ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет Информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

Лабораторная работа № 6

Дисциплина: API-технологии

Тема: Введение в парсинг данных

Выполнил: студент группы 221-375

Яковлев Р. А.

(Фамилия И.О.)

Дата, подпись _____ _____

(Дата) (Подпись)

Проверил: _____ _____

(Фамилия И.О., степень, звание) **(Оценка)**

Дата, подпись _____ _____

(Дата) (Подпись)

Замечания:

_____

_____

Москва

2025

Импортируемые библиотеки

В начале кода используются следующие библиотеки:

- sys: предоставляют функции и переменные, используемые для работы со средствами управления Python.
- time: используется для создания задержек в процессе выполнения кода.
- requests: библиотека для отправки HTTP-запросов к веб-серверу.
- BeautifulSoup из bs4: инструмент для парсинга HTML и XML документов.
- pandas: библиотека для работы с данными в табличной форме (DataFrame).
- matplotlib.pyplot: библиотека для построения графиков и визуализации данных.

Основные функции программы

1. Функция fetch_financial_data

```
def fetch_financial_data(ticker, field):
```

Параметры:

- ticker (строка): уникальный идентификатор компании на фондовом рынке (например, "AAPL" для Apple).
- field (строка): конкретный финансовый показатель, который необходимо извлечь (например, "Выручка").

Алгоритм работы:

1. Формирование ссылки на страницу:
   - Код создает URL для страницы с финансовыми данными, используя переданный тикер:
   - base_url = f"https://finance.yahoo.com/quote/{ticker}/financials?p={ticker}"
2. Установка заголовков:

- Задаются заголовки HTTP-запроса, чтобы смоделировать запрос из браузера:

3. Отправка запроса и обработка ответа:

- Используя библиотеку requests, код отправляет GET-запрос на полученный URL. Если сервер возвращает код состояния, отличный от 200, код генерирует исключение:
- if response.status_code != 200:
- raise Exception(f"Ошибка: Не удалось получить данные для ticker {ticker}. HTTP Status: {response.status_code}")

4. Парсинг HTML-контента:

- Ответ сервера (HTML) передается в BeautifulSoup для дальнейшего анализа:
- soup = BeautifulSoup(response.text, 'html.parser')

5. Извлечение таблицы и данных:

- Код ищет таблицы с классом tableBody yf-9ft13 и извлекает строки, в которых содержится искомое поле:
- tables = soup.find_all('div', class_='tableBody yf-9ft13')

6. Сбор значений:

- Если поле не найдено, генерируется исключение. В противном случае значения добавляются в список результатов:
- values = [col.text.strip() for col in row.find_all('div', class_=['column yf-t22klz', 'column yf-t22klz alt'])]

7. Обработка ошибок:

- Ошибки запросов и парсинга обрабатываются с выводом соответствующих сообщений.

Примеры использования

Программа предназначена для запуска через командную строку. Пример использования:

python financial.py AAPL Выручка

Этот вызов осуществит извлечение данных о выручке компании Apple, сохранит их в CSV-файл и построит график.

lr6.py

```
import sys
import time
import requests
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt



def fetch_financial_data(ticker, field):
    base_url = f"https://finance.yahoo.com/quote/{ticker}/financials?p={ticker}"
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
    }
```

