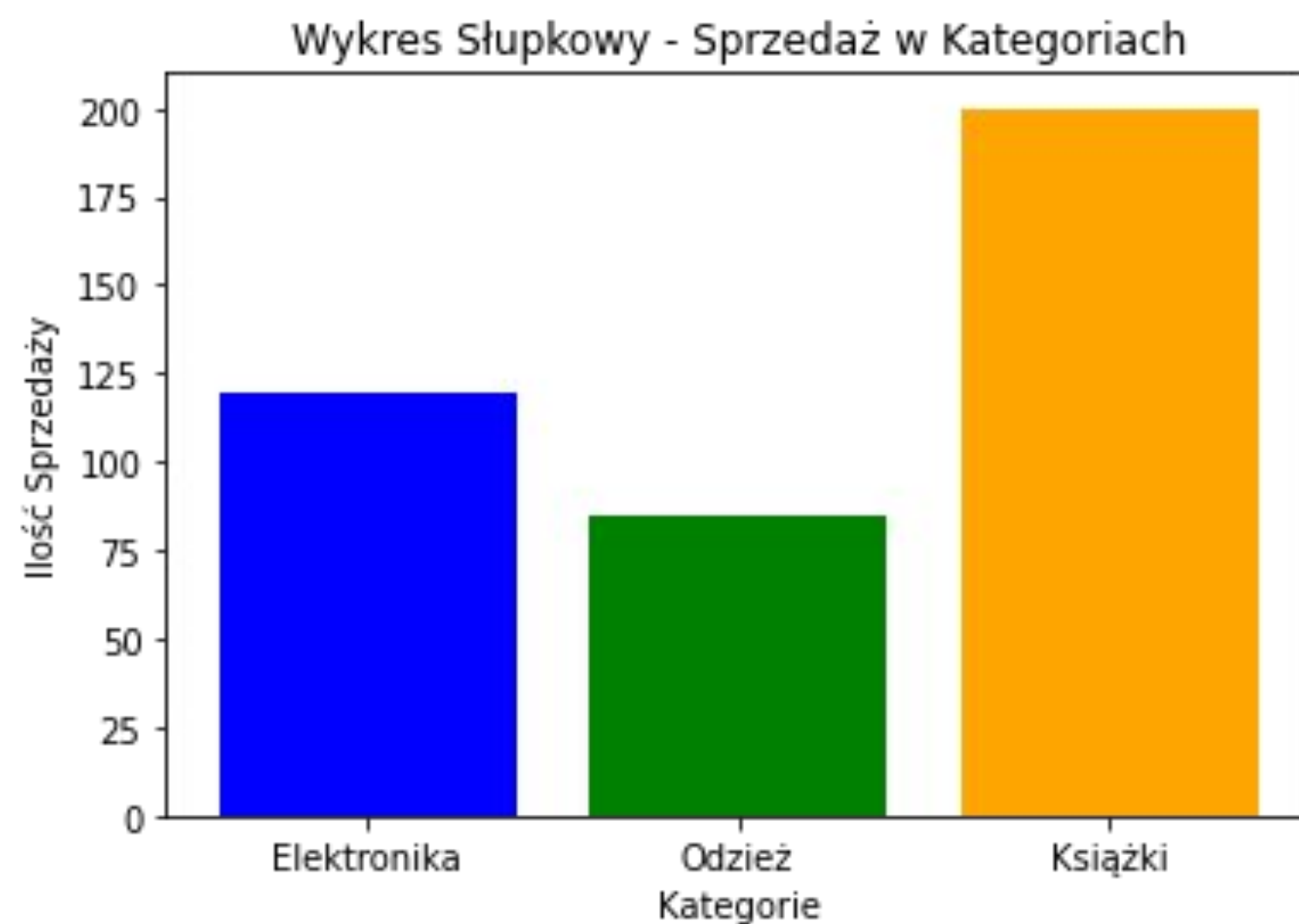




# Python Praca z Danymi

## matplotlib – wykres słupkowy

infoShare  
ACADEMY





# Python Praca z Danymi

## matplotlib – wykres słupkowy

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

```
kategorie = ['Elektronika', 'Odzież', 'Książki']  
sprzedaz = [120, 85, 200]
```

} dane o sprzedaży w  
trzech kategoriach

```
plt.bar(kategorie, sprzedaz, color=['blue', 'green', 'orange'])
```



tworzenie wykresu słupkowego

```
plt.xlabel('Kategorie')
```

```
plt.ylabel('Ilość Sprzedaży')
```

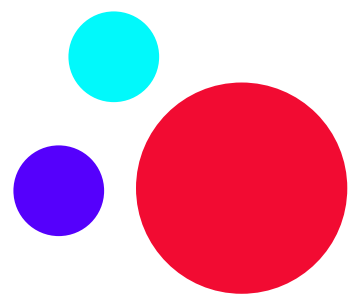
```
plt.title('Wykres Słupkowy - Sprzedaż w Kategoriach')
```

} dodawanie  
etykiet i tytułu

```
plt.show()
```



wyświetlenie wykresu



# Zadanie 9.15

## Wykres słupkowy (instrukcja)

Przyjmijmy, że wyniki testów matematycznych z trzech klas (A, B, C) zostały zebrane, a twoim zadaniem jest stworzenie wykresu słupkowego przedstawiającego średnie wyniki każdej z klas. Poniżej znajdują się wyniki testów dla każdego ucznia w poszczególnych

klasach:	Klasa A:	Klasa B:	Klasa C:
	Uczeń 1: 85	Uczeń 1: 90	Uczeń 1: 75
	Uczeń 2: 78	Uczeń 2: 82	Uczeń 2: 80
	Uczeń 3: 92	Uczeń 3: 78	Uczeń 3: 68
	Uczeń 4: 88	Uczeń 4: 85	Uczeń 4: 72
	Uczeń 5: 95	Uczeń 5: 88	Uczeń 5: 85

- Stwórz wykres słupkowy przedstawiający średnie wyniki matematyczne w każdej z klas.
- Oznacz osie x etykietami dla klas (A, B, C) i osie y etykietami jako 'Średni Wynik'.
- Użyj kolorów do odróżnienia słupków dla każdej z klas.
- Dodaj tytuł wykresu, który jednoznacznie opisuje przedstawione dane.



