ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«24» июня 2025 г.
Протокол №2

Рабочая программа дисциплины (модуля) «Разработка на Python. Углубленный»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Разработка*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2025*

Москва
2025

Электронный документ

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Разработка на Python. Углубленный» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по специальности 02.03.01 Математика и компьютерные науки, профиль Разработка, утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) дает умение программирования, используемым в различных областях, таких как веб-разработка, анализ данных и машинное обучение. Освоение Python позволяет эффективно решать сложные задачи в профессиональной деятельности.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 Математика и компьютерные науки, профиль Разработка и входит в обязательную часть Блока 1, как дисциплина по выбору.

Дисциплина (модуль) изучается на 1 курсе в 1 семестре.

Дисциплина (модуль) «Разработка на Python» включает три уровня: основной, углубленный и профессиональный, которые соответствуют навыкам и подготовке обучающихся; уровни определяются с помощью входного тестирования, по итогам которого обучающиеся распределяются на соответствующие уровни.

«Углубленный» уровень подходит обучающимся, знакомым с основами Python, которые хотят понять, какие задачи решает разработчик.

Цель изучения дисциплины (модуля): формирование у студентов навыков программирования на языке Python, включая разработку, отладку и оптимизацию приложений.

Задачи изучения дисциплины (модуля):

— освоение основ языка Python и принципы структурирования кода;
— понимание базовых принципов анализа и визуализации данных;
— структурирование решений с использованием функций и классов;
— выполнение базовых обработок и анализ данных;

В результате освоения дисциплины (модуля), обучающийся должен:

знать:

— основные концепции API: что такое API, принципы взаимодействия с REST API;
— JSON: структура, ключи, значения, вложенность и правила обработки;
— основные команды Git: создание репозиториев, ветвление, коммиты, слияние и разрешение конфликтов;
— основы работы с GitLab: создание проектов, управление ветками, MR;
— принципы разработки ботов, особенности взаимодействия с Bot API;
— принципы работы клиент-серверной архитектуры: роль фронтенда и бэкенда;
— основные возможности FastAPI: обработка маршрутов, работа с запросами и формирование ответов;
— как передавать данные между фронтендом и бэкендом через HTTP-запросы;
— принципы использования HTML-шаблонов и их связь с данными от бэкенда;

уметь:

— выполнять запросы к REST API с использованием библиотеки requests (GET, POST, PUT, DELETE);
— получать и обрабатывать данные из JSON-ответов API;
— писать базовые программы для автоматизации задач с использованием API;
— работать с Git для управления версионностью проекта: выполнять коммиты, переключаться между ветками, работать с удаленными репозиториями;

— разрабатывать боты с использованием библиотеки python-…..-bot;

— реализовывать базовые команды и обработчики событий для бота, работать с инлайн-кнопками и пользовательскими вводами;

— реализовывать API для обработки запросов от фронтенда;

— передавать данные между бэкендом и фронтендом, используя JSON или данные форм;

— настраивать маршруты FastAPI для взаимодействия с готовым фронтендом;

— интегрировать статические файлы (CSS, JavaScript) и HTML-шаблоны в проект FastAPI;

— разрабатывать базовую логику для обработки запросов от фронтенда, таких как добавление, удаление или отображение данных;

владеть:

— навыком разработки полнофункциональных ботов для решения прикладных задач;

— взаимодействием с несколькими API в рамках одной программы, комбинируя их результаты;

— созданием и ведением проектов в GitLab;

— организацией проектной структуры для разработки бота: модульный код, разделение логики API и взаимодействия с пользователем;

— разработкой проектов в команде с использованием Git: создавать MR, проводить код-ревью, разрешать конфликты при слиянии;

— написанием бэкенда для работы с готовым фронтендом, обеспечивающий корректное выполнение функционала.;

— организацией кода проекта так, чтобы отделить логику работы с данными от маршрутов;

— обработкой ошибки на стороне бэкенда и отправлять понятные сообщения фронтенду;

