ЧОУ ДПО «Центр инновационного развития
человеческого потенциала и управления знаниями»

КВАНТОРИУМ
Фотоника

г. Пермь, ул. 25 Октября, д.64
http://kvantorium-perm.ru
+7(342)214-42-69

Принято:
На заседании методического совета
Протокол № 11 от 11.08.2026 г.

Утверждаю:
Директор ЧОУ ДПО «ЦИРЧПиУЗ»
_____ П.И.Крендель
Приказ № 52 от 12.08.2026 г.

Дополнительная общеобразовательная общеразвивающая программа «IT-квантум. Продвинутый курс.»

Направленность: техническая
Возраст обучающихся: 12-17 лет
Срок реализации: 64 часа
Продвинутый уровень

Разработчик:
Субботин Евгений Сергеевич
Педагог дополнительного образования

Пермь, 2026 г.

ПРОГРАММА КУРСА «IT-квантум. Продвинутый курс»

Пояснительная записка

Курс рассчитан на учеников с базовыми знаниями в программировании, которые хотят расширить свои навыки и освоить современные технологии для создания востребованных программных решений. Курс позволяет углубиться в веб-разработку и повысить квалификацию как во frontend, так и в backend направлениях. Также курс будет полезен тем, кто хочет научиться работать с современными инструментами и писать более эффективный, масштабируемый код.

Программа для учащихся, стремящихся перейти на следующий уровень профессионального развития. Благодаря комплексному подходу, включающему как теорию, так и практические проекты, учащиеся смогут не только освоить новые инструменты, но и приобрести навык их применения в реальных задачах.

Этот курс будет особенно полезен тем, кто планирует работать в больших компаниях или командах, где востребованы знания современных библиотек, фреймворков, умение создавать и поддерживать сложные приложения, а также навыки работы с базами данных и API. Обучение также рассматривает важные аспекты DevOps и контейнеризации, что расширяет возможности специалиста по части автоматизации и деплоя приложений.

Кроме того, курс ориентирован на тех, кто хочет оставаться конкурентоспособным на рынке труда, постоянно обновляя и совершенствуя свои компетенции в быстро меняющейся сфере IT. Полученные знания открывают дорогу к новым карьерным возможностям и позволяют эффективно участвовать в разработке современных программных продуктов и систем.

Актуальность программы:

Программа актуальна в условиях стремительного развития информационных технологий и постоянного обновления инструментов разработки. Современный рынок IT требует от специалистов глубокого понимания не только базовых принципов программирования, но и владения передовыми технологиями для создания высококачественных, производительных и масштабируемых приложений. Освоение современных языков, фреймворков и методик работы с данными становится необходимостью для успешной карьеры и конкурентоспособности на рынке труда.

Кроме того, интеграция frontend и backend разработки, а также навыки взаимодействия с базами данных, API и DevOps-практиками позволяют создавать полнофункциональные продукты и обеспечивать их эффективное сопровождение. В свете роста востребованности комплексных специалистов, способных работать с современными веб-технологиями и средствами автоматизации, данный курс обеспечивает качественную подготовку, соответствующую актуальным требованиям IT-индустрии и перспективам цифровой трансформации.

Новизна программы:

Новизна программы заключается в комплексном подходе к обучению, объединяющем две ключевые и актуальные сферы современной разработки — современный

веб-фронтенд и бэкенд на JavaScript/TypeScript, а также Python с уклоном в Data Engineering и DevOps практики. Такой интегрированный формат позволяет слушателям получить навыки создания полноценных приложений от пользовательского интерфейса до серверной логики и обработки данных, что отражает современные требования рынка и практики разработки.

Кроме того, программа обновлена с учётом самых последних тенденций и инструментов, включая использование актуальных фреймворков (React или Vue), асинхронного программирования, контейнеризации с Docker, и современных подходов к тестированию и автоматизации. Такой подход помогает не только освоить теорию, но и получить практический опыт работы с технологиями, которые активно применяются в ведущих компаниях и стартапах, обеспечивая быстрое и качественное развитие профессиональных компетенций.

Педагогическая целесообразность:

Педагогическая целесообразность программы обусловлена её структурированным и поэтапным подходом к освоению сложных современных технологий, что обеспечивает максимально эффективное усвоение материала. Курс сочетает теоретические объяснения с практическими заданиями и проектной деятельностью, что способствует формированию прочных навыков и умению применять знания в реальных условиях. Такой баланс теории и практики способствует развитию критического мышления, самостоятельности и творческого подхода у обучающихся.

Кроме того, программа учитывает различные стили восприятия информации и уровень подготовки слушателей, предлагая адаптивные методы обучения, включая интерактивные занятия, групповые и индивидуальные проекты, а также регулярное тестирование и обратную связь. Это создаёт благоприятную образовательную среду, мотивирующую участников к постоянному развитию и приобретению компетенций, необходимых для успешной карьеры разработчика в условиях быстро меняющихся технологических реалий.

Цель программы:

Подготовить специалистов, способных разрабатывать современные полнофункциональные программные продукты с применением актуальных технологий фронтенд и бэкенд разработки, а также инструментов для обработки данных и автоматизации, обеспечивая высокий уровень профессиональной компетентности и готовность к работе в современных IT-командах.

Образовательные задачи:

- Обучить современным языкам программирования и технологиям (JavaScript/TypeScript, Python).
- Освоить основы и продвинутые приёмы разработки frontend и backend приложений.
- Познакомить с методами работы с базами данных, API и средствами аутентификации.
- Обучить технологиям тестирования, отладки и обеспечению качества кода.
- Ознакомить с инструментами контейнеризации и автоматизации разработки (Docker, CI/CD).

Развивающие задачи:

- Развивать аналитическое и алгоритмическое мышление.
- Формировать навыки самостоятельного решения сложных технических задач.
- Способствовать развитию критического мышления в оценке и выборе технических решений.
- Развивать умение работать в команде и использовать современные средства коммуникации.
- Формировать навыки планирования и реализации комплексных проектов.

