Цифриум

Технологическая карта

Умный код на Python. Базовый уровень

Модуль 3 | Тема 1 | Урок 1

Основы HTTP-запросов (библиотека requests)

Уважаемый коллега, обратите внимание!

Учёт присутствия ребёнка на уроке ведётся только по цифровому следу. Каждый ученик из группы на уроке должен передать цифровой след на нашей платформе.

Вы отвечаете за передачу следа учеником на нашей платформе во время урока.

Передавать цифровой след нужно в личном кабинете ученика.

Цифровой след засчитывается по итогу выполнения учеником трёх задач на платформе в ходе урока.

Обращаем внимание — в задачах с отправкой ответа в виде файла наша платформа поддерживает следующие форматы: .pdf, .doc, .png, .jpeg, .xlsx, .mp3, .py, .ipynb, .txt, .csv, .json, .xml, .sb3, .ino, .hex.

Контролируйте время урока, чтобы не пропустить момент, когда ученики должны залогиниться в личном кабинете и перейти к решению задач на платформе.

Проследите, чтобы каждый ученик залогинился в личном кабинете и выполнил 3 задачи на платформе. В противном случае, Вам не будет засчитано, что вы провели урок, занятие не будет оплачено, а ученику не будет выставлена отметка о присутствии на уроке.

Просим вас в ходе 45 минут урока проконтролировать, чтобы:

1. Каждый из учеников залогинился в личном кабинете.
2. Каждый ученик в разделе **«Решить задания»** отправил решения по 3 задачам — «Практическое задание 1», «Практическое задание 2» и «Практическое задание 3».

О занятии

Краткое содержание занятия:

Вы узнаете, как работает протокол HTTP, какие существуют методы HTTP-запросов, что означают коды ответа сервера. Также научитесь с помощью библиотеки requests в Python отправлять запросы и получать ответы.

Цель занятия:

Познакомиться с принципами работы протокола HTTP и научиться использовать библиотеку Python — requests для отправки простых HTTP-запросов и получения ответов веб-серверов

Задачи занятия:

1. Понять, что такое протокол HTTP и его принцип работы.
2. Познакомиться с библиотекой requests.
3. Рассмотреть основные методы HTTP-запросов.
4. Узнать про коды состояния ответов и способы их интерпретации.
5. Научиться использовать атрибуты в HTTP-запросах.

Необходимые материалы:

Для проведения занятия понадобятся: рабочая тетрадь, ваш документ/файл для записи терминов и выводов, ваш файл с расширением .py для написания скриптов, доступ к платформе Цифриум для самопроверки и ДЗ.

Термины:

HTTP (HyperText Transfer Protocol) — протокол обмена данными между веб-сервером и клиентскими приложениями. Он работает на прикладном уровне модели OSI, который используется для передачи гипертекста**.**

OSI (Open Systems Interconnection) — это модель взаимодействия открытых систем, разделяющая процесс передачи данных на семь уровней, каждый из которых выполняет определённые функции.

Запрос (Request) — пакет данных, который клиент отправляет серверу**.**

О занятии

Ответ (Response) — пакет данных, который сервер возвращает клиенту.

Requests — модуль в Python, который позволяет удобно и легко отправлять запросы по протоколу HTTP**.**

Коды состояния HTTP — специальные числовые коды, которые сервер отправляет в ответ на HTTP-запрос.

Темы и время

ЭТАП	ВРЕМЯ
Приветствие	5 мин.
Теория	25 мин.
Итоги занятия	5 мин.
Получение фидбека. Рефлексия. Проверка знаний. Домашнее задание	5 мин.
Вопрос-ответ	5 мин.
Итого	45 мин.

