УТВЕРЖДАЮ

Директор ООО «ТОП-ТАЙМ»
_____А.И. Даишева
М.П.

Приказ №5 от 14 июля 2024г.

ОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА
дополнительного образования, дополнительная профессиональная программа
профессиональной переподготовки
«Python-разработчик (Django, DRF)»

Г. Казань
2024

СОДЕРЖАНИЕ ПРОГРАММЫ:

1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1. Дополнительная профессиональная программа профессиональной переподготовки «Python-разработчик (Django, DRF)», разработана в соответствии с нормами:

- Федерального закона от 29 декабря 2012 г. № 273-ФЗ «Об образовании в Российской Федерации» (далее - Федеральный закон № 273-ФЗ);

- Приказа Министерства образования и науки России от 01.07.2013 г. № 499 «Порядок организации и осуществления образовательной деятельности по дополнительным профессиональным программам»;

- Приказа Министерства труда и социальной защиты Российской Федерации от 18 января 2017 года N 44н об утверждении профессионального стандарта «Разработчик Web и мультимедийных приложений»;

- Приказа Министерства здравоохранения и социального развития Российской Федерации от 11 января 2011 года №1н об утверждении Единого квалификационного справочника должностей руководителей, специалистов и служащих, раздел «Квалификационные характеристики должностей руководителей и специалистов высшего профессионального и дополнительного профессионального образования».

1.2. Структура и содержание рабочей программы представлена учебным планом по изучаемому предмету, календарным графиком, программами учебных предметов, планируемыми результатами освоения программы, условиями реализации, системой оценки результатов освоения программы.

В рабочей программе содержится перечень учебных предметов с указанием объемов времени, отводимых на освоение каждого предмета, включая объемы времени, отводимые на теоретическое обучение. В рабочей программе по учебному предмету раскрывается рекомендуемая последовательность изучения разделов и тем, указывается распределение учебных часов по разделам и темам. В рабочей программе учебного предмета приводится содержание предмета с учетом требований к результатам освоения в целом рабочей программы.

1.3. Целью Программы является изучить язык Python и научиться применять его для решения задач анализа данных и машинного обучения.

Задачи Программы:

- Изучить базовый синтаксис языка Python;
- Изучить основные стандартные модули языка Python;
- Изучить основы функционального программирования в Python;
- Изучить основы объектно-ориентированного программирования в Python.

1.4. Категории слушателей:

Программа курсов предназначена для лиц, имеющих среднее профессиональное и (или) высшее образование; а также лиц, получающих среднее профессиональное и (или) высшее образование.

Курс разработан для тех, кто

- хочет найти работу бэкенд-разработчика на Python в большой компании;
- готов уделить обучению и практическим занятиям в течении 6 месяцев;
- обладает хорошими базовыми знаниями о компьютерах, интернете и операционных системах.

1.5. Дополнительная профессиональная образовательная программа построена на специфических базовых **принципах обучения**:

• принцип непрерывности (опора на самообразование, освоение умений и навыков учения, развитие ценностных ориентаций в духе «учения через всю жизнь», широкое использование активных форм и методов обучения, подход к обучению как процессу

преобразования жизненного и профессионального опыта, гуманистический тип отношений участников образовательного процесса);

• синергетический принцип (признание субъективности человеческого знания и интересов личности, являющихся имманентным потенциалом саморазвития и самоорганизации личности, сотрудничество, взаимозависимость и личная автономия);

• андрагогический принцип (принцип опоры на жизненный и профессиональный опыт слушателей, индивидуализации обучения, диалогового сотрудничества, принцип вариативности).

1.6. Трудоемкость освоения программы: 270 ак. часов.

1.7. Форма обучения: очная.

1.8. Виды занятий:

Обучение осуществляется в форме занятий в группах:

-лекции;

- практические занятия;

- тестирование;

- итоговая дипломная работа.

1.9. Документ, выдаваемый после завершения обучения – диплом о профессиональной переподготовке.

2. Результат освоения программы

2.1. В результате обучения слушатели приобретают знания, необходимые для качественного совершенствования профессиональных компетенций.

Описание трудовых функций, входящих в профессиональный стандарт.

Обобщенные трудовые функции			**Трудовые функции**		
код	наименование	уровень квалификации	наименование	код	уровень (подуровень) квалификации
A	Техническая поддержка процессов создания (модификации) и сопровождения информационных ресурсов	4	Проверка и отладка программного кода	A/01.3	3
			Работа с системой контроля версий	A/02.3	3
			Верстка страниц ИР	A/03.4	4
			Кодирование на языках web-программирования	A/04.4	4
			Тестирование ИР с точки зрения логической целостности (корректность ссылок, работа элементов форм)	A/05.4	4
			Тестирование интеграции ИР с внешними сервисами и учетными системами	A/06.4	4
			Проведение работ по резервному копированию ИР	A/07.4	4
			Управление доступом к данным и установка прав пользователей ИР	A/08.4	4
			Регистрация и обработка запросов заказчика в службе технической поддержки в соответствии с трудовым заданием	A/09.4	4
B	Выполнение работ по созданию (модификации) и	5	Сбор предварительных данных для выявления требований к ИР	B/01.5	5

	сопровождению информационных ресурсов		Определение первоначальных требований заказчика к ИР и возможности их реализации	B/02.5	5
			Планирование коммуникаций с заказчиком в рамках типовых регламентов организации	B/03.5	5
			Проектирование разделов ИР	B/04.5	5
			Установка и настройка прикладного программного обеспечения и модулей	B/05.5	5
			Тестирование интеграции ИР с внешними сервисами и учетными системами с использованием взаимодействия компонентов распределенной системы	B/06.5	5
			Проведение и регламентация работ по резервному копированию и развертыванию резервной копии ИР	B/07.5	5
			Управление доступом к данным и определение уровней прав пользователей ИР	B/08.5	5
			Обеспечение безопасной и бесперебойной работы сайта	B/09.5	5
			Регистрация и обработка запросов заказчика в службе технической поддержки	B/10.5	5
			Разработка процедур интеграции программных модулей	B/11.5	5
			Осуществление интеграции программных модулей и компонент и верификации выпусков программного продукта	B/12.5	5
c	Управление работами по созданию (модификации) и сопровождению информационных ресурсов	6	Анализ и формализация требований к ИР	C/01.6	6
			Разработка технических спецификаций на ИР	C/02.6	6
			Проектирование ИР	C/03.6	6
			Тестирование ИР с точки зрения пользовательского удобства на основании данных о поведении пользователей	C/04.6	6
			Организация работ по обеспечению безопасной работы ИР	C/05.6	6
			Организация работ по интеграционному тестированию ИР с внешними сервисами и учетными системами	C/06.6	6
D	Управление процессами и проектами по созданию (модификации) информационных ресурсов	7	Управление процессом разработки программного обеспечения	D/01.6	6
			Руководство разработкой проектной и технической документации	D/02.6	6
			Руководство проектированием ИР	D/03.7	7

			Управление процессами оценки сложности, трудоемкости, сроков выполнения работ	D/04.7	7
			Руководство проверкой работоспособности ИР	D/05.7	7
			Экспертная оценка функционирования ИР и планирование методов его реализации	D/06.7	7

Слушатели должны знать:

- основы Backend-разработки на Python;
- основы работы с Django Rest Framework.

Слушатели должны уметь:

- создавать веб-приложений на Django.

2.2. Контроль и оценка результатов освоения программы осуществляется с помощью итоговой аттестации– защиты дипломной работы.

Обучение включает в себя теоретическую часть, тестирование, практические занятия.

Обучение завершается итоговой аттестацией с выдачей **диплома о профессиональной переподготовке.**

Учебные главы	Номер недели																							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
Введение в курс	1 ч																							
Основы Python	12 ч	12 ч	12 ч	12 ч	5 ч																			
Бэкинг на Django					7 ч	12 ч	12 ч	12 ч	11 ч															
API										12ч	12ч	12ч	12ч	6ч										
Инфраструктура														6ч	12 ч	12ч	12 ч	12 ч						
Дипломная работа																			11 ч	11 ч	10 ч	10ч	10 ч	
Итоговая аттестация																								2ч

3. Календарный учебный график.

Форма обучения: очная.
Общая продолжительность занятий: 270 ак. часов, 6 месяцев. Не более 14 часов в неделю. Занятия проходят 2-3 раза в неделю.

Сокращения:
Ч- часы

4.УЧЕБНЫЙ ПЛАН

дополнительной профессиональной программы профессиональной переподготовки
«Python-разработчик (Django, DRF)»

Срок обучения – 270 ак. часов.

Форма обучения – очная.

Требования к уровню образования: для лиц, имеющих среднее профессиональное и (или) высшее образование; а также лиц, получающих среднее профессиональное и (или) высшее образование.

