Web-разработчик на Python

Курс предназначен для подготовки специалиста, который сможет не только решать рядовые задачи бекенд-разработки, но и сделать с нуля современную фронтенд часть.

Длительность курса: 180 академических часов

1 Основы python и модульного тестирования

1	**Знакомство с курсом. Проектирование "хорошей" системы. Написание "чистого" кода**	правильно декомпозировать код; именовать переменные и функции; оценивать сложность кода и использовать статические анализаторы.
2	**Модули, библиотеки, пакеты**	собирать пакет; объяснить, что такое области видимости, локальные и глобальные переменные; импортировать данные из модулей и пакетов; пользоваться конструкцией if __name__ == '__main__'; объяснять разницу между модулем и пакетом

запомнить варианты импортов.

Домашние задания

1 Создать программу-поисковик

Цель: В этой самостоятельной работе тренируем умения:
1. Писать "чистый код"
2. Собирать пакеты

Чтобы:
Применять принципы написания чистого кода и сборку пакетов

Задача:

Создать программу поисковик (консольную)
Пользователь вводит текст запроса, поисковую систему (google.com, yandex.ru, ...), количество результатов, рекурсивный поиск или нет, формат вывода (в консоль, в файл json, в csv)
Программа находит в интернете начиная от стартовой точки все ссылки на веб-странице в заданном количестве (название ссылки и саму ссылку)
Если поиск не рекурсивный, то берем ссылки только из поисковика, если рекурсивный, то берем первую ссылку, переходим, находим там ссылки, переходим, ...
В зависимости от выбранного формата вывода сохраняем результат (текст ссылки: ссылка) либо в консоль либо в файл выбранного формата

0. Создать репозиторий для нового проекта (gitlab, github, ...)

1. Решить задачу
2. Обратить внимание на следующие принципы:

1. декомпозиция сверху вниз
2. srp - принцип единственной ответственности
3. термины предметной области
4. уменьшение зависимости
5. чистые функции

		6. цикломатическая сложность python3 -m mccabe --min 5 module.py flake8 --max-complexity 5 7. понятные названия у переменных, функций, классов, модулей 8. контекст ближе к коду (привязка к комитам, тикетам, комменты, документация, вики) 9. разумное использование фишек python 10. код на английском а не на python 11. фичеризм - не слишком гибко, не слишком жестко, обобщать когда используется 2 раза 12. тесты демонстрирующие не очевидное поведение 13. статический анализ кода pycodestyle, flake8, ast 4. Добавить setup.py для сборки программы в пакет 5. Сдать дз в виде ссылки на репозиторий
3	**Генераторы, тернарные операторы, исключения, декораторы**	писать генераторы; писать тернарные операторы; объяснить назначение декораторов; объяснить как писать декоратор; осознать назначение исключений; объяснить как обрабатывать исключения.
4	**Основы ООП. Принципы ООП**	создавать классы; создавать объекты классов; создавать свойства и методы класса; создавать методы доступа; объяснить как реализованы принципы ООП в python.
5	**ООП. Магические методы, утиная типизация, статические методы, методы класса**	писать код с применением основных магических методов и утиной типизации; объяснить, что такое магические методы и зачем они нужны; объяснить для чего нужны статические методы и методы класса и как их писать.

Домашние задания

1 Написать игру "Лото"

Цель: В этой самостоятельной работе тренируем умения:
1. разбивать программу на классы
2. у класса создавать свойства, методы, инициализатор
3. создавать объекты класса
4. создавать методы доступа
5. использовать магические методы

Эти умения нам понадобятся при написании программ на ООП

1. Создать новый проект "Игра лото"

2. Правила игры можно скачать тут: https://drive.google.com/open?id=1bDMcmVfhYaTUw3CB-MV8WtIIXdYEutvX

3. Написать игру лото

Возможные подходы к решению задачи:
1) Проектирование на основании предметной области. Подумать какие объекты есть в игре и какие из них можно перенести в программу. Для них создать классы с соответствующими свойствами и методами. Проверить каждый класс отдельно. Написать программу с помощью этих классов

2) Метод грубой силы + рефакторинг. Написать программу как получится. После этого с помощью принципа DRY убрать дублирование в коде

3) Процедурное программирование

4. Минимальные требования: 2 игрока - человек играет с компьютером
5. (Дополнительно *) возможность выбирать тип обоих игроков (компьютер или человек) таки

образом чтобы можно было играть: компьютер - человек, человек - человек, компьютер - компьютер
6. (Дополнительно *) возможность играть для любого количества игроков от 2 и более

