NIERELACYJNE ROZWIĄZANIA BAZODANOWE

WYKŁAD 6

UNIWERSYTET GDAŃSKI

AGENDA

- Pozyskiwanie informacji z Internetu
- Kolekcje stron internetowych w bazie NoSQL
- Przykłady pracy w NoSQL z poziomu języka Python

POZYSKIWANIE INFORMACJI Z INTERNETU

POZYSKIWANIE I PRZETWARZANIE BIG DATA

Statystyka cen

Ceny biletów lotniczych

Ceny nieruchomości

Elementy strony WWW

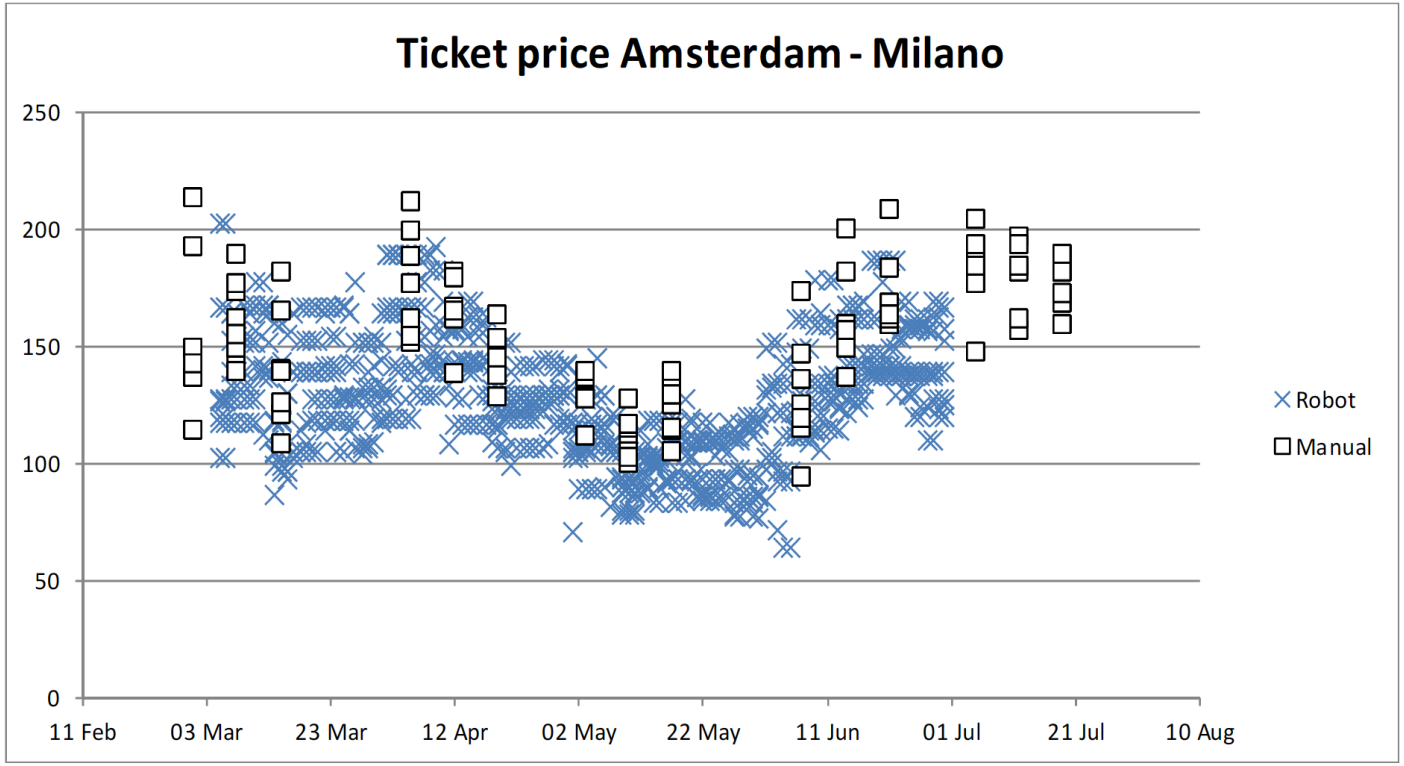
Media społecznościowe

Opinie / komentarze

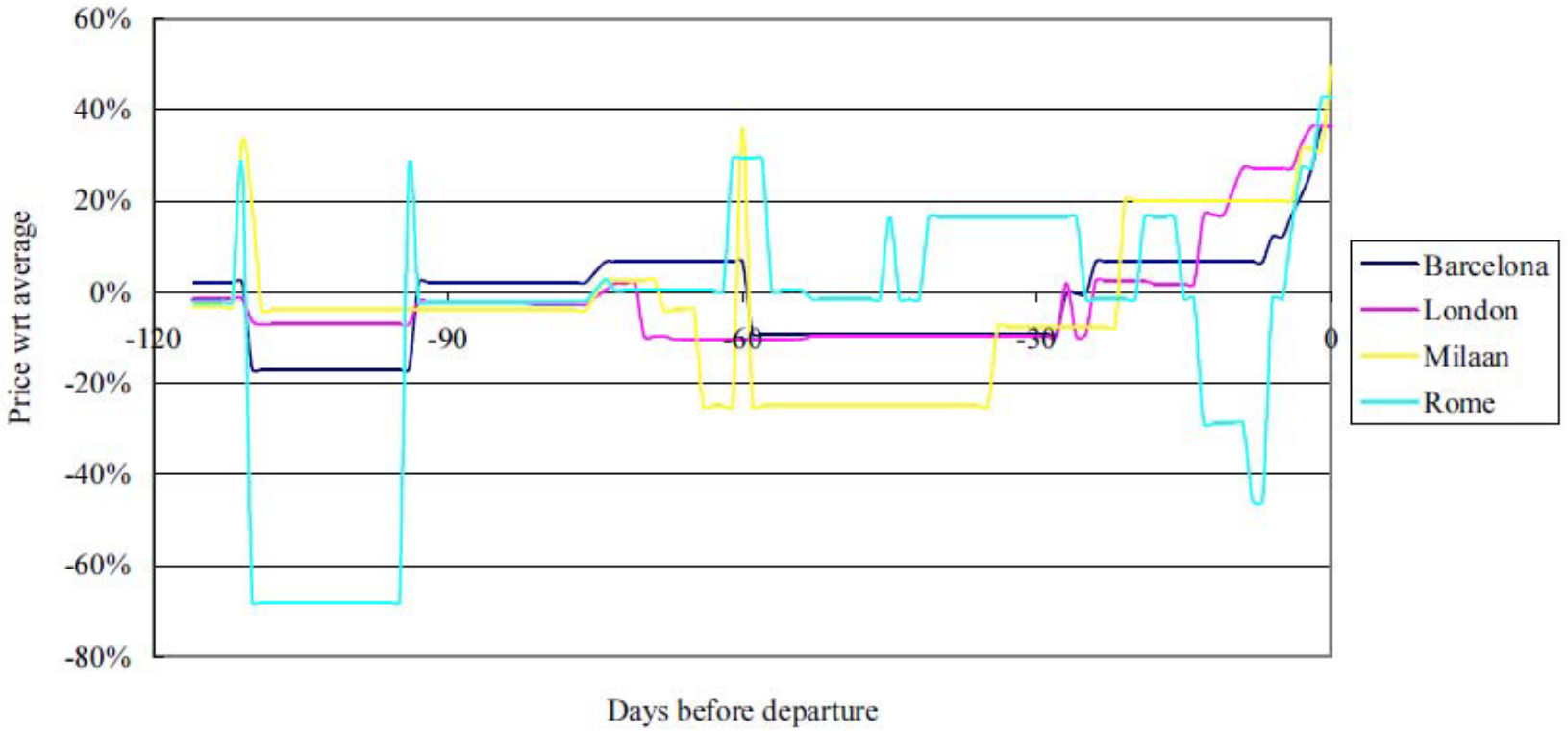
Inne informacje

PORÓWNANIE CPI I PRICESTATS

CENY BILETÓW LOTNICZYCH



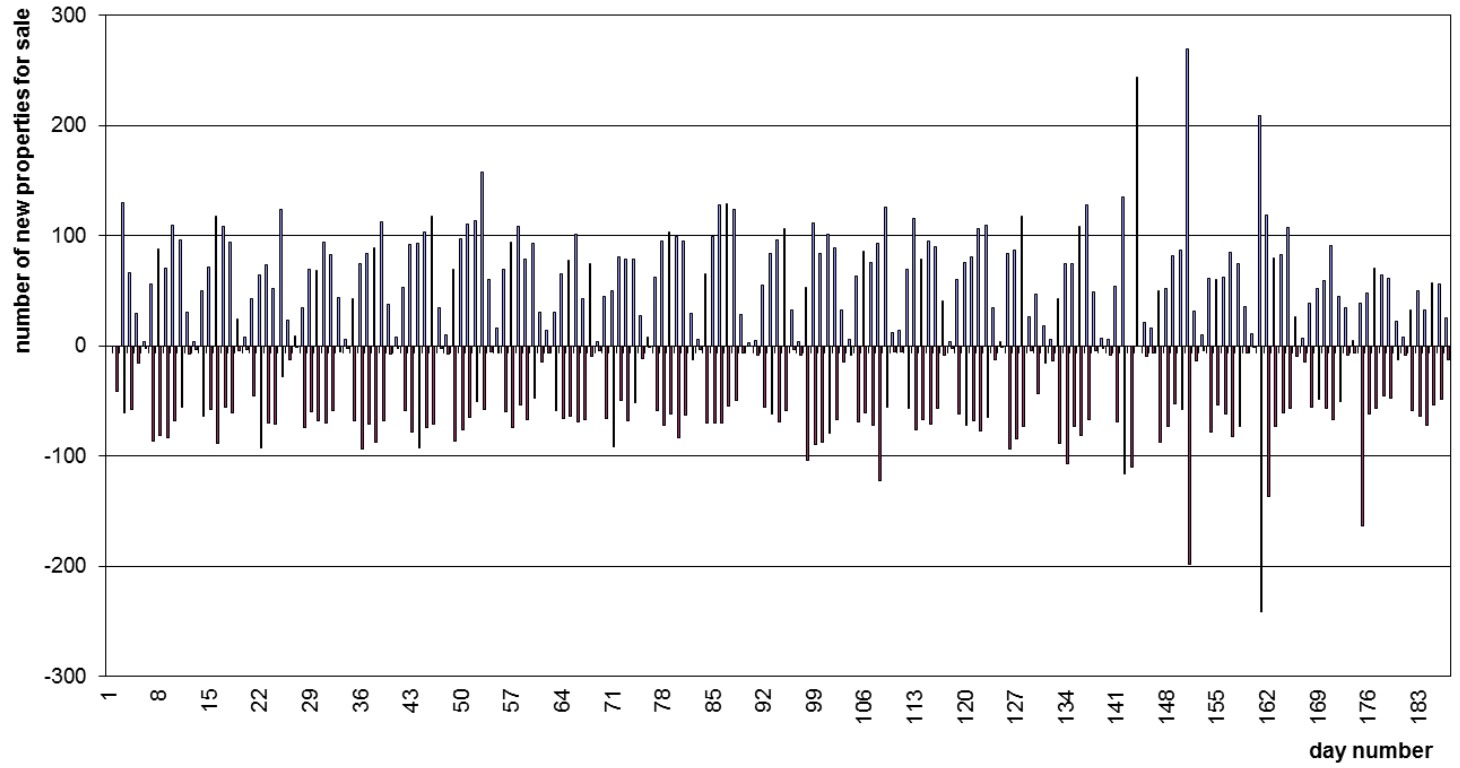
CENY BILETÓW LOTNICZYCH W ZALEŻNOŚCI OD LICZBY DNI PRZED WYLOTEM



DANE NT. CEN NIERUCHOMOŚCI

- Dane te są wykorzystywane do tworzenia statystyki rynku nieruchomości (dane takie jak adres, liczba pokoi, powierzchnia, cena).
- Konieczne jest przy tym zbadanie jak często przybywa nowych nieruchomości (podaż).
- Problem jaki może się pojawić to blokada robotów przez strony internetowe.
- Aspekty prawne pozyskiwania danych.

