Biblioteka pandas, część 1.

1. Struktury danych w bibliotece pandas.

Serie danych (Series) – to jednowymiarowa tablica z etykietami, która może przechowywać dowolny typ danych. Dokumentacja dla serii danych https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html

Ramka danych (DataFrame) – dwuwymiarowa struktura z etykietami, mogąca przechowywać kolumny z różnymi typami danych. Dokumentacja dla ramek - https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

Listing 1 – tworzenie Series i DataFrames

```
import pandas as pd
import numpy as np

#Series
s = pd.Series([1, 3, 5, np.nan, 6, 8])
print(s)
s = pd.Series([10, 12, 8, 14], index=['Ala', 'Marek',
'Wiesiek', 'Eleonora'])
print(s)


#DataFrame
#tworzenie dataframe na podstawie słownika
data = {'Kraj': ['Belgia', 'Indie', 'Brazylia'],
        'Stolica': ['Bruksela', 'New Delhi', 'Brasilia'],
        'Populacja': [11190846, 1303171035, 207847528]}
df = pd.DataFrame(data)
print(df)
#DataFrame przechowuje typ dla każdej kolumny co możemy
sprwadzić wpisując
print(df.dtypes)
#możemy również w prosty sposób stworzyć serię danych - czyli
próbki dla kolejnych
#dat, pomiarów czasu
daty = pd.date_range('20210324', periods=5)
print(daty)
df = pd.DataFrame(np.random.randn(5,4), index=daty,
columns=list('ABCD'))
print(df)

#biblioteka Pandas umożliwia również tworzenie DataFrame'ów z
zewnętrznych źródeł danych
#CSV, odczyt i zapis
df = pd.read_csv('dane.csv', header=0, sep=";",decimal=',')
print(df)
df.to_csv('plik.csv', index=False)
```

```
#Excel - wymagane są biblioteki xlrd oraz openpyxl
#przed importem trzeba je zainstalować
import xlrd
import openpyxl

xlsx = pd.ExcelFile('dane.xlsx')
df = pd.read_excel(xlsx, header=0)
print(df)
df.to_excel('wyniki.xlsx', sheet_name='arkusz pierwszy')
```

2. Pobieranie danych ze struktur

Pandas dostarcza wielu sposobów na pobieranie pojedynczych wartości, kolumn, wierszy lub zbiorów wartości na podstawie parametrów. Poniżej znajdują się 2 listingi z najbardziej popularnymi metodami.

```
import pandas as pd
import numpy as np


s = pd.Series([10, 12, 8, 14], index=['Ala', 'Marek',
'Wiesiek', 'Eleonora'])
print(s)

data = {'Kraj': ['Belgia', 'Indie', 'Brazylia'],
        'Stolica': ['Bruksela', 'New Delhi', 'Brasilia'],
        'Populacja': [11190846, 1303171035, 207847528]}
df = pd.DataFrame(data)
print(df)

#pojedynczy element serii, odnosimy się do nazwy indeksu
print(s['Wiesiek'])
#możemy również dostać się do wartości serii jak do pola klasy
print(s.Wiesiek)

#pojedynczy elemenet ramki danych, tak jak przy cięciu tablic,
tylko tu jest oparte na indeksach
print(df[0:1])
print("")
#kolumna po etykiecie
print(df['Populacja'])
#pobieranie pojedynczej wartości po indeksie wiersza i kolumny
print(df.iloc[[0][0]])
#pobieranie wartości po indeksie wiersza i etykiecie kolumny
print(df.loc[[0],["Kraj"]])
print(df.at[0,"Kraj"])
#podobnie jak w przypadku serii można odwoływać się do kolumn
jak do pól klasy
#dodatkowo print jest wywoływany jak w pętli dla każdego
elementu danej kolumny
print('kraj: ' + df.Kraj)
```

```
#Pandas posiada również funkcje pozwalające na losowe
pobieranie elementów lub
#w odniesieniu do procentowej wielkości całego zbioru

#jeden losowy element
print(df.sample())
#n losowych elementów
print(df.sample(2))
#ilość elementów procentowo, uwaga na zaokrąglenie
print(df.sample(frac=0.5))

#jeżeli potrzeba nam więcej próbek niż znajduje się w zbiorze
i dopuszczamy duplikaty
#to możemy użyć parametru replace, który będzie losował z
powtórzeniami
print(df.sample(n=10, replace=True))

#zamiast wyświetlać całą kolekcje możemy wyświetlić określoną
ilość elementów od początku kolekcji
#lub od jej końca
print(df.head())
print(df.head(2))
print(df.tail(1))

# Pandas jest też w stanie wyświetlić statystykę dla wartości,
które dana kolekcja zawiera, o ile są jakieś kolumny
# z danymi numerycznymi
print(df.describe())
# transpozycja to zmienna T kolekcji, podobnie jak w Numpy
print(df.T)
```

