Цифриум

Технологическая карта

Умный код на Python. Базовый уровень

Модуль 3 | Тема 1 | Урок 2

Простые публичные API (часть 1)

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, что такое API и как с его помощью приложения обмениваются данными. Разберёте, какие виды API существуют, а также научитесь отправлять простые GET-запросы к публичным API с помощью библиотеки requests.

Цель занятия:

Изучить, что такое API, научиться работать с простыми публичными API с использованием библиотеки requests и понять, как происходит обмен данными между приложениями и серверами.

Задачи занятия:

1. Разобраться, что такое API и как оно работает.
2. Понять отличие публичных API от приватных.
3. Изучить структуру URL-адреса при обращении к API.
4. Научиться отправлять простые GET-запросы к публичным API с помощью библиотеки requests.
5. Рассмотреть примеры популярных открытых API.

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:

API (Application Programming Interface) — набор правил и инструкций, с помощью которых различные программы могут обращаться к функциям и данным других программ или сервисов.

URI (Uniform Resource Identifier) — набор символов, который служит для обозначения какого-то ресурса, который может быть реальным (веб-страница, файл) или абстрактным (имя, идея); соответственно, такие ресурсы не обязательно должны быть доступны через сеть Интернет.

URL (Uniform Resource Locator) — это особый вид URI, который говорит, где именно находится ресурс в сети Интернет или в другой системе. Он

О занятии

показывает путь к ресурсу, чтобы компьютер мог его найти и получить к нему доступ: открыть страницу, скачать файл.

Параметры запросов к API — это пары «ключ-значение», которые добавляются в URL после знака вопроса «?». Параметры разделяются между собой символом амперсанда «&».

Партнерские API — это API, работающие как публичные, но доступные только авторизованным пользователям.

Приватные API (они же и внутренние) — это API, которые используются внутри компаний или для ограниченного круга пользователей. Доступ к ним часто защищён, и они представляют более чувствительные, специфичные данные.

Публичные API — это API, которые открыты для использования всеми желающими. Обычно они предоставляют доступ к общедоступным данным или сервисам.

Составные API — это API, в состав которых входит два или больше API. Такое решение позволяет решать сложные задачи и объединять возможности несколько серверов.

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	API, или программный интерфейс приложения, — это, по сути, контракт между сервисами. Он определяет строгий набор правил: как одна программа может запросить данные или действие у другой, какой формат запроса использовать и в каком виде будет получен ответ. В веб-контексте это чаще всего означает отправку HTTP-запросов на специальные адреса (эндпоинты) сервиса и получение данных в структурированном виде, обычно JSON. Это позволяет вашему приложению использовать готовую функциональность другого сервиса, например, получать погоду, курсы валют или публикации из соцсетей.
5	Принцип работы API основан на модели «клиент-сервер». Ваша программа выступает в роли клиента, который инициирует взаимодействие. Процесс состоит из четырёх последовательных этапов. Сначала клиент формирует и отправляет HTTP-запрос на определённый адрес (URI или URL) API. Этот запрос содержит инструкцию, например «получить список пользователей». Далее API, выступая как посредник, принимает этот запрос, проверяет его корректность и перенаправляет на соответствующий внутренний сервер или сервис для обработки.

Ход занятия

	Сервер выполняет необходимую логику: обращается к базе данных, проводит вычисления. После этого он формирует результат и передаёт его обратно API. Наконец, API получает результат от сервера, упаковывает его в стандартный формат, чаще всего JSON, и отправляет этот структурированный ответ обратно клиенту.
6	API классифицируются по уровню доступа и назначению. Основных вида четыре. Публичные (или открытые) API доступны для любого разработчика. Они обычно предоставляют бесплатный или условно-бесплатный доступ к общественным данным, например, погоде, биржевым котировкам, новостным лентам. Приватные API предназначены для внутреннего использования внутри компании или организации. Они не публикуются и защищают доступ к бизнес-логике и чувствительным данным. Партнёрские API занимают промежуточное положение: они открыты, но только для авторизованных сторонних разработчиков или бизнес-партнёров, часто после заключения соглашения. Составные API — это архитектурный подход, когда несколько отдельных API объединяются в один интерфейс для решения сложной задачи, что упрощает клиенту работу с несколькими сервисами одновременно.
7	Приведём конкретные примеры для каждого типа API. Публичный API — это, например, VK API. Любой разработчик может зарегистрировать приложение и использовать его для получения общедоступных данных из соцсети: информации о пользователях, постах, сообществах. Пример приватного API — внутренняя система компании, например, CRM. Её API используется только сотрудниками и

