

Python API, Web Scraping



Python API, Web Scraping

Agenda

1 Idea Web Scrapingu oraz pobierania danych z API

2 Dane w formacie .json

3 Łączenie z API

4 Korzystanie z beautifulsoup

5 Praca z plikami tekstowymi



Python API, Web Scraping

Agenda

1 Idea Web Scrapingu oraz pobierania danych z API

2 **Dane w formacie .json**

3 Łączenie z API

4 Korzystanie z beautifulsoup

5 Praca z plikami tekstowymi



Python API, Web Scraping

Agenda

1 Idea Web Scrapingu oraz pobierania danych z API

2 Dane w formacie .json

3 **Łączenie z API**

4 Korzystanie z beautifulsoup

5 Praca z plikami tekstowymi



Python API, Web Scraping

Agenda

1 Idea Web Scrapingu oraz pobierania danych z API

2 Dane w formacie .json

3 Łączenie z API

4 **Korzystanie z BeautifulSoup**

5 Praca z plikami tekstowymi



Python API, Web Scraping

Agenda

1 Idea Web Scrapingu oraz pobierania danych z API

2 Dane w formacie .json

3 Łączenie z API

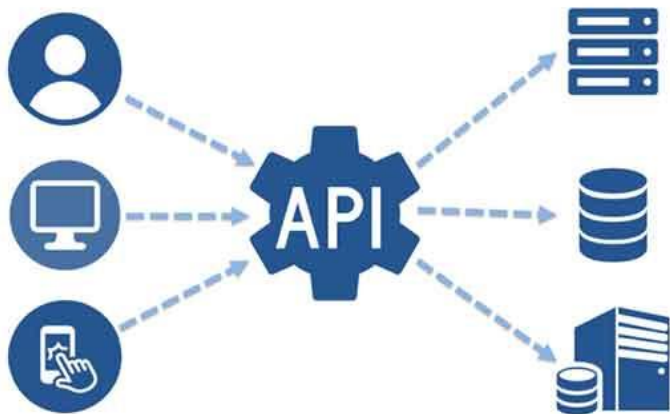
4 Korzystanie z beautifulsoup

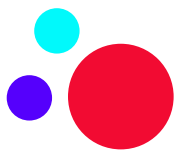
5 **Praca z plikami tekstowymi**



Python API, Web Scraping API

Application Programming Interface





Python API, Web Scraping

API – działanie



Application/Customer



Waiter/API



Kitchen/Server

info **Share**
ACADEMY



Python API, Web Scraping

API

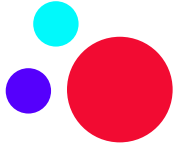
1. Integracja aplikacji
2. Usługi webowe
3. Platformy społecznościowe
4. Bankowość i finanse
5. Internet Rzeczy (IoT)
6. Rozwój oprogramowania
7. Geolokalizacja
8. Analiza danych



Python API, Web Scrapping

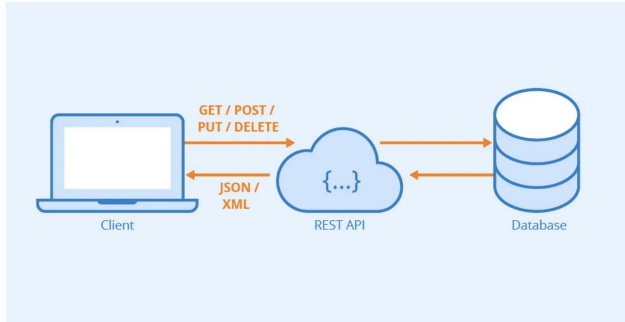
API – przykłady

1. RESTful API (Representational State Transfer)
2. SOAP API (Simple Object Access Protocol)
3. GraphQL API
4. JSON-RPC i XML-RPC
5. WebSocket API
6. Library-based API