Воспитательные задачи:

- Воспитывать профессиональную ответственность и внимательность к качеству работы.
- Формировать этические принципы и уважение к интеллектуальной собственности.
- Развивать настойчивость и терпение при преодолении сложных технических проблем.
- Воспитывать уважение к командной работе и коллективному успеху.
- Формировать готовность к постоянному обучению и профессиональному росту.

Категория слушателей:

- Возраст 12-17 лет.
  Виды программы: Дополнительная общеобразовательная общеразвивающая
  Уровень: продвинутый

Формат и структура занятий.

Организационные формы работы:

- Лекции и теоретические занятия.
- Практические лабораторные работы и кодинг-сессии.
- Групповые проекты и командная работа.
- Онлайн-вебинары и интерактивные мастер-классы.
- Самостоятельная работа и подготовка домашних заданий.

Структура занятий:

- Введение и постановка целей урока.
- Теоретический блок с объяснением ключевых концепций.
- Демонстрация практических примеров и приёмов.
- Выполнение практических заданий и работа над проектами.
- Обсуждение результатов, обратная связь и закрепление материала.

Особенности реализации программы:

- Интеграция теории и практики для формирования комплексных навыков.
- Адаптивность к уровню подготовки участников и постепенное усложнение материала.
- Использование современных инструментов и актуальных технологий.
- Акцент на проектную деятельность и решение реальных задач.
- Регулярное тестирование и получение обратной связи для контроля прогресса.

Методы реализации программы:

- Активное обучение через практические упражнения и проекты.
- Кейс-стади и анализ реальных ситуаций из IT-процессов.
- Демонстрация и разбор кода с подробными комментариями.
- Групповая дискуссия и коллективное решение проблем.
- Использование онлайн-платформ для самостоятельного обучения и контроля.

Ключевые аспекты программы:

- Освоение современных языков и технологий веб-разработки.
- Формирование навыков создания полноценных frontend и backend приложений.
- Изучение принципов DevOps, контейнеризации и автоматизации.
- Развитие проектного мышления и работы в команде.
- Подготовка к требованиям современного IT-рынка и карьерному росту.

Предметные результаты обучения:

- Владение современными языками программирования (JavaScript/TypeScript, Python).
- Умение создавать и поддерживать frontend и backend приложения с использованием актуальных фреймворков и библиотек.
- Навыки работы с базами данных, API и средствами аутентификации.
- Освоение технологий тестирования, отладки и обеспечения качества кода.
- Знание принципов контейнеризации и автоматизации с использованием Docker и CI/CD.

Метапредметные результаты обучения:

- Способность анализировать и решать комплексные технические задачи.
- Навыки самостоятельного обучения и использования современных информационных ресурсов.
- Умение работать в команде, эффективно коммуницировать и координировать совместную деятельность.
- Применение критического мышления при выборе и обосновании технических решений.
- Формирование проектного мышления и навыков планирования работы над программными продуктами.

Личностные результаты обучения:

- Развитие ответственности и дисциплины в выполнении профессиональных задач.
- Формирование интереса к постоянному профессиональному развитию и самообучению.
- Воспитание этических норм и уважительного отношения к коллегам и интеллектуальной собственности.
- Развитие настойчивости и терпения при решении сложных и нестандартных задач.
- Формирование мотивации к работе в коллективе и стремления к достижению общих целей.

В результате реализации программы учащиеся будут:

Знать:

- Современные языки программирования JavaScript/TypeScript и Python, их особенности и области применения.
- Основные технологии и инструменты для разработки фронтенд и бэкенд приложений.
- Методы работы с базами данных, API и системами аутентификации.
- Основные принципы автоматизации процессов разработки с помощью Docker и CI/CD.
- Подходы к тестированию, отладке и обеспечению качества программного обеспечения.

Уметь:

- Разрабатывать полноценные веб-приложения и сервисы с использованием современных технологий.
- Создавать и интегрировать API, реализовывать работу с базами данных и системами хранения данных.
- Настраивать и использовать инструменты автоматизации для ускорения разработки и деплоя.
- Проводить тестирование, отладку и оптимизацию кода для повышения его надежности и эффективности.
- Реализовывать командную работу над проектами с учетом современных методов организацией разработки.

Владеть:

- Практическими навыками создания и поддержки современных веб-приложений.
- Инструментами и методами автоматизации, контейнеризации и деплоя.
- Техниками поиска и исправления ошибок в программном коде.
- Способами проведения интеграционного и юнит-тестирования программных решений.
- Навыками организации работы в команде и эффективного взаимодействия в процессе разработки.

Практические умения:

- Разработка и развертывание полноценных frontend-приложений с использованием современных фреймворков.
- Создание и интеграция backend-сервисов и RESTful API.
- Работа с реляционными и нереляционными базами данных, выполнение запросов и управление данными.
- Настройка и использование контейнеризации с Docker для разработки и деплоя приложений.
- Проведение тестирования (юнит-тесты, интеграционные тесты) и отладки кода.
- Организация командной разработки с использованием систем контроля версий (Git) и инструментов CI/CD.

Критерии и способы определения результативности:

Предметные:

- Умение создавать веб-приложения с использованием современных технологий и инструментов.
- Владение навыками работы с базами данных, API и системами аутентификации.
- Способность писать структурированный, читаемый и тестируемый код.

- Реализация автоматизированных процессов разработки и деплоя с помощью Docker и CI/CD.
- Умение проводить тестирование и устранять ошибки в программных решениях.

Метапредметные:
- Способность анализировать технические задачи и самостоятельно находить решения.
- Навыки эффективной работы в команде и организации совместных проектов.
- Владение методами планирования и управления временем при реализации проектов.
- Умение применять информационные ресурсы для поиска решений и самообучения.
- Развитие критического мышления при оценке и выборе технологий и подходов.