Ход занятия

	внутренними сервисами для работы с клиентской базой, он недоступен извне. Партнёрский API можно увидеть в интеграции с банком. Интернет-магазин подключает API платёжного шлюза банка для приёма оплаты. Доступ к API есть только у авторизованных партнёров — магазинов, заключивших договор. Составной API используется в сложных приложениях. Например, в мобильном банке один запрос к API может сразу вернуть пользователю сводную информацию: баланс, историю операций и статус бонусного счёта, объединив данные из нескольких внутренних систем.
8	За что отвечает библиотека requests? Она предоставляет высокоуровневый интерфейс для отправки HTTP-запросов из Python-кода, избавляя от необходимости ручного формирования сетевых пакетов. Её функции, такие как get() или post(), позволяют легко взаимодействовать с веб-серверами. Какие существуют методы HTTP-запросов? Приведите примеры. Основные методы определяют тип операции над ресурсом. GET для получения данных (например, requests.get('https://api.example.com/data')). POST для отправки данных (отправка формы). PUT для полного обновления ресурса. PATCH для частичного обновления. DELETE для удаления ресурса.
9	Библиотека requests — это основной инструмент в Python для взаимодействия с веб-API. Она абстрагирует низкоуровневые детали сетевого обмена, позволяя сосредоточиться на логике работы с данными. С её помощью вы можете формировать любые типы запросов, гибко настраивая их через параметры: передавать данные в теле запроса, устанавливать необходимые заголовки, включая заголовки для авторизации.

Ход занятия

	Библиотека удобно работает с основными форматами обмена. Вы можете отправлять данные в формате JSON и так же легко получать и автоматически парсить JSON-ответы от сервера. Она также предоставляет механизмы для обработки ошибок, таких как таймауты или коды состояния HTTP 4xx/5xx, и поддерживает различные методы аутентификации для доступа к защищённым API.
10	Для корректной работы с API важно понимать два ключевых понятия: URI и URL. URI, или универсальный идентификатор ресурса, — это общий термин для любой строки, которая однозначно идентифицирует какой-либо ресурс. Ресурсом может быть что угодно: документ, изображение, сервис. URI — это, по сути, «имя» ресурса. URL, или универсальный указатель ресурса, — это частный, но самый распространённый случай URI. URL не просто называет ресурс, а указывает его местоположение в сети и способ доступа к нему. Он содержит протокол (например, http://), доменное имя и путь. Если проводить аналогию, URI — это имя человека, а URL — его полный почтовый адрес с индексом, по которому можно отправить письмо. В контексте API мы почти всегда работаем с URL как с конкретными адресами эндпоинтов.
11	Структура URL, с которым вы работаете в API, состоит из нескольких обязательных и опциональных компонентов. Первым идёт схема, чаще всего http или https. Она определяет протокол доступа. Далее следует доменное имя (или IP-адрес) сервера, к которому мы обращаемся. За ним идёт путь — это конкретный адрес ресурса на сервере, например /api/v1/users. Именно путь определяет эндпоинт API. После пути могут следовать параметры запроса. Они начинаются с символа ? и представляют собой пары ключ=значение, разделённые амперсандом &. Например,