— обеспечением корректной работы сервиса как единой системы: от взаимодействия фронтенда с бэкендом до вывода данных пользователю.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
ОПК-6.	Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-6.1.	Знает алгоритмы разработки, компьютерные программы, а также алгоритмы вычислительной математики
		ОПК-6.2.	Умеет разрабатывать математические программные продукты и комплексы с использованием современных технологий программирования
		ОПК-6.3.	Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований
ПК-3.	Способен использовать методы математического и алгоритмического моделирования при решении теоретических и прикладных задач	ПК-3.1.	Знает основные методы математического и алгоритмического моделирования, а также их применение для решения теоретических и прикладных задач
		ПК-3.2.	Умеет разрабатывать и применять математические модели и алгоритмы для решения различных задач, анализируя полученные результаты
		ПК-3.3.	Имеет практический опыт использования методов математического и алгоритмического моделирования в реальных проектах или исследованиях
ПК-4.	Способен под руководством специалиста более высокой категории разрабатывать программное обеспечение для решения прикладных задач в сфере	ПК-4.1.	Знает основные принципы разработки программного обеспечения и методы решения прикладных задач
		ПК-4.2.	Умеет применять языки программирования и инструменты разработки для создания программного обеспечения под руководством более опытного специалиста
		ПК-4.3.	Имеет практический опыт участия в проектах по разработке программного обеспечения, работая в команде под руководством специалиста более высокой категории

3. Тематический план

№п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы					ТКУ (текущий контроль успеваемости)
		Очная форма					
		Контактная работа			Контроль	Самостоятельная работа	
		Лекции	Семинары (практические занятия)	Консультации			
1	Инструменты и практика разработки	12	12	7	4	60	Коллоквиум Проект
2	Разработка веб приложения	12	12	7	4	60	Коллоквиум Проект
	Зачет с оценкой						
	Итого:	***24***	***24***	***14***	***8***	***120***	
Объем дисциплины (модуля) (в ак. ч.)		***190***					
Объем дисциплины (модуля) (в зач. ед.)		***5***					

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Инструменты и практика разработки	Git и GitLab. Протокол HTTP и REST API. Работа с библиотекой requests и JSON. Модульный код и разделение логики. Инлайн-кнопки и продвинутые обработчики
2	Разработка веб приложения	Основы клиент-серверной архитектуры. Знакомство с FastAPI. Работа с данными и настройка API. Интеграция HTML-шаблонов и статики. Обработка форм и передача данных. Обработка ошибок и обеспечение стабильности

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Харрисон, М. Как устроен Python. Гид для разработчиков, программистов и интересующихся : практическое руководство / М. Харрисон. - Санкт-Петербург : Питер, 2019. - 272 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-0906-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1760794.

2. Бейдер, Д. Чистый Python. Тонкости программирования для профи : практическое руководство / Д. Бейдер. - Санкт-Петербург : Питер, 2021. - 288 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-0803-9. - Текст : электронный. - URL: https://znanium.ru/catalog/product/1766370.

3. Ван Хорн II, Б. М. PyCharm: профессиональная работа на Python : практическое руководство / Б. М. Ван Хорн II, К. Нгуен ; пер. с англ. И. Л. Люско. – Москва : ДМК Пресс, 2024. - 620 с. – ISBN 978-5-93700-274-7. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2205063.

4. Равичандиран, С. Глубокое обучение с подкреплением на Python. OpenAI Gym и TensorFlow для профи : практическое руководство / С. Равичандиран. - Санкт-Петербург : Питер, 2020. - 320 с. - (Серия «Библиотека программиста»). - ISBN 978-5-4461-1251-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1756109.

Дополнительная литература:

1. Гниденко, И. Г. Технологии и методы программирования : учебник для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 241 с. — (Высшее образование). — ISBN 978-5-534-18130-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/581329.

2. Чернышев, С. А. Основы программирования на Python : учебник для вузов / С. А. Чернышев. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 349 с. — (Высшее образование). — ISBN 978-5-534-17139-6. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/567821.