Więcej przykładów pobierania elementów poprzez indeksowanie można znaleźć pod adresem https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html

Listing 3 – filtrowanie, grupowanie i agregowanie danych

```
s = pd.Series([10, 12, 8, 14], index=['Ala', 'Marek',
'Wiesiek', 'Eleonora'])
print(s)

data = {'Kraj': ['Belgia', 'Indie', 'Brazylia'],
        'Stolica': ['Bruksela', 'New Delhi', 'Brasilia'],
        'Populacja': [11190846, 1303171035, 207847528]}
df = pd.DataFrame(data)
print(df)
#wyświetla dane z serii gdzie wartość większa od 1
print(s[(s>9)])
# nieco inny efekt można osiągnąć wykorzystując funkcję where,
```

```
która zwraca kolekcję w oryginalnej wielkości
# (liczebności) elementów, ale wartości nie spełniające
warunku uzupełnia wartością NaN
print(s.where(s > 10))
#możemy też podać wartość, która zostanie podstawiona zamiast
NaN
print(s.where(s>10, 'za duże'))
# jeszcze inna własność where pozwala na modyfikowanie
oryginalnego zbioru (domyślnie zwaracana jest kopia)
seria = s.copy()
seria.where(seria > 10, 'za duże', inplace=True)
print("########")
print(seria)

#wyświetla dane z serii gdzie wartość nie jest większa od 10
print(s[~(s > 10)])
#warunki możemy też łączyć
print(s[(s < 13) & (s > 8)])

#warunki dla pobierania DataFrame
print(df[df['Populacja']>1200000000])
#bardziej skomplikowane warunki
print(df[((df.Populacja > 1000000) & (df.index.isin([0,2])))])

#inny przykład z listą dopuszczalnych wartości oraz isin
zwracająca wartości boolowskie
print('#########')
szukaj = ['Belgia', 'Brasilia']
print(df.isin(szukaj))

#zmiana, usuwanie i dodawanie danych

#w przypadku serii możemy dodać/zmienić wartość poprzez
odwołanie się do elementu serii przez klucz (indeks)
s['Wiesiek'] = 15
print(s.Wiesiek)
s['Alan'] = 16
print(s)

# podobna operacja dla DataFrame ma nieco inny efekt - wartość
ustawiona dla wszystkich kolumn
df.loc[3] = 'dodane'
print(df)
# ale można dodać wiersz w postaci listy
df.loc[4] = ['Polska', 'Warszawa', 38675467]
print(df)

# usuwanie danych można wykonać przez funkcję drop, ale
pamiętajmy, że operacja nie wykonuje się in-place więc
# zwracana jest kopia DataFrame z usuniętymi wartościami
```

```
new_df = df.drop([3])
print(new_df)
# więc jeżeli chcemy zmienić pierwotny zbiór dodajemy parametr
inplace=True
df.drop([3], inplace=True)
print(df)
# można usuwać również całe kolumny po nazwie indeksu, ale
wykonanie tej komendy uniemożliwi
# wykonanie dalszego kodu (można przetstować po zakomentowaniu
dalszej części listingu)
#df.drop('Kraj', axis=1, inplace=True)

# do DataFrame możemy dodawać również kolumny zamiast wierszy
df['Kontynent'] = ['Europa', 'Azja', 'Ameryka Południowa',
'Europa']
print(df)

# Pandas ma również własne funkcje sortowania danych
print(df.sort_values(by='Kraj'))

# grupowania
grouped = df.groupby(['Kontynent'])
print(grouped.get_group('Europa'))
# można też jak w SQL czy Excelu uruchomić funkcje agregujące
na danej kolumnie
print(df.groupby(['Kontynent']).agg({'Populacja':['sum']}))

# podobny mechanizm to sumy częściowe i tabele przestawne
znane z Excela, które w Pandas również mają swoje odpowiedniki
print("_____ sumy częściowe ______")
tabela =
pd.pivot_table(df,values=['Populacja'],index=['Kontynent'],col
umns=['Kraj'],aggfunc=np.sum,margins=True)
print(tabela.stack('Kraj'))
```