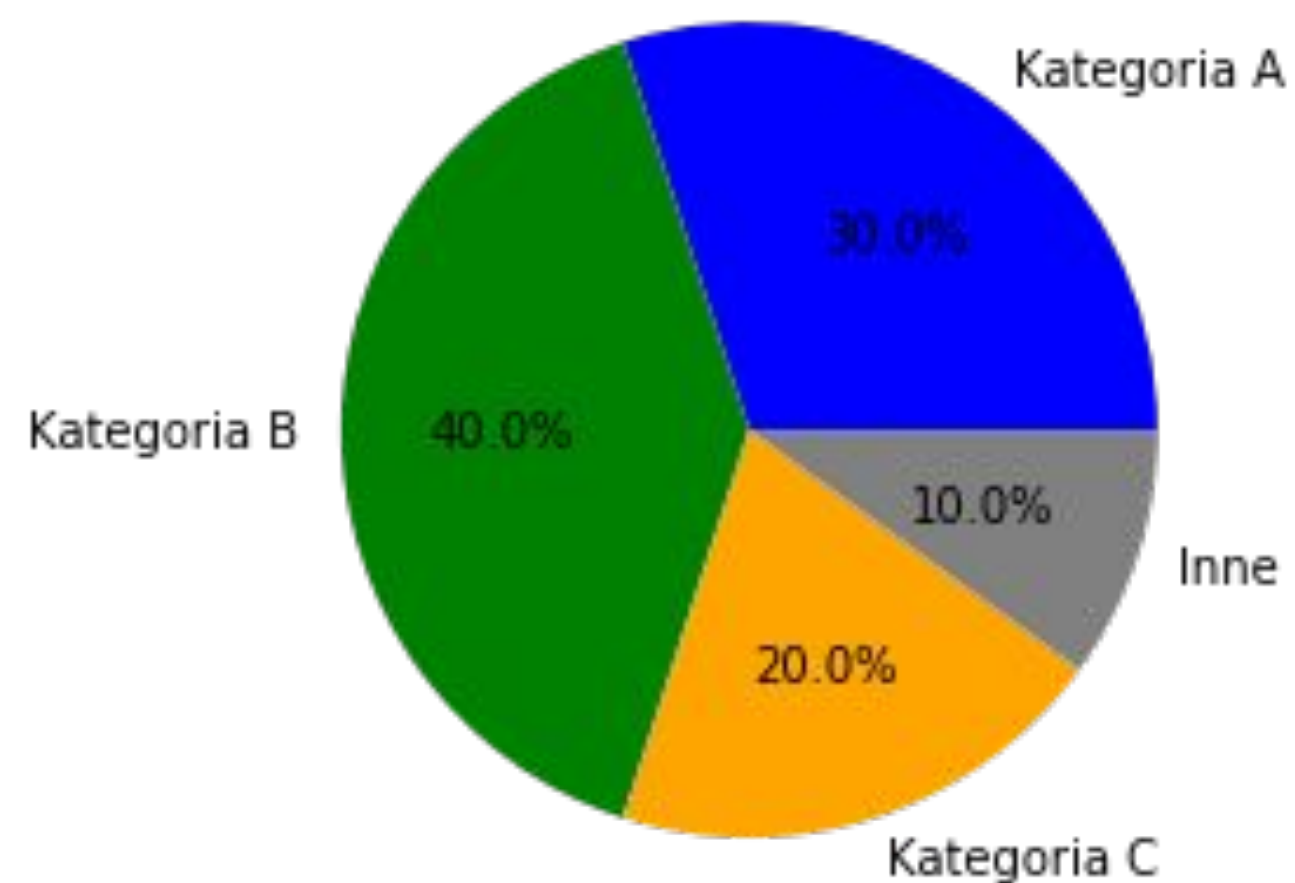


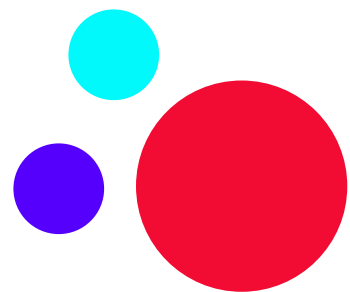
# Python Praca z Danymi

## matplotlib – wykres kołowy

infoShare  
ACADEMY

Udziały Procentowe w Całości





# Python Praca z Danymi

## matplotlib – wykres kołowy



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

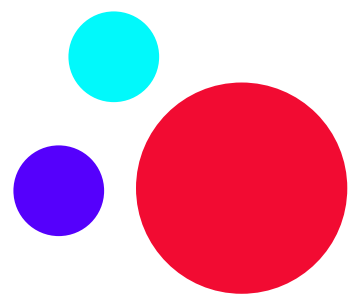
```
udzialy = [30, 40, 20, 10]
```

```
etykiety = ['Kategoria A', 'Kategoria B', 'Kategoria C', 'Inne']
```

```
plt.pie(udzialy, labels=etykiety, autopct='%1.1f%%', colors=['blue', 'green', 'orange', 'gray'])
```

```
plt.title('Udziały Procentowe w Całości')
```

```
plt.show()
```



# Zadanie 9.16

## Wykres kołowy (instrukcja)

Zakładamy, że prowadzisz analizę wydatków w swoim domowym budżecie. Masz informacje na temat procentowego udziału wydatków w kilku kategoriach. Twoim zadaniem jest stworzenie wykresu kołowego, który przedstawi te udziały procentowe. Poniżej znajdziesz dane dotyczące procentowego udziału w trzech kategoriach:

Jedzenie: 35%      Mieszkanie: 40%      Rozrywka: 25%

Treść zadania:

1. Stwórz wykres kołowy przedstawiający procentowy udział wydatków w trzech kategoriach.
2. Dodaj etykiety dla każdej kategorii na wykresie.
3. Skonfiguruj kolor kawałków wykresu.
4. Dodaj procentowe etykiety do kawałków wykresu.

