

# Wizualizacja danych - wykład 5

dr Piotr Jastrzębski

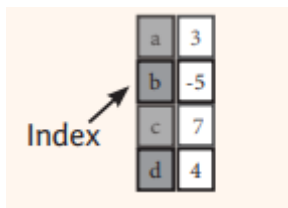
# Biblioteka Pandas

# Biblioteka Pandas

Import:

```
import pandas as pd
```

## Seria - Series

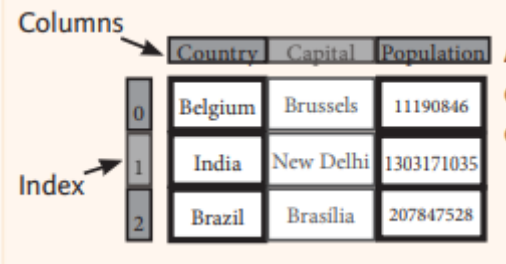


The diagram illustrates a Pandas Series. It consists of a vertical column of four cells. The first column contains index labels 'a', 'b', 'c', and 'd'. The second column contains numerical values '3', '-5', '7', and '4'. An arrow labeled 'Index' points to the 'b' label in the second row.

a	3
b	-5
c	7
d	4

Rysunek 1

## Ramka danych - DataFrame



The diagram illustrates a DataFrame structure. It features a grid of data with three columns and three rows. The columns are labeled 'Country', 'Capital', and 'Population'. The rows are indexed from 0 to 2. An arrow labeled 'Columns' points to the top row, and an arrow labeled 'Index' points to the first column.

	Country	Capital	Population
0	Belgium	Brussels	11190846
1	India	New Delhi	1303171035
2	Brazil	Brasilia	207847528

Rysunek 2

```
import pandas as pd
import numpy as np

s = pd.Series([3, -5, 7, 4])
print(s)
```

```
## 0    3
## 1   -5
## 2    7
## 3    4
## dtype: int64
```

```
print(s.values)
```

```
## [ 3 -5  7  4]
```

```
print(type(s.values))
```

```
## <class 'numpy.ndarray'>
```

```
t = np.sort(s.values)  
print(t)
```

```
## [-5  3  4  7]
```

```
print(s.index)
```

```
## RangeIndex(start=0, stop=4, step=1)
```

```
print(type(s.index))
```

```
## <class 'pandas.core.indexes.range.RangeIndex'>
```



```
s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])  
print(s)
```

```
## a      3  
## b     -5  
## c      7  
## d      4  
## dtype: int64
```

```
print(s['b'])
```

```
## -5
```

```
s['b'] = 8  
print(s)
```

```
## a      3  
## b      8  
## c      7  
## d      4  
## dtype: int64
```

```
print(s[s>5])
```

```
## b      8
```

```
## c      7
```

```
## dtype: int64
```

```
print(s*2)
```

```
## a      6  
## b     16  
## c     14  
## d      8  
## dtype: int64
```

```
print(np.sin(s))
```

```
## a      0.141120  
## b      0.989358  
## c      0.656987  
## d     -0.756802  
## dtype: float64
```

```
d = {'key1': 350, 'key2': 700, 'key3': 70}
s = pd.Series(d)
print(s)
```

```
## key1      350
## key2      700
## key3       70
## dtype: int64
```

```
d = {'key1': 350, 'key2': 700, 'key3': 70}
k = ['key0', 'key2', 'key3', 'key1']
s = pd.Series(d, index=k)
print(s)
```

```
## key0      NaN
## key2      700.0
## key3       70.0
## key1      350.0
## dtype: float64
```

```
pd.isnull(s)
```

```
## key0      True  
## key2      False  
## key3      False  
## key1      False  
## dtype: bool
```

```
pd.notnull(s)
```

```
## key0      False  
## key2      True  
## key3      True  
## key1      True  
## dtype: bool
```

```
s.isnull()
```

```
## key0      True  
## key2      False  
## key3      False  
## key1      False  
## dtype: bool
```

```
s.notnull()
```

```
## key0      False  
## key2      True  
## key3      True  
## key1      True  
## dtype: bool
```



```
s.name = "Wartość"  
s.index.name = "Klucz"  
print(s)
```

```
## Klucz  
## key0      NaN  
## key2      700.0  
## key3      70.0  
## key1      350.0  
## Name: Wartość, dtype: float64
```

```
data = {'Country': ['Belgium', 'India', 'Brazil'],  
        'Capital': ['Brussels', 'New Delhi', 'Brasília'],  
        'Population': [11190846, 1303171035, 207847528]}  
frame = pd.DataFrame(data)  
print(frame)
```

```
##      Country      Capital  Population  
## 0  Belgium  Brussels    11190846  
## 1   India  New Delhi   1303171035  
## 2   Brazil  Brasília   207847528
```

```
df = pd.DataFrame(data, columns=['Country', 'Capital',  
                                'Population'])  
print(df)
```

```
##      Country      Capital  Population  
## 0  Belgium  Brussels    11190846  
## 1   India  New Delhi    1303171035  
## 2   Brazil  Brasília    207847528
```

```
print(df.iloc[[0],[0]])
```

```
##      Country  
## 0  Belgium
```

```
print(df.loc[[0], ['Country']])
```

```
##      Country  
## 0  Belgium
```

```
print(df.loc[2])
```

```
## Country          Brazil
## Capital           Brasília
## Population       207847528
## Name: 2, dtype: object
```

```
print(df.loc[:, 'Capital'])
```

```
## 0    Brussels
## 1    New Delhi
## 2    Brasília
## Name: Capital, dtype: object
```

```
print(df.loc[1, 'Capital'])
```

```
## New Delhi
```

```
print(df[df['Population']>1200000000])
```

```
##   Country      Capital  Population
## 1   India  New Delhi  1303171035
```

```
print(df.drop('Country', axis=1))
```

```
##      Capital  Population
## 0  Brussels   11190846
## 1  New Delhi  1303171035
## 2  Brasília  207847528
```

```
print(df.shape)
```

```
## (3, 3)
```

```
print(df.index)
```

```
## RangeIndex(start=0, stop=3, step=1)
```

```
print(df.columns)
```

```
## Index(['Country', 'Capital', 'Population'], dtype='object')
```



```
print(df.info())
```

```
## <class 'pandas.core.frame.DataFrame'>
## RangeIndex: 3 entries, 0 to 2
## Data columns (total 3 columns):
## Country      3 non-null object
## Capital      3 non-null object
## Population    3 non-null int64
## dtypes: int64(1), object(2)
## memory usage: 152.0+ bytes
## None
```

```
print(df.count())
```

```
## Country      3
## Capital      3
## Population    3
## dtype: int64
```