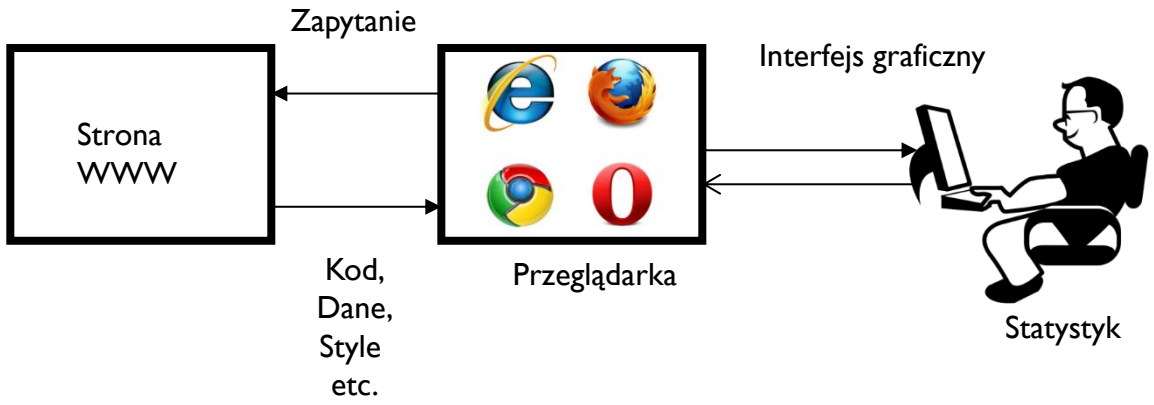
LICZBA NOWYCH OFERT NIERUCHOMOŚCI



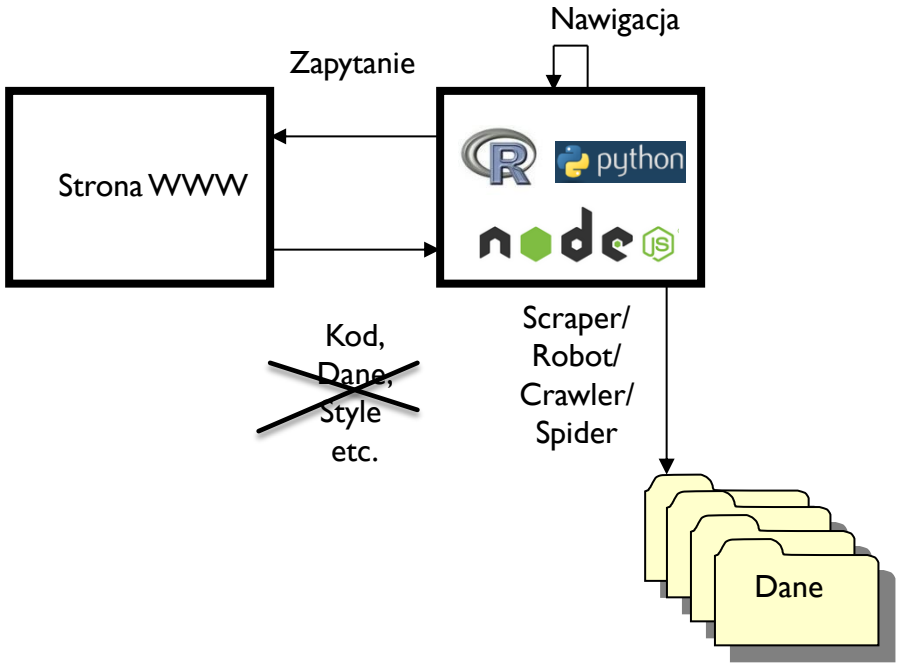
STATYSTYKA CEN

- Zbierane dane mogą dotyczyć np. nazwy, opisu, kategorii, koloru oraz rozmiaru.
- Kilkaset tys. obserwacji dziennie dla różnych stron internetowych.
- Testowane są już narzędzia do rozpoznawania produktów na podstawie obrazów.
- Ceny online vs. ceny w sklepach stacjonarnych.
- Rozpoznawanie produktów i cen na podstawie zdjęć paragonów sklepowych.