№ п/п	**Наименование предметов/тем/глав**	**Всего ак.часов**	**Из них**		
			Теоретич. Занятия (ак.час)	**Тестирование, практические занятия (ак.час)**	**Контроль знаний**
1	Глава 1. Введение в курс	1	1	**-**	**-**
2	Глава 2. Основы Python	53	37	**16**	**-**
3	Тема 2.1 Знакомство с Python	3	3	-	**-**
4	Тема 2.2 Интерпретатор vs компилятор	3	3	-	**-**
5	Тема 2.3 Установка интерпретатора Python	3	3	-	**-**
6	Тема 2.4 Командная строка	3	2	1	**Тестирование**
7	Тема 2.5 Git и GitHub	3	2	1	**Тестирование, задание**
8	Тема 2.6 Настройка окружения	3	2	1	**Тестирование, задание**
9	Тема 2.7 IDE	3	2	-	**Задание**
10	Тема 2.8 Коллекции в Python	3	2	1	**Тестирование, практика**
11	Тема 2.9 Словари	3	2	1	**Тестирование**
12	Тема 2.10 Множества	3	2	1	**Тестирование**
13	Тема 2.11 Строки и форматирование	3	2	1	**Тестирование, задание**

14	Тема 2.12 Библиотеки и модули	3	2	1	**Тестирование, задание**
15	Тема 2.13 Ветвления (if, elif, else)	3	2	1	**Тестирование, задание**
16	Тема 2.14 Циклы	2	1	1	**Тестирование, задание**
17	Тема 2.15 Функции	2	1	1	**Тестирование, задание**
18	Тема 2.16 Типы данных в Python	2	1	1	**Тестирование, задание**
19	Тема 2.17 Переменные и память	2	1	1	**Тестирование, задание**
20	Тема 2.18 Типизация	2	1	1	**Тестирование, задание**
21	Тема 2.19 Классы. ООП	2	1	1	**Тестирование, задание**
22	Тема 2.20 Три кита ООП	2	1	1	**Тестирование, проектная работа**
23	Глава 3. Бэкинг на Django	54	45	9	**-**
24	Тема 3.1 Знакомство с Django	3	2	1	**Тестирование**
25	Тема 3.2 Создание Django-проекта	3	2	1	**Тестирование, задание**
26	Тема 3.3 Пути и view-функции	3	2	1	**Тестирование, задание**
27	Тема 3.4 HTML и шаблоны Django	3	2	1	**Тестирование, задание**
28	Тема 3.5 Введение в базы данных	3	2	1	**Тестирование**
29	Тема 3.6 SQL: INSERT, UPDATE, SELECT, DELETE	3	2	1	**Тестирование**
30	Тема 3.7 SQL: Сортировка, ограничения и сдвиг выборки (LIMIT, OFFSET)	3	2	1	**Тестирование, задание**
31	Тема 3.8 SQL: Группировка записей в выборке	3	2	1	**Тестирование, задание**
32	Тема 3.9 Работа с базой из Python	3	3	-	**-**
33	Тема 3.10 Программы для работы с базами данных	3	3	-	**-**
34	Тема 3.11 Отношения между таблицами	3	3	-	**-**
35	Тема 3.12 Django ORM	3	2	1	**Тестирование**
36	Тема 3.13 Админка Django	3	3	-	**-**
37	Тема 3.14 Получение данных из бд через Django ORM	3	3	-	**-**
38	Тема 3.15 Запрос к связанным моделям	3	3	**-**	**-**
39	Тема 3.16 Django-формы	3	3	**-**	**-**
40	Тема 3.17 Защита формы от атак. CSRF-токен	2	2	**-**	**-**

41	Тема 3.18 Работа с изображениями в формах	2	2	-	-
42	Тема 3.19 Тестирование	2	2	-	-
43	Глава 4. API	54	53	1	-
44	Тема 4.1 Что такое API	4	4	-	-
45	Тема 4.2 Работа с внешними API	5	5	-	-
46	Тема 4.3 Клиент-API Telegram	4	4	-	-
47	Тема 4.4 Создание телеграм бота	5	4	1	Задание
48	Тема 4.5 API TMDB	4	4	-	Задание
49	Тема 4.6 Django Rest Framework	4	4	-	-
50	Тема 4.7 Сериализаторы для связанных моделей	4	4	-	-
51	Тема 4.8 Права доступа - Permissions	4	4	-	-
52	Тема 4.9 Ограничение количества запросов Throttling	4	4	-	-
53	Тема 4.10 Пагинация в DRF	4	4	-	-
54	Тема 4.11 Фильтрация, сортировка, поиск	4	4	-	-
55	Тема 4.12 Документация API. Swagger	4	4	-	-
56	Тема 4.13 CORS политика	4	4	-	-
57	Глава 5. Инфраструктура	54	52	2	-
58	Тема 5.1 DevOps. Работа с сервером	5	3	2	Тестирование
59	Тема 5.2 Подключение к удаленной машине	5	5	-	-
60	Тема 5.3 Первый деплой	5	5	-	-
61	Тема 5.4 Веб-прокси. NGINX	5	5	-	-
62	Тема 5.5 Доменное имя	5	5	-	-
63	Тема 5.6 Шифрование. HTTPS	5	5	-	-
64	Тема 5.7 Мониторинг и сбор ошибок	6	5	-	
65	Тема 5.8 Виртуализация. Контейнеризация	6	5	-	-
66	Тема 5.9 Docker	6	5	-	-
67	Тема 5.10 Запуск приложения в docker-контейнере	6	5	-	Тестирование
68	Глава 6. Дипломная работа	52	-	52	-
69	Итоговая аттестация	2	-	2	Защита дипломной работы
70	Итого	270	189	81	-

5. РАБОЧАЯ ПРОГРАММА

Глава 1. Введение в курс

Деятельность \ проекты. Достижения. Команда. Перспективы\планы.
Зачем создан этот курс. Как построена работа. Кому подойдет курс. Какие знания будут получены.

Глава 2. Основы Python

Тема 2.1 Знакомство с Python

Описания языка. История появления. Основные характеристики Python. Пример простейшей программы на Python.

Тема 2.2 Интерпретатор vs компилятор

Компилятор и интерпретатор. Программный код. Типы интерпретаторов.

Тема 2.3 Установка интерпретатора Python

Инструкция по установке интерпретаторов Python. Способы установки интерпретаторов через утилиту PyEnv.

Тема 2.4 Командная строка

Командная строка. Командные оболочки Bash и Zsh. Запуск терминала. Навигация.
Тестирование:

1) Какая команда используется для вывода ключей команды:

1 -help
2. ---man
3. ---help
4. man ---help

2) Какая команда используется для сортировки директорий и файлов по времени последнего доступа:

1. ls -t
2. ls -u
3. ls –X
4. ls- –access

3) Какая команда выведет все файлы, включая скрытые и отсортирует их по дате изменения:

1. ls -alt

2. ls -la
3. ls -a
4. ls -ll

4) Вы находитесь в директории /home/Users/newUser. Как посмотреть содержимое директории /Users/newUser/pictures/photo?
1. cd /home/Users/newUser/pictures/photo
Ls.
2. cd /home/Users/newUser/pictures/photo
pwd
3.• cd ..
cd newUser/pictures/photo
ls
4.cd pictures/photo
pwd

Тема 2.5 Git и GitHub

Введение. Система версионирования Git. Ключевые аспекты Git и его функциональность. Версионирование. Работа с несколькими разработчиками. Работа с ветками (Branching). Резервное копирование и восстановление. Отслеживание изменений. Работа в офлайн-режиме. GitHub и GitLab. Виды использования. Модели лицензирования. Интеграция с CI/CD. 4 Функциональные возможности. Функции для командной работы.
Установка **Git** Windows. Linux. MacOS.
Первоначальная настройка Git. Настройка имени пользователя. Выбор редактора. Настройка ветки по умолчанию. Проверка настроек.
Создание репозитория на GitHub в GitLab. Файл Readme.
Тестирование:
1. Вопрос: Как создать новый Git репозиторий?
- a) `git init`
- b) `git create`
- c) `git new`
- d) `git start`
2. Вопрос: Как добавить все измененные файлы в индекс (staging area) в Git?
- a) `git commit -a`
- b) `git add .`
- c) `git stage -A`
- d) `git update`
3. Вопрос: Какой командой можно проверить статус изменений в рабочей директории Git?
- a) `git log`
- b) `git status`
- c) `git changes`
- d) `git check`
4. Вопрос: Каким образом можно клонировать (скопировать) удаленный Git репозиторий на локальную машину?
- a) `git pull`
- b) `git copy`
- c) `git clone`
- d) `git download`
Практическое задание:
1.Зарегистрируйтесь на Gitlab
2.Создайте новый проект

3.Клонируйте его локально (git clone)
4.Добавьте скприт на питон, откомментируйте измения и отправьте на сервер

Тема 2.6 Настройка окружения

Виртуальное окружение. Venv (виртуальное окружение). Изоляция. Управление зависимостями. Совместимость версий. Чистая среда разработки. Легкость репликации.
Тестирование:
1. Вопрос: Какая команда используется для создания виртуального окружения в Python?
a. `create venv`
b. `python venv create`
c. `venv init`
d. `python -m venv venv`
2. Вопрос: Как активировать виртуальное окружение в операционной системе Windows?
a.`source venv/bin/activate`
b.`activate venv`
c.`venv/activate`
d.`venv\Scripts\activate`
3. Вопрос: Как установить зависимости из файла `requirements.txt` в виртуальном окружении?
a.`pip install requirements.txt`
b.`pip install -r requirements.txt`
c.`install -r requirements.txt`
d.`python install requirements.txt`
4. Вопрос: Как деактивировать виртуальное окружение в любой операционной системе?
a.`deactivate`
b.`end venv`
c.`exit venv`
d.`stop venv`
Практическое задание:
1.Установите и активируйте виртуальное окружение
2.Установите библиотеку https://pypi.org/project/tmdbv3api/ версии 1.9.0
3.Создайте файл requirements.txt, в которых поместите список установленных библиотек
4.Деактивируйте виртуальное окружение.

Тема 2.7 IDE

IDE, или интегрированная среда разработки (Integrated Development Environment). Основные компоненты IDE. Текстовый редактор. Отладчик. Система управления версиями. Средства анализа кода и профилирования.
Установка и настройка.
Практическое задание:
1.Установите Pycharm.
2.Создайте новый проект.
3.Убедитесь что проект запустился в виртуальном окружении.

Тема 2.8 Коллекции в Python

Задачи программирования. Встроенные типы коллекций.
Списки Python. Чем полезны. Важные особенности списка. Способы объявления списков. Методы работы со списками.