Личностные:
- Формирование ответственности за качество выполняемых задач.
- Проявление инициативы и мотивации к постоянному профессиональному росту.
- Воспитание уважения к коллегам и этики в профессиональной деятельности.
- Развитие настойчивости и терпения при преодолении сложностей.
- Готовность к обучению и адаптации к новым условиям работы и технологиям.

Способы определения результативности

Формирующие методы оценки результативности:
- Непрерывный контроль выполнения практических заданий и проектов.
- Проведение регулярных устных и письменных опросов для проверки усвоения материала.
- Анализ качества выполнения домашних заданий и промежуточных проектов.
- Фидбек и консультации во время занятий для своевременного выявления трудностей.
- Использование тестов и квизов по ключевым темам в процессе обучения.

Итоговые методы оценки результативности:
- Итоговый экзамен или тестирование по теоретической части курса.
- Представление и защита практических проектов или работ.
- Анализ выполнения итогового контрольного задания, связанного с реальной задачей.
- Оценка портфолио выполненных проектов и их соответствие поставленным целям.
- Самооценка и рефлексия учащихся о достигнутых результатах и приобретённых навыках.

Итог:

Комплексный анализ результатов с учетом данных методов и критериев позволяет определить эффективность программы, выявить сильные стороны и определить направления для дальнейшего развития учебного процесса. Регулярное формирующее и итоговое оценивание способствует не только контрольной функции, но и развитию самостоятельности, мотивации и интереса у обучающихся.

Учебный план

1 год обучения		
№ модуля	Название модуля	Количество часов

		Всего	Теория	Практика
1.	Современная веб-разработка на JavaScript (Frontend + Backend)	32	10	22
2.	Python для разработки и Data Engineering	32	9	23
	ИТОГО	64	19	45

Учебно-тематический план модуля «Современная веб-разработка на JavaScript (Frontend + Backend)»

№	Тема занятия	Количество часов			Форма контроля
		Теория	Практика	Всего	
1	Введение в JavaScript и современную веб-разработку	0,5	1,5	2	Устный опрос, тест на базовое понимание
2	Основы синтаксиса JavaScript	0,5	1,5	2	Практическое задание: базовые операции
3	Управляющие конструкции и работа с массивами	0,5	1,5	2	Выполнение задачи с использованием циклов
4	Функции и область видимости	0,5	1,5	2	Короткий тест + практика: написание функций
5	Объекты в JavaScript	0,5	1,5	2	Практическое задание на работу с объектами
6	Работа с DOM: базовая манипуляция	0,5	1,5	2	Домашнее задание с динамическим элементом
7	События в браузере и обработчики событий	0,5	1,5	2	Практическое упражнение с обработчиками
8	Асинхронность в JS: коллбеки и промисы	0,5	1,5	2	Тест + написание асинхронной функции
9	Введение в Node.js и установка окружения	0,5	1,5	2	Мини-проект: запуск hello world сервера
10	Работа с файлами и модулями в Node.js	0,5	1,5	2	Практическая работа: чтение и запись в файл
11	Создание простого веб-сервера на Node.js	0,5	1,5	2	Практическое задание: сервер с одним маршрутом
12	Введение в Express.js	0,5	1,5	2	Практическое задание: создание маршрутов Express
13	Работа с REST API	0,5	1,5	2	Тест + создание простого CRUD API

14	Взаимодействие Frontend с Backend	0,5	1,5	2	Практическое задание: интеграция frontend с API
15	Основы React.js	0,5	1,5	2	Практическое задание: написать простой компонент React
16	Итоговый проект и тестирование	0	2	2	Объединение frontend и backend, итоговая проверка знаний
	ИТОГО:	7,5	24,5	32	

Учебно-тематический план модуля «Python для разработки и Data Engineering»

№	Тема занятия	Количество часов			Форма контроля
		Теория	Практика	Всего	
1	Введение в Python: установка, запуск, синтаксис	0,5	1,5	2	Краткий тест (онлайн-викторина)
2	Управляющие конструкции: условные операторы и циклы	0,5	1,5	2	Практическое задание (написать скрипт)
3	Функции и модули	0,5	1,5	2	Проверка кода (Code review)
4	Работа со списками и словарями	0,5	1,5	2	Практическое задание
5	Обработка исключений и отладка	0,5	1,5	2	Краткий тест
6	Введение в библиотеки для работы с данными (Pandas)	0,5	1,5	2	Практическое задание (работа с таблицей)
7	Работа с файлами: чтение и запись	0,5	1,5	2	Практическое задание
8	Регулярные выражения и обработка текстовых данных	0,5	1,5	2	Краткий тест + практика
9	Работа с базами данных и SQL	0,5	1,5	2	Практическое задание (запросы)
10	Введение в ETL-процессы	0,5	1,5	2	Аналитическое задание
11	Многопоточность и асинхронность в Python	0,5	1,5	2	Практическое задание
12	Веб-запросы и API интеграция	0,5	1,5	2	Практическое задание
13	Обработка больших данных: основы	0,5	1,5	2	Краткий тест
14	Контейнеризация приложений (Docker)	0,5	1,5	2	Практическое задание (создать Dockerfile)
15	Итоговый проект: сборка и обработка данных	0,5	1,5	2	Промежуточная защита проекта
16	Итоговое занятие и защита проектов	0	2	2	Итоговая защита + тест

	ИТОГО:	7,5	24,5	32	

Общая характеристика содержания программы

Модуль 1: «Современная веб-разработка на JavaScript (Frontend + Backend» (16 занятий / 32 часа)

1. Введение в JavaScript и современную веб-разработку

Теория:
- Что такое JavaScript
- История и роль JS в веб-разработке
- Обзор технологий: Frontend и Backend
- Инструменты разработки: редакторы, браузеры, Node.js

Практика:
- Установка Node.js и редактора кода
- Создание первого скрипта "Hello, World!"