3. Гаско, Р. Простой Python просто с нуля / Р. Гаско ; под ред. Н. Ю. Комлева. - Москва : СОЛОН-ПРЕСС, 2023. - 256 с. - (Серия «Программирование»). - ISBN 978-5-91359-334-4. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2185854.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и

индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;
— компьютерной техникой;
— механическими калькуляторами;
— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое

Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Разработка на Python. Углубленный» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, консультации, коллоквиумы и проекты, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Участие в семинаре – активная работа студента на семинаре, его ответы на вопросы преподавателя и участие в дискуссии.

Для успешного участия в семинаре студентам рекомендуется заранее ознакомиться с темой обсуждения, прочитать необходимые материалы и подготовить вопросы. Важно активно слушать и вовлекаться в дискуссию, высказывая свои мнения и аргументируя их.

При ответах на вопросы преподавателя стоит быть уверенным, четким и логичным, опираясь на изученный материал. Также полезно поддерживать диалог с однокурсниками, чтобы обогатить обсуждение и расширить свои знания.

Консультации – структурированные встречи, на которых преподаватели предоставляют индивидуальную или групповую помощь в освоении учебного материала, обсуждении вопросов и решении проблем, возникающих в процессе обучения.

Консультации могут включать разъяснение сложных тем, подготовку к экзаменам и помощь в выполнении проектных работ, что способствует более глубокому пониманию предмета и улучшению академической успеваемости.

Коллоквиум – устные ответы на вопросы, список которых известен студенту заранее.

В процессе подготовки к коллоквиуму необходимо проанализировать учебные материалы, ознакомившись с лекциями, учебниками и дополнительными источниками, акцентируя внимание на ключевых темах. Рекомендуется создать структурированные конспекты, выделяя основные идеи, термины и формулы.

Проект – это целенаправленная деятельность, имеющая определенные цели, задачи и временные рамки, в результате которой создается уникальный продукт или услуга.

Для успешной подготовки проекта рекомендуется следует выполнять следующие рекомендации:

- четко определите цель и задачи проекта, чтобы понимать, какой результат вы хотите достичь;
- составьте план работы, разбив проект на этапы с указанием сроков выполнения каждого из них;
- используйте разнообразные источники информации и инструменты для исследования темы, чтобы обеспечить качественную основу для вашего проекта;
- регулярно проверяйте прогресс и вносите коррективы в план, если это необходимо, чтобы оставаться на правильном пути к завершению проекта.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Бонусные баллы — это оценки, которые студенты могут получить за выполнение дополнительных заданий.

Формат бонусных баллов позволяет студентам улучшить общую оценку по курсу и стимулирует углубленное изучение материала.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине «Разработка на Python. Углубленный»

Оценивание уровня учебных достижений, обучающихся по дисциплине (модулю), осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме

зачета с оценкой, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом:

Десятибалльная оценка	**Пятибалльная оценка**	**Оценка за зачет**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Зачтено	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	Зачтено	
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет
6	Хорошо	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			связывать теоретические аспекты предмета с практическими задачами.
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине, но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	Зачтено	
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

Дисциплина (модуль) «Разработка на Python. Углубленный» оценивается следующим образом:

Активность	Вес	Количество	Описание
Коллоквиум	30%	2	Устные ответы на вопросы, список которых известен студенту заранее
Проекты	70%	6	Каждый проект разбивается на этапы и оценивается по заданным критериям

Формула расчёта итоговой оценки по дисциплине (модулю) «Разработка на Python. Углубленный»: «0,3 × среднее за коллоквиум + 0,6 × среднее за проекты».

При изучении дисциплины (модуля) так же возможно получение бонусных баллов.

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные задания для коллоквиума

Коллоквиум состоит из трёх шагов, связанных между собой:

1. **Анализ кода**
   Ты рассказываешь, что делает программа, объясняешь логику её работы, находишь потенциальные ошибки. Проверяется умение “читать код глазами”, объяснять, как он выполняется, и замечать проблемы.
2. **Дебаг (поиск и исправление ошибок)**
   Преподаватель предлагает исправить найденные ошибки — например, заменить неверный HTTP-метод, добавить обработку исключений или корректно передать данные в запрос. Проверяется внимание к деталям, знание синтаксиса и принципов HTTP/REST, понимание типичных ошибок.
3. **Дописать часть кода (мини-задача)**
   Нужно реализовать недостающий фрагмент — например, добавить новую команду бота, обработчик, функцию или часть логики. Проверяется умение применять базовые принципы написания кода: функции, структуры данных, работа с API и JSON.