Python API, Web Scrapping

REST API





Python API, Web Scrapping

REST API – działanie



Trzecia książka z półki

URL

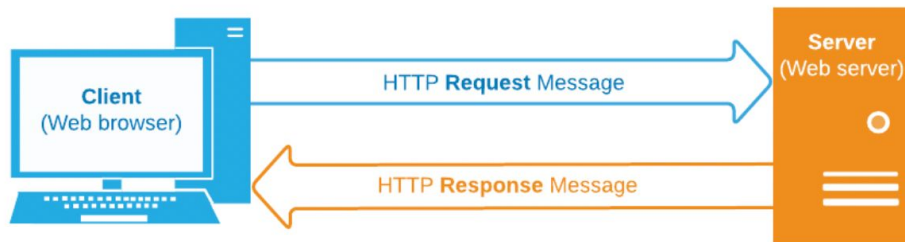
/book/3

/book/3/page/35



Python API, Web Scrapping

Protokół HTTP

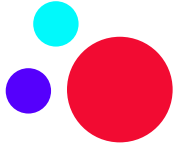




Python API, Web Scrapping

Typy zapytań HTTP

Metoda HTTP	Akcja API	Działanie
GET	READ	Odczytanie danych z zasobu
POST	CREATE	Utworzenie wskazanego zasobu
PUT/PATCH	UPDATE	Aktualizacja istniejącego zasobu
DELETE	DELETE	Usunięcie zasobu



Python API, Web Scrapping

Format JSON

POST /api/users

Content-Type: application/json

```
{  
  "username": "example_user",  
  "email": "user@example.com"  
}
```

← format JSON



Python API, Web Scraping

Zalety formatu JSON

```
{  
  "employeeId": 101,  
  "name": "John Doe",  
  "position": "Developer",  
  "department": {  
    "id": 1,  
    "name": "IT"  
  }  
}
```

← złożone struktury danych



Python API, Web Scraping requests

POST /api/users

Host: example.com

Content-Type: application/json


```
{  
  "username": "new_user",  
  "email": "new_user@example.com"  
}
```

} request



Python API, Web Scrapping endpoint

Posts

GET	/posts/	Get Posts	
POST	/posts/	Create Posts	
GET	/posts/{id}	Get Post	
PUT	/posts/{id}	Update Post	
DELETE	/posts/{id}	Delete Post	



Python API, Web Scraping

Biblioteka requests



info **Share**
ACADEMY



Python API, Web Scraping

Biblioteka requests – zastosowania

Przekazywanie parametrów:

```
import requests
```

```
response = requests.get("https://example.com/api/data",  
params={"page": 1, "limit": 10})
```



Python API, Web Scraping

Biblioteka requests – zastosowania

Obsługa nagłówków:

```
import requests
```

```
headers = {"Authorization": "Bearer my_token", "Accept": "application/json"}  
response = requests.get("https://example.com/api/data", headers=headers)
```



Python API, Web Scraping

Biblioteka requests – zastosowania

Obsługa Danych w Formie JSON:

```
import requests
```

```
import json
```

```
data = {"key": "value"}
```

```
response = requests.post("https://example.com/api/post",  
json=data)
```




Python API, Web Scrapping

Biblioteka requests – zastosowania

Obsługa błędów i wyjątków:

```
import requests
```

```
response = requests.get("https://example.com/api/data")
```

```
if response.status_code == 200:
```

```
    print("Request successful")
```

```
else:
```

```
    print(f"Request failed with status code {response.status_code}")
```



Python API, Web Scraping

requests – zapytanie i wczytywanie odpowiedzi

Wczytywanie odpowiedzi API do JSON :

```
import requests
```

```
r = requests.get('https://some.api.com/endpoint')
```

```
# dekodowanie odpowiedzi w JSON do Pythonowej struktury danych  
(słownik / lista)
```

```
response = r.json()
```

```
print(response)
```



Python API, Web Scraping

Parametry w zapytaniu

Parametry w zapytaniu:

```
import requests
```

```
payload = {'jakis_parametr': 'jakas_wartosc'}
```

```
r = requests.get('https://some.api.com/endpoint', params=payload)
```



URL

```
https://some.api.com/endpoint?jakis_parametr=jakas_wartosc
```



Zadanie 10.1 (instrukcja)

1. Korzystając z TheCatAPI
<https://developers.thecatapi.com/> pobierz z niego
zdjęcie losowego kota i wstaw je w notebooku.