2. Основы синтаксиса JavaScript

Теория:
- Типы данных: number, string, boolean, null, undefined, symbol, object
- Переменные: var, let, const
- Операторы: арифметические, сравнения, логические

Практика:
- Написание скрипта с объявлениями переменных и базовыми операциями

3. Управляющие конструкции и работа с массивами

Теория:
- Условия: if, else, switch
- Циклы: for, while, do-while
- Методы массивов: push, pop, map, filter, reduce

Практика:
- Решение задач с использованием условий и циклов
- Обработка массивов с помощью методов

4. Функции и область видимости

Теория:
- Объявление функций: function, стрелочные функции
- Параметры и возвращаемое значение
- Замыкания и область видимости переменных

Практика:
- Создание функций разных видов
- Написание функций с использованием замыканий

5. Объекты в JavaScript

Теория:

- - Создание и свойства объектов
- - Методы объектов
- - Контекст: this

Практика:

- - Создание объектов и вызов методов
- - Манипуляция свойствами объектов

6. Работа с DOM: базовые операции

Теория:

- Структура DOM
- Навигация по DOM: getElementById, querySelector
- Создание и удаление элементов

Практика:

- Динамическое добавление и изменение элементов на странице

7. События в браузере и обработчики

Теория:

- Виды событий (click, input, submit и др.)
- Назначение обработчиков событий
- Делегирование событий

Практика:

- Реализация интерактивных элементов с помощью событий

8. Асинхронность в JS: коллбеки и промисы

Теория:

- Модель асинхронности
- Коллбеки
- Объекты Promise
- async/await синтаксис

Практика:

- Создание асинхронных функций
- Обработка данных с помощью промисов и async/await

9. Введение в Node.js и настройка окружения

Теория:

- Что такое Node.js
- Установка Node.js и npm
- Работа с командной строкой

Практика:

- Запуск простого скрипта на Node.js ("Hello, World!")

10. Работа с файлами и модулями в Node.js

Теория:

- Работа с файловой системой (fs)
- Модули CommonJS

Практика:

- Чтение и запись в файлы
- Создание своих модулей и подключение их

11. Создание простого веб-сервера на Node.js

Теория:
- HTTP-модуль
- Обработка запросов и ответов

Практика:
- Создание базового сервера, отдающего простую страницу или сообщение

12. Введение в Express.js

Теория:
- Что такое Express.js
- Установка и настройка
- Маршрутизация и middleware

Практика:
- Создание маршрутов, обработка запросов через Express

13. Основы REST API

Теория:
- Что такое REST
- Методы HTTP: GET, POST, PUT, DELETE
- Организация API

Практика:
- Создание простого CRUD API на Express

14. Взаимодействие Frontend с Backend

Теория:
- AJAX, Fetch API
- Отправка запросов и обработка ответов
- Работа с JSON

Практика:
- Создать страницу, которая взаимодействует с API через Fetch

15. Основы React.js

Теория:
- Что такое компоненты
- Props и state
- Работа cJSX

Практика:
- Создать простой React-компонент и отобразить его

16. Итоговый проект и тестирование

Теория:
- Объединение frontend и backend
- Общие принципы разработки проекта

Практика:

- Разработка мини-проекта, интеграция API + интерфейс
- Защита проекта, проход по чек-листам

Модуль 2: «Python для разработки и Data Engineering» (16 занятий / 32 часа)

1. Введение в Python и основы синтаксиса

Теория:

- Установка Python, структура программы
- Типы данных: числа, строки, булевы значения
- Переменные и их создание
- Базовые операции: вывод, арифметика, комментарии

Практика:

- Написать программу, которая выводит приветственное сообщение
- Создать переменные разных типов и вывести их значения

2. Управляющие конструкции: условия и циклы

Теория:

- Условные операторы: if, elif, else
- Циклы: for, while
- Вложенные циклы и условия
- Операторы прерывания: break, continue

Практика:

- Реализовать программу поиска простых чисел или обработку списка с условиями и циклами

3. Функции и модули

Теория:

- Объявление функций с аргументами и возвратом
- Использование встроенных модулей (например, math)
- Импортирование собственных и сторонних модулей

Практика:

- Создать функцию для вычисления площади фигуры
- Импортировать модуль math и использовать его функции в своем коде

4. Работа с коллекциями: списки и словари

Теория:

- Создание и изменение списков
- Методы списков: append, pop, sort
- Создание и работа со словарями: добавление, удаление элементов

Практика:

- Обработать список данных, например, сортировать или фильтровать
- Создать словарь с информацией о пользователях и вывести его содержимое

5. Обработка исключений и основы отладки

Теория:

- Конструкция try/except для обработки ошибок
- Виды ошибок и ловля конкретных исключений
- Методы поиска и устранения ошибок (print, IDE debugger)

Практика:

- Написать программу с возможными ошибками и обработать их
- Использовать отладчик или print для поиска и исправления ошибок

6. Введение в работу с Pandas: Series и DataFrame

Теория:

- Структуры данных Pandas: Series и DataFrame
- Создание данных, просмотр и базовые операции (выборки, фильтрации)

Практика:

- Загрузить небольшие табличные данные
- Выполнить простую фильтрацию или агрегирование

7. Работа с файлами — чтение и запись

Теория:

- Операции с файлами: open, read, write
- Работа с CSV-файлами через модуль csv или pandas

Практика:

- Загрузить данные из файла, обработать их и сохранить результат в новый файл CSV

8. Регулярные выражения

Теория:

- Синтаксис регулярных выражений (шаблоны)
- Основные функции модуля re: search, match, findall, sub
- Примеры шаблонов для поиска и замены текста

Практика:

