Języki programowania – Python
Biblioteka Pandas

Marzena Nowakowska

Wydział Zarządzania i Modelowania Komputerowego
Politechnika Świętokrzyska

Budynek C, p. 3.21

spimn@tu.kielce.pl

Informacja podstawowa

Moduł (pakiet, biblioteka) **Pandas** pracuje głównie ze strukturami tabelarycznymi, które mogą zawierać dane różnych typów. Należy go zaimportować, aby móc korzystać z jej zasobów, np:

```
import pandas as pd          # częstsze, odwołanie: pd.funkcja()
```

Pandas zapewnia dostęp do danych (wykorzystywanych w analityce) o typach:

- **Series** (szereg, seria) dla danych jednowymiarowych
- **DataFrame** (ramka danych) dla wielowymiarowych

Każdy obiekt ww. typu definiuje obiekt, który jest kontenerem i może zawierać dane różnych typów. Jednak zaleca się zachowanie unifikacji; dla ramki danych w obrębie każdej kolumny.

Uwaga: może istnieć potrzeba instalacji biblioteki, co można zrealizować z wykorzystaniem programu pip w oknie konsoli Spydera:

```
pip install pandas
```

Obiekt *Series* w Pandas

Pandas Series (seria) to **jednowymiarowa struktura danych**, podobna do kolumny w tabeli. Najważniejsze cechy:

- przechowuje dane różnych typów (np. liczby, teksty, daty),
- ma indeks, którego wartości identyfikują każdy element serii danych (indeksowanie domyślne: liczby całkowite od zera),
- pozwala wykonywać operacje matematyczne i analizy,
- jest podstawowym budulcem obiektu DataFrame (który składa się z wielu Series).

Przykłady tworzenia serii danych

```
import pandas as pd
s1 = pd.Series([1, 3.2, 0.1, 2.])
s2 = pd.Series([[1, 3.2, 0.1, 2],'alhambra', 33, 3])
s3 = pd.Series({'f':1, 'g':5, 'h':3, 5:[1,2,3]})
s4 = pd.Series(([[1, 3.2, 0.1, 2]])
```

Index	0
0	1
1	3.2
2	0.1
3	2

s1[0] → 1

Index	0
0	[1, 3.2, 0.1, 2]
1	alhambra
2	33
3	3

s2[1]→"alhambra"

Index	0
f	1
g	5
h	3
5	[1, 2, 3]

s3['g'] → 5

Index	0
0	[1, 3.2, 0.1, 2]

s4[0] → [1, 3.2, 0.1, 2]

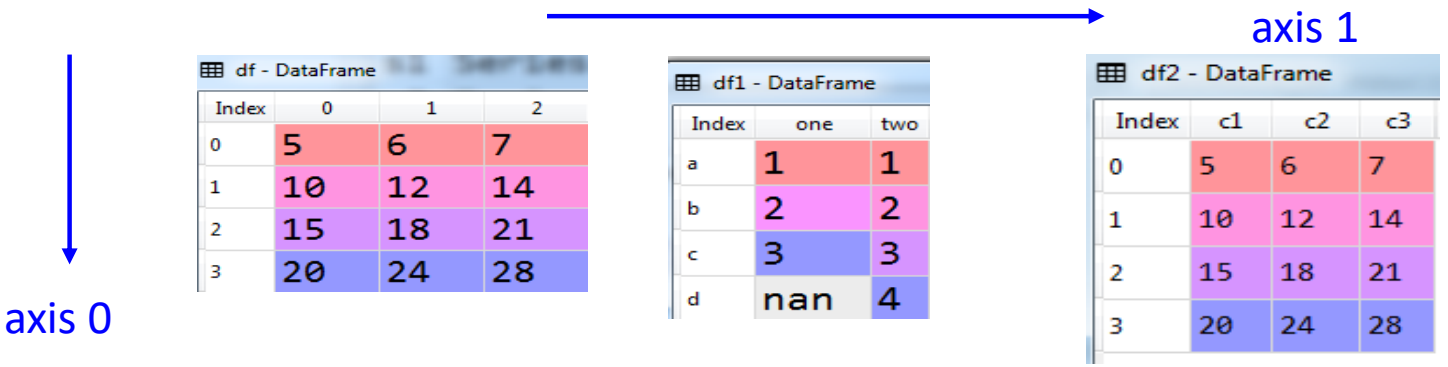
Obiekt *DataFrame* w Pandas

Pandas DataFrame (ramka danych) to **dwuwymiarowa tabela danych** w bibliotece Pandas, podobna do arkusza kalkulacyjnego lub tabeli w bazie danych. Najważniejsze cechy:

- składa się z wielu kolumn typu Series, z własnymi nazwami,
- posiada indeksy wierszy i etykiety kolumn,
- umożliwia filtrowanie, grupowanie i analizę danych,
- obsługuje różne typy danych w różnych kolumnach.