Ход занятия

Номер слайда	Пояснение к слайду
0	**Титульный слайд занятия**
1	**Повторение прошлого занятия**
2	**Анонс текущего занятия**
3	**Постановка цели и задач**
4	Прямо сейчас поговорим о фундаментальной вещи — протоколе HTTP. Это буквально язык общения вашего браузера с любым сайтом. Вот вы вводите адрес в строку — и в этот момент браузер на HTTP говорит серверу: «Дай мне эту страницу». А сервер ему в ответ на том же языке отправляет HTML, картинки и всё остальное. Название раскрывает суть: это протокол для передачи гипертекста, то есть связанных ссылками документов. Чтобы понять, где он работает,поговорим о модели OSI — это теоретическая схема из семи уровней, которая описывает, как данные путешествуют по сети. HTTP находится на самом верхнем, прикладном уровне. Это уровень наших с вами приложений — браузеров, мессенджеров. Ему неважно, как данные идут по проводу или по Wi-Fi, его задача — договориться о том, что передавать. А на практике мы будем общаться с серверами не через браузер, а из Python-кода. Для этого есть отличная библиотека — requests. С её помощью мы сможем программно отправлять запросы и смотреть, что нам серверы отвечают, пощупав HTTP своими руками.
5	Работа HTTP строится по схеме «запрос-ответ». Клиент, например браузер, формирует и отправляет HTTP-запрос на сервер. В нём указывается, какой ресурс нужен и какое действие выполнить.

Ход занятия

	Сервер обрабатывает этот запрос и возвращает HTTP-ответ. В ответе содержится статус (успех или код ошибки) и, если запрос выполнен, запрашиваемые данные: HTML-страница, изображение или другие файлы. Давайте детально разберём структуру обеих частей этого обмена: из чего состоит запрос и из чего состоит ответ. На практике мы будем работать с ними с помощью библиотеки requests в Python.
6	Теперь разберём структуру HTTP-запроса и ответа. Начнём с запроса. HTTP-запрос — это строго форматированный пакет данных. Его первая и обязательная часть — стартовая строка. В ней указывается метод запроса, например GET или POST, URL ресурса и версия протокола. После стартовой строки идут заголовки. Это служебная информация в формате «ключ: значение», которая сообщает серверу детали: тип клиента, допустимые форматы данных, куки и так далее. Заголовки отделяются от тела запроса одной обязательной пустой строкой. Тело сообщения присутствует не всегда, а только в запросах, которые отправляют данные на сервер, например при отправке формы методом POST.
7	Теперь рассмотрим структуру HTTP-ответа от сервера. Ответ начинается со строки статуса. В ней указывается версия протокола, трёхзначный числовой код состояния и его текстовое пояснение. Например, «200 OK» для успешного запроса или «404 Not Found». После строки статуса следуют HTTP-заголовки ответа. Они содержат метаинформацию: тип и размер возвращаемых данных, параметры кэширования, дату, данные о сервере. Как и в запросе, заголовки отделяются от тела ответа одной

Ход занятия

	пустой строкой. Тело ответа содержит непосредственно запрошенные данные — HTML-код страницы, содержимое файла или JSON-объект. Тело может быть пустым, например, при некоторых кодах перенаправления.
8	Для работы с HTTP в Python используется библиотека requests. Она предоставляет высокоуровневый интерфейс для отправки запросов, скрывая низкоуровневые детали работы с сокетами и ручное формирование пакетов. Библиотека позволяет выполнять все основные операции: отправлять запросы различными методами, управлять их заголовками, работать с параметрами URL и данными форм. Она также поддерживает работу с cookies, сессиями, позволяет выполнять базовую и сложную аутентификацию, загружать файлы и задавать параметры производительности, такие как таймауты.
9	Перед началом работы необходимо установить библиотеку. Это делается через пакетный менеджер pip. Откройте терминал или командную строку и выполните соответствующую команду. Если у вас Windows, используйте команду pip install requests. Для пользователей Linux или macOS команда обычно выглядит как pip3 install requests, чтобы явно указать на менеджер для Python 3. После успешной установки библиотеку нужно подключить в ваш Python-скрипт. Для этого в начале файла с расширением .py пропишите строку import requests. Теперь все функции библиотеки доступны для использования.
10	HTTP-методы определяют тип операции, которую клиент хочет выполнить с ресурсом на сервере. Они указываются в стартовой строке запроса и являются ключевой частью его семантики. Основные методы, с которыми вы будете работать: GET для получения данных, POST для отправки и создания новых данных, PUT для полного обновления ресурса, PATCH для