- Найти все email-адреса в тексте
- Заменить определённые слова или фразы по шаблону

9. Работа с базами данных и SQL из Python

Теория:

- Основные понятия SQL (выборка, вставка, обновление данных)
- Подключение к базе данных через sqlite3, SQLAlchemy или подобные библиотеки
- Выполнение запросов из Python

Практика:

- Создать небольшую базу данных и выполнить SELECT, INSERT и UPDATE запросы из скрипта

10. Основы ETL-процессов (Extract, Transform, Load)

Теория:

- Что такое ETL и его этапы
- Инструменты и подходы
- Простая схема реализации на Python

Практика:

- Написать скрипт для извлечения данных из файла, их преобразования (например, очистки или агрегации) и загрузки в новый файл или базу

11. Многопоточность и асинхронное программирование

Теория:
- Основы threading и asyncio
- Отличия и области применения каждого подхода
- Примеры применения: скачивание нескольких ресурсов, асинхронные API-запросы

Практика:
- Реализовать пример скачивания нескольких веб-страниц или вызов API в параллельном режиме

12. Работа с HTTP и API

Теория:
- Основы протокола HTTP, методы GET, POST
- Использование библиотеки requests для отправки запросов
- Обработка JSON-ответов

Практика:
- Получить данные с публичного API (например, погоды)
- Обработать и вывести интересующие параметры

13. Работа с большими объемами данных и оптимизация

Теория:
- Методы повышения скорости обработки данных, профилирование кода
- Использование генераторов, эффективных структур данных
- Подходы к обработке объемных данных в памяти

Практика:
- Оптимизировать существующий скрипт обработки данных
- Провести сравнение времени выполнения до и после оптимизации

14. Контейнеризация с Docker

Теория:
- Что такое Docker, основные понятия
- Создание Dockerfile и сборка образа
- Запуск контейнера и работа внутри него

Практика:
- Создать Docker-образ с Python-программой
- Запустить контейнер и проверить его работу

15. Итоговый проект — архитектура и подготовка к защите

Теория:
- Основы проектирования решений в Data Engineering
- Интеграция навыков, структура проекта
- Лучшие практики разработки и документирования

Практика:
- Начать работу над индивидуальным или групповым проектом

- Подготовить описание и план для предварительной защиты

16. Защита проектов и обзор курса

Теория:
- Презентация итоговых решений
- Вопросы и ответы, обмен опытом

Практика:
- Защитить проект перед группой или экспертом
- Обсудить подходы и возможности для дальнейшего развития

№	Тема урока	Количество часов (теория / практика)	Решаемая проблема в рамках НТИ	Цель	Задачи	Результаты		Оборудование	Контроль
						Soft skills	Hard skills		
Модуль 1. Современная веб-разработка на JavaScript (Frontend + Backend) (32 часа)									
1	Введение в JavaScript и современную веб-разработку	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Ознакомить учащихся с основами JavaScript и ролью языка в веб-разработке.	- Понять, что такое JavaScript и где он используется. - Познакомиться с основными инструментами разработки. - Создать первый простой скрипт.	- Восприятие новой информации. - Внимательность к деталям.	- Установка Node.js и редактора кода. - Написание базового JS-кода.	- Компьютер с доступом в интернет. - Редактор кода (VS Code или аналогичный). - Установленный Node.js.	Устный опрос, тест на базовое понимание
2	Основы синтаксиса JavaScript	0,5/1,5	Недостаточная подготовленность молодых специалистов к	Изучить базовые конструкции и типы	- Ознакомиться с типами	- Логическое мышление. - Усидчивость.	- Объявление переменных с var, let, const.	- Компьютер с доступом в интернет.	Практическое задание:

			современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	данных JavaScript.	данных и переменными. - Научиться использовать операторы.		- Применение арифметических и логических операторов.	- Редактор кода (VS Code или аналогичный). - Установленный Node.js.	базовые операции
3	Управляющие конструкции и работа с массивами	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в	Освоить условия, циклы и работу с массивами.	- Реализовывать ветвления (if, switch). - Использовать циклы (for, while). - Применять методы массивов.	- Аналитическое мышление. - Настойчивость.	- Написание условий и циклов. - Манипуляции с массивами.	- Компьютер с доступом в интернет. - Редактор кода (VS Code или аналогичный). - Установленный Node.js.	Выполнение задачи с использованием циклов

			высокотехнологичных сферах.						
4	Функции и область видимости	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Научиться создавать функции и понимать их область видимости.	- Ознакомиться с различными способами объявления функций. - Понять замыкания.	- Самоорганизация. - Внимание к деталям.	- Создание функций (function, стрелочные). - Работа с замыканиями.	- Компьютер с доступом в интернет. - Редактор кода (VS Code или аналогичный). - Установленный Node.js.	Короткий тест + практика: написание функций
5	Объекты в JavaScript	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что	Изучить работу с объектами и их методами.	- Создавать объекты и добавлять свойства. - Понимать контекст this.	- Критическое мышление. - Внимательность.	- Создание и изменение объектов. - Использование методов объектов.	- Компьютер. - Редактор кода. - Среда выполнения.	Практическое задание на работу с объектами

			затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.						
6	Работа с DOM: базовая манипуляция	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Познакомить с манипуляциями DOM.	- Изучить структуру DOM. - Научиться изменять HTML-элементы динамически.	- Творческий подход. - Внимательность.	- Использование методов querySelector, getElementById. - Добавление и удаление элементов.	- Компьютер. - Редактор кода. - Среда выполнения.	Домашнее задание с динамическим элементом
7	События в браузере и	0,5/1,5	Недостаточная подготовленность молодых специалистов к	Изучить работу с событиями в браузере.	- Понимание основных событий.	- Реакция на обратную связь.	- Назначение слушателей событий.	- Компьютер. - Редактор кода.	Практическое упражнение с

