

Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет»

Документ подписан
простой электронной подписью
Информация о владельце:
ФИО: Лустенков Михаил Евгеньевич
Должность: Ректор
Дата: 09.09.2026

УТВЕРЖДАЮ

Ректор Белорусско-Российского
университета

М.Е. Лустенков

22.09.2026

Регистрационный № УД-090304/6.1.0.23р

РЕШЕНИЕ ПРИКЛАДНЫХ ЗАДАЧ НА ЯЗЫКЕ PYTHON
(наименование дисциплины)

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) Разработка программно-информационных систем

Квалификация Бакалавр

	Форма обучения
	Очная
Курс	2, 3
Семестр	4, 5
Лекции, часы	32
Лабораторные работы, часы	68
Экзамен, семестр	5
Зачет, семестр	4
Курсовая работа, семестр	5
Контактная работа по учебным занятиям, часы	100
Самостоятельная работа, часы	116
Всего часов / зачетных единиц	216/6

Кафедра-разработчик программы: «Программное обеспечение информационных технологий»

(название кафедры)

Составитель: Кутузов Виктор Владимировича, канд. техн. наук, доцент

(И.О. Фамилия, ученая степень, ученое звание)

Могилев, 2026

Рабочая программа дисциплины составлена в соответствии с федеральным государственным образовательным стандартом высшего образования - бакалавриат по направлению подготовки 09.03.04 «Программная инженерия» утвержденным приказом № 920 от 19.09.2017 г., учебными планами рег. №090301-2.2 утвержденными 18.02.2026.

Рассмотрена и рекомендована к утверждению кафедрой «Программное обеспечение информационных технологий» 02.03.2026, протокол № 8.

Зав. кафедрой  В. В. Кутузов

Одобрена и рекомендована к утверждению Научно-методическим советом
Белорусско-Российского университета

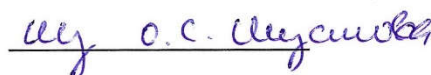
15.04.2025, протокол № 6.

Рецензент:

И. В. Акиншева, заведующая кафедрой программного обеспечения информационных технологий МГУ им. А.А. Кулешова, канд. техн. наук, доцент

Рабочая программа дисциплины согласована

Ведущий библиотекарь



Начальник учебно-методического
отдела

 О.Е. Печковская

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

1.1 Цель дисциплины

Целью изучения дисциплины является формирование у обучающихся профессиональных компетенций в области промышленной разработки прикладного программного обеспечения на языке Python, включая устойчивые навыки проектирования, отладки, оптимизации и документирования программных решений с учётом требований информационной безопасности, актуальных отраслевых стандартов и реальных прикладных задач.

1.2 Планируемые результаты изучения дисциплины

В результате освоения дисциплины студент должен

знать:

- синтаксис, семантику, систему типов, основные коллекции и модель памяти Python, а также возможности объектно-ориентированного и функционального программирования в языке;
- механизмы обработки исключений, контекстные менеджеры, методы логирования, отладки и профилирования кода;
- принципы работы с данными: форматы файлов и сериализации, библиотеки анализа и визуализации данных, реляционные и нереляционные СУБД, ORM SQLAlchemy и миграции Alembic;
- архитектуру веб-приложений и REST API, протокол HTTP, методы аутентификации, асинхронное программирование, разработку ботов, основы компьютерного зрения, больших языковых моделей (LLM, RAG) и требования информационной безопасности;
- требования информационной безопасности при написании и развертывании кода.

уметь:

- проектировать и реализовывать чистый, эффективный и типобезопасный код на Python с применением ООП, функциональных возможностей конструкций и ключевых библиотек экосистемы;
- загружать, очищать, анализировать и визуализировать данные с помощью Pandas, NumPy и Matplotlib, а также работать с файловыми форматами и базами данных;
- решать прикладные задачи, задачи компьютерного зрения, создавать AI-приложения с использованием LLM и RAG-конвейеров, а также разрабатывать ботов для популярных мессенджеров;
- проводить отладку, рефакторинг и тестирование кода;
- оформлять результаты разработки в соответствии с действующими стандартами и требованиями.