В конце могут быть **дополнительные теоретические вопросы** по теме — например, про работу callback-запросов, обработку ошибок, структурирование кода и т.д.

Как оцениваем

Коллоквиум оценивается **по процессу работы с кодом**, а не по готовому решению. Главное — показать понимание, логику и грамотный ход мыслей.

Критерий	**Что проверяется**	**Пример хорошего ответа**
Понимание логики программы	Насколько точно студент объясняет, что делает код, какие модули и принципы используются	«Функция add_task делает POST-запрос в API GitLab для создания задачи, но в коде сейчас используется GET, поэтому сервер вернёт ошибку»
Умение исправлять ошибки	Замечает ли студент проблемы (метод, токен, обработка исключений, структура данных) и может ли предложить решение	«Токен не стоит хранить в коде — лучше вынести его в .env и подгружать через переменные окружения»
Практическое мышление	Способность предложить улучшение или дополнение	«Для /list можно сделать requests.get, распарсить response.json() и пройтись по задачам циклом»
Аргументация	Умение объяснить, почему именно так нужно исправить или реализовать	«Код 201 означает, что объект успешно создан, поэтому его нужно проверять после POST-запроса»

Как подготовиться

- Повтори темы: работа с **HTTP-запросами**, **модулями requests и json**, основы **Telegram Bot API**, обработку ошибок (try/except), и принципы **REST**.
- Попробуй самостоятельно объяснить, как работает готовый код — шаг за шагом.
- Обрати внимание, **как правильно обрабатывать ошибки и проверять ответы от API**.
- Подумай, как можно улучшить читаемость и структуру кода.

Коллоквиум — это не проверка на "знание синтаксиса", а возможность показать, **что ты умеешь думать как разработчик**: видишь логику, понимаешь, где ошибка, и можешь предложить адекватное решение.

Как проходит коллоквиум

Коллоквиум — это итоговая оценка твоих навыков разработки на Python, включающая как проверку теоретической базы, так и решение практической задачи в условиях, максимально приближённых к реальной работе.

В этот раз мы уходим от устного формата и субъективной оценки — весь коллоквиум построен так, чтобы результаты каждого студента были измеримыми, прозрачными и объективными.

Коллоквиум длится **3 часа** и проходит в два этапа: теоретический тест и практическая часть с задачей в проекте.

Слоты для коллоквиума есть у вас в календарях. Сдать коллоквиум в другой слот нельзя!

Формат и тайминг

Общая длительность: 3 часа.

1. Организационный блок — 20 минут

- Брифинг и объяснение правил.
- Раздача вариантов теоретического теста.
- Ответы на вопросы по проведению.

2. Теоретическая часть — 30 минут

- Письменный тест с вопросами разных типов.

3. Перерыв — 10 минут

4. Подготовка к практике — 10 минут

- Получение варианта задания.
- Запуск прокторинга.
- Подготовка окружения.

5. Практическая часть — 100 минут

- Работа с проектом: нужно реализовать указанный в TODO функционал.
- Проверка осуществляется автоматическими тестами.
- На практической части можно использовать IDE, **без установленных ИИ агентов и копилотов** (Cursor использовать нельзя!)
- Практическую часть можно написать только при написании теоеретической части

6. Завершение — 10 минут

- Формирование архива с решением.
- Загрузка в LMS.

Как оцениваем

Главный принцип — **объективность и прозрачность**.

Теория: оценивается по чётким критериям.

Практика: оценивается исключительно автотестами — никакой субъективной интерпретации.

Итоговая оценка складывается из:

- 40% — теория
- 60% — практика

Примерные задания по проекту

Цель проекта

Создать Telegram-бота, который по команде `/wiki <термин>` находит определение в Wikipedia и отправляет его пользователю.