	обработчики событий		современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.		- Создание обработчиков.	- Внимание к деталям.	- Делегирование событий.	- Среда выполнения.	обработчиками
8	Асинхронность в JS: коллбеки и промисы	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в	Понять основы асинхронного программирования.	- Изучить коллбеки и Promise. - Использовать async/await.	- Терпение. - Аналитическое мышление.	- Создание асинхронного кода. - Работа с Promise и async/await.	- Компьютер. - Редактор кода. - Среда выполнения.	Тест + написание асинхронной функции

			высокотехнологичных сферах.						
9	Введение в Node.js и установка окружения	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Ознакомить с одной из популярных сред выполнения JS.	- Установить Node.js и npm. - Выполнить скрипт в Node.js.	- Самостоятельность. - Навыки работы с терминалом.	- Установка ПО. - Запуск скриптов Node.js.	- Компьютер. - Редактор кода. - Среда выполнения.	Мини-проект: запуск hello world сервера
10	Работа с файлами и модулями в Node.js	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что	Изучить работу с файловой системой и модулями.	- Чтение и запись файлов. - Создание и экспорт модулей.	- Внимательность. - Усидчивость.	- Использование fs-модуля. - CommonJS модули.	- Компьютер. - Редактор кода. - Среда выполнения.	Практическая работа: чтение и запись в файл

			затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.						
11	Создание простого веб-сервера на Node.js	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Научиться создавать и запускать сервер.	- Использовать HTTP-модуль. - Обрабатывать запросы и ответы	- Логическое мышление. - Способность к экспериментам.	- Создание сервера. - Работа с HTTP-запросами.	- Компьютер. - Редактор кода. - Среда выполнения.	Практическое задание: сервер с одним маршрутом
12	Введение в Express.js	0,5/1,5	Недостаточная подготовленность молодых специалистов к	Ознакомить с Express.js и базовой	- Установить Express.	- Усидчивость. - Умение работать с документацией.	- Использование Express.	- Компьютер. - Редактор кода.	Практическое задание: создание

			современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	маршрутизацией.	- Написать маршруты и middleware.		- Создание маршрутов.	- Среда выполнения.	маршрутов Express
13	Работа с REST API	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в	Изучить принципы REST и HTTP методы.	- Понять архитектуру REST. - Реализовать CRUD операции.	- Аналитика. - Коммуникация (для проектной работы).	- Создание REST API на Express. - Работа с JSON.	- Компьютер. - Редактор кода. - Среда выполнения.	Тест + создание простого CRUD API

			высокотехнологичных сферах.						
14	Взаимодействие Frontend с Backend	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Научиться отправлять запросы с frontend к backend.	- Использовать Fetch API. - Обрабатывать ответы сервера.	- Творческий подход. - Внимание.	- Отправка запросов AJAX. - Работа с JSON.	- Компьютер. - Редактор кода. - Среда выполнения.	Практическое задание: интеграция frontend с API
15	Основы React.js	0,5/1,5	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что	Познакомить с базовыми принципами React.	- Создать компонент. - Использовать props и state.	- Творческое мышление. - Самообучение.	- Работа с JSX. - Создание компонентов React.	- Компьютер. - Редактор кода. - Среда выполнения.	Практическое задание: написать простой компонент React

			затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.						
16	Итоговый проект и тестирование	0/2	Недостаточная подготовленность молодых специалистов к современным требованиям веб-и fullstack-разработки, что затрудняет их интеграцию в инновационные ИТ-проекты, работающие в высокотехнологичных сферах.	Применить полученные знания для создания полноценного проекта.	- Разработать mini fullstack-приложение. - Провести тестирование.	- Тайм-менеджмент. - Командная работа (если проект в группе). - Решение проблем.	- Интеграция frontend и backend. - Тестирование и отладка кода.	- Компьютер. - Средства разработки. - Средства тестирования (браузер, Postman).	Объединение frontend и backend, итоговая проверка знаний
Модуль 2. Python для разработки и Data Engineering (32 часа)									

17	Введение в Python: установка, запуск, синтаксис	0,5/1,5		Познакомить учеников с основами Python: выполнить установку среды, понять способ запуска программ и освоить базовый синтаксис языка.	- Установить и настроить Python и среду разработки (например, VS Code или Anaconda) - Запустить интерпретатор Python и написать первую программу - Ознакомиться с основными элементами синтаксиса: комментарии, вывод, переменные - Познакомиться со структурой простых программ на Python	- Навыки самостоятельной установки программного обеспечения - Внимательность и аккуратность при конфигурации среды разработки - Навыки постановки целей и планирования этапов обучения - Способность задавать уточняющие вопросы и искать решения проблем - Умение работать с документацией и руководствами пользователя	- Умение устанавливать и настраивать Python и IDE - Знание основ синтаксиса Python - Понимание структуры первой программы на Python	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Краткий тест (онлайн-викторина)

18	Управляющие конструкции: условные операторы и циклы	0,5/1,5		Освоить управление потоком выполнения программ с помощью условий и циклов.	- Изучить условные операторы if, elif, else - Освоить циклы for и while - Использовать операторы break и continue - Реализовать вложенные управляющие конструкции	- Логическое мышление - Внимательность к деталям - Способность писать читаемый и структурированный код	- Умение работать с условиями и циклами в Python - Понимание принципов управления потоком выполнения	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Практическое задание (написать скрипт)
19	Функции и модули	0,5/1,5		Научиться создавать функции и использовать модули для повторного использования кода.	- Объявлять и вызывать функции с параметрами и возвращаемыми значениями - Изучить области видимости переменных - Импортировать и	- Мышление модульное и системное - Навыки документирования и комментирования кода	- Создание и вызов функций в Python - Работа с модулями и пакетами	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Проверка кода (Code review)