Python API, Web Scrapping

Kod odpowiedzi

Atrybuty odpowiedzi – kod odpowiedzi:

```
import requests
```

```
r = requests.get('https://some.api.com/endpoint')
```

```
# kod odpowiedzi
```

```
r.status_code
```



Python API, Web Scrapping

Kod odpowiedzi

- **2xx** (Success)
- **3xx** (Redirection)
- **4xx** (Client Error)
- **5xx** (Server Error)



Python API, Web Scraping

request - reason

Atrybuty odpowiedzi - kod odpowiedzi:

```
import requests  
r = requests.get('https://some.api.com/endpoint') # reason  
r.reason
```




Python API, Web Scrapping

Kody odpowiedzi HTTP

Oficjalny rejestr:

<https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml>

Bardziej przyjazna lista:

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

<https://http.cat/>

<https://httpstatusdogs.com/>



Zadanie 10.2 (instrukcja)

1. Korzystając z API Hacker News
<https://github.com/HackerNews/API> podaj tytuł aktualnie
najbardziej popularnego linku (story) i przez kogo został
dodany.



Python API, Web Scraping

API – autoryzacja

Dostęp do API może być zabezpieczony przez np.:

API keys

OAuth <https://oauth.net/code/python/>



Zadanie 10.3 (instrukcja)

Korzystając z API The Metropolitan Museum of Art Collection

<https://metmuseum.github.io/> podaj:

1. Ile departamentów posiada muzeum.
2. Ile obiektów posiada departament sztuki średniowiecznej.
3. Jaki objectID posiada obraz "Śmierć Sokratesa"

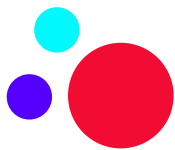
<https://joyofmuseums.com/museums/united-states-of-america/new-york-museums/metropolitan-museum-of-art/highlights-of-the-met/the-death-of-socrates-by-jacques-louis-david/> (tutaj trochę trzeba będzie też pogłównkować jakie parametry wyszukiwania podać, żeby dostać interesujące nas wyniki - da się zrobić to tak, żeby na pierwszym miejscu było ID interesującego nas obrazu).



Zadanie 10.4 (instrukcja)

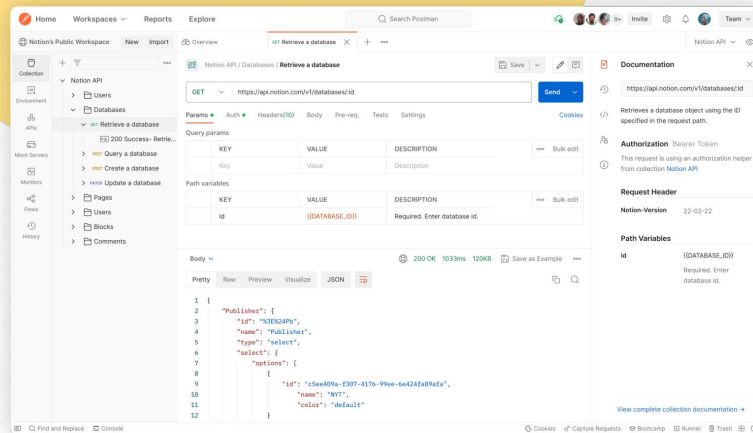
1. Użyj API Open Library <https://openlibrary.org/developers/api> aby:
 - znaleźć OLID (Open Library ID) wszystkich edycji książki "Automatic Information Organization and Retrieval"
<https://openlibrary.org/dev/docs/api/search> (podpowiedź: 'edition_key', sam OLID wygląda np. tak OL234546M),
 - użyć znalezionych OLID do znalezienia i wypisania URLi strony książki na stronie Open Library <https://openlibrary.org/dev/docs/api/books> (Podpowiedź: 'info_url'),
 - wyniki zapisz w słowniku w formacie z OLID jako kluczem i URLelem jako wartością użyj metody sub() z biblioteki re, aby przygotować tytuł książki do użycia w URLu <https://docs.python.org/3/library/re.html#re.sub> Słownik, który otrzymacie na końcu powinien wyglądać tak:

```
{ 'OL21159198M': 'http://openlibrary.org/books/OL21159198M/Automatic_information_organization_and_retrieval',  
  'OL5617209M': 'http://openlibrary.org/books/OL5617209M/Automatic_information_organization_and_retrieval.' }
```



Python API, Web Scrapping

Postman





Python API, Web Scrapping

API do nauki

<https://mixedanalytics.com/blog/list-actually-free-open-no-auth-needed-apis/>

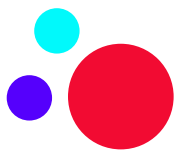
Big List of Free Open APIs

The APIs below can be accessed using any method:

- your web browser (just click on the sample URLs to load them)
- any modern coding language
- [cURL](#) for the command line
- no-code API clients like [Swagger](#), [Postman](#), or [Insomnia](#)
- Mixed Analytics' own [API Connector](#) for Google Sheets

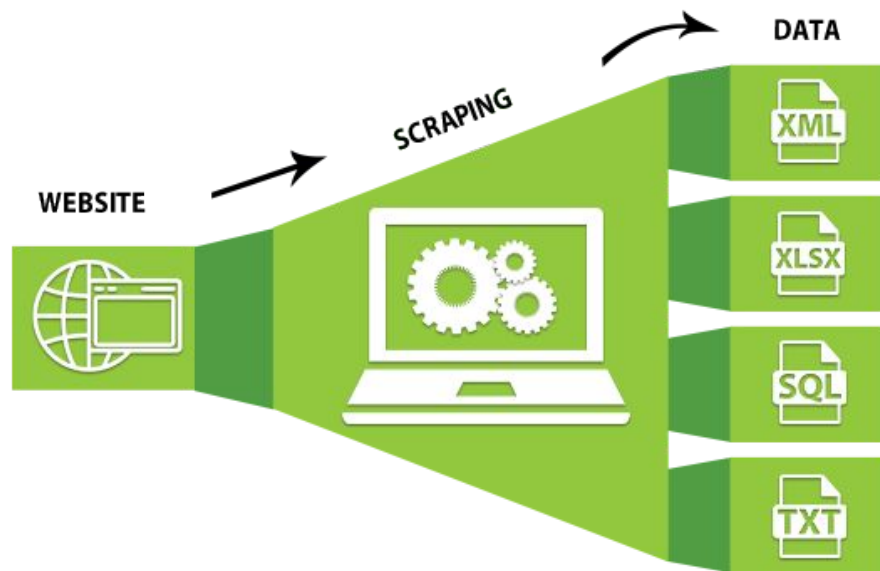
Filter list by category, name, or description

#	CATEGORY	API NAME	DESCRIPTION	SAMPLE URL
1	API Directories	APIs.guru	OpenAPI API directory	https://api.apis.guru/v2/list.json
2	API Directories	Public APIs	List of public APIs	https://api.publicapis.org/entries
3	Art & Images	Art Institute of Chicago	Artwork from the museum	https://api.artic.edu/api/v1/artworks/search?q=cats



Python API, Web Scraping

Web Scraping



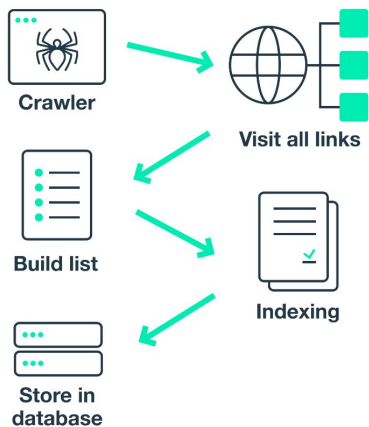


Python API, Web Scraping

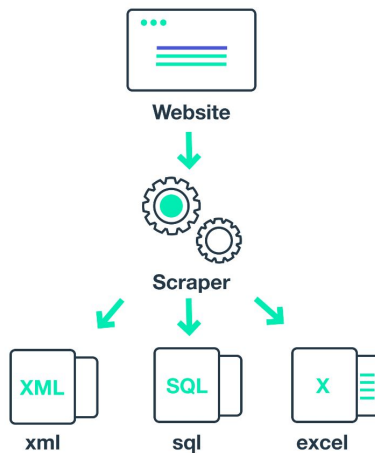
Web Scraping a Web Crawling?

infoShare
ACADEMY

Web Crawling



Web Scraping





Python API, Web Scraping

Web Scraping – korzyści

info **Share**
ACADEMY

WEB SCRAPING

Web Scraping Applications





Python API, Web Scraping

Kiedy nie scrapować?