7. Сдать дз в виде ссылки на репозиторий

Для тренировки предлагаю пользоваться принципами "чистого" когда, которые разбирали на первом занятии

6	**Введение в автотесты. pytest**	писать тесты для функций и классов на pytest; запускать тесты; объяснить, что такое автоматизированное тестирование; осознать зачем нужно автоматизированное тестирование; запомнить плюсы и минусы библиотеки pytest; объяснить для чего нужны методы setup, teardown. Домашние задания 1 Покрыть предыдущие дз тестами Цель: В этой самостоятельной работе тренируем умения: 1. пользоваться библиотекой pytest 2. писать тесты для имеющихся функций и методов 3. запускать тесты Для того чтобы проверять код программы автоматически, писать чистый код, создавать оптимальную архитектуру программы 1. В проекте "Лото" написать тесты 2. Если написать тесты не удается, попробуйте разбить программу так, чтобы в ней было больше чистых функций 3. В качестве тренировки можно попробовать написать тесты для парсера из 1-го дз 4. Сдать дз в качестве ссылки на репозиторий с проектом

2 Создаем свой блог и начинаем создавать обучающий сайт. База данных и ORM, web-фреймворки Flask и Django. MVC, MVT

1	**Основы реляционных БД. Работа с sqlite**	подключаться к базе данных с помощью python; писать select запросы в базу данных из python; понять назначение реляционных баз данных; объяснить как хранятся данные в базе; осознать зачем нужны нормальные формы; запомнить типы запросов в базу.
2	**ORM, SQLAlchemy**	создавать модели данных с помощью SQLAlchemy для заданной предметной области; объяснить, что такое ORM, для чего он используется; объяснить как делать основные запросы в базу данных с помощью ORM. Домашние задания 1 Создание моделей данных для сайта "Мой блог" на выбранную тему Цель: В этой самостоятельной работе тренируем умения: 1. Создавать модели данных 2. Создавать связи между моделями 3. Работать с сессией 4. Делать простые запросы Смысл: Для того чтобы работать с SQLAlchemy в проектах с базой данных. Понимать как работать с orm Создать модели Post, Tag для сайта "Мой блог" на тему (ваша тема). Для пользователя можно использовать стандартную модель User. Установить связи между моделями. Добавить некоторые данные. Выбрать все посты конкретного пользователя с 2-мя любыми тегами

1. Создать новый проект "Мой блог", по нему будет 3 домашних задания. Рекомендуется создать для этого проекта отдельный репозиторий
2. Придумать тему блога. Она может быть любая какая вам более интересна (например экзотические птицы, занятия workout-ом, искусство, ...)
3. С помощью SQLAlchemy создать модели данных для блога, например (Post, User, ...) и все другие, которые вы считаете важными
4. Установить связи между моделями
5. В качестве примера ввести некоторые данные
6. Выбрать все посты конкретного пользователя, попробовать сделать другие запросы (Рекомендуется сделать это в виде тестов pytest, можно просто с помощью print)
7. Сдать дз в виде ссылки на репозиторий

3	**Знакомство с Front-end частью курса. Основы HTML, CSS, методологии верстки. Немного Bootstrap 4**	писать css селекторы; запомнить устройство http, web, rest; осознать назначение кодов ответа; объяснить как связаны html, css, js и из чего они состоят; проанализировать какие есть способы разработки css; объяснить как пользоваться bootstrap4. Домашние задания 1 Сделать верстку для сайта "Мой блог" Цель: Сделать верстку для сайта Мой блог. Страницы: главная, все посты, 1 пост

4	**Введение в werkzeug; Flask**	запустить тестовый сервер на Flask; проанализировать как связаны view и шаблоны; объяснить как работает шаблонизатор, что это такое; объяснить зачем нужны Blueprint и как их использовать; создать небольшой проект на Flask.
5	**Werkzeug; Flask + SQLAlchemy. Работа с моделями данных**	добавлять модели и базу данных в проект на Flask; проанализировать паттерн MVC и зачем он нужен; настроить Flask для работы с SQLAlchemy; объяснить как сохранять и получать данные. Домашние задания 1 Сделать сайт "Мой блог" на Flask + SQLAlchemy Цель: Сделать сайт "Мой блог" на Flask + SQLAlchemy. Страницы: главная, посты, 1 пост

6	**Django settings, orm, админка, миграции, superuser**	добавлять модели и базу данных в проект на Django; делать миграции данных; сохранять данные в базу; объяснить как создавать проект на Django; запомнить из чего состоит проект; осознать, что такое миграции и зачем они нужны; посмотреть на стандартную админку. Домашние задания 1 Обучающий сайт на выбранную тему Цель: Обучающий сайт (тема обучения может быть любая). Сделать модели данных для сайта. Примерная работа сайта. На сайте есть список курсов, каждый курс ведет преподаватель, студент может записаться на курс. На курсе есть некоторое количество занятий по расписанию.
7	**Тестирование django приложений. Тестирование моделей. mixer для создания фейковых данных**	тестировать django-приложения; запускать тесты; объяснить для чего setUp и tearDown; создавать фейковые данные с помощью mixer.