					использовать сторонние модули - Создавать собственные модули				
20	Работа со списками и словарями	0,5/1,5		Освоить обработку и манипуляцию данными с использованием списков и словарей.	- Создавать и изменять списки и словари - Использовать методы и функции для работы с коллекциями - Проводить итерации по элементам коллекций - Решать практические задачи с использованием этих структур	- Аналитическое мышление - Умение структурировать данные	- Работа со списками и словарями в Python - Использование методов добавления, удаления, поиска элементов	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Практическое задание
21	Обработка исключений и отладка	0,5/1,5		Обучить работе с ошибками и инструментам отладки кода.	- Понимать типы ошибок в Python - Использовать	- Настойчивость при поиске ошибок	- Работа с исключениями в Python	- Компьютеры - Необходимое программное обеспечение: - Python	Краткий тест

					try-except для обработки исключений - Применять основное отладочное ПО и встроенные инструменты IDE - Писать устойчивый к ошибкам код	- Внимание к деталям - Критическое мышление	- Использование отладчика IDE	- Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	
22	Введение в библиотеки для работы с данными (Pandas)	0,5/1,5		Познакомить с библиотекой Pandas для обработки табличных данных.	- Установить и импортировать Pandas - Создавать и работать с DataFrame и Series - Загружать и сохранять данные в CSV, Excel - Применять базовые функции фильтрации и агрегации	- Навыки критического анализа данных - Усидчивость при работе с большими массивами данных	- Использование Pandas для анализа данных	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Практическое задание (работа с таблицей)

23	Работа с файлами: чтение и запись	0,5/1,5		Научить работу с файлами для импорта и экспорта данных.	- Чтение и запись текстовых файлов - Обработка CSV и JSON файлов - Использование контекстных менеджеров (with) - Решение практических задач по работе с файлами	- Внимательность при работе с данными - Самостоятельность	- Работа с файлами в Python - Использование стандартных библиотек	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Практическое задание
24	Регулярные выражения и обработка текстовых данных	0,5/1,5		Обучить поиску и обработке текстовых данных с помощью регулярных выражений.	- Синтаксис регулярных выражений - Использование модуля re в Python - Поиск, замена и валидация текста - Решение задач с реальными	- Внимательность к деталям - Способность к анализу паттернов	- Работа с регулярными выражениями в Python	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к	Краткий тест + практика

					текстовыми данными			документации и руководствам	
25	Работа с базами данных и SQL	0,5/1,5		Освоить основы работы с реляционными базами данных и SQL на Python.	- Установка и подключение к СУБД (например, SQLite, PostgreSQL) - Выполнение базовых SQL-запросов (SELECT, INSERT, UPDATE, DELETE) - Использование библиотек (sqlite3, SQLAlchemy) - Импорт и экспорт данных между Python и БД	- Логическое мышление - Организация данных	- Основы SQL - Подключение и взаимодействие с базами данных на Python	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Практическое задание (запросы)
26	Введение в ETL-процессы	0,5/1,5		Понять основы Extract-Transform-Load (ETL) и их	- Объяснение этапов ETL - Примеры инструментов	- Проектное мышление - Решение комплексных задач	- Знание ETL-концепций - Реализация ETL на Python	- Компьютеры - Необходимое программное обеспечение: - Python	Аналитическое задание

				роль в Data Engineering.	и реализации на Python - Построение простого ETL-пайплайна - Обработка и преобразование данных			- Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	
27	Многопоточность и асинхронность в Python	0,5/1,5		Познакомить с многопоточностью и асинхронным выполнением в Python.	- Понимание потоков и процессов - Использование threading и multiprocessing - Основы async/await - Примеры оптимизации задач	- Критическое мышление - Решение задач повышения производительности	- Работа с потоками и асинхронным кодом	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Практическое задание
28	Веб-запросы и API интеграция	0,5/1,5		Научиться получать и отправлять	- Использование библиотеки requests	- Навыки коммуникации с технической документацией	- Работа с REST API через Python	- Компьютеры - Необходимое программное обеспечение:	Практическое задание

				данные через API.	- Формирование GET и POST запросов - Парсинг ответов (JSON и др.) - Работа с внешними сервисами	- Гибкость в работе с разными источниками данных		- Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	
29	Обработка больших данных: основы	0,5/1,5		Дать представление о работе с большими данными и инструментах Python для этого.	- Введение в Apache Spark и PySpark - Основные концепции распределенной обработки - Написание простых Spark-программ на Python - Основы эффективности и масштабируемости	- Аналитическое мышление - Умение работать с абстрактными концепциями	- Знание PySpark (базово)	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Краткий тест
30	Контейнеризация	0,5/1,5		Познакомить с технологией Docker для	- Установка и базовое	- Ориентация в новых технологиях	- Docker основы	- Компьютеры	Практическое задание

	приложений (Docker)			упаковки и развертывания приложений.	знакомство с Docker - Создание Dockerfile для Python-приложения - Запуск контейнеров - Понимание контейнеризации в Data Engineering	- Самостоятельное изучение		- Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	(создать Dockerfile)
31	Итоговый проект: сборка и обработка данных	0,5/1,5		Закрепить знания на практике, реализовав полный пайплайн сбора и обработки данных.	- Спроектировать проект с использованием приобретённых технологий - Реализовать чтение, обработку и сохранение данных - Использовать функции, модули, Pandas,	- Командная работа (если групповая) - Тайм-менеджмент - Критическое и творческое мышление	- Комплексное применение изученных технологий	- Компьютеры - Необходимое программное обеспечение: - Python - Среда разработки (например, Visual Studio Code, PyCharm или Anaconda) - Браузер для доступа к документации и руководствам	Промежуточная защита проекта