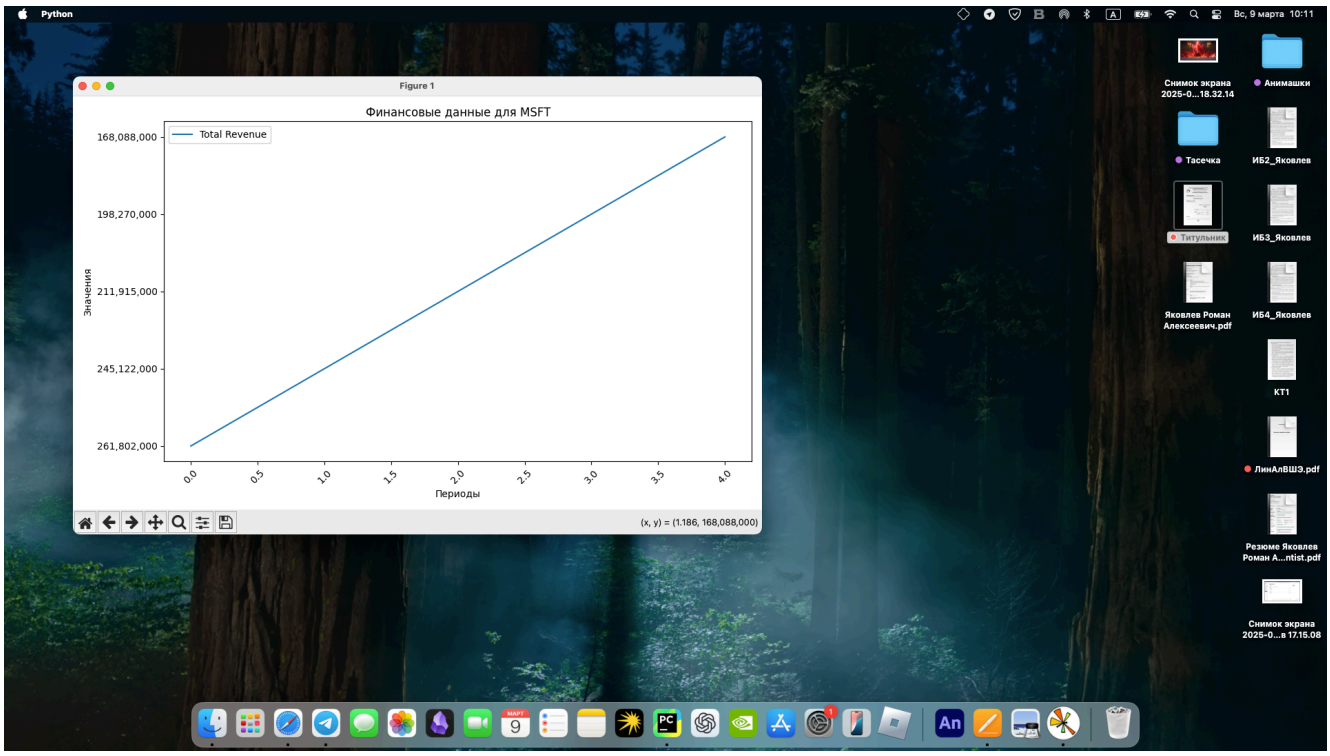
```
    try:
        response = requests.get(base_url, headers=headers, timeout=10)
        if response.status_code != 200:
            raise Exception(
                f"Ошибка: Не удалось получить данные для ticker {ticker}. HTTP Status:
{response.status_code}")

        soup = BeautifulSoup(response.text, 'html.parser')
        tables = soup.find_all('div', class_='tableBody yf-9ft13')

        if not tables:
                raise Exception(f"Ошибка: Таблица финансовых данных не найдена для
ticker {ticker}.")

        results = []
        for table in tables:
            rows = table.find_all('div', class_='row lv-0 yf-t22klz')
            for row in rows:
                row_title = row.find('div', class_='rowTitle yf-t22klz')
                if row_title and row_title.text.strip() == field:
                    values = [col.text.strip() for col in
                              row.find_all('div', class_=['column yf-t22klz', 'column yf-t22klz
alt'])]
```

```
                results.append({'Field': field, 'Values': values})
                return results

        raise Exception(f"Ошибка: Поле '{field}' не найдено для ticker {ticker}.")

    except requests.exceptions.RequestException as e:
        raise Exception(f"Сетевая ошибка при получении данных: {e}")
    except Exception as e:
        raise Exception(f"Ошибка при получении данных: {str(e)}")


def save_to_csv(data, ticker):
    df = pd.DataFrame(data)
    file_name = f"{ticker}_financial_data.csv"
    df.to_csv(file_name, index=False)
    print(f"Данные сохранены в файл {file_name}")


def plot_data(data, ticker):
    plt.figure(figsize=(10, 6))
    for entry in data:
        plt.plot(entry['Values'], label=entry['Field'])
    plt.xlabel('Периоды')
```

```
    plt.ylabel('Значения')
    plt.title(f'Финансовые данные для {ticker}')
    plt.legend()
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.savefig(f"{ticker}_financial_data.png")
    plt.show()


if __name__ == "__main__":
    if len(sys.argv) != 3:
        print("Использование: ./financial.py 'TICKER' 'FIELD'")
        sys.exit(1)

    ticker = sys.argv[1]
    field = sys.argv[2]

    time.sleep(5)
    try:
        result = fetch_financial_data(ticker, field)
        save_to_csv(result, ticker)
        plot_data(result, ticker)
```

```
except Exception as e:
    print(e)
    sys.exit(1)
```

Картиночки



Ссылка на репозиторий с кодом:

https://github.com/Ry0u14iY0Ru/APIshechki.git