Тестирование:
1) Как создать пустой список в Python?
Варианты ответов:
1. empty_list = {}
2. empty_list = []
3. empty_list = new_list()
4. empty_list = ""

2) Как получить длину (количество элементов) списка?
Варианты ответов:
1. list.length()
2. len(list)
3. list.size()
4. list.len()

3) Какой индекс у первого элемента списка?
Варианты ответов:
1. 0
2. 1
3. -1
4. Нет индекса
5. Зависит от способа объявления

4) Какой метод используется для добавления элемента "item" в конец списка "my_list"?
Варианты ответов:
1. my_list.add(item)
2. my_list.insert(item)
3. my_list.append(item)
4. my_list.extend(item)

5) Как удалить элемент "value" из списка "my_list"?
Варианты ответов:
1. my_list.remove(value)
2. my_list.delete(value)
3. my_list.pop(value)
4. my_list.drop(value)
5. my_list.del(value)

Практика:
1.Создайте список фильмов, с названием films, содержащий элементы в именно в таком порядке: “Аватар”, “Человек-паук”, “Дюна”, “Титаник”. После того, как создадите список films, выведите его содержимое в консоль.
2.Пора посмотреть фильм “Дюна”. Допишите код так, чтобы ваша программа напечатала фразу “Сегодня вечером посмотрю фильм Дюна”, при этом название фильма нужно получить из списка выше, взяв его по индексу.
3.Добавим немного функциональности к просмотру фильмов. Научимся считать фильмы.
●Объявите переменную count и сохраните в ней количество фильмов, воспользовавшись специальной функцией
●Выведите на экран строку “У тебя {количество} фильмов в избранных”, где “{количество}” – значение переменной count. Слова “У тебя” и “фильмов в избранных” в коде должны быть в кавычках, это строки!

Тема 2.9 Словари

Словари в Python. Ключи в словаре. Основные операции со словарями. Преимущества словарей.

Тестирование:

1) Что такое словарь в Python 3.7 или старше?
1. Упорядоченная коллекция элементов типа ключ-значение.
2. Неупорядоченная коллекция пар ключ-значение.
3. Структура данных для хранения только чисел.
4. Специальный тип строкового значения.

2) Какой из следующих типов данных может быть использован в качестве ключа в словаре?
1. Список
2. Целое число (integer)
3. Множество (set)
4. Функция

3) Как можно получить значение из словаря по ключу?
1. С помощью индексации, как в списках.
2. С помощью оператора . и имени ключа.
3. Используя функцию get() и передавая ключ в качестве аргумента.
4. Значения из словаря получить невозможно, можно только добавлять ключи.

4) Что произойдет, если вы попытаетесь добавить элемент с уже существующим ключом в словарь?
1. Будет создан новый ключ с таким же именем.
2. Старое значение для этого ключа будет заменено новым.
3. Python выдаст ошибку и завершит выполнение программы.
4. Словарь автоматически увеличится в размере.
5) Как удалить элемент из словаря по ключу?
1. Используя функцию remove() и указав ключ.
2. С помощью оператора delete и имени ключа.
3. С помощью функции pop() и передав ключ в качестве аргумента.
4. Элементы в словаре нельзя удалять, только перезаписывать.

Тема 2.10 Множества

Множество (set, сеты). Создание множества. Удаление элементов из множества. Проверка наличия элемента в множестве. Операции объединения, пересечения и разности множеств. Подсчет количества элементов в множестве. Очистка множества. Копирование множества. Подмножества и надмножества.

Тестирование:

1.Что характеризует множество (set) в Python?
a) Элементы могут дублироваться.
b) Элементы хранятся в порядке добавления.
c) Элементы могут быть уникальными и не имеют определенного порядка.
d) Элементы автоматически сортируются по алфавиту.

2. Какие операции доступны для множества (set) в Python?
a) Только объединение.
b) Только пересечение.

c) Только разность.
d) Объединение, пересечение, разность и дополнение.

3.Что произойдет, если вы попытаетесь добавить дублирующийся
элемент в множество?
a) Множество выбросит исключение.
b) Дублирующийся элемент будет добавлен в множество.
c) Дублирующийся элемент не будет добавлен в множество.
d) Множество станет неизменяемым.

4.Какой тип данных должны иметь элементы множества (set) в Python?
a) Только числа.
b) Любые хешируемые (immutable, неизменяемые) объекты, такие как числа,
строки, кортежи.
c) Только изменяемые объекты.
d) Только списки.

Тема 2.11 Строки и форматирование

Строки (strings) в Python. Конкатенация строк. Доступ к символам и подстрокам. Методы строк. Форматирование строк.

Тестирование:
1. Какой метод строки используется для преобразования всей строки в
верхний регистр?
1) .upper()
2) .capitalize()
3) .lower()
4) .title()
5) .casefold()

2. Какой оператор используется для конкатенации (соединения) двух
строк в Python?
1) *
2) -
3) /
4) +
5) &

3. Как можно создать многострочную строку в Python?
1) multiline()
2) '"многострока"'
3) "многострока"
4) 'многострока'

Практическое задание:
Вы получили небольшой набор данных о фильмах в виде текстовых строк.
Ваша задача - написать программу на Python, которая обрабатывает эти строки, извлекая различную информацию о фильмах.
На вход подаются одна строка в которой содержится информация о фильмах разделенных знаком запятой. В строке содержится название, год выпуска в скобках, режиссер и жанр.
Пример строки:
Властелин колец (2003) Майкл Бэй, фэнтези

Выведите на экран год самого старого и самого нового фильма через запятую, если информации недостаточно, введите пустую строку. А также в новой строке список всех режиссеров через запятую.

Тема 2.12 Библиотеки и модули

Модули в Python. Полный импорт всего модуля import module и выборочный импорт.
Импорт и использование библиотек.
Тестирование:
1. Какой модуль Python используется для работы с операционной системой, включая доступ к файловой системе и выполнение команд в терминале?
A) os
B) math
C) requests
D) datetime
2. Какой модуль Python предоставляет функции для генерации случайных чисел и выборок?
A) random
B) sys
C) argparse
D) json
3. Для чего используется модуль requests в Python?
A) Для выполнения HTTP-запросов.
B) Для работы с регулярными выражениями.
C) Для сериализации и десериализации данных в формате JSON.
D) Для создания статических графиков.

4. Какой командой подключить внешний модуль в Python?
A) module_import
B) input
C) library
D) lib
E) import
5. Для чего предназначен модуль numpy?
A) Для парсинга HTML и XML документов.
B) Для работы с операционной системой.
C) Для создания статических, анимированных и интерактивных графиков.
D) Для поддержки многомерных массивов и математических операций над ними.

Практическое задание:
Задание:
У нас есть стакан, который представляет из себя цилиндр, с высотой H и радиусом R. Напишите программу на Python, которая запрашивает у пользователя ввод радиуса и высоту и выводит на экран его объем. Для выполнения расчетов используйте функции из модуля math. Значение Пи так же получите из этого модуля. Полученный результат округлите до 1 знака после запятой.
Например:
H=10
R=5
V=785,4

Тема 2.13 Ветвления (if, elif, else)

Ветвления. Условные операторы.
Тестирование:
1. Какой результат будет выведен после выполнения следующего кода?
1) x = 15
2) if x > 5:
3) if x < 10:
print("Меньше 10")
4) elif x < 20:
print("Меньше 20")
5) else:
print("20 или больше")
Варианты ответов:
1. "Меньше 10"
2. "Меньше 20"
3. "20 или больше"
4. "Меньше 10" и "Меньше 20"
5. "Меньше 20" и "20 или больше"

2) Какой результат будет выведен после выполнения следующего кода?
1. x = 7
2. if x > 5:
3. if x < 10:
print("Между 5 и 10")
4. else:
print("Больше или равно 10")
5.else:
print("Меньше или равно 5")
Варианты ответов:
1. Между 5 и 10"
2. "Больше или равно 10"
3. "Меньше или равно 5"
4. "Между 5 и 10" и "Больше или равно 10"
5. "Между 5 и 10" и "Меньше или равно 5"

3): Какой результат будет выведен после выполнения следующего кода?
1. x = 25
2. y = 12
3. if x > 20 and y > 10:
print("Оба условия выполняются")
4. elif x > 20 or y > 10:
print("Хотя бы одно из условий выполняется")
5. else:
print("Ни одно из условий не выполняется")
Варианты ответов:
1. "Оба условия выполняются"
2. "Хотя бы одно из условий выполняется"
3 "Ни одно из условий не выполняется"
4 "Оба условия выполняются" и "Хотя бы одно из условий выполняется"
5 "Хотя бы одно из условий выполняется" и "Ни одно из условий не выполняется".