Przykłady tworzenie ramek danych

```
import pandas as pd
df=pd.DataFrame([[ 5,  6,  7], [10, 12, 14], [15, 18, 21], [20, 24, 28]])
d = { "one": pd.Series([1.0, 2.0, 3.0], index=["a", "b", "c"]), "two": pd.Series([1.0, 2.0, 3.0, 4.0], index=["a", "b", "c", "d"]) }; df1=pd.DataFrame(d)
df2=pd.DataFrame(([[ 5,  6,  7], [10, 12, 14], [15, 18, 21], [20, 24, 28]]), columns=("c1", "c2", "c3"))
```



Dostęp do danych *DataFrame* w Pandas

df - DataFrame

Index	0	1	2
0	5	6	7
1	10	12	14
2	15	18	21
3	20	24	28

df1 - DataFrame

Index	one	two
a	1	1
b	2	2
c	3	3
d	nan	4

df2 - DataFrame

Index	c1	c2	c3
0	5	6	7
1	10	12	14
2	15	18	21
3	20	24	28

Dostęp do kolumny:

DataFrame[nr_kolumny], DataFrame["nazwa_kolumny"]

Dostęp do wiersza – za pomocą atrybutu *iloc* i indeksu lub *loc* i etykiety:

DataFrame.iloc[nr_wiersza], DataFrame.loc["nazwa_wiersza"]

Dostęp do ponumerowanego elementu ramki:

DataFrame.iat[nr_wiersza, nr_kolumny]

DataFrame.loc[nr_wiersza, nr_kolumny]

Dostęp do nazwanego elementu ramki:

DataFrame.at["nazwa_wiersza", "nazwa_kolumny"]

DataFrame.loc[nr_wiersza, "nazwa_kolumny"]

Przykłady:

df.iat[0, 1] → 6
df2.loc[1][2] → 14
df1.at["b", "two"] → 2
df1.loc['d']['two'] → 4
df1.loc['d', 'two'] → 4
df2.loc[0]['c1'] → 5
df2.loc[0, 'c1'] → 5

df[0]

0
5
10
15
20

df1["one"]

one
1
2
3
nan

df.iloc[1]

1	10	12	14

df1.loc["c"]

c	3	3

Wybrane składowe *DataFrame* w Pandas

DataFrame.index wykaz indeksów (nazw lub numerów) ramki danych

DataFrame.columns wykaz kolumn (nazw lub numerów) ramki danych

DataFrame.set_index() nadanie wierszom nazw (indeksów)

DataFrame.rename() zmiana nazw kolumn i wierszy

DataFrame.loc[len(DataFrame.index)]=[*wart_1*, …, *wart_k*]
dodanie wiersza na końcu ramki

DataFrame.append(DataFrame)
dodanie ramki danych do innej ramki danych

DataFrame['*Nowa_kolumna*']=[*wart_1*, …, *wart_n*]
dodanie kolumny do ramki danych

Data.Frame.insert(), DataFrame.assign()
dodanie kolumny lub kolumn

Data.Frame.drop() usunięcie kolumn lub wierszy (parametr *axis*)

Odczyt pliku Excela do ramki danych
Wybrane parametry

pandas.read_excel(*file*, *sheet_name=0*, *header=0*, *usecols=None*, *index_col=None*)

Odczyt pojedynczego arkusza lub zbioru arkuszy Excela do obiektu typu odpowiednio *DataFrame* lub słownika arkuszy. Parametry:

file – wartość typu *str* definiująca plik do odczytu

sheet_name – łańcuch (np. *'Arkusz1'*, liczba całkowita (np. 1) lub lista (np. [0, 1, *'ArkuszXX'*] określające arkusz (lub arkusze) do wczytania

header – liczba całkowita lub lista liczb całkowitych. Wykaz numerów wierszy arkusza, których zawartość ma być wykorzystana jako nazwy kolumn ramki danych.

usecols – łańcuch. Określa, które kolumny arkusza mają być wczytane do ramki danych

index_col – liczba/tekst lub lista liczb/tekstów. Ustala indeks ramki (numer lub nazwa kolumny) na etapie wczytywania pliku (pierwsza kolumna ma numer 0)