владеть:

- навыками промышленной разработки на Python с соблюдением PEP 8, контролем версий (Git), управлением окружениями (Poetry) и инструментами отладки и профилирования;
- навыками полного цикла работы с данными: от импорта и предобработки до визуализации и экспорта результатов в различных форматах;
- методами формализации прикладных задач и их программной реализации;
- практическими приёмами обеспечения надёжности: написание тестов всех уровней, настройка CI/CD-пайплайнов и автоматизация развертывания приложений.

1.3 Место дисциплины в системе подготовки

Дисциплина «Решение прикладных задач на языке Python» относится к Блоку 1 «Дисциплины (модули)», обязательная часть Блока 1.

Перечень учебных дисциплин, изучаемых ранее, усвоение которых необходимо для изучения данной дисциплины:

- Информатика;
- Программирование;
- Базы данных.

Перечень учебных дисциплин (циклов дисциплин), которые будут опираться на данную дисциплину:

- Основы информационной безопасности;
- Тестирование и отладка программного обеспечения;
- Экспертные системы и системы искусственного интеллекта.

Кроме того, знания, полученные при изучении дисциплины на лабораторных занятиях будут применены при прохождении первой и второй технологической (проектно-технологической) практик, а также при выполнении выпускной квалификационной работы и дальнейшей профессиональной деятельности.

1.4 Требования к освоению дисциплины

Освоение данной дисциплины должно обеспечивать формирование следующих компетенций:

Коды формируемых компетенций	Наименования формируемых компетенций для направления подготовки 09.03.04 «Программная инженерия»
ОПК-3	Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности
ОПК-7	Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой

2 СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Вклад дисциплины в формирование результатов обучения выпускника (компетенций) и достижение обобщенных результатов обучения происходит путём освоения содержания обучения и достижения частных результатов обучения, описанных в данном разделе.

2.1 Содержание дисциплины

Номер тем	Наименование тем	Содержание	Коды формируемых компетенций
Тема 1	Введение в Python	1. Основы Python. Место в рейтингах, области применения, философия Zen of Python. 2. Сравнение с другими языками программирования (C#, Java, JavaScript и другими) : интерпретируемость vs компилируемость, динамическая типизация vs статическая, GIL (Global Interpreter Lock). 3. Области примени Python. 4. Основы базового синтаксиса: переменные, типы данных, базовые операции, ввод/вывод. 5. Стандарты оформления кода: PEP 8, автоматическое форматирование и линтинг (ruff, black, isort), подключение pre-commit hooks для проверок до коммита. Настройка проверок как часть требований безопасного кодирования. 6. Документирование с первого дня: docstrings в Google/NumPy стилях, генерация документации Sphinx как обязательный элемент профессионального кода.	ОПК-3 ОПК-7