DZIAŁANIE – SPOSÓB RĘCZNY



DZIAŁANIE – SPOSÓB AUTOMATYCZNY



PODSUMOWANIE DOTYCZĄCE WEB SCRAPINGU

- Web scraping zależy od rozpoznania właściwych i aktualnych stron internetowych (pierwszy etap oceny stron).
- Zmiana wzorca wyszukiwania danych może spowodować, że szereg czasowy będzie nieporównywalny. Dynamiczny charakter Internetu sprawia, że należy monitorować prawidłowe działanie wzorców.
- Od strony prawnej należy zabezpieczyć się poprzez informowanie właścicieli stron internetowych o prowadzonych działaniach w zakresie webscrapingu.
- Pomocna jest również strona www.robotstxt.org, gdzie zawarto najnowsze informacje nt. oznaczania stron internetowych dla robotów.
- Ta sama analiza danych – np. klastry – może dostarczać każdego dnia innych wyników.
- Problem masowego pozyskania danych – na znajdujących się w bazie ok. 500 tys. polskich stron prawidłowo można pobrać ok. 390 tys.

ĆWICZENIE 1. PLIK ROBOTS.TXT

- http://www.robotstxt.org/robotstxt.html

- Ćwiczenie.
  - Sprawdzić następujące strony WWW pod względem możliwości przeprowadzenia webscrapingu:
    - http://allegro.pl
    - http://ug.edu.pl
    - http://www.otodom.pl

PRZYKŁAD 1. UKŁAD STRONY WWW

- Przykładowe strony WWW zawierające różną strukturę z danymi:
  - Tabele
  - Kontenery
  - Klasy
  - CSS
  - HTML
  - JSON
  - …

KONSTRUKCJA STRONY WWW

<a>	hiperłącze
<p>	akapit
<h?>	nagłówek, gdzie "?" 1-6
	sekcja
	linia tekstu
/	nienumerowana lista, element listy
<table>	tabela
<tr> / <td>	wiersz tabeli/komórka tabeli
<body>	ciało strony – widoczne w przeglądarce
<html>	całość dokumentu HTML

- http://www.w3schools.com/tags/

PRAWNE ASPEKTY

- Ochrona danych osobowych
  - General Data Protection Regulation (GDPR) https://gdpr.eu/tag/gdpr/
- Ochrona baz danych
  - USTAWA z dnia 27 lipca 2001 r. o ochronie baz danych
  - Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases
- Plik robots.txt

PRZYKŁAD 2. INSPEKCJA DANYCH NA STRONIE INTERNETOWEJ

- Inspekcja danych na stronie internetowej poprzez przeglądanie źródła strony oraz za pomocą „Inspect element”.

DEMONSTRACJA – BIBLIOTEKA REQUESTS

- Załadowanie danych z Internetu
  - import requests
  - data=requests.get("http://wikipedia.com")
  - print(data.text)

- Czy coś jest na stronie internetowej?
  - if "abc" in data.text:
  - print("Na stronie jest abc")

ZADANIE NA PODSTAWIE DEMONSTRACJI

- r = requests.get("http://_____")

- Sprawdzić, czy poniższe słowa kluczowe występują na stronie?
  - html
  - body

ĆWICZENIE 2. INSPEKCJA STRONY INTERNETOWEJ

- Znaleźć jakie elementy na stronie:
  - wp.pl
- odpowiadają za wyświetlanie tematów aktualności?

ETAP I

1. import requests
2. from bs4 import BeautifulSoup
3. page=requests.get("http://....")

ETAP 2

1. soup = BeautifulSoup(page.text, 'html.parser’)
2. dane = soup.findAll('', attrs={'':''})
3. for d in dane:
4. print (d.text.strip())

ETAP 3

```
1. import csv
2. from datetime import datetime
3. with open('data.csv', 'a') as csv_file:
4.     writer = csv.writer(csv_file)
5.     for d in dane:
6.         writer.writerow ([d.text.strip(),datetime.now()])
```

WEB SCRAPING W PYTHONIE

KOLEKCJE STRON INTERNETOWYCH W NOSQL

UNIWERSYTET GDAŃSKI

DEMONSTRACJA. BAZA NOSQL

- Dokumenty jako dane nieustrukturyzowane.