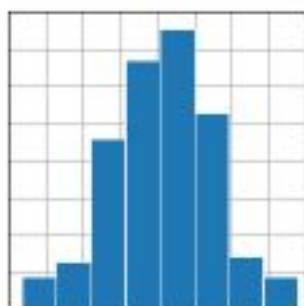




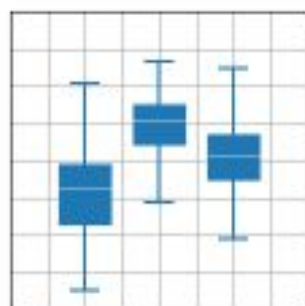
# Python Praca z Danymi

matplotlib – wykresy do analizy rozkładu,  
trendów i zależności

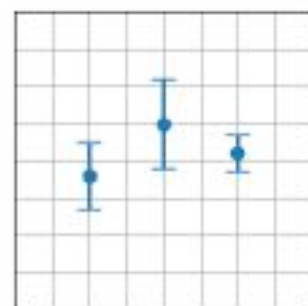
infoShare  
ACADEMY



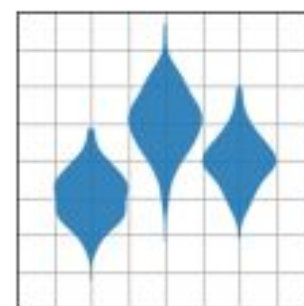
hist(x)



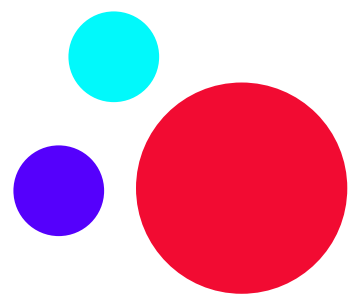
boxplot(X)



errorbar(x, y, yerr, xerr)



violinplot(D)