		- REPL (Read-Eval-Print Loop), Jupyter Notebook, IDE (PyCharm, VS Code с расширениями). Google Coolab. - Установка и настройка окружения: ruenv, виртуальные окружения (venv, poetry), IDE (PyCharm, VS Code), инициализация Git-репозитория, правила коммитов, первый push на GitHub. Настройка защиты ветки main и обязательного Code Review. - Непрерывная интеграция: настройка GitHub Actions для автоматического запуска ruff и black при каждом push – формирование привычки к чистому коду с первой недели. Проверка безопасности (pip-audit). - Роль LLM и AI-ассистентов в разработке на Python: как использовать правильно и когда это вредит обучению, а когда целесообразно использовать. - Cursor, OpenAI Codex, Anthropic Claude Code, Gemini Code Assist, GitHub Copilot Workspace, VS Code + GitHub Copilot, JetBrains PyCharm + AI Assistant и другие среды.	
Тема 2	Типы данных, переменные, коллекции и управляющие конструкции	- Типы данных, переменные, коллекции и управляющие конструкции - Модель памяти Python: всё – объект, переменные – ссылки (сравнение с value/reference types в C#) - Числовые типы: int (неограниченная точность), float, complex, Decimal, Fraction - Строки: неизменяемость, f-строки, многострочные строки, raw-строки - Булев тип и концепция truthiness/falsiness (что является True/False) - None: булев тип и концепция truthiness/falsiness (что есть True/False в условиях), аналог null в C#, но с особенностями - Безопасность типов: как динамическая типизация может привести к уязвимостям (инъекции через невалидированные данные), роль туру в предотвращении логических ошибок - Условные конструкции: if/elif/else, тернарный оператор, match/case (structural pattern matching из Python 3.10+) - Циклы: for (итерация по объектам, а не индексный цикл), while, else в циклах - Базовые коллекции: list (списки, динамический массив), tuple (кортеж, неизменяемый, хешируемый), dict (хеш-таблица, словарь), set (уникальность), slicing (срезы). - Паттерны безопасной работы с коллекциями: использование defaultdict вместо ручных проверок, защита от KeyError при доступе к внешним данным - Comprehensions как замена LINQ: list comprehension, dict comprehension, set comprehension - Статическая типизация на практике: type hints для переменных и коллекций (list[int], dict[str, float]), проверка типов с туру для раннего обнаружения ошибок. Роль типизации в предотвращении уязвимостей. - Модульное тестирование: первые тесты с pytest для функций работы с коллекциями, включая тесты на некорректные входные данные и граничные значения, покрытие не ниже 80%.	ОПК-3 ОПК-7

		Настройка автоматического запуска тестов в GitHub Actions при push. 15. Операторы walrus (:=), unpacking, swap без временной переменной	
Тема 3	Функции, замыкания, декораторы и генераторы	1. Функции, замыкания, декораторы и генераторы 2. Функции как объекты первого класса: передача в аргументы, возврат из функций 3. Параметры: позиционные, именованные, *args, **kwargs, keyword-only аргументы 4. Мутабельные значения по умолчанию (def f(a, lst=[])) – классическая ловушка Python 5. Области видимости: LEGB-правило (Local, Enclosing, Global, Built-in) 6. Замыкания (closures): что это, зачем нужны, пример счётчика 7. Декораторы: как работают, написание собственных декораторов, декораторы с аргументами 8. Стандартные декораторы: @property, @staticmethod, @classmethod, @functools.wraps, @lru_cache. Важность @wraps для сохранения метаданных, что критично для аудита безопасности. 9. Декораторы для безопасности: @require_auth, @rate_limit, @validate_input – практические примеры применения декораторов для защиты функций 10. Генераторы: yield, yield from, генераторные выражения; ленивые вычисления, экономия памяти 11. Генераторы для безопасной обработки больших данных: потоковое чтение файлов с шифрованием «на лету», предотвращение переполнения памяти 12. Документирование функций и генераторов: полное описание назначения, параметров, возвращаемых значений и исключений в docstrings. 13. Тестирование декораторов и генераторов: изоляция побочных эффектов, mock-объекты, проверка корректности ленивых вычислений, тестирование освобождения ресурсов генераторов и обработки исключений. 14. Автоматизация в CI/CD: включение туру и rpydocstyle в GitHub Actions для проверки типов и документации при каждом push, а также запуск тестов с анализом покрытия. 15. Модули collections и itertools. Модуль collections: Counter, defaultdict, deque, namedtuple, OrderedDict. Модуль itertools: chain, product, combinations, permutations, groupby, islice	ОПК-3 ОПК-7
Тема 4	ООП в Python	1. ООП в Python 2. Классы в Python vs C#: синтаксические и концептуальные различия 3. __init__, self (вместо this), атрибуты класса vs атрибуты экземпляра 4. Инкапсуляция по-питоновски: нет private/protected, есть конвенция _ и __ (name mangling), свойства как механизм контроля доступа и валидации. 5. Свойства через @property (аналог get/set в C#) 6. Наследование: множественное наследование, MRO (Method Resolution Order), алгоритм C3-линеаризации	ОПК-3 ОПК-7