- Przykład - strona internetowa.

- {
- ”url”:”…”,
- ”contents”:”…”,
- ”date”:”…”
- }

BAZA STRON WWW W NOSQL

NA ZIELONO ZAZNACZONO JSON JAKO PRZYKŁAD DOKUMENTU NOSQL
HIERARCHIA: BAZA.KOLEKCJA.DOKUMENT

```
import requests
from datetime import datetime
bazastron=klient['bazaStron']
strony=['http://wp.pl','http://wzr.pl','http://ug.edu.pl']
kolekcja=bazastron.strony
for s in strony:
    r=requests.get(s)
    strona = {
        "url":s,
        "html":r.text,
        "data":datetime.today().strftime("%d/%m/%Y")
    }
    rezultat=kolekcja.insert_one(strona)
    print("Wstawiono ",s,rezultat.inserted_id)
print("Zakończyłem web scraping")
```

ZADANIA NOSQL – CZĘŚĆ I

- Zadanie 1. Zaimportuj bibliotekę pymongo i zdefiniuj obiekt klienta.
- Zadanie 2. Podłącz się do lokalnego serwera (localhost) na porcie 27017, protokół mongodb.

- Zadanie 3. Zdefiniuj lub utworz bazę baza_TwojeImieN
- Zadanie 4. Wstaw dane studenta: nr indeksu 200200, imię: Jan, nazwisko: Kowalski
- Zadanie 5. Analogicznie dodaj trzech kolejnych studentów.
- Zadanie 6. Wyszukaj studenta o nazwisku Nowak.
- Zadanie 7. Wyszukaj wszystkich studentów o nazwisku Nowak.
- Zadanie 8. Zaimportuj dane z pliku XLS do bazy mongodb.
- Zadanie 9. Wyświetl zawartość kolekcji.

ROZWIĄZANIA

- #Zadanie 1. Zaimportuj bibliotekę pymongo i zdefiniuj obiekt klienta.
- from pymongo import MongoClient
- klient=MongoClient()
- #Zadanie 2. Podłącz się do lokalnego serwera (localhost) na porcie
- #27017, protokół mongodb.
- klient=MongoClient('mongodb://localhost:27017')
- #Zadanie 3. Zdefiniuj lub utworz bazę baza_TwojeImieN
- bd=klient['baza_JM']

ROZWIĄZANIA

```
#Zadanie 4. Wstaw dane studenta: nr indeksu 200200,
#imię: Jan, nazwisko: Kowalski
kolekcja=bd.student
dokument1={
    'indeks':'200200',
    'imie':'Jan',
    'nazwisko':'Kowalski'
}
rezultat=kolekcja.insert_one(dokument1)
print('Wstawiono',rezultat.inserted_id)
dane=bd.student.find()
for d in dane:
    print(d)
```

ROZWIĄZANIA

```
#Zadanie 5. Analogicznie dodaj trzech kolejnych studentów.
dokument2={
    'indeks':'876765',
    'imie':'Anna',
    'nazwisko':'Wiśniewska'
}
dokument3={
    'index':'453234',
    'imie':'Marian',
    'nazwisko':'Lewandowski'
}
dokument4={
    'indeks':'876543',
    'imie':'Maria',
    'nazwisko':'Kowalska'
}
dokumenty=[dokument2,dokument3,dokument4]
rezultat=kolekcja.insert_many(dokumenty)
print('Wstawiono',rezultat.inserted_ids)
```

ROZWIĄZANIA

```
dane=bd.student.find()
for d in dane:
    print(d)
```

ROZWIĄZANIA

- # Zadanie 6. Wyszukaj studenta o nazwisku Nowak.
- wyszukiwanie=kolekcja.find_one({'nazwisko':'Kowalski'})
- print(wyszukiwanie)
- # Zadanie 7. Wyszukaj wszystkich studentów o nazwisku Nowak.
- wynik=kolekcja.find({'nazwisko':'Kowalski'})
- for w in wynik:
- print(w['imie'])