Практическое задание:

Задание: Калькулятор рейтинга фильма
Нам предстоит написать программу-калькулятор оценок фильма. Пользователь выставляет свои оценки фильму по нескольким параметрам, который суммируются между собой, в результате получается число от 0 до 100.
Программа должна запрашивать у пользователя его баллы и выводить соответствующую оценку согласно следующей шкале:
●90 и выше баллов: Оценка 5
●80 - 89 баллов: Оценка 4
●70 - 79 баллов: Оценка 3
●60 - 69 баллов: Оценка 2
●Ниже 60 баллов: Оценка 1
Инструкции:
Попросите пользователя ввести свои баллы с клавиатуры.
Используйте условные операторы (if, elif, else) для определения оценки в зависимости от введенных баллов.
Выведите на экран оценку, соответствующую введенным баллам.
Пример вывода программы:
Введите количество баллов: 75
Ваша оценка: 3
Введите количество баллов: 92
Ваша оценка: 5

Тема 2.14 Циклы

Типы циклов: Цикл for, Цикл while. Прерывание и продолжение циклов. Оператор else в циклах for и while. Цикл "for" с оператором else. Цикл "while" с оператором else.
Тестирование:
1. Какой тип цикла используется для итерации по последовательности (например, списку) в Python?
а) loop
б) for
в) repeat
г) iterate
2. Какой оператор используется для прерывания выполнения цикла в Python?
а) end
б) finish
в) break
г) stop
3. В каком случае блок кода после оператора else в цикле for выполнится?
а) Всегда, независимо от того, был ли цикл прерван или завершился нормально.
б) Только если цикл завершился нормально (без прерывания).
в) Только если цикл был прерван с помощью оператора break.
г) Блок кода после else в цикле for не существует.
4. Какой оператор используется для перехода к следующей итерации цикла, пропуская текущую итерацию?
а) pass
б) continue
в) next
г) skip
5. Какой тип цикла используется для повторения блока кода, пока условие остается истинным?
а) repeat

б) while
в) iterate
г) loop

Практическое задание:
Задание на for:
Напишите программу, которая принимает на вход строку s (предложение) от пользователя и затем выводит на экран, через запятую, все слова из этой строки, длина которых больше 3 символов. Задачу необходимо выполнить с помощью цикла for, слова могут отделять друг от друга любые печатные символы, кроме цифр. Слова надо всегда выводить в нижнем регистре (маленькими буквами).
Пример:
s = “Пайтон, это отличный и гибкий язык программирования! С его помощью
можно решать разные задачи (например по работе со строками) ”
Вывод:
пайтон,отличный,гибкий,язык,программирования,помощью,можно,решать,разн
ые,задачи,например,работе,строками
Задание на while:
Задание: Напишите программу, которая будет считать сумму цифр введенных пользователем целых положительных чисел. Программа должна продолжать запрашивать ввод числа, пока не будет введено отрицательное число. После ввода отрицательного числа, программа должна вывести сумму всех цифр введенных чисел. Обратите внимание, пользователь может вводить несколько чисел. Последнее введенное отрицательное число не считается.
Пример:
123
Вывод:
6
Пример:
123
44
583
12
-1
Вывод:
33
Пример:
-5
Вывод:
-

Тема 2.15 Функции

Функции в Python. Аргументы функции. Возвращаемое значение. Вызов функции. Аргументы по умолчанию. Передача аргументов по имени. Произвольное количество аргументов. Лямбда функции.
Тестирование:
1. Что такое функция в Python?
1) Отдельный файл с кодом, который можно импортировать в другие
программы.
2) Метод для определения переменных в Python.
3) Блок кода, который выполняет определенную задачу, может принимать
аргументы и возвращать результат.
4) Оператор для объявления классов в Python.
2. Для чего используется ключевое слово return в функциях Python?

1) Для вывода текста на экран.
2) Для обработки исключений в функциях.
3) Для завершения выполнения функции и возврата результата.
4) Для объявления новой переменной внутри функции.
3. Что такое аргументы функции?
1) Это переменные, которые объявляются внутри функции.
2) Это специальные ключевые слова, используемые для определения функций.
3) Это значения, передаваемые функции при ее вызове для обработки.
4) Это имена, которые можно использовать для вызова функции.
4. Какие преимущества предоставляют функции в
программировании?
1) Функции не позволяют делать вычисления.
2) Функции делают код менее читаемым.
3) Функции увеличивают сложность программы.
4) Функции улучшают модульность, повторное использование кода и
организацию программы.
5. Что такое лямбда-функции в Python?
1) Функции, которые работают только с целыми числами.
2) Функции, которые могут быть вызваны только один раз.
3) Анонимные функции, которые могут содержать только одно выражение.
4) Функции, которые автоматически выполняются при запуске программы.

Практические задания:
Вы работаете над сервисом в онлайн-кинотеатре, который считает цены на покупку фильмов пользователем при различных условиях. Вам необходимо создать функцию calc_film_price, которая будет принимать на вход цену фильма и процент скидки, а затем вычислять и возвращать цену с учетом этой скидки. Цена должна округляться до целого числа по математическим правилам.
Также, вы должны предусмотреть случаи, когда переданная скидка больше 50%, в таком случае, скидка автоматически устанавливается на 50%. Обратите внимание, что в нашу функцию, по ошибке может прийти отрицательная цена или скидка, а так-же случайно скидка может быть назначена больше 100%.

Тема 2.16 Типы данных в Python

Переменная в языке программирования. Числовые типы данных. Целые числа (int). Числа с плавающей точкой (float). Комплексные числа (complex). Строки. Списки. Кортежи. Множества. Словари.
Тестирование:
1. Какой из следующих типов данных в Python используется для
представления целых чисел?
а) float
б) int
в) str
г) bool
2. Какой тип данных в Python предназначен для хранения десятичных чисел
с плавающей точкой?
а) str
б) float
в) int
г) list
3. Какой из перечисленных типов данных используется для представления
последовательности символов?

а) bool
б) int
в) str
г) dict
4. Какой тип данных в Python позволяет хранить упорядоченные и изменяемые коллекции элементов?
а) tuple
б) set
в) list
г) str
5. Какой из нижеперечисленных типов данных используется для хранения двоичных значений "Истина" или "Ложь"?
а) dict
б) int
в) bool
г) float

Практическое задание:
Калькулятор стоимости фильмов
Введение: для онлайн-кинотеатра нужно написать программу которая будет считать стоимость арендованных фильмов. Каждый пользователь добавляет фильмы себе в корзину и указывает, на какое количество дней он хочет арендовать фильм.
Вам нужно написать программу, которая запрашивает у пользователя информацию о каждом фильме (артикул, цена и количество дней аренды) и затем вычисляет общую стоимость. Данные о аренде получаются в виде строки 1:2:3,2:2:3,3:2:3
Пример входных данных: 1:2:3, 3:2:1
В качестве выходные данные должна быть итоговая стоимость аренды

Тема 2.17 Переменные и память

Память. Краткая теория. Как это работает.
Тестирование:
1. Когда мы говорим “управление памятью в Python”, какой тип памяти мы имеем ввиду?
A) оперативная память
B) операционная память
C) твердотельные накопители
D) все перечисленные ответы
2. Что такое "циклический сборщик мусора" в Python?
A) Механизм, который автоматически удаляет все циклы из программы.
B) Модуль для создания циклических структур данных в Python.
C) Процесс, автоматически освобождает память, занятую неиспользуемыми объектами.
D) Тип данных, используемый для обхода циклов в программах.
3. Что такое "круговая ссылка" в контексте управления памятью в Python?
A) Ссылка на функцию внутри класса.
B) Ссылка на объект, созданный с помощью ключевого слова circle.
C) Ссылка между двумя объектами, что приводит к утечке памяти, так как они ссылаются друг на друга.
D) Ссылка на документацию Python, связанную с управлением памятью.
4. Что такое "ссылочный счетчик" (reference count) в контексте управления памятью в Python?
A) Число ссылок, указывающих на определенный объект в памяти.

B) Счетчик обращений к документации Python.
C) Количество файлов, открытых в текущем сеансе Python.
D) Число переменных, определенных в глобальной области видимости.

Переменные. Стандартные типы данных в Python. Объявление и присваивание переменных. Моржовый оператор. Названия переменных в Python.
Тестирование:
Тесты
Вопрос: Какое имя переменной не допустимо в Python?
a) 123variable
b) _my_variable
c) variable$1
d) string
Вопрос: Какой тип данных используется для хранения целых чисел?
a) int
b) float
c) str
d) bool
Вопрос: Как изменить значение переменной x на 5?
a) x = 5
b) 5 = x
c) x := 5
d) x <-- 5
e) a и c
Вопрос: Как удалить переменную my_var из памяти?
a) delete my_var
b) remove(my_var)
c) del my_var
d) my_var.delete()
Вопрос: Как получить длину строки, хранящейся в переменной text?
a) text.length()
b) length(text)
c) len(text)
d) text.size()

Практическое задание:
Задача:
Создайте переменные для хранения следующей информации:
●Имя пользователя
●Возраст пользователя
●Город, в котором живет пользователь
Выведите на экран информацию о пользователе, используя созданные
переменные.
Например
Имя пользователя: John
Возраст пользователя: 30
Город пользователя: New York
Измените значение переменной "Возраст пользователя" на новое значение.
Вычислите сумму возраста пользователя и некоторой другой переменной (например, 10) и сохраните результат в новую переменную.
Выведите на экран новое значение переменной из задачи 4, предварительно преобразовав его в строку (строковое представление).

Создайте новую переменную, которая будет хранить информацию о длине имени пользователя (количество символов в имени).

Тема 2.18 Типизация

Динамическая типизация. Строгая типизация. Зачем нужна динамическая типизация. Зачем нужна строгая типизация.

Тестирование:

1. Какая система типизации характеризует язык программирования Python?
a) Статическая типизация
b) Динамическая типизация
c) Слабая типизация
d) Сильная типизация
2. Какой оператор используется для проверки, является ли объект экземпляром определенного класса или его подкласса в Python?
a) isclass()
b) isinstance()
c) istype()
d) isobject()
3. Что делает функция type() в Python?
a) Возвращает тип объекта
b) Проверяет, является ли объект экземпляром класса
c) Устанавливает новый тип объекта
d) Определяет тип переменной
4. Какие аннотации используются для указания ожидаемых типов аргументов и возвращаемых значений функций в Python 3.5 и более поздних версиях?
a) @args и @returns
b) :param и :return
c) -> и ::
d) @param и @return

5. Какой инструмент позволяет проводить статическую проверку типов в Python и основан на аннотациях типов?
a) PyTest
b) PyType
c) MyPy
d) PyCheck

Практическое задание:

Задание: Создание функции для расчета среднего значения.

Цель этого задания - создать функцию для вычисления среднего значения списка чисел, в то время как использование аннотаций типов и проверки типов поможет нам обеспечить корректную работу функции.