		7. Миксины как паттерн (Mixin Pattern) 8. Магические методы (dunder methods): <code>__str__</code>, <code>__repr__</code>, <code>__eq__</code>, <code>__hash__</code>, <code>__lt__</code>, <code>__len__</code>, <code>__getitem__</code>, <code>__call__</code>, <code>__enter__</code>, <code>__exit__</code> 9. Протоколы (duck typing), абстрактные классы (abc.ABC), typing.Protocol. 10. Проектирование с учётом SOLID: единственная ответственность, открытость/закрытость, подстановка. Примеры. Безопасные архитектурные решения через инверсию зависимостей. 11. Документирование классов и методов: полные docstrings с примерами использования – формирование привычки промышленного описания интерфейсов. 12. Модульное тестирование классов: изоляция через mock, проверка инвариантов и граничных состояний, тестирование безопасности геттеров/сеттеров, использование фикстур pytest. 13. dataclasses, slots для оптимизации памяти. Сравнение с Pydantic для безопасной сериализации.	
Тема 5	Обработка ошибок, тестирование и отладка	1. Обработка ошибок, контекстные менеджеры, отладка 2. Философия Python: EAFP (Easier to Ask Forgiveness than Permission) vs LBYL (Look Before You Leap) 3. Иерархия исключений в Python: BaseException → Exception → конкретные исключения 4. Полный синтаксис try/except/else/finally 5. Перехват нескольких исключений и доступ к объекту исключения (except (TypeError, ValueError) as e, атрибуты исключения, args, traceback) 6. Создание пользовательских исключений: лучшие практики (наследование от Exception, а не от BaseException; именование с суффиксом Error; передача контекста) 7. Цепочки исключений: raise ... from ... 8. Контекстные менеджеры: протокол <code>__enter__</code>/<code>__exit__</code> (управление ресурсами через with, гарантированное освобождение, параметры exit: exc_type, exc_val, exc_tb) 9. Модуль contextlib: contextmanager, closing, suppress, redirect_stdout (декоратор @contextmanager для генераторов, suppress для игнорирования конкретных исключений) 10. Логирование: модуль logging – уровни, форматирование, хэндлеры (DEBUG, INFO, WARNING, ERROR, CRITICAL; StreamHandler, FileHandler, RotatingFileHandler; почему не print) 11. Безопасность обработки ошибок: почему нельзя раскрывать детали исключений пользователю (information disclosure), как логировать ошибки без утечки чувствительных данных 12. Продвинутое логирование: логгеры, иерархия, фильтры, конфигурация (logging.getLogger(name), dictConfig, structlog для структурированных логов, корреляция запросов). Настройка фильтров для исключения секретных данных из логов. 13. Интеграция с мониторингом и CI: сбор логов в контейнере, настройка алертинга на критические ошибки. 14. Покрывание кода тестами: анализ pytest-cov в	ОПК-3 ОПК-7

