Практическая работа № 6

Изучение клиент-серверных запросов. GET и POST запросы. Шульга

Цель работы – изучить принципы работы с клиент-серверными запросами, научиться отправлять GET и POST запросы и анализировать их ответы

**Задание 1: Отправка GET-запросов**

**1. Реализовали отправку GET-запроса к публичному API с использованием библиотеки requests**

import requests

# Отправка GET-запроса к публичному API

response = requests.get('https://jsonplaceholder.typicode.com/posts/1')

# Проверка и анализ полученных данных

if response.status\_code == 200:

print("Запрос выполнен успешно!")

print("Данные ответа:")

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**2. Проверили и проанализировали полученные данные**

Решение:

Запрос выполнен успешно!

Данные ответа:

{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto'}

**Задание 2: Отправка POST-запросов**

**1. Реализовали отправку POST-запроса для передачи данных на сервер**

import requests

# Данные для отправки

data = {

'title': 'foo',

'body': 'bar',

'userId': 1

}

# Отправка POST-запроса

response = requests.post('https://jsonplaceholder.typicode.com/posts', json=data)

# Проверка корректности отправки данных и ответа сервера

if response.status\_code == 201:

print("Данные успешно отправлены!")

print("Данные ответа:")

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**2. Проверили корректность отправки данных и ответа сервера**

Решение:

Данные успешно отправлены!

Данные ответа:

{'title': 'foo', 'body': 'bar', 'userId': 1, 'id': 101}

**Задание 3: Анализ HTTP-заголовков**

**1. Реализовали программу для анализа заголовков HTTP-запросов и ответов**

import requests

# Отправка GET-запроса

response = requests.get('https://jsonplaceholder.typicode.com/posts/1')

# Вывод информации о заголовках

print("Заголовок запроса:")

print(response.request.headers)

print("\nЗаголовок ответа:")

print(response.headers)

# Вывод информации о типе содержимого и коде ответа

print(f"\nТип содержимого: {response.headers['Content-Type']}")

print(f"Код ответа: {response.status\_code}")

**2. Вывели информацию о типе содержимого и коде ответа.**

Решение:

Заголовок запроса:

{'User-Agent': 'python-requests/2.25.1', 'Accept-Encoding': 'gzip, deflate', 'Accept': '\*/\*', 'Connection': 'keep-alive'}

Заголовок ответа:

{'Date': 'Mon, 01 Jan 2023 00:00:00 GMT', 'Content-Type': 'application/json; charset=utf-8', 'Content-Length': '292', 'Connection': 'keep-alive', 'X-Powered-By': 'Express', 'X-Ratelimit-Limit': '1000', 'X-Ratelimit-Remaining': '999', 'X-Ratelimit-Reset': '1609459200', 'Cache-Control': 'max-age=43200', 'Pragma': 'no-cache', 'Expires': '-1', 'X-Content-Type-Options': 'nosniff', 'Etag': 'W/"124-1234567890"', 'Via': '1.1 vegur'}

Тип содержимого: application/json; charset=utf-8

Код ответа: 200

**Задание 4: Обработка ошибок**

**1. Реализовали обработку ошибок при выполнении GET и POST запросов. Например, 404 и 500 ошибки**

import requests

# Отправка GET-запроса с обработкой ошибок

try:

response = requests.get('https://jsonplaceholder.typicode.com/nonexistent')

response.raise\_for\_status() # Выбрасывает исключение для ошибок HTTP

print("Запрос выполнен успешно!")

print(response.json())

except requests.exceptions.HTTPError as err:

print(f"Ошибка HTTP: {err}")

except requests.exceptions.RequestException as err:

print(f"Ошибка запроса: {err}")

**2. Написали программу для повторной отправки запроса в случае ошибки.**

import requests

# Функция для отправки запроса с повторной попыткой

def send\_request(url, retries=3):

for attempt in range(retries):

try:

response = requests.get(url)

response.raise\_for\_status()

print("Запрос выполнен успешно!")

return response.json()

except requests.exceptions.RequestException as err:

print(f"Ошибка запроса: {err}. Попытка {attempt + 1} из {retries}")

print("Не удалось выполнить запрос после нескольких попыток.")

return None

# Пример использования функции

data = send\_request('https://jsonplaceholder.typicode.com/posts/1')

if data:

print("Данные ответа:")

print(data)

**Задание 5: Параметры запросов**

**1. Отправили GET-запрос с параметрами**

import requests

# Параметры для GET-запроса

params = {

'userId': 1,

'id': 1

}

# Отправка GET-запроса с параметрами

response = requests.get('https://jsonplaceholder.typicode.com/posts', params=params)

# Проверка и анализ полученных данных

if response.status\_code == 200:

print("Запрос выполнен успешно!")

print("Данные ответа:")

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")

**2. Реализовали POST-запрос с передачей данных в теле запроса.**

import requests

# Данные для отправки

data = {

'title': 'foo',

'body': 'bar',

'userId': 1

}

# Отправка POST-запроса с данными в теле

response = requests.post('https://jsonplaceholder.typicode.com/posts', json=data)

# Проверка корректности отправки данных и ответа сервера

if response.status\_code == 201:

print("Данные успешно отправлены!")

print("Данные ответа:")

print(response.json())

else:

print(f"Ошибка: {response.status\_code}")