1.Создайте новый файл Python с именем average_calculator.py.
2.Создайте функцию calculate_average, которая принимает в качестве аргумента список чисел и возвращает их среднее значение. Используйте аннотации типов для указания ожидаемого типа аргумента и возвращаемого значения.
3.Обеспечьте проверку типов внутри функции. Убедитесь, что переданный аргумент является списком чисел. Если тип аргумента не соответствует ожидаемому, выведите сообщение об ошибке и верните None.
4.Реализуйте вычисление среднего значения чисел в списке и верните результат.
5.Создайте второй файл Python с именем main.py.

6.В main.py создайте список чисел и вызовите функцию calculate_average из average_calculator.py, передав этому списку чисел в качестве аргумента.
7.Проверьте результат выполнения функции и выведите среднее значение на экран.
8.Запустите main.py и убедитесь, что функция правильно рассчитывает
среднее значение и обрабатывает неправильный тип аргумента.

Тема 2.19 Классы. ООП

Классы в объектно-ориентированном программировании (ООП) Python. Что такое класс. Зачем нужны классы. Создание класса. Атрибуты класса. Методы класса. Создание объектов. Доступ к атрибутам и методам объекта. Методы new и init.
Тестирование:
1. Что такое класс в Python?
a) Это модуль Python.
b) Это объект, созданный из другого класса.
c) Это пользовательский тип данных, определяющий атрибуты и методы.
d) Это функция для создания объектов.

2. Как объявляется класс в Python?
a) def MyClass:
b) set MyClass():
c) class MyClass:
d) class():

3. Что такое метод класса в Python?

a) Это переменная, хранящая данные объекта.
b) Это функция, определенная внутри класса, которая выполняет операции с данными объекта.
c) Это встроенная функция Python для создания объектов.
d) Это ключевое слово для наследования классов.

4.Как создать объект на основе класса в Python?

a) new_object = class_name();
b) object = new class_name();
c) new_object = Class(class_name);
d) object = class_name();

Практическое задание:
Задание: Управление банковским счетом
Создайте класс BankAccount для управления банковским счетом. В этой задаче мы не указываем какуют то конкретную валюту. Банковский счет должен иметь следующие атрибуты и методы:
Атрибуты класса:
account_number (уникальный номер счета, число из 5 цифр)
account_holder (имя владельца счета, строка длиной до 100 символов)
balance (баланс счета)
Методы класса:
__init__(self, account_number, account_holder, initial_balance):
Конструктор класса, который инициализирует атрибуты account_number, account_holder и balance при создании экземпляра класса.

Обратите внимание, что initial_balance - это начальный баланс счета. Имя владельца счета может состоять только из букв и пробелов. Никакие другие знаки не разрешаются.
deposit(self, amount): Метод для внесения денег на счет. Метод должен увеличивать баланс на указанную сумму amount. Сумма может быть дробным числом.
withdraw(self, amount): Метод для снятия денег со счета. Метод должен проверить, что на счете достаточно средств для снятия указанной суммы amount.
Если средств недостаточно, выведите сообщение об ошибке. В противном случае снимите указанную сумму и обновите баланс.
get_balance(self): Метод, который возвращает текущий баланс на счете.
display_account_info(self): Метод, который выводит информацию о счете, включая номер счета, имя владельца и текущий баланс.

Тема 2.20 Три кита ООП

Три основных принципа объектно-ориентированного программирования (ООП). Инкапсуляция (Encapsulation). Наследование (Inheritance). Полиморфизм (Polymorphism). Mro и множественное наследование.
Тестирование:
1. Какой модификатор доступа используется для атрибутов или методов, которые доступны только внутри класса и его подклассов?
a) public
b) protected
c) private
d) global

2. Какой принцип ООП позволяет создавать новые классы на основе существующих классов?
a) Инкапсуляция
b) Полиморфизм
*c) Наследование
d) Абстракция

3. Какая функция используется для создания экземпляра класса в Python?
a) create()
b) new()
c) instance()
d) init()

4. Что такое полиморфизм в контексте ООП в Python?
a) Возможность скрыть детали реализации класса
b) Возможность создать новый класс на основе существующего
c) Возможность объектов разных классов иметь общий интерфейс и различное поведение
d) Возможность ограничить доступ к атрибутам и методам класса

Проектная финальная работа спринта
Задача: Онлайн кинотеатр
Вы разрабатываете систему для онлайн-кинотеатра. Вам необходимо создать классы, которые описывают основные компоненты этой системы.
Вам нужно реализовать классы для фильмов и пользователей:
1.Класс "Фильм" (Film):
У каждого фильма есть: название, жанр, продолжительность (целое число минут, по умолчанию 90), рейтинг (целое число от 0 до 10), возрастной рейтинг (целое число от 0 до 18)
Методы:
Конструктор `__init__`,

Метод «get_info».
2.Класс "Зритель" (User):
- Атрибуты: имя, возраст.
- Методы: конструктор __init__,
метод для покупки билета на сеанс buy_film,
метод для просмотра информации о фильме movie_info.
Важно учесть следующее:
- Зритель не может посмотреть фильм, если его возраст меньше рейтинга фильма.
- Зритель должен иметь возможность купить билет на сеанс, если фильм доступен, и у него есть достаточно средств.
- Фильмы имеют разные жанры (например, "комедия", "драма", "фантастика"), и зритель может быть заинтересован в определенных жанрах.
- Зритель может посмотреть информацию о фильме, а также узнать расписание сеансов.
Задача:
1. Реализовать описанные классы с учетом принципов ООП (инкапсуляция, наследование, абстракция, полиморфизм).
2. Создать несколько объектов классов "Фильм" и "Зритель".
3. Продемонстрировать взаимодействие объектов, включая покупку фильмов, просмотр фильмов и получение информации о них.

Глава 3. Бэкинг на Django

Тема 3.1 Знакомство с Django

Django. Применение. Версии Django. Основные особенности Django.
Тестирование:
1. Что представляет собой архитектурный шаблон Model-View-Controller (MVC) в Django?
a) Модель, Вид и Контроллер - это разные URL-адреса в Django.
b) MVC - это фреймворк для разработки мобильных приложений.
c) Модель представляет данные, Вид отвечает за бизнес-логику, а Контроллер управляет URL-ами и запросами.
d) Модель, Вид и Контроллер - это разные части HTML-страницы.

2. Каким образом Django обеспечивает безопасность от атаки CSRF (межсайтовой подделки запросов)?
a) Django автоматически отключает все запросы от других доменов.
b) Django добавляет уникальные токены CSRF к формам и проверяет их при отправке запросов.
c) Django просит пользователя ввести капчу перед отправкой каждого запроса.
d) Django требует, чтобы пользователи были аутентифицированы, чтобы предотвратить атаки CSRF.

3. Какой компонент Django отвечает за взаимодействие с базой данных?
a) Вид (View)
b) Контроллер (Controller)
c) Модель (Model) и ORM
d) Шаблон (Template)

4. Какие из следующих функциональных возможностей предоставляют административные интерфейсы в Django?
a) Создание и управление пользователями и их ролями.
b) Редактирование HTML-кода в реальном времени.
c) Управление настройками сервера хостинга.
d) Добавление сторонних библиотек к проекту Django.

Тема 3.2 Создание Django-проекта

Структура проекта Django. Настройки проекта (settings). Маршруты (urls). База данных (database). Статические файлы (static) и Медиа-файлы (media). Шаблоны (templates). Приложения (apps).

Тестирование:

1.Что такое миграции в Django?

a) Фреймворк Django работает без миграций
b) Это способ перемещения файлов
c) Это файлы для изменения базы данных
d) Это путь для перемещения проектов

2. Какая актуальная версия фрэймворка Django:

a) 1
b) 5
c) 4
d) 2

3.С помощью какой команды можно запустить сервер для разработки?

a) manage.py runpython
b) manage.py runserver
c) manage.py startserver
d) manage.py startweb

Практическое задание

1.Запустите виртуальное окружение
2.Установите Django
3.Создайте новый проект
4.Создайте примените миграции
5.Запустите тестовый сервер

Тема 3.3 Пути и view-функции

Пути (URL Patterns) в Django. View-функции. Именованные пути (Named URLs). Передача параметров в view-функции. Рендеринг шаблонов.

Тестирование:

1. Какие файлы в Django используются для определения путей и связи URL-адресов с view-функциями?

1. settings.py
2. models.py
3. urls.py
4. views.py

2. Как определить именованный путь в Django?

1. С помощью атрибута path в модели данных.
2. С помощью декоратора @named_url над view-функцией.
3. С помощью аргумента name в определении пути.
4. Именованные пути не поддерживаются в Django.

3. Какой модуль в Django используется для рендеринга HTML-шаблонов?

1. django.views.render
2. django.template

3. django.shortcuts
4. django.views.html

Практическое задание
1. Создайте вью, которая будет возвращать строку “Hello world!” и выводить его в браузере, в качестве респонса можно использовать HttpResponse
2. Добавьте роут для обработки вью, страница должна быть доступна по адресу http://localhost:8000/hello-world/

Тема 3.4 HTML и шаблоны Django

Шаблоны Django. Создание шаблона. Вставка данных. Управление логикой. Расширение шаблонов. Дочерний шаблон. Статические файлы.
Тестирование:
1.Вопрос: Какой символ используется в Django-шаблонах для вывода переменной на веб-странице?
a) {{ }}
b) []
c) ()
d) << >>

2.Вопрос: Какой атрибут используется для задания ссылки на внешний CSS-файл в HTML?
a) style
b) script
c) css
d) link

3.Вопрос: Какой HTTP-метод обычно используется для отправки данных формы на сервер в HTML?
a) GET
b) POST
c) PUT
d) DELETE

Практическое задание:

Создайте шаблон html страницы, на который будут выводиться фильмы из заранее подготовленного списка. Фильм состоит из следующих полей: название, жанр, год выпуска и описание. При создании необходимо использовать view-функцию, а также css стили.