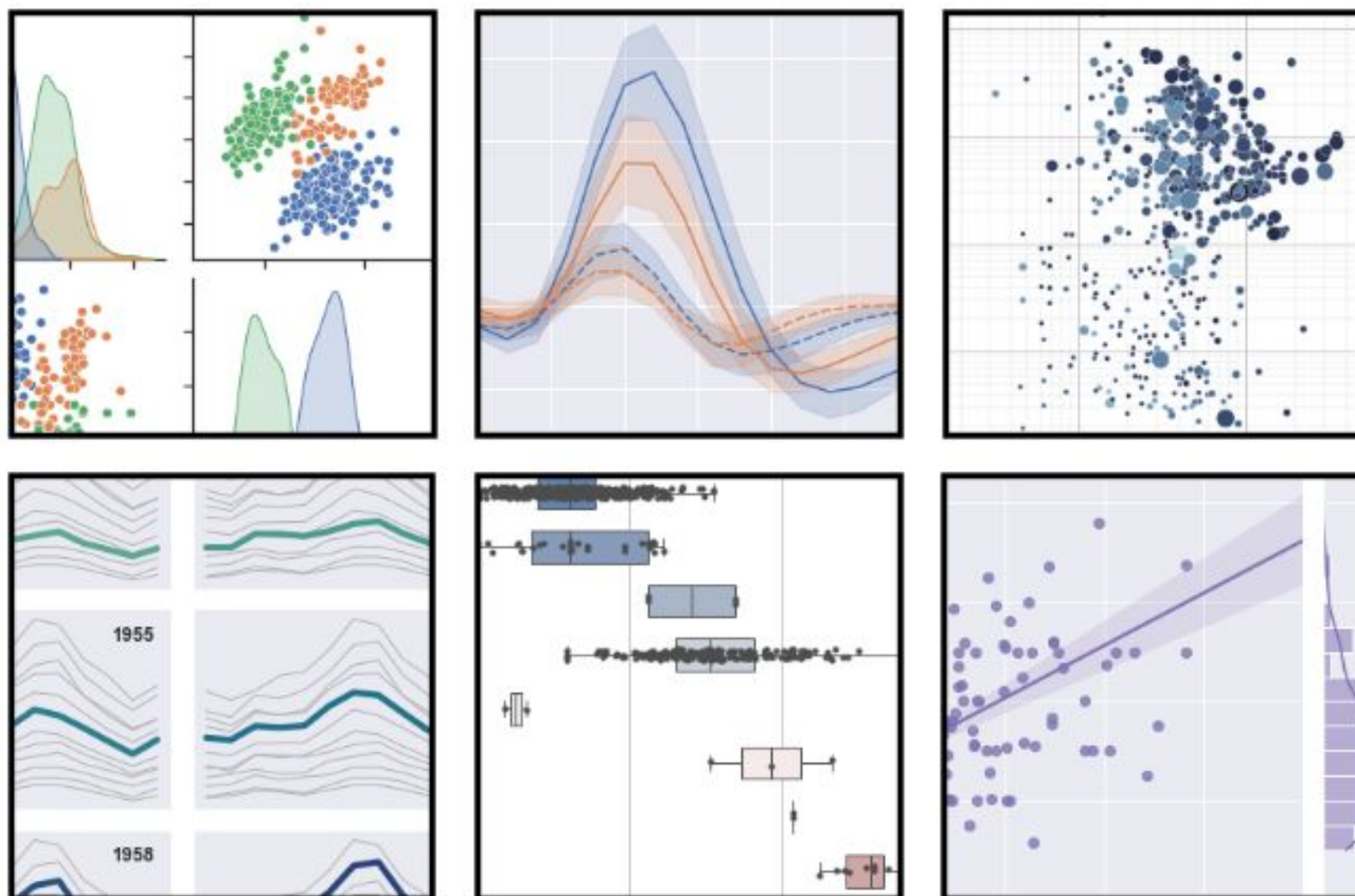


# Python Praca z Danyimi

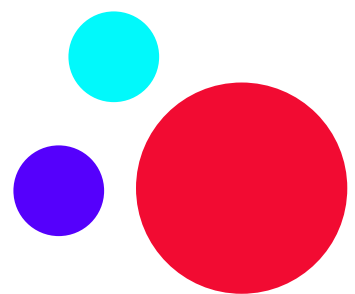
seaborn – nakładka matplotlib



info **Share**  
ACADEMY

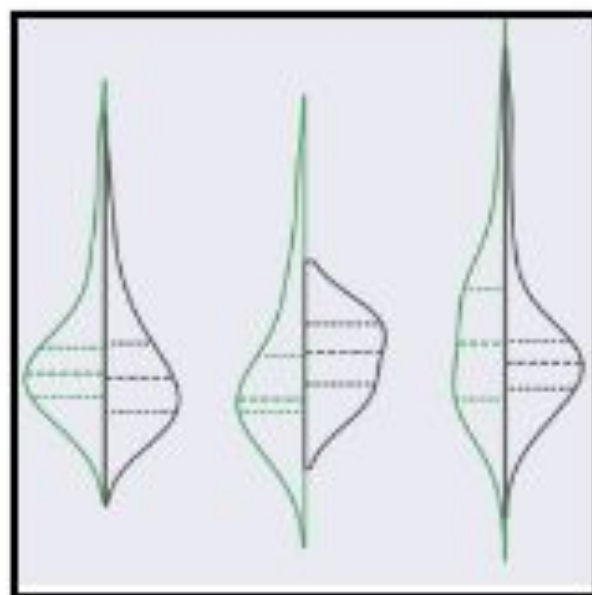




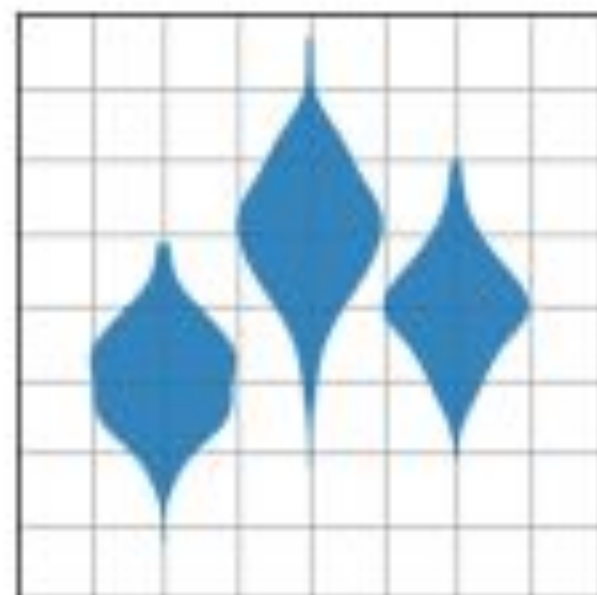


# Python Praca z Danyymi

## seaborn



?