Przykłady (uwaga: ustalić folder roboczy)

```
dc = pd.read_excel("vehicles.xlsx", sheet_name=["Old Europe", "New Europe"],
        header=0, usecols="A,H:L") # słownik z ramkami danych
df1 = dc["Old Europe"] : df2=dc["New Europe"]  # wyodrębnienie ramek danych
df3 = pd.read_excel("vehicles.xlsx",  sheet_name=3,  header=0, index_col=0,
      usecols=[0, 2, 3,5])  # wczytanie ramki danych
```

Zapis ramki danych do pliku Excela
Wybrane parametry

DataFrame.to_excel(*excel_writer, sheet_name='Sheet1’, columns=None, header=True, index=True*)

Zapisuje ramkę danych do pliku *Excela*. Parametry:
excel_writer – ścieżka do pliku lub istniejący obiekt *ExcelWriter*.
sheet_name – nazwa arkusza typu *str*, do którego jest zapisywana ramka danych. Domyślnie *'Sheet1*'.
columns – sekwencja (np. lista napisów) definiująca kolumny do zapisu
header – dana logiczna informująca, czy w pliku mają być zapisane nazwy kolumn (*True*) czy nie (False)
index – dana logiczna informująca, czy w pliku mają być zapisane indeksy (*True*) czy nie (False)

Uwaga: DataFrame.to_excel() nie ma wbudowanego parametru, który automatycznie ostrzega lub blokuje zapis, jeżeli plik już istnieje.

Przykład (uwaga: ustalić folder roboczy)

```
df4 = df2.drop([2011, 2012, 2013], axis=1) # operacja po kolumnach
df4.to_excel('wynik.xlsx', sheet_name = 'Nowa Europa wybrane')
```

Kontrola zapisu do pliku Excela

pandas.ExcelWriter(path-file_name, engine='openpyxl', mode='w',
if_sheet_exists='error')

Klasa (narzędzie) w bibliotece pandas, do tworzenia i zapisywania plików Excel (XLSX, XLS) z danych zawartych w ramkach danych. Działa jak pośrednik między Pandas a plikiem Excel. Parametry:

path-file_name – ścieżka do pliku wraz z jego nazwą.

engine – silnik zapisu

mode – tryb zapisu:

- 'w' (write) nadpisanie pliku
- 'a' (append) dopisanie arkusza do istniejącego skoroszytu.

if_sheet_exists – działanie, jeżeli arkusz o podanej nazwie już istnieje:

- 'error' zgłoszenie błędu, jeżeli arkusz istnieje (domyślnie); chroni przed niezamierzonym nadpisaniem danych, działa w trybie dopisywani a (mode="a"), najczęściej z silnikiem openpyxl
- 'new' utworzenie nowego arkusza z unikalną nazwą (np. Sheet1)
- 'replace' — nadpisz istniejący arkusz.

Uwaga: pandas.ExcelWriter nie ma wbudowanego parametru, który automatycznie ostrzega lub blokuje zapis, jeśli plik już istnieje.

Rozwiązanie dopisania ramki danych do pliku Excela - podanie nazwy dołączanego arkusza

```
import pandas as pd
# Nazwa pliku Excela
plik = "Skoroszyt.xlsx"

# Nazwa nowego arkusza
arkusz = "Ramka"

dfs = pd.DataFrame({
    "name": ["Anna", "Bartek", "Celina"],
    "age": [25, 30, 22]
})

# Zapis do nowego arkusza w istniejącym pliku
with pd.ExcelWriter(plik, mode="a", if_sheet_exists="new") as writer:
    dfs.to_excel(writer, sheet_name = arkusz, index=False)
```

Pętla po ramce danych

```
df  = pd.read_excel("titanic.xlsx")
```

Pętla po ramce danych z wykorzystaniem atrybutu *index* (id wierszy)

```
for ind in df.index:
        print(df['name'][ind], df['age'][ind])
```

Pętla po ramce danych z wykorzystaniem metody *iterrows* (zwraca *Series* dla każdego wiersza ramki); iterowanie po wierszach