Тема 3.5 Введение в базы данных

Базы данных. С какими базами данных работает Django. основные понятия и принципы, связанные с базами данных.
Тестирование:
1. Что такое SQL?
A) Структурированный язык запросов к базе данных.
B) Системный язык программирования.
C) Современный язык маркировки.
D) Системный язык разметки.

2. Что такое первичный ключ (Primary Key) в реляционных базах данных?
A) Уникальный индекс, определенный для каждой таблицы.

B) Уникальное имя для столбца в таблице.
C) Основной столбец, содержащий основные данные.
D) Столбец, содержащий только числовые значения.

3. Какой тип баз данных наиболее подходит для хранения неструктурированных данных, таких как текст, изображения и видео?
A) Реляционные базы данных.
B) NoSQL базы данных.
C) SQLite базы данных.
D) Иерархические базы данных.

4. Что такое транзакция (Transaction) в контексте баз данных?
A) Программный код для выполнения операций с данными.
B) Отчет о состоянии базы данных.
C) Группа операций, выполняемых как единое целое.
D) Запись в системном журнале базы данных.

Тема 3.6 SQL: INSERT, UPDATE, SELECT, DELETE

CRUD. Create (Создание). Read (Чтение). Update (Обновление). Delete (Удаление). SQL (Structured Query Language).
Тестирование:
1. Какая SQL-операция используется для добавления новой записи в таблицу?
A) INSERT
B) UPDATE
C) SELECT
D) DELETE

2. Какая SQL-операция используется для изменения существующей записи в таблице?
A) INSERT
B) UPDATE
C) SELECT
D) DELETE

3. Какая SQL-операция используется для извлечения данных из таблицы?
A) INSERT
B) UPDATE
C) SELECT
D) DELETE

4. Какая SQL-операция используется для удаления записи из таблицы?
A) INSERT
B) UPDATE
C) SELECT
D) DELETE

Тема 3.7 SQL: Сортировка, ограничения и сдвиг выборки (LIMIT, OFFSET)

Основные команды и ключевые слова. Сортировка (ORDER BY). Ограничение количества записей (LIMIT). Сдвиг выборки (OFFSET).
Тестирование:
1. Какой оператор SQL используется для сортировки результатов запроса?
A) WHERE
B) ORDER BY*

C) GROUP BY
D) SELECT DISTINCT

2. Для чего используется ключевое слово SQL "LIMIT"?
A) Для ограничения количества строк, возвращаемых запросом*
B) Для фильтрации строк в таблице
C) Для объединения таблиц
D) Для группировки данных

3. Какой оператор SQL используется для сдвига начала выборки результатов запроса?
A) OFFSET*
B) LIMIT
C) WHERE
D) GROUP BY

4. Как правильно сортировать данные в убывающем порядке (по убыванию значений) в SQL?
A) ORDER BY DESC*
B) ORDER BY ASC
C) SORT DESC
D) DESCENDING ORDER BY

Практическое задание:
Допустим, у вас есть база данных с информацией о фильмах и их рейтингах.
Используйте таблицу с именем "movies" с соответствующими столбцами, такими как "title" (название фильма), "year" (год выпуска), "rating" (рейтинг) и другими, если необходимо.
Вам необходимо написать SQL-запросы для выполнения следующих задач:

1. Получите список фильмов, отсортированных по алфавиту в порядке возрастания.

2. Получите список фильмов, отсортированных по году выпуска в порядке убывания (сначала самые новые).
3. Выведите 10 самых высоко оцененных фильмов (с наивысшим рейтингом), отсортированных по убыванию рейтинга.
4. Получите список фильмов, выпущенных в период с 2010 по 2020 год, отсортированных по году выпуска в порядке возрастания.
5. Выведите фильмы, начиная с 11 по 20 в списке, отсортированные по алфавиту в порядке возрастания названий.
Подготовьте 5 соответствующих запросов и убедитесь, что ваши SQL-запросы правильно выполняют указанные задачи.

Тема 3.8 SQL: Группировка записей в выборке

Группировка записей в SQL. Оператор GROUP BY. Агрегатные функции.
Тестирование:
1. Какой SQL-оператор используется для группировки записей в результате запроса?
A) GROUP BY*
B) WHERE
C) ORDER BY
D) SELECT DISTINCT
2. Какая агрегатная функция SQL используется для подсчета количества записей в группе?
A) SUM
B) AVG

C) COUNT*
D) MAX

3. Если вы хотите найти среднюю цену продуктов в каждой категории, какой SQL-запрос вы используете?
A) SELECT AVG(price) FROM products GROUP BY category_id;
B) SELECT SUM(price) FROM products GROUP BY category_id;
C) SELECT COUNT(price) FROM products GROUP BY category_id;
D) SELECT MAX(price) FROM products GROUP BY category_id;

4. Какой SQL-запрос используется для нахождения максимальной цены продукта в каждой категории?
A) SELECT AVG(price) FROM products GROUP BY category_id;
B) SELECT SUM(price) FROM products GROUP BY category_id;
C) SELECT COUNT(price) FROM products GROUP BY category_id;
D) SELECT MAX(price) FROM products GROUP BY category_id*.

Практическое задание:
У вас есть таблица "orders" с информацией о заказах в интернет-магазине. Таблица имеет следующие столбцы:

- order_id (идентификатор заказа)
- customer_id (идентификатор клиента)
- order_date (дата заказа)
- order_amount (сумма заказа)

Ваша задача - написать SQL-запросы для выполнения следующих задач:
1. Найдите общее количество заказов для каждого клиента и выведите результаты вместе с их идентификаторами клиентов.
2. Найдите сумму всех заказов, размещенных в месяце июнь 2023 года.
3. Определите месяц с наибольшим количеством заказов, и выведите эту информацию.

4. Найдите клиента (и его идентификатор), у которого наибольшая общая сумма заказов.
5. Найдите клиента (и его идентификатор), который сделал наибольшее количество заказов.
Убедитесь, что ваши SQL-запросы правильно используют оператор GROUP BY и агрегатные функции для группировки и анализа данных. Ваш запрос должен корректно выполнять указанные задачи.

Тема 3.9 Работа с базой из Python

Подключения к базе данных SQLite и выполнение SQL-запросов в Python.

Тема 3.10 Программы для работы с базами данных

DBeaver. Основные особенности DBeaver.

Тема 3.11 Отношения между таблицами

Отношения “один к одному”. Отношения “многие к одному”. Объединение таблиц JOIN. Отношения “многие ко многим”. Изменения таблиц в базе.

Тема 3.12 Django ORM

Django ORM (Object-Relational Mapping). Модели (Models) в Django ORM. Миграции (Migrations). QuerySet. Фильтрация и сортировка в Django ORM. Отношения. Агрегатные функции. Сигналы (Signals). Многозадачность и безопасность.
Тестирование:
Вопрос 1: Что представляет собой Django ORM?
A) Язык программирования для веб-разработки.

B) Интерфейс для создания веб-дизайна.
C) Библиотека для взаимодействия с базой данных с использованием Python и абстракции объектно-реляционного отображения (ORM).
D) Графический редактор для создания изображений.

Вопрос 2: Какие преимущества предоставляет Django ORM?
A) Он позволяет создавать сложные анимации на веб-страницах.
B) Он автоматически генерирует весь фронтенд вашего веб-приложения.
C) Он упрощает взаимодействие с базой данных, предоставляя абстракцию ORM*.
D) Он предоставляет средства для создания веб-серверов.

Вопрос 3: Что такое QuerySet в Django ORM?
A) Это средство для создания графических запросов к базе данных.
B) Это специальный класс для создания моделей.
C) Это миграция базы данных.
D) Это объект, представляющий запрос к базе данных и используется для выборки данных, фильтрации и агрегации*.

Вопрос 4: Какие виды отношений между моделями поддерживает Django ORM?
A) Только однозначные отношения (OneToOne).
B) Только многозначные отношения (ManyToMany).
C) Только отношения "многие-ко-многим" (ManyToMany) и "один-ко-многим" (ForeignKey).
D) Все перечисленные выше варианты*.

Тема 3.13 Админка Django

Django admin. Генерация интерфейса. Аутентификация и авторизация. Пользовательские действия. Поиск и фильтрация. Поддержка различных типов полей. Расширяемость. Создание администратора для доступа к админ-панели Django. Регистрация моделей в админке. Настройка интерфейса: фильтры, поиск, отображение на странице, инлайны. Локализация и перевод.

Тема 3.14 Получение данных из бд через Django ORM

Получение данных SELECT … FROM через ORM. Простой SELECT ... FROM:. Фильтрация WHERE(filter/exclude). Условия AND, OR, NOT. Q-объекты. Сортировка ORDER BY, LIMIT/OFFSET. Ограничение (LIMIT/OFFSET). Комбинирование сортировки и ограничения.

Тема 3.15 Запрос к связанным моделям

Select_related. Prefetch_related.

Тема 3.16 Django-формы

Работа с формами в Django. Создание формы. Отображение формы в HTML-шаблоне. Обработка данных формы. Класс Forms.
Формы на основе моделей forms.ModelForm

Тема 3.17 Защита формы от атак. CSRF-токен

CSRF (Cross-Site Request Forgery). CSRF-токен (или маркер).

Тема 3.18 Работа с изображениями в формах

CBV. view-классы вместо view-функций. Создание класса представления (CBV). Подключение класса представления к URL. Создание шаблона. Дополнительная настройка.
Работа с пользователями в Django. Создание модели пользователя. Аутентификация и вход. Страница авторизации и регистрации. Декораторы. Декоратор login_required.