info **Share**  
ACADEMY





# Python Praca z Danyymi

## seaborn - instalacja

`pip install seaborn`

`conda install seaborn`

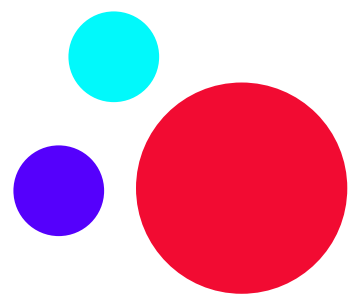
} Instalacja biblioteki

`import seaborn as sns`



Import biblioteki

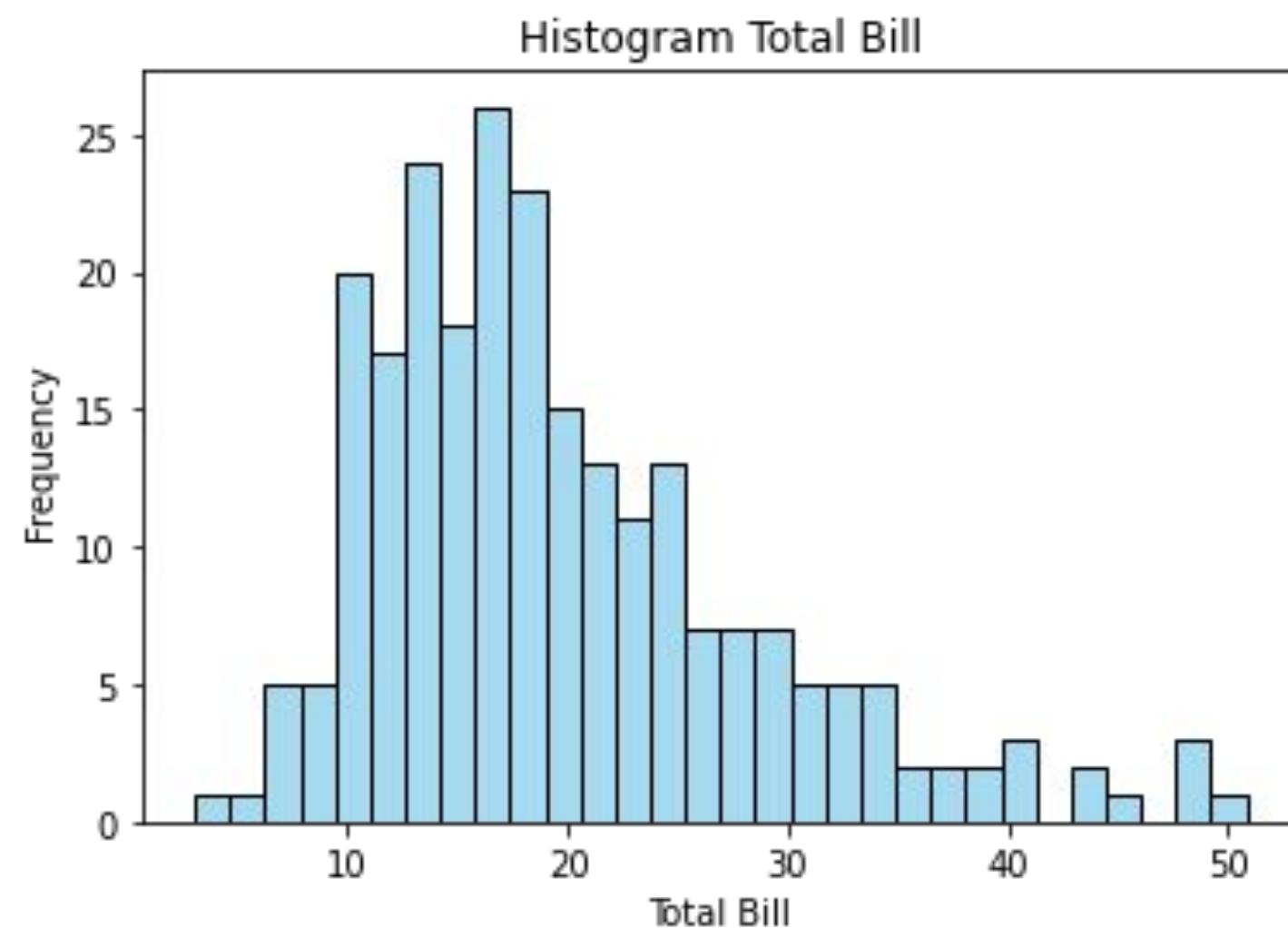


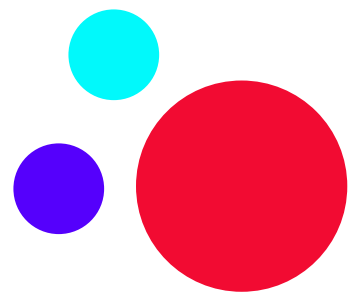


# Python Praca z Danyami

seaborn - histogram

info **Share**  
ACADEMY





# Python Praca z Danyymi

## seaborn - histogram

```
import seaborn as sns
```

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

```
data = sns.load_dataset("tips")
```

```
sns.histplot(data["total_bill"], bins=30, kde=False, color='skyblue')
```

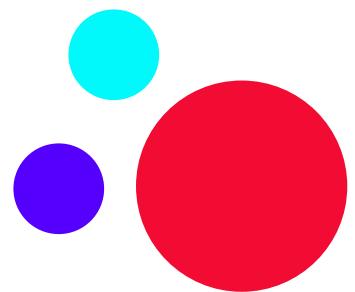
```
plt.title('Histogram Total Bill')
```

```
plt.xlabel('Total Bill')
```

```
plt.ylabel('Frequency')
```

```
plt.show()
```



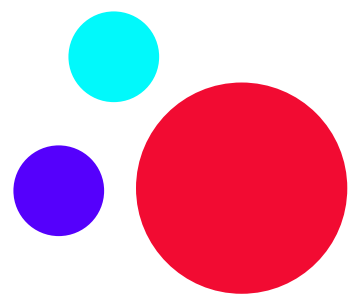


# Zadanie 9.17

## Histogram (instrukcja)

1. Załaduj zbiór danych "flights" z biblioteki Seaborn.
2. Przygotuj histogram dla kolumny "passengers".
3. Dostosuj liczbę przedziałów (bins) i kolor histogramu.
4. Dodaj tytuł wykresu oraz opisy osi.
5. Wyświetl ostateczny histogram.

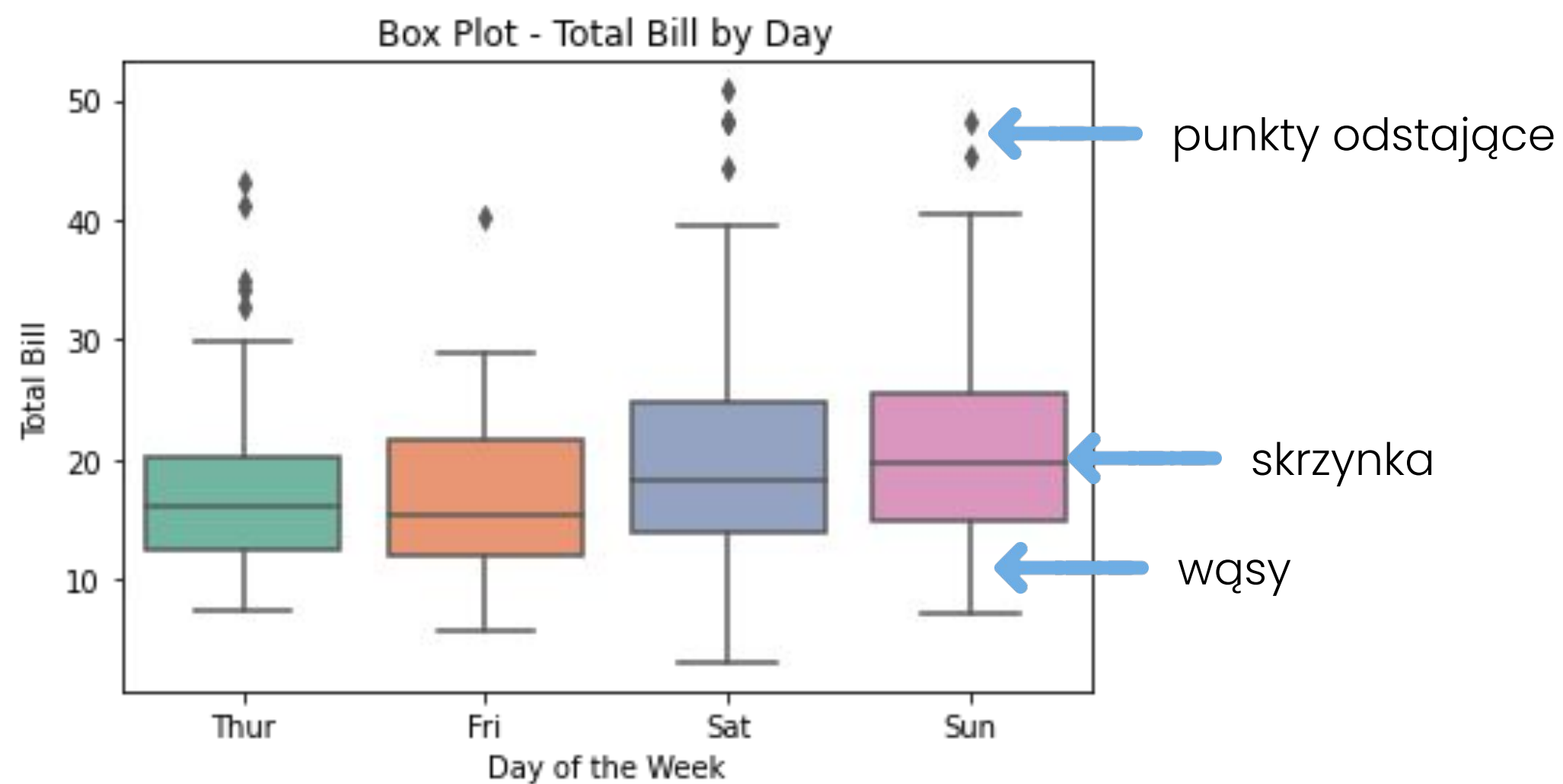


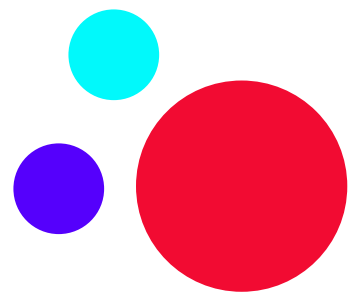


# Python Praca z Danymi

seaborn - boxploty

info **Share**  
ACADEMY





# Python Praca z Danymi

## seaborn – boxploty

```
import seaborn as sns
```

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

```
dane_restauracyjne = sns.load_dataset("tips")
```

```
sns.boxplot(x="day", y="total_bill", data=dane_restauracyjne, palette="Set2")
```