8 **Django cbv, шаблоны, наследование шаблонов**

разобраться как работает шаблонизатор django;
объяснить для чего и как использовать наследование шаблонов;
осознать что такое cbv в django;
объяснить какие классы из cbv используются для crud;
разобраться для чего нужны классы View и TemplateView;
осознать для чего нужны Mixins и как они позволяют расширять стандартные классы.

Домашние задания

1 Страницы для создания, удаления, редактирования, просмотра 1-го курса и списка курсов

Цель:

3 Создаем backend для обучающего сайта. REST API, django-rest-framework, GraphQL, оптимизация работы с базой данных

1	**Django forms. Наследование моделей. Абстрактные классы и proxy в django**	взаимодействовать с пользователем с помощью Django Forms; проанализировать различные варианты форм; объяснить как можно настраивать форму; разобраться с наследованием моделей в Django; проанализировать варианты наследования.
2	**Азы работы с очередями задач**	разобраться зачем нужны очереди задачи; настроить rq и redis; создавать задачи; запускать задачи по отдельности и по расписанию; делать в проекте на django. Домашние задания 1 Добавить страницу с контактами Цель: Добавить страницу с контактами На странице создать форму для отправки сообщения После отправки формы отправлять письмо на почту администратора и второе письмо на почту указанную в форме Отправку писем реализовать через очередь задач

3	**Введение в django-rest-framework**	объяснить зачем нужен rest framework; установить rest framework; работать с APIView; объяснить для чего и как используются сериализаторы; создать CRUD для модели данных.
4	**Django-rest-api авторизация**	проанализировать варианты авторизации с django-rest-framework; объяснить в каком случае какой вариант используется; реализовать некоторые варианты; объяснить как происходит авторизация по JWT и Oauth2. Домашние задания 1 Создать rest-api для сайта Цель: Создать rest-api для сайта. Реализовать авторизацию пользователя
5	**Тестирование django приложений. Тестирование views. Тестирование api**	использовать тестовый клиент для тестирования view в django; объяснить, что можно проверять на странице; писать тесты для api.

6	**API. GraphQL и его реализация в Python. GraphQL и Django**	разобраться зачем нужен GraphQL; объяснить как он реализован в python; объяснить как создавать схему; проанализировать варианты использования GraphQL; фильтровать данные с GraphQL; изменять (мутировать) данные. Домашние задания 1 С помощью GraphQL создать схему Цель: С помощью GraphQL создать схему, которая позволяет получать одновременно курсы, преподавателей и всех студентов записанных на курс
7	**Django m2m, select_related/prefetch_related, django debug toolbar, faker**	объяснить зачем нужен django-debug-toolbar; установить и настроить; настроить админку для manytomany; создавать management commands скрипты; создавать случайные данные с помощью faker; добавлять many_to_many записи; объяснить зачем нужны prefetch_related и select_related и в чем их разница.

8	**Django ORM, оптимизация работы с БД**	писать запросы с применением F-объектов; оптимизировать запросы с помощью exists; оптимизировать запросы с помощью cashed_property; объяснить для чего и как использовать bulk update, iterator в queryset, аннотации. Домашние задания 1 Оптимизировать работу с базой данных. Написать отчет Цель: Оптимизировать работу с базой данных, используя изученные средства. Написать отчет. Как было до оптимизации, какое средство использовалось, что стало после оптимизации
9	**Code review бэкенд части приложения**	делать code review; проанализировать слабые места своей работы; запомнить best practice.