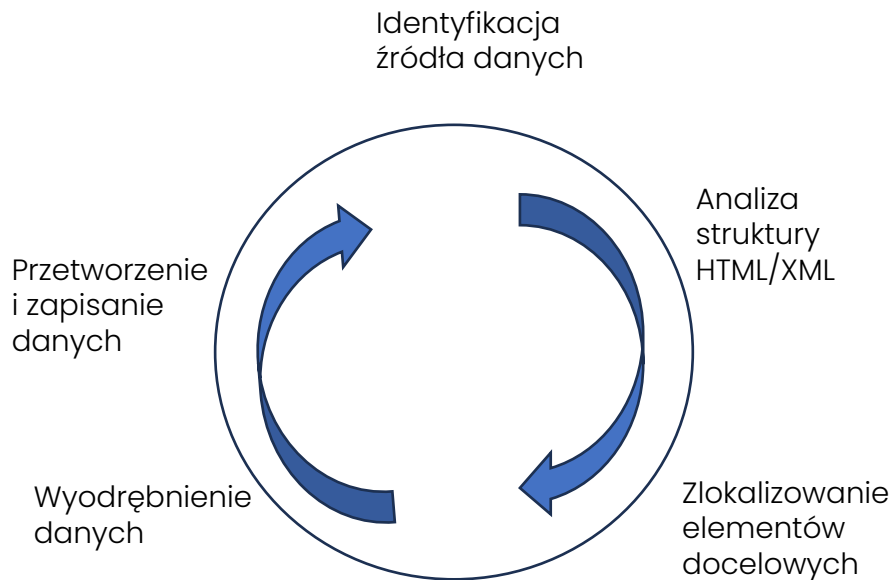


info **Share**
ACADEMY



Python API, Web Scraping

Jak działa Web Scraping?





Python API, Web Scraping

HTML, XML

HTML:

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Strona Przykładowa</title>  
  </head>  
  <body>  
  
    <h1>Witaj, Świecie!</h1>  
    <p>To jest przykładowa strona internetowa.</p>  
  
  </body>  
</html>
```

XML:

```
<person>  
  <name>John Doe</name>  
  <age>30</age>  
  <city>New York</city>  
</person>
```



Python API, Web Scraping

Techniki Web Scrapingu

1. Analiza Struktury HTML/XML
2. XPath i CSS Selectors
3. Użycie API
4. Automatyzacja przeglądarki
5. Tokenizacja HTML
6. Regular Expressions
7. Headless Browsing
8. RPA (Robotic Process Automation)
9. Web Scraping Frameworks



Python API, Web Scraping

Metody i narzędzia Web Scrapingu

BeautifulSoup

 Selenium

 Scrapy



Python API, Web Scraping

Dobre praktyki

1. Zgoda i warunki użytkowania
2. Częstotliwość zapytań
3. Szacunek dla zasobów serwera
4. Zgoda na scrapowanie
5. Respektowanie Robot.txt
6. Unikanie przetwarzania danych osobowych
7. Zgoda na powtarzalne scrapowanie
8. Dokładność i integrowalność danych



Python API, Web Scraping

Biblioteka BeautifulSoup

BeautifulSoup





Python API, Web Scraping

Ekstrakcja tytułów z witryny

```
from bs4 import BeautifulSoup
import requests

url = 'https://example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

titles = soup.find_all('h2')

for title in titles:
    print(title.text)
```



Zadanie 10.5 (instrukcja)

1. Twoim zadaniem jest dokonanie ekstrakcji tytułów artykułów z witryny internetowej. Użyj biblioteki BeautifulSoup, aby znaleźć i wydobyć tytuły artykułów z witryny. Witryna do analizy to `"https://en.wikipedia.org/wiki/Main_Page"`.



Python API, Web Scrapping

Ekstrakcja danych z HTML

Wczytywanie odpowiedzi API do JSON:

```
import requests
```

```
from bs4 import BeautifulSoup
```

```
r = requests.get('https://jakasstrona.com/jakis/url')
```

```
soup = BeautifulSoup(r.content, 'html.parser')
```



Python API, Web Scrapping

Ekstrakcja URL

Znajdź wszystkie linki na stronie:

```
from bs4 import BeautifulSoup
import requests

url = 'https://example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

links = soup.find_all('a')

for link in links:
    print(link.get('href'))
```



Python API, Web Scraping

Ekstrakcja URL

Znajdź linki w konkretnym bloku HTML (np. div z klasą 'content'):

```
from bs4 import BeautifulSoup
import requests

url = 'https://example.com'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

content_div = soup.find('div', class_='content')
links_in_content = content_div.find_all('a')

for link in links_in_content:
    print(link.get('href'))
```



Python API, Web Scraping

Ekstrakcja URL

Znajdź linki z określonym tekstem:

```
from bs4 import BeautifulSoup  
import requests
```

```
url = 'https://example.com'  
response = requests.get(url)  
soup = BeautifulSoup(response.text, 'html.parser')
```