Ход занятия

	частичного обновления и DELETE для удаления. Каждый метод имеет строго определённое назначение, и сервер обрабатывает запросы в зависимости от использованного метода.
11	Метод GET — самый распространённый. Он предназначен для получения данных от сервера. При выполнении GET-запроса параметры передаются в самой строке URL, а у запроса, как правило, нет тела. В библиотеке requests для отправки GET-запроса используется функция get(). В неё передаётся URL целевого ресурса. Функция возвращает объект Response, который содержит весь ответ сервера. На слайде приведён базовый пример: создание запроса к тестовому сервису httpbin.org и вывод полученного объекта ответа.
12	Метод POST используется для отправки данных на сервер, например, при отправке формы. В отличие от GET, данные в POST-запросе обычно передаются в теле запроса, а не в URL. В библиотеке requests для этого применяется функция post(). Через параметр data или json передаются отправляемые данные в виде словаря. На примере видно: мы отправляем POST-запрос на https://httpbin.org/post с данными {'name': 'Ivan'}. Сервер httpbin в ответ возвращает JSON-объект, содержащий наши отправленные данные, а также информацию о запросе, например, заголовки.
13	Метод PUT предназначен для полного обновления существующего ресурса или его создания, если он отсутствует по указанному URL. Его ключевое отличие от POST в идемпотентности: повторный одинаковый PUT-запрос не должен изменять состояние системы. В библиотеке requests метод вызывается функцией put(). Данные для обновления ресурса передаются через параметр data или json. В примере мы отправляем запрос на тестовый сервер с данными {'status': 'updated'}, что имитирует обновление состояния ресурса. Сервер возвращает ответ,

Ход занятия

	подтверждающий операцию.
14	Метод DELETE предназначен для удаления указанного ресурса на сервере. Как и GET, он обычно не содержит тела запроса — целевой ресурс определяется URL. Для отправки DELETE-запроса в библиотеке requests используется функция delete(). В примере мы отправляем такой запрос на тестовый эндпоинт. Сервер httpbin.org обрабатывает его и возвращает стандартный ответ, подтверждающий, что запрос на удаление был получен и обработан.
15	Метод PATCH используется для частичного обновления ресурса. В отличие от PUT, который заменяет ресурс целиком, PATCH применяется, когда нужно изменить только некоторые поля, например, обновить пароль пользователя, не затрагивая его имя или email. В библиотеке requests для этого применяется функция patch(). Данные для частичного обновления передаются через параметр data или json. В примере мы отправляем новый пароль на сервер, который должен обновить только это поле в ресурсе. Сервер возвращает ответ с подтверждением и отправленными данными.
16	Коды состояния HTTP — это стандартные трёхзначные числа в строке статуса ответа сервера. Они мгновенно сообщают клиенту о результате обработки его запроса. Все коды разделены на пять классов. Первые цифры 1xx — информационные, указывают, что запрос принят к обработке. Коды 2xx означают успех, самый известный — 200 OK. Коды 3xx — перенаправления, требующие от клиента дополнительного действия. Коды 4xx — ошибки на стороне клиента, например, 404 Not Found или 403 Forbidden. Коды 5xx — ошибки на стороне сервера, такие как 500 Internal Server Error.