4 Начинаем создавать frontend часть обучающего сайта, получаем данные с backend. Основы html, css, js, ES6, node.js, webpack, ajax

1	**Основы JS: типы данных, операторы, объекты, работа с DOM и браузером**	разобраться с типами данных и приведением типов в js операторами, обработкой ошибок, циклами, условными операторами; DOM и BOM; разобраться с обработчиками событий.
2	**Продвинутый JS: ООП в JS, прототипирование, асинхронность**	разобраться с ООП в js; разобраться с объектами и функциями; объяснить разницу конструкторов в es5 и es6; разобраться с прототипами; объяснить разницу методов в es5 и es6; объяснить разницу в наследовании в es5 и es6; разобраться с асинхронностью и моделью памяти.
3	**ES6, NodeJS окружение, babel + webpack, транспайлинг**	объяснить разницу кода в ES6; разобраться с деструкцией и распаковкой; проанализировать объекты в ES6, getters, setters; разобраться с import, export; объяснить зачем нужен node.js, npm, babel, webpack; разобраться с настройкой проекта. Домашние задания 1 Сборка UI с помощью webpack, форма логина и регистрации Цель: Сборка UI с помощью webpack, форма логина и регистрации, проксирование вызовов на back-end

4	**CSS препроцессоры". "fetch || axios || $.ajax для REST запросов, модульность**	написать код на less; объяснить зачем нужны css препроцессоры; установить less в webpack; объяснить зачем нужны ajax, axios, fetch и в чем их разница. Домашние задания 1 Сделать страницу курсов и при ее загрузке получать данные из api с помощью fetch или axios или ajax Цель: Сделать страницу курсов и при ее загрузке получать данные из api с помощью fetch или axios или ajax и выводить на страницу динамически
5	**Код ревью frontend части**	объяснить слабые места своей работы; запомнить best practice.
6	**Основы React, JSX, компоненты React**	создавать приложение на react; создавать react-компоненты; объяснить, что такое компонентный подходи и зачем он нужен; разобраться со структурой react-приложения; проанализировать как работает render; разобраться с virtual DOM. Домашние задания 1 Сделать главную страницу на react

5 Создаем SPA приложение на React, собираем все воедино. React, Redux, SPA, тестирование в js, docker

1 **State и props, data-flow в React-компонентах**

менять state компонента по событию;
запомнить варианты использования props;
запомнить какие есть доступные события;
объяснить, что такое обертка на event и зачем она нужна;
объяснить, что такое state и для чего он используется;
объяснить разницу между state и props;
проанализировать принцип data-flow.

---

2 **Жизненный цикл React-компонент**

проанализировать этапы жизненного цикла react-компонента;
объяснить в каком методе лучше делать загрузку данных с сервера и почему;
разобраться с загрузкой данных через fetch.

Домашние задания

1 Сделать страницу курсов, одного курса и записи на курс на react

---

3 **Состояние приложения. Flux & Redux**

объяснить зачем нужны Flux и Redux и в чем их разница;
запомнить роли во Flux;
запомнить реализации Flux;
разобраться с работой Redux и его основными принципами.

Домашние задания

1 Перевести все страницы сайта на react

4	**Code review frontend части на react**	проанализировать слабые места своей работы; best practice.
5	**Routing в React. SPA**	объяснить, что такое SPA и для чего он используется; рассмотреть Routing в react; разобраться как создать SPA приложение. Домашние задания 1 Организация всего приложения в виде SPA
6	**Тестирование JS приложений**	проанализировать инструменты тестирования в js; писать тесты; запускать тесты. Домашние задания 1 Написание unit-тестов для UI и back-end
7	**Введение в docker, docker-compose**	объяснить, что такое docker и для чего он нужен; проанализировать плюсы и минусы docker; разобраться с примерами настроек docker-контейнера; собирать docker-контейнер для django проекта проанализировать основные команды docker, понять как они работают. Домашние задания 1 Завернуть проект в докер-контейнеры

8	**Контектсные процессоры и middleware в django. Подведение итогов**	писать контекстные процессоры; объяснить строение middleware в django оценить результаты обучения на курсе.

6 Работа с макетами

1	**Основные ошибки при инспектировании макетов**	разобрать самые типичные ошибки при верстке с макетов от дизайнеров; смотреть сетку в макете; инспектировать цвета, размеры текста и элементов.
2	**Рабочая машина под названием Figma**	работать с редактором Figma; использовать панели программы при работе с макетами от дизайнеров; правильно экспортировать графику; комментировать в редакторе.

7 Проект

1	**Выбор темы и организация проектной работы**	выбрать и обсудить тему проектной работы; спланировать работу над проектом; ознакомиться с регламентом работы над проектом. Домашние задания 1 Выбор проекта и дальнейшая работа с ним
2	**Консультация по проектам и домашним заданиям**	получить ответы на вопросы по проекту, ДЗ и по курсу.
3	**Защита проектных работ**	защитить проект и получить рекомендации экспертов.