```
links_with_specific_text = soup.find_all('a', text='Czytaj więcej')
```

```
for link in links_with_specific_text:  
    print(link.get('href'))
```




Zadanie 10.6 (instrukcja)

1. Używając BeautifulSoup podaj najbardziej popularny aktualny news (story) na Hacker News

<https://news.ycombinator.com/>:

- tytuł,
- URL.



Python API, Web Scraping

Web Scraping – podsumowanie



Website



Web Scraping



Structured Data



Python API, Web Scrapping

Obsługa plików

```
import os

os.chdir('C:\\Python\\Folder1')
plik = 'plik2.txt'
x = open(plik, 'r')
```



Python API, Web Scrapping

Zmienne plikowe

Otwarcie pliku do odczytu ('r')

file = open('example.txt', 'r') ← zmienna plikowa file

Wykonanie operacji na pliku

content = file.read()

Zamknięcie pliku

file.close()



Python API, Web Scrapping

Otwieranie pliku

`file = open('example.txt', 'r')` ← Otwarcie pliku do odczytu ('r')



Python API, Web Scraping

Tryby otwierania pliku

'r': Odczyt (domyślny tryb).

'w': Zapis (utworzy nowy plik lub nadpisze istniejący).

'a': Dodawanie (doda dane do końca istniejącego pliku).

'b': Tryb binarny (np. 'rb' lub 'wb').

'x': Utwórz plik, jeśli nie istnieje.



Python API, Web Scraping

Ścieżka do pliku

filepath = "example.txt" ← Ścieżka relatywna

filepath = "E:\\blog\\example.txt" ← Ścieżka absolutna



Python API, Web Scrapping

Kodowanie pliku

1. ASCII (American Standard Code for Information Interchange)
2. Unicode
3. UTF-8 (Unicode Transformation Format - 8-bit)



```
with open('example.txt', 'r', encoding='utf-8') as file:  
    content = file.read()
```




Python API, Web Scraping

Odczyt z pliku

Odczyt całej zawartości:

with open('example.txt', 'r') as file:

```
content = file.read()
```

```
print(content)
```



Zadanie 10.7 (instrukcja)

1. Stwórz plik tekstowy i umieść tam dowolny tekst.
2. Po zapisaniu tekstu, otwórz plik "moj_tekst.txt" i odczytaj jego zawartość.
3. Wyświetl odczytany tekst na ekranie.



Python API, Web Scrapping

Czytanie pliku liniami

with open('example.txt', 'r') as file:

```
    line = file.readline()
```

```
    print(line)
```



Zadanie 10.8 (instrukcja)

1. Stwórz plik tekstowy i umieść tam dowolny tekst (wielolinijkowy).
2. Po zapisaniu tekstu, otwórz plik "moj_tekst.txt" i odczytaj jego zawartość linijka po linijce.
3. Wyświetl odczytany tekst na ekranie.



Python API, Web Scrapping

Zapis do pliku

with open('example.txt', 'w') as file:

```
file.write("To jest przykładowy tekst do zapisu w pliku.")
```



```
lines = ["Linia 1", "Linia 2", "Linia 3"]
```

zapis tekstu do pliku

with open('example.txt', 'w') as file:

```
file.writelines(lines)
```



zapis listy linii do pliku

info **Share**
ACADEMY



Zadanie 10.9 (instrukcja)

1. Skorzystaj ze stworzonych w poprzednim zadaniu plików i dopisz liniijkę tekstu do nich.
2. Dodaj do pliku wiele linii tekstu jednocześnie (wykorzystaj do tego listę napisów). Upewnij się, że każdy element z listy jest w nowej linii.



Python API, Web Scraping

Zamykanie pliku

Użycie bloku with:

with open('example.txt', 'w') as file:

```
file.write("To jest przykładowy tekst do zapisu w pliku.")
```



Po opuszczeniu bloku with, plik zostanie automatycznie zamknięty.

Użycie metody close():

```
file = open('example.txt', 'w')
```

```
file.write("Inny przykładowy tekst.")
```

```
file.close()
```



Ręczne zamknięcie pliku.



Zadanie 10.10 (instrukcja)

1. Otwórz i zamknij pliki z poprzednich zadań. Zastosuj dwie metody do zamknięcia plików.



Python API, Web Scraping

Podsumowanie