Тема 3.19 Тестирование

Тестирование в Python. Обеспечение качества. Поддержание стабильности. .Улучшение документации. Упрощение рефакторинга. Автоматизация тестирования. Увеличение уверенности. Взаимодействие в команде. Экономия времени и ресурсов.
Unittest в django. Unittest vs Pytest. Pytest. Pytest в Django.

Глава 4. API

Тема 4.1 Что такое API

Что такое API. Как подключиться к **API.**

Тема 4.2 Работа с внешними API

Введение.

Тема 4.3 Клиент-API Telegram

Подходы для создания телеграмм ботов: официальное api от телеграмм, сторонние библиотеки.
python-telegram-bot
pyTelegramBotAPI
aiogram

Тема 4.4 Создание телеграм бота

Регистрация бота. Подключение библиотеки. Написание бота.
Практическое задание:
Доработайте телеграмм бота, таким образом, чтобы при запросе “Какой курс доллара”, телеграмм бот обращался на любой внешний сервис, получал курс обмена доллара к рублю и выводил его в диалог пользователю.

Тема 4.5 API TMDB

API The Movie Database (TMDb).
Практическое задание:
Модифицируйте код выше, таким образом, чтобы на экран выводилось название фильмы на русском языке.

Тема 4.6 Django Rest Framework

Сериализаторы (Serializers) в Django REST framework (DRF). Пример использования сериализаторов в DRF. View-функции. View-функции на основе функций. View-функции на основе классов (наследуя от APIView). View-функции на основе классов (используя Mixins и Generics).
View-классы. APIView. GenericAPIView. Классы, основанные на CRUD операциях. ModelViewSet.
Роутеры, Вьюсеты.

Тема 4.7 Сериализаторы для связанных моделей

Аутентификация по токену. JWT. Djoser.

Тема 4.8 Права доступа – Permissions

IsAuthenticated. IsAdminUser. IsAuthenticatedOrReadOnly. IsOwnerOrReadOnly. DjangoModelPermissions. DjangoObjectPermissions. Как настроить права доступа в DRF.

Тема 4.9 Ограничение количества запросов Throttling

Способ 1.Используя Django REST framework (DRF). Способ 2. Используя стандартный middleware.

Тема 4.10 Пагинация в DRF

Использование встроенных стилей пагинации. PageNumberPagination. LimitOffsetPagination. CursorPagination. CustomPagination. Использование других стилей пагинации. Настраиваемая пагинация.

Тема 4.11 Фильтрация, сортировка, поиск

Фильтрация (Filtering). Сортировка (Sorting). Поиск (Search). Комбинированные операции.

Тема 4.12 Документация API. Swagger

Документация API. Swagger. Как можно настроить Swagger в Django REST framework.

Тема 4.13 CORS политика

ORS (Cross-Origin Resource Sharing). Какие задачи решает CORS.

Глава 5. Инфраструктура

Тема 5.1 DevOps. Работа с сервером

DevOps (сокращение от Development и Operations). Главная цель DevOps. Основные принципы и практики DevOps.
Тестирование:
1. Каково определение DevOps?

a) Методология разработки приложений.
b) Система автоматической финансовой отчетности.
c) Совместная методология, объединяющая разработку и операции.
d) Вид программного обеспечения для виртуализации.

2. Какая является главной целью DevOps?
a) Увеличение числа сотрудников в организации.
b) Сокращение бизнес-процессов.
c) Улучшение сотрудничества между отделами разработки и операций.
d) Увеличение сложности программных приложений.

3. Какая из нижеперечисленных практик не является частью DevOps?

a) Непрерывная интеграция и непрерывая доставка (CI/CD).
b) Автоматическое тестирование и доставка пиццы.
c) Мониторинг производительности приложений.
d) Автоматизированное развертывание приложений.

4. Какие преимущества может предоставить внедрение DevOps?
a) Замедление процесса разработки.
b) Увеличение числа ошибок в приложении.
c) Улучшение качества и скорости поставки программного обеспечения.
d) Увеличение количества изолированных команд.

Тема 5.2 Подключение к удаленной машине

Учетная запись на удаленном сервере. SSH-клиент. IP-адрес. Пароль или ключ.

Тема 5.3 Первый деплой

Что такое окружение развёртывания? Выбор хостинг провайдера. Подготовка веб-сайта к публикации.

Тема 5.4 Веб-прокси. NGINX

NGINX (произносится "энджин-экс"). Причин,ы по которым NGINX широко используется и является популярным выбором для многих веб-приложений.

Тема 5.5 Доменное имя

Доменное имя. Регистратор доменных имен.

Тема 5.6 Шифрование. HTTPS

HTTPS (HyperText Transfer Protocol Secure). Основные принципы HTTPS. Шифрование данных. Идентификация сервера. Целостность данных. Использование порта 443.

Тема 5.7 Мониторинг и сбор ошибок

Sentry. Почему Sentry полезен для разработчиков. Sentry и Django.

Тема 5.8 Виртуализация. Контейнеризация

Виртуализация и контейнеризация. Выбор между виртуализацией и контейнеризацией.

Тема 5.9 Docker

Docker.

Тема 5.10 Запуск приложения в docker-контейнере

Установке Docker на Windows. Инструкция по установке WSL. Установка на Docker на macOS
Тестирование:

1. Вы установили Docker на свой компьютер. Какие инструменты на нём появились?

a. Докер-демон
b. Докер-клиент
c. Докер-джин
d. Докер-контейнер

2. Вы установили Docker на свой компьютер. Как в контексте работы с контейнерами называется ваш компьютер (или виртуальная машина, в которой установлен Docker)?

a. Докер-хост
b. Докер-контейнер
c. Докер
d. Контейнеровоз

Глава 6. Дипломная работа

Подготовка дипломной работы.
За основу дипломной работы возьмем проект InnovaTV. Структура проекта должна быть примерно такая же как и в проектной работе из главы Docker. Единственное что добавится - это внешний и внутренний прокси-сервер Nginx. В качестве веб-сервера берем Gunicorn, в качестве базы данных - PostgreSQL.

Структура проекта:
innova-tv

├── LICENSE

```
├── README.md
├── backend
│   ├── Dockerfile
│   ├── api
│   ├── backend
│   ├── manage.py
│   ├── requirements.txt
│   ├── .env
├── infra
    ├── docker-compose.yml
    ├── gateway
        ├── Dockerfile
        ├── nginx.conf
```

Пример docker-compose.yml:

```
# Файл docker-compose.yml
version: '3'
volumes:
  pg_data:
  # Новый volume — для статических файлов
  static:
services:
  db:
    image: postgres:13.10
    env_file: .env
    volumes:
      - pg_data:/var/lib/postgresql/data
```

```
  backend:
    build: ../backend/
    env_file: .env
    # Тут подключаем volume к backend
    volumes:
      - static:/backend_static
  gateway:
    build: ./gateway/
    # А тут подключаем volume со статикой к gateway
    volumes
      - static:/staticfiles
    ports:
      - 8000:80
```

Для медиа файлов необходимо сделать по аналогии со статик файлами, необходимо создать volume для медиа с нужными директориями внутри каждого сервиса.

Для nginx.conf файла должны быть следующие настройки:

```
server {
 listen 80;
  # Запросы по адресам /api/... перенаправляй в контейнер backend
  location /api/ {
    # Это и есть нужная строка:
    # при перенаправлении запроса в контейнер backend
    # подменить адрес "backend" в заголовке запроса
    # на тот адрес, который пользователь ввёл в браузере
    proxy_set_header Host $http_host;
    proxy_pass http://backend:8000/api/;
```

```
  }
  location /admin/ {
    # И в этом блоке то же самое:
    proxy_set_header Host $http_host;
    proxy_pass http://backend:8000/admin/;
  }
  location / {
    alias /staticfiles/;
    index index.html;
  }
  # Также нужно добавить блок для медиа файлов
}
```

Данная конфигурация должна быть скопирована внутрь контейнера в специальную директорию для работы с nginx. Docker файл должен быть следующим:

```
FROM nginx:1.22.1

COPY nginx.conf /etc/nginx/templates/default.conf.template
```

После запуска docker compose up -d –build должен подняться контейнер gateway с нужными нам конфигурациями.

Для того что бы собрать статику в Django-проекте и перенести в другую директорию можно использовать следующие команды:

```
# Собрать статику Django

docker compose exec backend python manage.py collectstatic

# Статика приложения в контейнере backend

# будет собрана в директорию /app/collected_static/.

# Теперь из этой директории копируем статику в /backend_static/static/;

# эта статика попадёт на volume static в папку /static/:

docker compose exec backend cp -r /app/collected_static/. /backend_static/static/
```

1.Как должно работать веб приложение
Основная структура проекта должна быть такая:

Авторизация/регистрация
Главная страница
Карточка фильма/сериала
Избранное
Профиль пользователя
Поиск фильма/сериала

Авторизация/регистрация
Нужно реализовать апи для авторизации и регистрации, для этого желательно использовать JWT-токены, в качестве библиотеки можно взять djoser/drf-simplejwt. Дополнительно необходимо реализовать апи по подтверждению почты(отправка кода подтверждения на почту).

Главная страница

На главной странице должны выдаваться все фильмы и сериалы. Необходимо реализовать апи для вывода фильмов и сериалов с фильтром на типа контента - film/serial(get-параметр category=film/serial). Обязательно использовать пагинацию. Сортировка должна быть по дате добавления контента. Доступ на главную страницу открыт для всех пользователей. На главной странице выводим только необходимые поля, например наименование, год выпуска, превью, рейтинг. Также для обычных пользователей необходимо отдавать только опубликованный контент. Для пользователя с ролью ADMIN, необходимо отдавать все фильмы и сериалы, включая неопубликованные.

Карточка фильма/сериала
В карточке фильма/сериала в апи отдаем постер, превью, год релиза, рейтинг(должно подсчитываться на бэке), сезоны(если есть), серии(если есть).