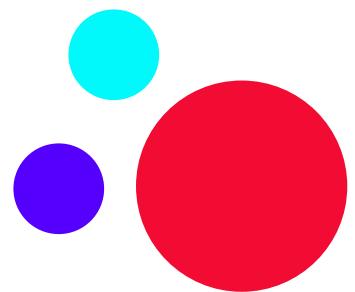
```
plt.title('Box Plot – Total Bill by Day')
```

```
plt.xlabel('Day of the Week')
```

```
plt.ylabel('Total Bill')
```

```
plt.show()
```





# Zadanie 9.18

## boxplot (instrukcja)

Przygotuj box plot, aby zobaczyć rozkład cen biletów w zależności od klasy podróźnej na statku Titanic.

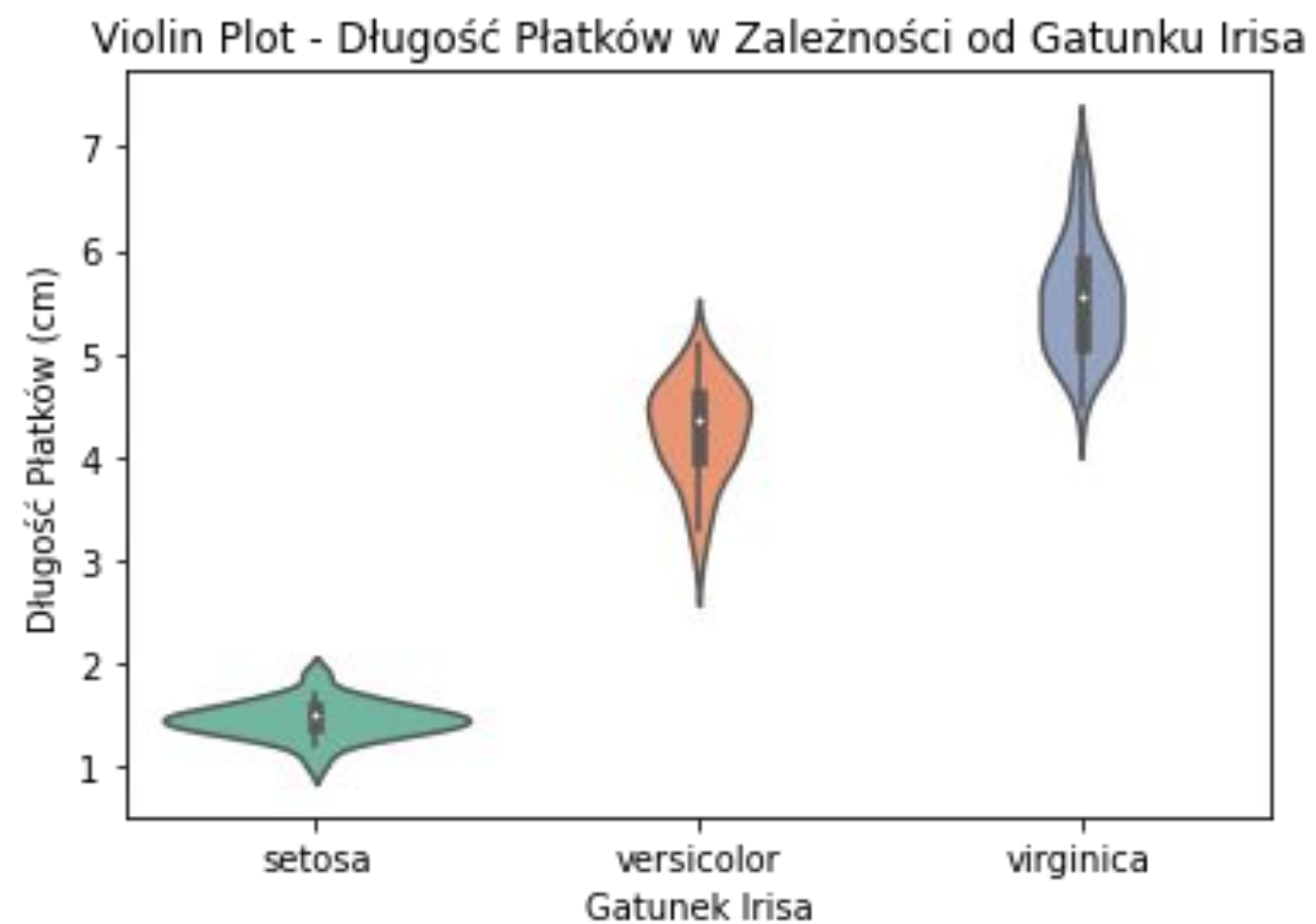
1. Załaduj zbiór danych "titanic" z biblioteki Seaborn.
2. Przygotuj box plot dla kolumny "fare" (cena biletu).
3. Podziel wykres na trzy pudełka, przedstawiające różne klasy podróźne.
4. Dostosuj paletę kolorów, aby podkreślić różnice między pudełkami.
5. Dodaj tytuł wykresu i opisy osi.
6. Wyświetl ostateczny box plot.