Ход занятия

17	Приведём конкретные примеры кодов. 100 Continue — информационный код, сервер готов принять тело запроса. 200 OK — стандартный ответ при успешном выполнении GET или POST. 301 Moved Permanently — код перенаправления, указывающий на новый постоянный URL ресурса. 403 Forbidden — ошибка клиента, означающая, что сервер понял запрос, но отказывается его авторизовать. 503 Service Unavailable — ошибка сервера, сигнализирующая о временной невозможности обработать запрос из-за перегрузки или технических работ.
18	После отправки запроса библиотека requests возвращает объект Response. Его атрибуты содержат всю информацию об ответе сервера. status_code возвращает числовой код состояния HTTP, например 200 или 404. url показывает итоговый URL, по которому был выполнен запрос, с учётом возможных перенаправлений. text содержит тело ответа в виде строки, подходящее для HTML или текста. content возвращает тело в виде байтов, что необходимо для работы с бинарными данными, например, изображениями. Метод .json() парсит тело ответа, если оно в формате JSON, и возвращает словарь Python.
19	Рассмотрим примеры использования атрибутов ответа. В первом примере после GET-запроса мы выводим код состояния через r.status_code и итоговый URL через r.url. Это базовая диагностика запроса. Во втором примере запрашивается HTML-страница. Мы получаем её содержимое как строку через r.text и выводим первые 200 символов для предварительного просмотра. В третьем примере запрашиваются данные в формате JSON. Метод r.json() автоматически преобразует ответ в словарь Python, что позволяет обращаться к данным по ключам.

Ход занятия

	Здесь мы получаем значение title из структуры ответа.

Практика

Вопросы для самопроверки

1. Что такое протокол HTTP и как он работает?
2. Чем отличается метод PUT от POST?
3. Что вернет r.status_code при успешном запросе?

Практическое задание

Первичная настройка

Установим библиотеку через консоль командой pip:

pip install requests # для тех, кто использует Windows

pip3 install requests # для тех, кто использует Linux/MacOS

Для дальнейшей работы необходимо в рабочем файле импортировать библиотеку

import requests

Задача 1: Напишите GET-запрос к сайту https://httpbin.org и выведите статус-код ответа и содержимое ответа в виде текста

Задача 2: Напишите POST-запрос на сайт https://httpbin.org, передав имя и возраст, а также выведите полученный словарь с помощью json()

Задача 3: Напишите PUT-запрос на сайт https://httpbin.org с телом {'status': 'updated'}, а также выведите статус-код и тело ответа в текстовом формате

Задача 4: Напишите GET-запрос к сайту https://цифриум.рф/ и выведите первые 300 символов HTML-кода страницы

Задача 5: Напишите GET-запрос к сайту https://httpbin.org, проверьте, что ответ содержит JSON, затем выведите:

- статус-код;
- URL запроса;
- ключи в JSON-ответе.

Если ответ не в формате JSON, выведите сообщение об ошибке.

Практика

Проверь себя по следующим критериям:

- Код выполняется без ошибок.
- Использованы нужные методы HTTP-запроса.
- Запросы возвращают ожидаемые данные.
- Ответы выведены в консоль.

Примеры выполнения заданий

Ответы на вопросы самопроверки

1. Необходимы: интерпретатор Python и текстовый редактор/IDE.
2. Командой python --version в терминале.
3. Интерпретатор выполняет код, IDE предоставляет среду для разработки.

Задача 1:

```
import requests

r = requests.get('https://httpbin.org/get')

print(r.status_code)

print(r.text)
```

вывод:

```
200

{

 "args": {},

 "headers": {

  "Accept": "*/*",

  "Accept-Encoding": "gzip, deflate",

  "Host": "httpbin.org",

  "User-Agent": "python-requests/2.31.0",

  "X-Amzn-Trace-Id": "Root=1-686545c4-729d4f195373eb8d5036f5e8"

 },

 "origin": "57.129.38.230",

 "url": "https://httpbin.org/get"
```