Ход занятия

	?page=2&limit=10. Это самый распространённый способ передать серверу дополнительные данные для фильтрации, сортировки или пагинации прямо в адресной строке GET-запроса. Последний необязательный компонент — фрагмент после #, который обычно используется в браузерах для навигации внутри страницы и в контексте API применяется редко.
12	Рассмотрим конкретные примеры URL с параметрами для API. Первый пример: https://api.example.com/users/?id=123&sort=desc. Здесь эндпоинт — /users/. Параметры запроса начинаются после ?. Параметр id=123 указывает серверу, что нам нужны данные конкретного пользователя с идентификатором 123. Параметр sort=desc задаёт порядок сортировки результатов — по убыванию. Второй пример имитирует запрос к интернет-магазину: https://shop.com/products?category=books&page=2. Эндпоинт /products возвращает список товаров. Параметр category=books фильтрует этот список, оставляя только книги. Параметр page=2 реализует пагинацию, запрашивая вторую страницу результатов, так как товаров может быть много. Таким образом, параметры в строке запроса — это удобный способ уточнить, какие именно данные и в каком виде мы хотим получить от API.
13	Теперь перейдём к практике. Мы будем отправлять GET-запросы к публичным API. В качестве примера возьмём сервис randomuser.me, который генерирует случайные данные профилей пользователей. Наш первый шаг — импортировать библиотеку requests. Затем мы отправляем GET-запрос по указанному URL, используя функцию requests.get(). Этот запрос вернёт объект ответа.

Ход занятия

	Чтобы увидеть сырые данные, которые прислал сервер, мы обратимся к атрибуту .text этого объекта и выведем его. В результате вы увидите структурированный ответ, скорее всего в формате JSON, содержащий сгенерированный профиль: имя, email, фотографию и другую информацию. Это базовый пример обращения к эндпоинту API без параметров.
14	Здесь используется другой публичный API — GitHub. Мы отправляем GET-запрос к эндпоинту https://api.github.com/users/octocat. Этот URL следует определённой структуре: база api.github.com, путь /users/ и параметр пути octocat, который указывает на конкретного пользователя. Сервер GitHub вернёт JSON-объект с публичной информацией об этом пользователе: его имя, количество репозиториев, дату создания аккаунта и так далее. Чтобы автоматически преобразовать ответ из формата JSON в словарь Python, мы используем не .text, а метод .json() объекта ответа. Это позволит нам легко обращаться к данным по ключам, например, response.json()['public_repos'].
15	Теперь усложним запрос. Мы обращаемся к другому эндпоинту GitHub API: https://api.github.com/users/octocat/repos. Структура пути здесь логична: /users/ указывает на работу с пользователями, octocat — на конкретного пользователя, а /repos — на ресурс, представляющий список его репозиториев. Этот GET-запрос вернёт не один объект, а JSON-массив. Каждый элемент этого массива будет словарём с подробной информацией об одном репозитории: его названием, описанием, языком программирования, количеством звёзд. Используя метод .json(), мы получаем этот массив в виде списка Python, с которым далее можно работать: перебирать в цикле, извлекать конкретные данные или фильтровать

Ход занятия

	результаты.
16	Теперь рассмотрим примеры популярных публичных API, которые можно использовать в реальных проектах. Первый пример — GisMeteo API. Это специализированный сервис для получения метеоданных. Он предоставляет как текущую погоду, так и прогнозы. API позволяет запрашивать информацию по географическим координатам или ID населённого пункта. Его основные возможности включают детальный прогноз по часам и дням, получение текущих условий, а также поддержку ответов на разных языках. Это делает API удобным для интеграции в сайты, мобильные приложения или сервисы, связанные с логистикой и путешествиями. Важный момент: в отличие от предыдущих примеров, для доступа к GisMeteo API требуется предварительная регистрация разработчика, создание приложения в личном кабинете и получение уникального API-ключа, который необходимо передавать в каждом запросе для авторизации.
17	Второй пример — API Яндекс Карт. Это мощный набор инструментов для работы с геоданными. Он предоставляет функциональность для встраивания интерактивных карт на сайт, поиска объектов и организаций, построения пешеходных и автомобильных маршрутов, а также для геокодирования — преобразования адреса в географические координаты и наоборот. Такое API используется в самых разных сервисах: от доставки и такси до туристических порталов и сайтов недвижимости, где важно показать локацию. Как и в случае с GisMeteo, для использования API Яндекс Карт необходима авторизация. Нужно зарегистрироваться в сервисе для разработчиков, создать проект и получить уникальный API-ключ, который будет идентифицировать