# Uzupełnianie braków

```
s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])  
s2 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])  
print(s+s2)
```

```
## a      10.0  
## b       NaN  
## c       5.0  
## d       7.0  
## dtype: float64
```

```
print(s.add(s2, fill_value=0))
```

```
## a      10.0  
## b      -5.0  
## c       5.0  
## d       7.0  
## dtype: float64
```

```
print(s.mul(s2, fill_value=2))
```

```
## a      21.0  
## b     -10.0  
## c     -14.0  
## d      12.0  
## dtype: float64
```

# “Tidy data”

# Koncepcja

Koncepcja czyszczenia danych (ang. tidy data):

- WICKHAM, Hadley . Tidy Data. Journal of Statistical Software, [S.l.], v. 59, Issue 10, p. 1 - 23, sep. 2014. ISSN 1548-7660. Available at: <https://www.jstatsoft.org/v059/i10>. Date accessed: 25 oct. 2018. doi:<http://dx.doi.org/10.18637/jss.v059.i10>.

# Zasady "czystych danych"

Idealne dane są zaprezentowane w tabeli:

Imię	Wiek	Wzrost	Kolor oczu
Adam	26	167	Brązowe
Sylwia	34	164	Piwne
Tomasz	42	183	Niebieskie

Na co powinniśmy zwrócić uwagę?

- jedna obserwacja (jednostka statystyczna) = jeden wiersz w tabeli/macierzy/ramce danych
- wartości danej cechy znajdują się w kolumnach
- jeden typ/rodzaj obserwacji w jednej tabeli/macierzy/ramce danych

# Przykłady nieuporządkowanych danych

Imię	Wiek	Wzrost	Brązowe	Niebieskie	Piwne
Adam	26	167	1	0	0
Sylwia	34	164	0	0	1
Tomasz	42	183	0	1	0

**Nagłówki kolumn muszą odpowiadać cechom, a nie wartościom zmiennych.**

# Kod do analizy

https:  
[//gist.github.com/pjastr/309281eedf2ca5d0425b26e8d12eaa6f](https://gist.github.com/pjastr/309281eedf2ca5d0425b26e8d12eaa6f)



# Bibliografia

- [https://s3.amazonaws.com/assets.datacamp.com/blog\\_assets/PandasPythonForDataScience.pdf](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PandasPythonForDataScience.pdf), dostęp online 5.4.2019.