Примеры выполнения заданий

}

Задача 2:

```
import requests

r = requests.post('https://httpbin.org/post', data={'name': 'Ivan', 'age': 18})

print(r.json())
```

вывод:

{'args': {}, 'data': '', 'files': {}, 'form': {'age': '18', 'name': 'Ivan'}, 'headers': {'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Content-Length': '16', 'Content-Type': 'application/x-www-form-urlencoded', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.31.0', 'X-Amzn-Trace-Id': 'Root=1-6865479d-5ef685773eb219ae4a17879a'}, 'json': None, 'origin': '57.129.38.230', 'url': 'https://httpbin.org/post'}

Задача 3:

```
import requests

r = requests.put('https://httpbin.org/put', data={'status': 'updated'})

print(r.status_code)

print(r.text)
```

вывод:

```
200

{

  "args": {},

  "data": "",

  "files": {},

  "form": {

    "status": "updated"
```

Примеры выполнения заданий

},

"headers": {

"Accept": "*/*",

"Accept-Encoding": "gzip, deflate",

"Content-Length": "14",

"Content-Type": "application/x-www-form-urlencoded",

"Host": "httpbin.org",

"User-Agent": "python-requests/2.31.0",

"X-Amzn-Trace-Id": "Root=1-686547ee-462e0b2a6854abd61af8259a"

},

"json": null,

"origin": "57.129.38.230",

"url": "https://httpbin.org/put"

}

Задача 4:

```
import requests

r = requests.get('https://цифриум.рф/')

print(r.text[:300])
```

вывод:

<!DOCTYPE html> <html lang="ru"> <head> <meta charset="utf-8" /> <meta http-equiv="Content-Type" content="text/html; charset=utf-8" /> <meta name="viewport" content="width=device-width, initial-scale=1.0" /> <!--metatextblock--> <title>Цифриум</title> <meta property="og:url" content="http://xn--h1aa

Примеры выполнения заданий

Задача 5:

```
import requests

r = requests.get('https://httpbin.org/json')

print('Статус-код:', r.status_code)

print('URL запроса:', r.url)

try:

  data = r.json()

  print('Ключи JSON-ответа:', list(data.keys()))

except ValueError:

  print('Ответ не в формате JSON')
```

вывод:

```
Статус-код: 200

URL запроса: https://httpbin.org/json

Ключи JSON-ответа: ['slideshow']
```

Сегодня на занятии вы:

- Узнали, как работает протокол HTTP.
- Изучили основные методы HTTP-запросов: GET, POST и др.
- Разобрали, что означают коды ответа сервера.
- Научились использовать библиотеку requests для отправки запросов.
- Попрактиковались в получении и обработке ответов от веб-серверов.

На следующем занятии вы:

- Разберёте, что такое API и чем публичные API отличаются от приватных.
- Поймете принцип работы API и как приложения обмениваются данными.
- Изучите структуру URL и построение запросов к публичным API.
- Научитесь отправлять простые запросы к API с помощью Python.
- Познакомитесь с популярными публичными API.

Получение фидбека. Рефлексия. Проверка знаний

Попросите ребят решить интерактивные задачи по теме урока.

Ответы к интерактивным задачам

Практическое задание 1

Какой код состояния HTTP означает, что запрос успешно обработан?

- 100
- 301
- 200
- 403

Ответ: 200

Практическое задание 2

Какой атрибут объекта response в библиотеке requests возвращает тело ответа в виде байтов?

- text
- content
- json()
- url

Ответ: content

Практическое задание 3

Ниже предложен фрагмент кода. Разработчику необходимо изменить только пароль пользователя,
не трогая остальные данные. Какое слово пропущено?

```
import requests

r = requests._____('https://httpbin.org/patch', data={'password':
'new_password'})

print(r)
```

Ответ: patch

Ответы к интерактивным задачам

Вопрос-ответ

Если во время подготовки или проведения урока возникают вопросы, их необходимо адресовать в методический чат.