Ход занятия

	ваше приложение в каждом запросе к сервису.
18	Третий пример — API HeadHunter (hh.ru). Это профессиональный инструмент для работы с данными рынка труда. Его возможности широки: можно получать и анализировать вакансии, искать резюме, получать информацию о компаниях-работодателях. Компании могут использовать API для размещения актуальных вакансий на своём корпоративном сайте. Особенность HH API в том, что он позволяет строить сложные запросы с фильтрацией по множеству параметров: профессии, региону, опыту работы, зарплате. В примере на слайде показан простой запрос https://api.hh.ru/vacancies?text=python, который возвращает список вакансий, связанных с Python. Параметр text в строке запроса задаёт поисковый запрос. Для части эндпоинтов HH API является публичным и не требует ключа для доступа к открытым данным, таким как вакансии. Для работы с персональными данными или от лица компании требуется OAuth-авторизация.

Практика

Вопросы для самопроверки

1. Что такое API и какую роль играет в обмене данными между программами?
2. В чем отличие между публичными и приватными API?
3. Что такое URL и из каких частей он состоит при обращении к API?
4. Чем библиотека requests полезна при работе с API в Python?

Практическое задание

Задача 1: Разберите URL: https://api.example.com/users?age=25&active=true и объясните:

- схему;
- домен;
- путь;
- параметры запроса.

Задача 2: Составьте URL для запроса вакансий с ключевым словом «developer» и городом «Москва».

Подсказка:

Список параметров для API HeadHunter доступен в официальной документации на GitHub: https://github.com/hhru/api

В разделе «Вакансии» в пункте «Поиск по вакансиям», где описаны параметры поиска.

Задача 3: Отправьте GET-запрос к randomuser.me и выведите JSON.

Задача 4: Получите данные пользователя GitHub - octocat, а также выведите его логин и количество репозиториев.

Задача 5: API Agify - сервис, который предсказывает возраст по имени, а именно анализирует реальные данные и делает предположение, сколько лет человеку с указанным именем.

Формулировка задания:

Отправьте GET-запрос на сайт Agify для своего имени и выведите предсказанный возраст.

Практика

Проверь себя по следующим критериям:

- Корректно разобран URL на части и правильно сформирован URL с параметрами.
- Код выполняется без ошибок.
- Запросы отправлены на корректные и существующие URL-адреса API.
- В случае работы с параметрами URL, они корректно сформированы и учитываются в запросах.
- Ответы выведены в консоль.
- Для задач с разбором JSON правильно извлечены и отображены нужные поля.

Примеры выполнения заданий

Ответы на вопросы самопроверки

1. Что такое API и какую роль играет в обмене данными между программами?

API — это набор правил и интерфейсов, позволяющий одной программе запрашивать данные или функциональность у другой. Он выступает в роли посредника, стандартизируя обмен данными (чаще через HTTP/JSON), что позволяет разным приложениям взаимодействовать без знания внутренней реализации друг друга.

2. В чем отличие между публичными и приватными API?

Публичные API открыты для всех разработчиков, часто бесплатны и предоставляют доступ к общественным данным (погода, курсы валют). Приватные API используются внутри организаций для внутренних систем, защищены и не доступны извне.

3. Что такое URL и из каких частей он состоит при обращении к API?

URL — это адрес ресурса в сети. При обращении к API он включает:

Схему (https://)

Домен сервера (api.example.com)

Путь к эндпоинту (/users/)

Параметры запроса (после ?, например ?id=123)

4. Чем библиотека requests полезна при работе с API в Python?

Requests упрощает отправку HTTP-запросов к API, автоматизируя формирование запросов, обработку заголовков, авторизацию и парсинг ответов (JSON/XML). Она избавляет от низкоуровневой работы с сокетами и позволяет сосредоточиться на логике взаимодействия с API.