Главная цель: научиться интегрировать Python-приложение с Telegram Bot API, используя библиотеку `telebot`.

План работы

Шаг 1. Получить токен бота

1. Открой Telegram и найди бота @BotFather
2. Отправь команду `/newbot`
3. Следуй инструкциям: придумай имя и username для бота
4. Скопируй полученный токен (выглядит так: `1234567890:ABCdefGHIjklMNOpqrsTUVwxyz`)

Важно: Токен — это секретный ключ. Не публикуй его в коде!

Шаг 2. Изучить структуру проекта

```
project_root/
├── smart_handbook
│   ├── __init__.py
│   ├── api_clients              # Пакет для работы с API
│   │   ├── __init__.py
│   │   └── wikipedia_client.py  # Функционал, который был
реализован в проекте 1
│   └── cli                      # Пакет для консольного
интерфейса
│       ├── __init__.py
│       └── main.py
├── telegram_bot                 # Пакет для работы с Telegram
│   ├── __init__.py
│   ├── bot.py                   # Основной файл бота
│   ├── state.py                 # Файл для хранения состояний
```

```
бота
│   └── handlers                  # Пакет для обработчиков
│       ├── __init__.py
│       ├── callback_handlers.py # Обработчики коллбэков
│       └── command_handlers.py  # Обработчики команд
├── README.md                     # Инструкция запуска и описание
API
├── .env                          # Файл с переменными окружения
├── example.env                   # Пример файла с переменными
окружения
├── requirements.txt              # Зависимости проекта
└── README.md                     # Инструкция по запуску и
описание проекта
```

Создай файл `.env`**:**

```
TG_BOT_TOKEN=твой_токен_от_BotFather
```

Шаг 3. Установить зависимости

Создай виртуальное окружение и установи зависимости в проект `telebot`**,** `requests`**,** `python-dotenv` **и др.)**

Не забудь зафиксировать зависимости в файле `requirements.txt`

Шаг 4. Реализовать обработчики команд

Файл `telegram_bot/handlers/command_handlers.py`

Напиши функцию `register_handlers(bot)`, которая регистрирует все обработчики команд.

Что нужно реализовать:

1. **Команда** `/start`
   - Приветствует пользователя
   - Подсказывает, как использовать бота
   - Пример ответа: `"Привет! Я Умный Справочник. Чтобы получить определение, используйте команду /wiki <термин>. Например: /wiki Интеграл"`
2. **Команда** `/help`
   - Показывает справку по использованию
   - Пример ответа: `"Я могу найти краткое определение по любому термину из Wikipedia. Просто используйте команду /wiki <термин>.\nПример: /wiki Эйлер"`
3. **Команда** `/wiki <термин>`
   - Извлекает термин из текста команды
   - Если термин не указан отправляет подсказку: `"Пожалуйста, укажите термин для поиска. Например: /wiki Интеграл"`
   - Вызывает `wikipedia_client.get_summary(term, lang="ru")`
   - Если найдено отправляет определение
   - Если не найдено (None) отправляет: `"Термин '{term}' не найден в Wikipedia."`
   - При ошибке сети/API отправляет: `"Ошибка при обращении к сервису. Попробуйте позже."`
4. **Неизвестная команда** (любая команда, начинающаяся с `/`)
   - Ответ: `"Неизвестная команда. Используйте /wiki <термин>."`

Шаг 5. Реализовать основной файл бота

Файл `telegram_bot/bot.py`

Этот файл запускает бота и регистрирует обработчики.