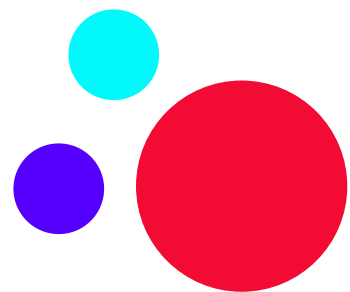


# Python Praca z Danymi

seaborn – violin ploty

infoShare  
ACADEMY





# Python Praca z Danymi

## seaborn – violin ploty

```
import seaborn as sns
```

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

```
iris_data = sns.load_dataset("iris")
```

```
sns.violinplot(x="species", y="petal_length", data=iris_data, palette="Set2")
```

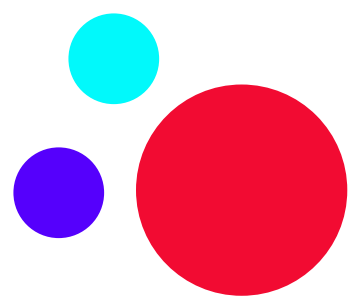
```
plt.title('Violin Plot – Długość Płatków w Zależności od Gatunku Irisa')
```

```
plt.xlabel('Gatunek Irisa')
```

```
plt.ylabel('Długość Płatków (cm)')
```

```
plt.show()
```





# Zadanie 9.19

## violin plot (instrukcja)

Przygotuj violin plot, aby zobaczyć rozkład długości płetw pingwinów w zależności od ich gatunku.

1. Załaduj zbiór danych "penguins" z biblioteki Seaborn.
2. Przygotuj violin plot dla kolumny "species" (gatunek) na osi X i "flipper\_length\_mm" (długość płetwy) na osi Y.
3. Dostosuj paletę kolorów, aby podkreślić różnice między gatunkami.
4. Dodaj tytuł wykresu i opisy osi.
5. Wyświetl ostateczny violin plot.

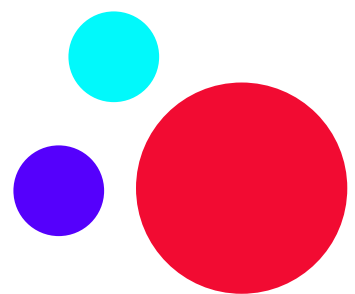


# Zadanie 9.20

## Podsumowanie (instrukcja)

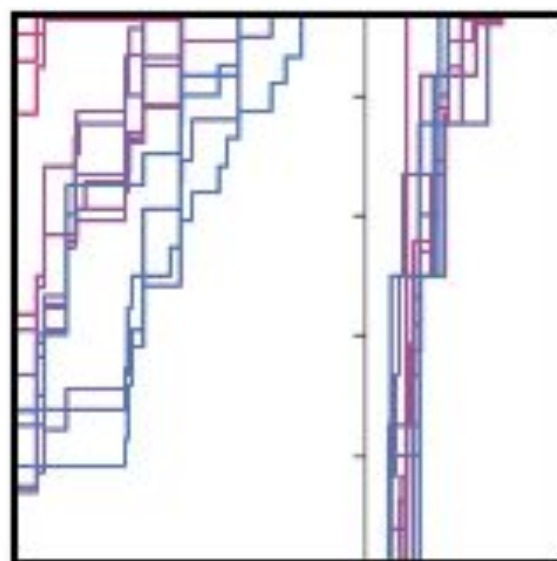
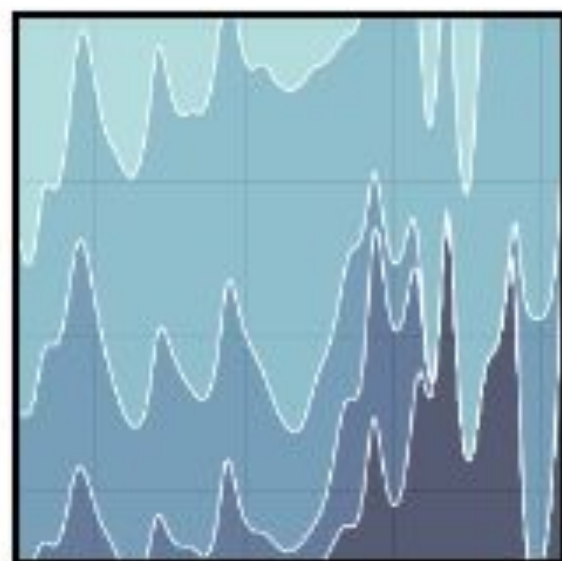
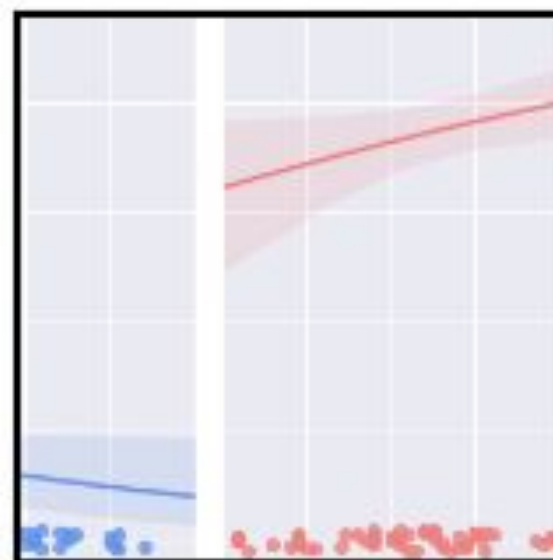
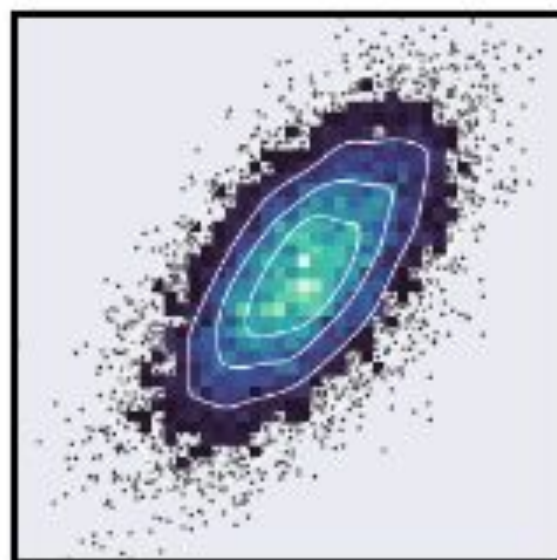
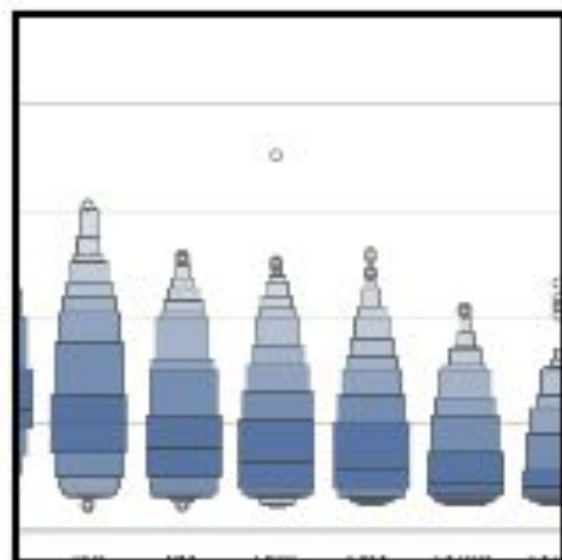
1. **Załadowanie Danych:** Wczytaj zbiór danych "car\_crashes" dostępny w bibliotece Seaborn.
2. **Analiza i Przetwarzanie Danych:** Sprawdź podstawowe informacje o danych, takie jak liczba wierszy, kolumn, typy danych itp. Zidentyfikuj i usuń ewentualne duplikaty. Przetwórz dane, jeśli to konieczne (np. zamień wartości null, dostosuj typy danych).
3. **Eksploracyjna Analiza Danych:** Stwórz histogram dla rozkładu liczby wypadków. Wygeneruj wykres punktowy, aby zobaczyć zależność między prędkością a średnią liczbą wypadków. Przy użyciu Seaborn, stwórz pair plot, aby zbadać korelacje między różnymi cechami.
4. **Statystyki i Grupowanie:** Oblicz średnią, medianę i odchylenie standardowe dla liczby wypadków. Utwórz wykres słupkowy przedstawiający średnią liczbę wypadków dla różnych stanów.
5. **Violin Plot:** Użyj Seaborn do stworzenia violin plotu przedstawiającego rozkład liczby wypadków w zależności od rodzaju alkoholu (prawne lub nielegalne).
6. **Podsumowanie:** Na podstawie przeprowadzonych analiz, sformułuj kilka wniosków dotyczących wypadków samochodowych.