Задача 1:

Схема: https — протокол передачи данных

Домен: api.example.com — адрес сервера

Путь: /users — ресурс API

Примеры выполнения заданий

Параметры запроса:

age=25 — фильтрация по возрасту

active=true — только активные пользователи

Задача 2:

URL: https://api.hh.ru/vacancies?text=developer&area=1

Параметры:

text=developer — ключевые слова,

area=1 — код Москвы в API HeadHunter

Задача 3:

import requests

response = requests.get("https://randomuser.me/api/")

print(response.json())

вывод:

{'results': [{'gender': 'female', 'name': {'title': 'Mrs', 'first': 'Roline', 'last': 'De Putter'}, 'location': {'street': {'number': 523, 'name': 'Buitenwater'}, 'city': 'Langweer', 'state': 'Noord-Holland', 'country': 'Netherlands', 'postcode': '0081 SI', 'coordinates': {'latitude': '36.8754', 'longitude': '73.4398'}, 'timezone': {'offset': '+3:00', 'description': 'Baghdad, Riyadh, Moscow, St. Petersburg'}}, 'email': 'roline.deputter@example.com', 'login': {'uuid': 'c288037a-bc24-47f7-829c-d77f76b1dfb9', 'username': 'silversnake518', 'password': 'mooney', 'salt': 'VksB0Esh', 'md5': '35b7e8d9d66f4a6428e22dd5e8402f97', 'sha1': '50a9923ff669bd31d74760eedda1825e5ba42778', 'sha256': '97d88e0f990dc1df80f7f312c4577fe760d9b61728fab34e3b899750ac6d6b74'}, 'dob': {'date': '1970-09-05T21:37:31.041Z', 'age': 54}, 'registered': {'date': '2021-10-16T22:59:38.290Z', 'age': 3}, 'phone': '(0579) 469263', 'cell': '(06) 48832383', 'id': {'name': 'BSN', 'value': '39233797'}, 'picture': {'large': 'https://randomuser.me/api/portraits/women/45.jpg', 'medium': 'https://randomuser.me/api/portraits/med/women/45.jpg', 'thumbnail': 'https://randomuser.me/api/portraits/thumb/women/45.jpg'}, 'nat': 'NL'}], 'info': {'seed': '89a28e3b818ec666', 'results': 1, 'page': 1, 'version': '1.4'}}

Примеры выполнения заданий

Задача 4:

```
import requests

response = requests.get("https://api.github.com/users/octocat")

data = response.json()

print(f"Логин: {data['login']}")

print(f"Публичных репозиториев: {data['public_repos']}")
```

вывод:

Логин: octocat

Публичных репозиториев: 8

Задача 5:

```
import requests

response = requests.get("https://api.agify.io/?name=Darya")

data = response.json()

print(f"Имя: {data['name']}, Предполагаемый возраст: {data['age']}")
```

вывод:

Имя: Darya, Предполагаемый возраст: 37

Сегодня на занятии вы:

- Разобрали, что такое API и чем публичные API отличаются от приватных.
- Поняли принцип работы API и как приложения обмениваются данными.
- Изучили структуру URI, URL и построение запросов к публичным API.
- Научились отправлять простые запросы к API с помощью Python.
- Познакомились с популярными публичными API.

На следующем занятии вы:

- Познакомитесь с понятием пагинации.
- Научитесь получать данные частями из API.
- Освоите методы разбора и извлечения информации из сложных JSON-ответов.
- Узнаете, как обрабатывать ошибки при работе с API и писать более устойчивый код на Python.
- Рассмотрите ограничения публичных API.

Получение фидбека. Рефлексия. Проверка знаний

Попросите ребят решить интерактивные задачи по теме урока

Ответы к интерактивным задачам

Практическое задание 1

Что такое API?

- Язык программирования для создания веб-приложений
- **Набор правил и инструкций для взаимодействия программ**
- База данных для хранения информации
- Протокол передачи файлов по сети

Практическое задание 2

Какой компонент URL указывает на местоположение ресурса на сервере?

- Схема
- Домен
- **Путь**
- Запрос

Практическое задание 3

Нужно отправить HTTP GET-запрос на сайт randomuser.me (который генерирует случайные данные пользователей) и вывести полученный ответ в формате JSON.Ниже представлен фрагмент программы. введите слово, которое пропущено

requests

Ответы к интерактивным задачам

Вопрос-ответ

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