ROZWIĄZANIA

```
# Zadanie 8. Zaimportuj dane z pliku XLS do bazy mongodb.
import pandas as pd
df=pd.read_excel('dane.xlsx')
df.head()
for index, row in df.iterrows():
    dokument={
        'imie':row['imie'],
        'nazwisko':row['nazwisko'],
        'wiek':row['wiek']
    }
    rezultat=kolekcja.insert_one(dokument)
    print('wstawiono',rezultat.inserted_id)
print('Zakończono wstawianie')
```

ROZWIĄZANIA

- bd=klient['baza']
- df=pd.DataFrame.from_records(bd.student.find())
- # Zadanie 9. Wyświetl zawartość kolekcji.
- wynik=kolekcja.find({})
- for w in wynik:
- print(w)

ZADANIA NOSQL – CZĘŚĆ II

- CZĘŚĆ II. Przygotowanie stron WWW.
- Zadanie 1. Przygotuj kolekcję w celu pobierania stron internetowych i ich zawartości - adres, opis, zawartosc, data itp.
- Zadanie 2. Zrób web scraping stron internetowych i zapisz je w bazie NoSQL.
- Zadanie 3. Wyświetl jedną z podanych stron internetowych.
- Zadanie 4. Przeprowadź analizę danych.

ROZWIĄZANIA

- #CZĘŚĆ I. Przygotowanie stron WWW.
- #Zadanie 1. Przygotuj kolekcję w celu pobierania
- #stron internetowych i ich zawartości - adres, opis, zawartosc, data itp.
- from pymongo import MongoClient
- import pandas as pd
- import requests
- from datetime import datetime
- import time
- # Python req 0.1
- class WWWtoNoSQL:
- def __init__(self):
- self.klient=MongoClient("mongodb://localhost:27017")
- self.db=self.klient['S5205_2024'] # baza danych
- self.kolekcja=self.db.www # kolekcja

ROZWIĄZANIA

- #Zadanie 2. Zrób prosty web scraping stron internetowych
- #i zapisz je w bazie NoSQL. "}
- def pobierzStrone(self,url): ######################## Z M I A N A
- headers={"User-Agent":"Mozilla/5.0 (Windows NT 10.0;Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/122.0.0.0 Safari/537.36"}
- strona={
- "url":url,
- "content":requests.get(url,headers=headers).text, ####### requests ... .text
- "date":datetime.now()
- }
- rezultat=self.kolekcja.insert_one(strona) # wstawienie dokumentu JSON do NoSQL
- print("Wstawiono",url,rezultat.inserted_id) # potwierdzenia wstawienia

ROZWIĄZANIA

- # tworzymy obiekt z klasy
- w=WWWtoNoSQL()
- # wywołujemy metodę
- w.pobierzStrone("https://ug.edu.pl")

ROZWIĄZANIA

- #Zadanie 3. Wyświetl jedną z podanych stron internetowych.
- klient=MongoClient("mongodb://localhost:27017")
- db=klient['S5205_2024']

- # znalezienie wszystkich URLi pobranych stron
- kursor=db.www.find({})
- for k in kursor:
- print(k['url'])

ROZWIĄZANIA

- df=pd.DataFrame(list(db.www.find({})))
- df.head()
- df.tail(2)
- df.url.count()
- df.groupby('url').count()
- def rozmiar_strony(x):
- return len(str(x))
- df['rozmiar_strony']=df['content'].apply(rozmiar_strony)
- df.nlargest(3,'rozmiar_strony')
- df.nsmallest(3,'rozmiar_strony')

ROZWIĄZANIA

```
import numpy as np # potrzebne do określenia wartości null, czyli NaN
def czy_link_youtube(content):
    if "youtube.com" in content:
        return 1
    elif "youtu.be" in content:
        return 1
    else:
        return np.nan
```

PYTANIE

- Podczas pobierania całych stron inte, jakiego rodzaju informacje nie muszą się znaleźć w bazie NoSQL?
  - rozmiar strony internetowej
  - data web scrapingu
  - adres URL scrapowanej strony
  - renderowana zawartość strony internetowej