Также должен быть следующие эндпоинты:
Эндпоинт для получения потока для контента(информация где лежит файл с видео контентом на сдн для проигрывания в плеере). Для данного эндпоинта должны быть ограничены права доступа - доступ можно получить только если пользователь авторизован. Если у пользователя роль ADMIN, то он должен получить доступ до стримов, если же нет, то в апи возвращать 403.
Эндпоинт для оценки контента(для высчитывания рейтинга)

Избранное
Нужно написать апи для добавления фильма/сериала в избранное и получения списка избранных. Данные можно отдавать такие же как и на главной странице. Доступно только авторизованным пользователям.

Профиль пользователя
В апи профиля пользователя нужно возвращать данные пользователя(имя, фамилия, почта, аватарка). Также необходимо реализовать апи по добавлению/смене аватарки пользователя.

Поиск фильмов/сериалов
Необходимо реализовать апи для поиска контента по схожести наименования контента, можно использовать полнотекстовый поиск от базы данных PostgreSQL. В ответе возвращать такой же набор полей как на главной странице.

2.Основные сущности в проекте
Пользователь - необходимо расширить абстрактную модель из “коробки” Django-приложения новыми полями(аватарка, роль).

Контент - основная сущность, у него должно быть поле “типа контента” - фильм/сериал. Помимо поля тип контента, должны быть еще следующие поля:
наименование контента - строковое поле, текст на кириллице
оригинальное название - оригинальное название контента
краткое описание - текстовое поле
год выпуска - год релиза контента, тип int
постер - картинка
превью - картинка для превью контента
продолжительность - целое число, продолжительность контента в секундах
страна производитель - строковое поле
опубликован - булево поле
дата добавления - дата/время добавления контента

Сезон - сущность для хранения сезонов(если тип контента - сериал), основные поля следующие:
наименование сезона - строковое поле
контент - внешняя ссылка на сущность Контент
номер сезона - тип int
дата добавления - дата/время добавления сезона

Серия - сущность для хранения серии(если тип контента - сериал). В серии должны быть следующие поля:
наименование серии - строковое поле
сезон - ссылка на сущность Сезон
номер сезона - тип int
дата добавления - дата/время добавления серии

Избранное - сущность для хранения избранного контента пользователя. Сущность должна содержать 2 основных поля - это ссылка на контент и ссылка на пользователя. Данную сущность можно реализовать через м2м поле, либо выделить сущность в отдельную модель.

Оценки - сущность для хранения оценок контента от пользователя, на основе этой сущности строится рейтинг контента. Поля следующие:

пользователь - ссылка на сущность Пользователь
контент - ссылка на сущность Контент
оценка - тип int, максимальное число - 10, минимальное

6. УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ

6.1. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Условия реализации обеспечивают: достижение планируемых результатов освоения Программы в полном объеме; соответствие применяемых форм, средств и методов обучения.

Материально –технические условия реализации дополнительной профессиональной программы:
Учебный кабинет 1 этаж помещение 3:
компьютеры - 9 штук (6 аймак / 3 виндоус);
столы - 11 штук;
стулья - 11 штук;
шкафы для книг - 2 штуки;
тумбы под технику - 2 штуки.

Учебный кабинет 1 этаж помещение 2:
шкаф - 1 штука;
столы - 15 штук;
компьютеры - 15 штук (3 айм / 12 виндоус);
стулья - 15 штук;
лампы у мониторов - 15 штук;
экран / телевизор – 1 штука;
доска для записей с рулоном – 1 штука.

Учебный класс 1 этаж помещение 8:
парты - 12 штук;
телевизор - 1 штука;
стулья – 24 штуки;
тумба для техники - 2 штуки
ноутбук преподавателя – 1 штука.

Учебный класс 2 этаж помещение 5:
стол – 2 штуки;
стулья – 4 штуки;
экран – 1 штука;

шкаф для хранения – 1 штука;
пуфик – 2 штуки.

Лекторий 2 этаж помещение 4:
стулья – 50 штук;
стол – 1 штука;
проектор – 1 штука;
шкаф – 1 штука.

Учебный класс 2 этаж помещение 7:
телевизор – 1 штука;
столы – 14 штук;
стулья – 25 штук;
компьютеры (аймак) – 7 штук;
ноутбук преподавателя – 1 штук.

Учебный класс 2 этаж помещение 2:
мини парты – 14 штук;
стулья – 32 штуки;
стол – 1 штука;
шкаф – 1 штука;
тумба для хранения материалов – 2 штуки.

6.2. КАДРОВОЕ ОБЕСПЕЧЕНИЕ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА

Требования к квалификации педагогических кадров, обеспечивающих обучение по программе: Высшее профессиональное образование и стаж работы в образовательном учреждении не менее 1 года, при наличии послевузовского профессионального образования (аспирантура, ординатура, адъюнктура) или ученой степени кандидата наук - без предъявления требований к стажу работы.

6.3. ОБЩИЕ ТРЕБОВАНИЯ К ОРГАНИЗАЦИИ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА

Программа реализуется в очной форме.

В процессе обучения основными формами являются: занятия, включающие лекции, практические занятия и оценочный материал (тестирование). Тематика лекций и заданий соответствует содержанию программы.

Организационно-педагогические условия реализации программы должны обеспечивать ее реализацию в полном объеме, соответствие качества подготовки обучающихся установленным требованиям, соответствие применяемых форм, средств, методов обучения, возрастным особенностям, способностям, интересам и потребностям обучающихся.

6.4. СИСТЕМА ОЦЕНКИ РЕЗУЛЬТАТОВ ОСВОЕНИЯ ПРОГРАММЫ

Форма приема итоговой аттестации – защита дипломной работы.

Экзаменующийся на право получения диплома о профессиональной переподготовке должен защитить проект в течении 120 минут.

Обучающийся допускается к итоговой аттестации после изучение всех тем программы.

Результаты итогового экзамена оцениваются по четырех бальной системе: «Отлично», «Хорошо», «Удовлетворительно», «Неудовлетворительно».

Оценки проставляются в зависимости от правильности и полноты ответа на вопросы:

- «Отлично»;
– «Хорошо»;
– «Удовлетворительно»;
– «Неудовлетворительно».

Оценка «отлично» (задание выполнено более чем на 92%) выставляется обучающемуся, если он глубоко и прочно усвоил материал курса, не затрудняется с ответами на вопросы, уверенно защищает свой проект, с уверенностью отвечает на дополнительные вопросы по проекту;

Оценка «хорошо» (задание выполнено более чем на 75%) выставляется обучающемуся, если он твердо знает материал курса, с небольшими недочетами подготовил проект, не допускает существенных неточностей в ответе на вопрос, правильно применяет теоретические и практические знания;

Оценка «удовлетворительно» (задание выполнено не менее чем на 50%) выставляется обучающемуся, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, испытывает затруднения при защите проекта;

Оценка «неудовлетворительно» (задание выполнено менее чем на 50%) выставляется обучающемуся, который не знает значительной части материала, допускает существенные ошибки, неуверенно, с большими затруднениями отвечает на вопросы или не справляется с ними самостоятельно, не может защитить проект.

Если обучающийся получил оценку «неудовлетворительно», ему дается возможность дополнительно подготовиться и не ранее чем через 2 недели заявиться на итоговую аттестацию.

По результату успешной сдачи итоговой аттестации, учащемуся выдается диплом о профессиональной переподготовке

В случае, если слушатель не может пройти итоговую аттестацию по уважительным причинам (болезнь, производственная необходимость и др.), которые могут быть подтверждены соответствующими документами, ему могут быть перенесены сроки прохождения итоговой аттестации на основе личного заявления.

Лицам, не сдавшим итоговую аттестацию, выдается Справка об обучении.

7. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ОБУЧЕНИЯ, СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1.Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
2. Златопольский Д.М. Основы программирования на языке Python. – М.: ДМК Пресс, 2017. – 284 с.
3. Лутц М. Программирование на Python, том I, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
4. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с.
5. Гэддис Т. Начинаем программировать на Python. – 4-е изд.: Пер. с англ. – СПб.: БХВ-Петербург, 2019. – 768 с.
6. Лучано Рамальо Python. К вершинам мастерства. – М.: ДМК Пресс, 2016. – 768 с.
7. Свейгарт, Эл. Автоматизация рутиных задач с помощью Python: практическое руководство для начинающих. Пер. с англ. — М.: Вильямс, 2016. – 592 с.
8. Рейтц К., Шлюссер Т. Автостопом по Python. – СПб.: Питер, 2017. – 336 с.: ил. – (Серия «Бестселлеры O'Reilly»).
9. Любанович Билл Простой Python. Современный стиль программирования. – СПб.: Питер, 2016. – 480 с.: – (Серия «Бестсепперы O'Reilly»).
10. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
11. Прохоренок Н.А. Python 3 и PyQt. Разработка приложений. – СПб.: БХВ-Петербург, 2012. – 704 с.
12. Пилгрим Марк. Погружение в Python 3 (Dive into Python 3 на русском)
13. Прохоренок Н.А. Самое необходимое. — СПб.: БХВ-Петербург, 2011. — 416 с.

8. ОЦЕНОЧНЫЙ МАТЕРИАЛ
к итоговой аттестации

Защита дипломной работы.
За основу дипломной работы возьмем проект InnovaTV. Структура проекта должна быть примерно такая же как и в проектной работе из главы Docker. Единственное что добавится - это внешний и внутренний прокси-сервер Nginx. В качестве веб-сервера берем Gunicorn, в качестве базы данных - PostgreSQL.
Требования к дипломной работе отображены в главе 6 рабочей программы.

Обучающийся демонстрирует свой проект и защищает его. Преподаватель может задавать дополнительные вопросы по пройдённому материалу и/или по проекту.