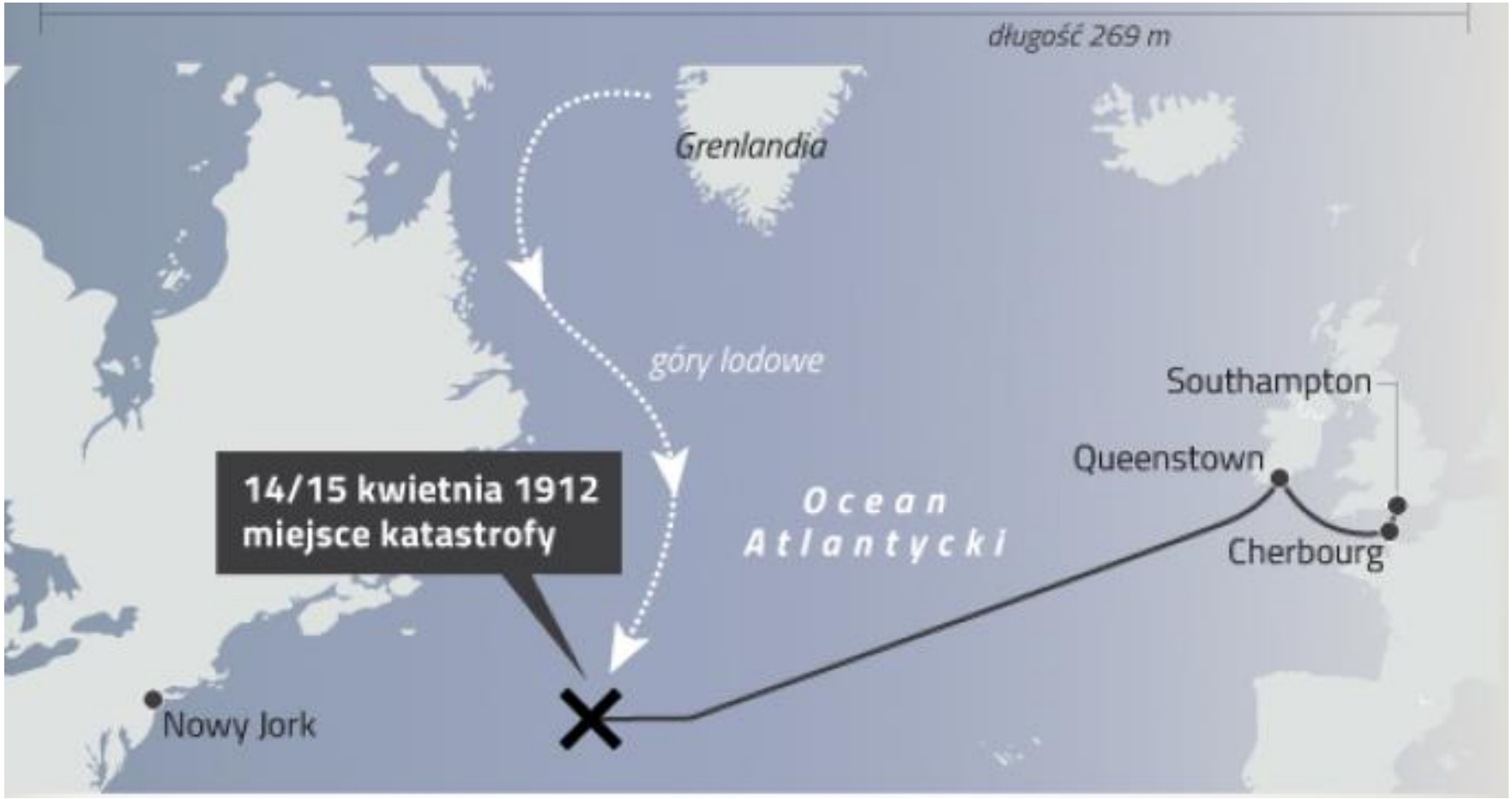
```
for index, row in df.iterrows():
        print(row["name"], row["age"])
```

metoda df.iterrows() zwraca kolejne pary (index, row), gdzie:

- index → to etykieta (indeks) wiersza z ramki danych
- row → to obiekt typu pandas.Series, który zawiera dane z tego wiersza (analogia do wiersza w tabeli bazy danych; dostęp do elementów tego wiersza jest poprzez nazwy kolumn)

Demo: Zad_06b-Titanic.py

Trasa Titanica



Katastrofę przeżyło 710 (32%) pasażerów spośród 2 224

Zbiór danych *titanic*

https://analityk.edu.pl/zbior-danych-titanic-analiza-w-python/#google_vignette

Opis zbioru danych o pasażerach Titanica

survived – czy pasażer przeżył (1-tak, 0-nie)
class – klasa, którą podróżował (1-najlepsza)
name – imię, nazwisko
sex – płeć
age – wiek
sibsp – Number of Siblings (liczba rodzeństwa)/Spouses Aboard (Małżonkowie na pokładzie)
parch – liczba dzieci lub rodziców
ticket – numer biletu
fare – cena biletu
cabin – numer kajuty
embarked – port, w którym osoba wsiadła na pokład (S = Southampton; C = Cherbourg; Q = Queenstown)
boat – numer łódki, którą osoba się ewakuowała
body – numer identyfikacyjny znalezionego ciała

Dygresja. Wartości logiczne mają odpowiedniki liczbowe:
False → 0 oraz *True* →1