					работу с файлами и API				
32	Итоговое занятие и защита проектов	0/2		Презентовать и защитить проекты, получить обратную связь.	- Демонстрация проектов - Обсуждение решений и трудностей - Получение рекомендаций и отзывов	- Навыки презентации и коммуникации - Открытость к критике - Самоанализ	- Умение презентовать технические проекты	- ПК с проектором или экраном - Средства презентации (PowerPoint, Jupyter Notebook и т.п.)	Итоговая защита + тест

УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММЫ

А) Учебно-методическое обеспечение программы

1. Пример: [GitHub — Course Syllabus Python Data Engineering](https://github.com/username/python-data-engineering-syllabus) (пример шаблона)
2. Методические рекомендации для преподавателей
2. Руководство по преподаванию Python и Data Engineering:
3. [DataCamp Instructor Resources](https://www.datacamp.com/instructor-resources)
4. Пошаговые конспекты лекций и презентации по Python
5. [Python для начинающих — презентации от Stepik](https://stepik.org/course/67/promo)
3. Учебники и пособия по Python
6. - «Изучаем Python» (Марк Лутц) — [Ozon](https://www.ozon.ru/product/izuchaem-python-5-e-izdanie-195031551/)
7. «Python Programming: An Introduction to Computer Science» (John Zelle) — [официальный сайт](http://mcsp.wartburg.edu/zelle/python/)
4. Практические задания и кейсы для самостоятельной работы
8. [LeetCode Python задачи](https://leetcode.com/problemset/all/?difficulty=Easy&topicSlugs=python)
9. [HackerRank Python](https://www.hackerrank.com/domains/tutorials/10-days-of-python)
5. Наборы контрольных и тестовых заданий
10. [Test and Quiz Examples on Python — GitHub](https://github.com/allenwzhou/python-unittest-examples)
6. Примеры готового кода и скриптов
11. [Awesome Python Scripts Collection](https://github.com/krzjoa/awesome-python-scripts)
7. Руководства по установке и настройке программного обеспечения
12. - Python: [Официальный сайт Python](https://www.python.org/downloads/)
13. IDE PyCharm: [JetBrains PyCharm](https://www.jetbrains.com/pycharm/download/)
8. Технические документации и справочники по стандартным библиотекам Python
14. [Официальная документация Python](https://docs.python.org/3/library/index.html)
9. Учебные проекты и практические работы
15. [Real Python Projects](https://realpython.com/tutorials/projects/)
10. Видеолекции и обучающие ролики
16. [Канал «Corey Schafer» (YouTube)](https://www.youtube.com/user/schafer5)
17. [Курс «Python для анализа данных» на Coursera](https://www.coursera.org/learn/python-for-data-science)
11. Инструкции по работе с инструментами отладки и профилирования кода
18. [Python Debugger (pdb) Guide](https://docs.python.org/3/library/pdb.html)
12. Материалы по работе с базами данных и SQL
19. [SQL для Data Engineering — Tutorialspoint](https://www.tutorialspoint.com/sql/)
13. Методики и презентации по обработке больших данных и ETL-процессам
20. [Apache Airflow Documentation](https://airflow.apache.org/docs/)
21. [ETL Best Practices — Databricks](https://databricks.com/blog/etl-best-practices)
14. Материалы по использованию Docker и контейнеризации приложений
22. [Docker для разработчиков Python](https://docs.docker.com/samples/python/)
15. Технические задания для итогового проекта курса
23. Пример: [Data Engineering Capstone Project — GitHub](https://github.com/username/data-engineering-capstone)

16. Рекомендации по самостоятельному обучению и расширению знаний

24. [Awesome Python Learning
Resources](https://github.com/jackfranklin/awesome-python-learning)

Б) Информационное обеспечение программы

1. Учебная программа и учебный план курса
Пример: [Sample Syllabus Python Data Engineering
(GitHub)](https://github.com/krzjoa/python-data-engineering-syllabus)
2. Методические рекомендации для преподавателей
[Ресурсы для преподавателей от
DataCamp](https://www.datacamp.com/instructor-resources)
3. Учебно-методические материалы (лекции, презентации, пособия)
[Stepik — Курс «Программирование на Python»](https://stepik.org/course/67)
4. Учебные пособия и справочные материалы
- «Изучаем Python» (Марк Лутц) —
[Ozon](https://www.ozon.ru/product/izuchaem-python-5-e-izdanie-195031551/)
- Официальная документация Python — [Python Docs](https://docs.python.org/3/)
5. Лабораторные и практические задания
[HackerRank — Решение задач на
Python](https://www.hackerrank.com/domains/tutorials/10-days-of-python)
6. Информационные ресурсы и платформы для самостоятельного обучения
[Coursera — Python for Everybody](https://www.coursera.org/specializations/python)
7. Образцы кода и проекты
[GitHub — Awesome Python Scripts](https://github.com/krzjoa/awesome-python-scripts)
8. Технические руководства и документация по инструментам
- [Docker для Python](https://docs.docker.com/samples/python/)
- [Apache Airflow Documentation](https://airflow.apache.org/docs/)
9. Источники и наборы данных для практики
[Kaggle Datasets](https://www.kaggle.com/datasets)
10. Онлайн-платформы для совместной работы и контроля версий
[GitHub](https://github.com/)
[Google Colab](https://colab.research.google.com/)
11. Ресурсы для повышения квалификации и профессионального сообщества
[Stack Overflow Python Tag](https://stackoverflow.com/questions/tagged/python)
[PyData Community](https://pydata.org/)
12. Системы оценки знаний и обратной связи
[Moodle](https://moodle.org/) — платформа для организации онлайн-курсов и тестирования
13. Консультационные и технические службы поддержки
Форумы и чаты:
- [Python-ru Forum](https://forum.python.org.ru/)
- [Reddit r/learnpython](https://www.reddit.com/r/learnpython/)
14. Информация об условиях лицензирования и авторских правах
[Creative Commons](https://creativecommons.org/licenses/) — лицензии для учебных материалов
[Python Software Foundation License](https://docs.python.org/3/license.html)