Что нужно реализовать:

```
import os
from pathlib import Path

import telebot
from dotenv import load_dotenv

from telegram_bot.handlers.command_handlers import
register_handlers


def main() -> None:
    """Запускает бота."""
    # TODO:
    # 1. Загрузи .env файл через load_dotenv()
    # 2. Получи токен из os.getenv("TG_BOT_TOKEN")
    # 3. Проверь, что токен не пустой (выведи сообщение об
ошибке и return)
    # 4. Создай бота через telebot.TeleBot(token)
    # 5. Зарегистрируй обработчики через register_handlers(bot)
    # 6. Выведи сообщение "Бот запущен. Ожидание команд..."
    # 7. Запусти бота через bot.polling(none_stop=True)
    # 8. Обработай ошибку неверного токена через
telebot.apihelper.ApiException
    pass


if __name__ == "__main__":
    main()
```

Шаг 6. Написать README.md

Создай понятное описание проекта и инструкцию по запуску.

Шаг 7. Протестировать бота

1. Запусти бота: `python -m telegram_bot.bot`
2. Открой бота в Telegram (найди по username, который указал в BotFather)
3. Протестируй команды:
    - `/start` — должно прийти приветствие
    - `/help` — должна прийти справка
    - `/wiki Питон` — должно прийти определение
    - `/wiki несуществующийтермин123` — должно прийти "не найден"
    - `/unknown` — должно прийти "Неизвестная команда"

Критерии приёмки

- [ ] Бот запускается без ошибок
- [ ] Команда `/start` возвращает приветствие
- [ ] Команда `/help` возвращает справку
- [ ] Команда `/wiki <термин>` находит и возвращает определение
- [ ] Команда `/wiki` без термина возвращает подсказку
- [ ] Команда `/wiki несуществующийтермин` возвращает "не найден"
- [ ] Неизвестная команда возвращает соответствующее сообщение
- [ ] Токен хранится в `.env` файле
- [ ] Есть `example.env` с пустым токеном
- [ ] `README.md` содержит инструкцию по запуску

Качество кода:

- [ ] Код соответствует PEP8
- [ ] Все функции имеют docstring
- [ ] Нет дублирования кода
- [ ] `wikipedia_client.py` используется через импорт (не копируется)

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Назовите основную HTTP-метод, используемую для получения данных.	GET / POST / PUT	ОПК-6
2	Укажите основную команду Git для добавления файлов в индекс.	add / commit / push	ОПК-6
3	Назовите уровень в архитектуре REST API, отвечающий за ресурсы.	ресурсы / представление / контроллер	ОПК-6
4	Укажите основную функцию библиотеки requests для отправки данных на сервер.	post / get / put	ОПК-6
5	Студент, назовите протокол для обмена данными в веб-разработке, используемый для REST API.	HTTP / HTTPS / FTP	ОПК-6
6	Укажите принцип модульного кода, обеспечивающий разделение логики.	разделение логики / инкапсуляция / абстракция	ОПК-6
7	Назовите компонент клиент-серверной архитектуры, отвечающий за хранение данных.	сервер / база данных / клиент	ПК-3
8	Укажите тип данных в JSON, представляющий список.	array / object / string	ПК-3
9	Назовите этап работы с FastAPI, связанный с созданием эндпоинтов.	эндпоинты / установка / тестирование	ПК-3
10	Укажите способ интеграции HTML-шаблонов в FastAPI, использующий Jinja2.	Jinja2 / встроенный / внешний	ПК-3
11	Назовите тип ошибки в веб-приложениях, связанный с отсутствием ресурса.	404 / 500 / 403	ПК-3
12	Укажите метод обработки форм в FastAPI для получения данных из POST-запроса.	POST / GET / PUT	ПК-3
13	Назовите принцип разработки ПО, обеспечивающий модульность.	модульность / инкапсуляция / наследование	ПК-4
14	Укажите инструмент для работы с GitLab, для управления ветками.	ветки / репозитории / merge	ПК-4
15	Назовите шаг для настройки API в FastAPI, связанный с валидацией данных.	валидация / маршрутизация / документация	ПК-4
16	Укажите способ обработки ошибок в FastAPI, использующий исключения.	исключения / коды / логи	ПК-4
17	Назовите этап разработки бота с инлайн-кнопками, для тестирования обработчиков.	тестирование / разработка / интеграция	ПК-4
18	Укажите метод обеспечения стабильности веб-приложения, связанный с логированием.	логирование / мониторинг / резервное копирование	ПК-4