# Python Praca z Danyymi

seaborn – podsumowanie



info **Share**  
ACADEMY





# Python Praca z Danymi

Stworzenie środowiska do analizy danych

```
conda create --name moje_srodowisko python=3.8
```

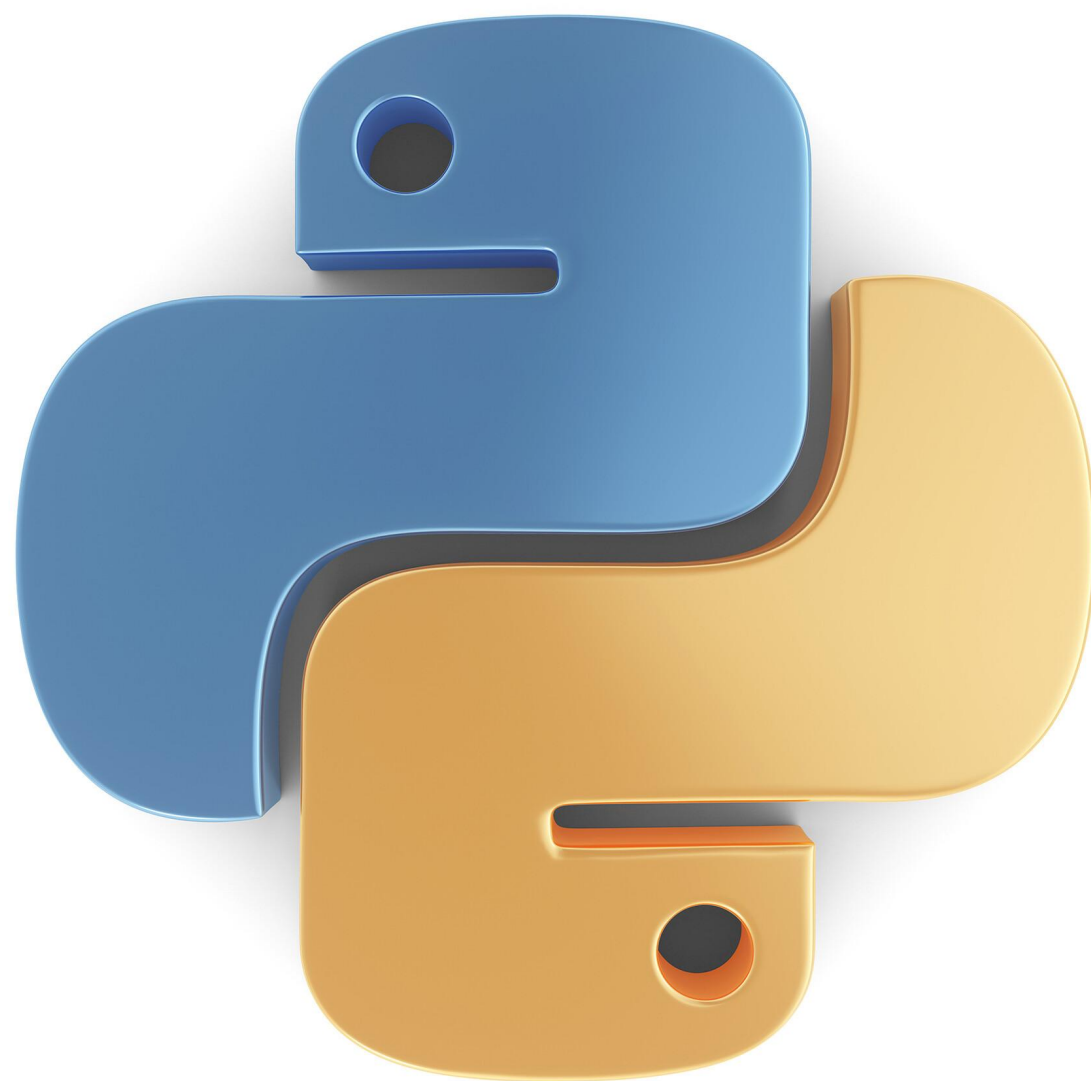
```
conda activate moje_srodowisko
```

```
conda install pandas numpy matplotlib seaborn
```



# Python Praca z Danymi

## Podsumowanie



**infoShare**  
ACADEMY