		CI/CD, установка пороговых значений, блокировка pull request при падении покрытия. 15. Отладка: pdb, breakpoint(), отладка в IDE 16. Assertions: когда использовать, когда не использовать. Почему нельзя полагаться на assert для проверок безопасности (флаг -O). 17. Профилирование: cProfile, timeit, line_profiler, memory_profiler (поиск узких мест, построение профилирование, анализ использования памяти, визуализация результатов)	
Тема 6	Работа с файлами, форматами данных и сериализация	1. Работа с файлами, форматами данных и сериализация 2. Работа с файлами: open(), режимы, кодировки (UTF-8, cp1251), BOM 3. Модуль pathlib: объектно-ориентированная работа с путями (замена os.path) 4. CSV: модуль csv, DictReader/DictWriter 5. JSON: json.dumps/loads, сериализация пользовательских объектов, JSONEncoder 6. XML: xml.etree.ElementTree, lxml (кратко) 7. YAML: библиотека PyYAML 8. TOML: tomlib (встроен с Python 3.11) 9. Pickle: бинарная сериализация Python-объектов, риски безопасности, категорический запрет распаковки недоверенных данных; безопасные альтернативы. 10. Работа с архивами: zipfile, tarfile, gzip 11. Работа с Excel: openpyxl 12. Поточковая обработка больших файлов через генераторы. 13. Безопасное обращение с данными: хеширование, шифрование конфиденциальных файлов, исключение хранения секретов в открытом виде, проверка целостности. 14. Валидация загружаемых данных: применение Rydantic для проверки структуры и типов при импорте из CSV/JSON, гарантия целостности. 15. Сравнение форматов: JSON vs YAML vs TOML с точки зрения безопасности (риск code execution в YAML), производительности и читаемости — обоснование выбора 16. Тестирование сериализации и потоковой обработки: использование временных файлов и фикстур, проверка корректности восстановления объектов, тестирование отказа при небезопасных операциях.	ОПК-3 ОПК-7
Тема 7	Обработка данных и визуализация: NumPy, Pandas, Matplotlib/ Seaborn	1. Введение в экосистему визуализации и анализа данных Python: NumPy, Pandas, Matplotlib/Seaborn, Plotly и др. 2. NumPy: зачем нужен, векторизация и производительность (сравнение с чистыми циклами Python, философия array-oriented computing, примеры); Индексация и операции с массивами; Агрегации и линейная алгебра в NumPy 3. Pandas: DataFrame, загрузка данных, фильтрация, группировка и агрегация, объединение данных, обработка пропусков, временные ряды, оптимизации 4. Визуализация: Matplotlib (figure, axes, subplots) и Seaborn (статистические графики) 5. Примеры обработки и визуализации данных 6. Валидация данных на входе: использование Great Expectations или pandera для проверки схемы и	ОПК-3 ОПК-7

		качества данных перед обработкой – предотвращение «мусор на входе – мусор на выходе» 7. Тестирование обработки данных: проверка инвариантов датасетов, unit-тесты трансформаций, smoke-тесты визуализаций. 8. Воспроизводимость исследований: версионирование данных с DVC, синхронизация с Git	
Тема 8	Работа с базами данных	1. Работа с базами данных 2. Типы баз данных: реляционные (SQL) vs нереляционные (NoSQL). 3. Когда использовать БД, а когда файлы/кэш. 4. Обзор экосистемы: драйверы, ORM, инструменты миграций.) 5. DB-API 2.0: стандартный интерфейс для работы с БД в Python. PEP 249: основа всего. Жизненный цикл запроса: connect() -> cursor() -> execute() -> fetchone()/fetchall() -> close(). Обработка исключений (стандартная иерархия Error, Warning). Контекстные менеджеры для автоматического закрытия соединений и курсоров. 6. SQLite: встроенная БД, модуль sqlite3, когда подходит 7. MS SQL Server: драйвер pyodbc и не только. Подключение к MS SQL Server (локальному и облачному Azure SQL) с помощью pyodbc. Настройка connection string: Integrated Security (Windows Auth) vs SQL Auth. Альтернатива: pymssql (исторически), современный sqlalchemy с драйвером pyodbc. 8. PostgreSQL: psycopg2/psycopg3, подключение, пулы соединений. Пул соединений: от наивного до psycopg_pool (встроенный), и внешние типа PgBouncer. 9. Облачные базы данных (Cloud Free Tiers). SQL: Supabase (PostgreSQL); Neon (Serverless PostgreSQL); PlanetScale (MySQL). NoSQL: MongoDB Atlas; Redis Cloud (Upstash). 10. SQL-инъекции: как возникают, параметризованные запросы. Разбор механики атаки (Union-Based, Boolean-Blind). Второй порядок инъекций (Second-Order SQL Injection). Защита: только параметризованные запросы и использование sql.Identifier с валидацией по белому списку для динамических имён. Примеры не параметризованных уязвимостей (f-strings, %-форматирование, конкатенация строк). Что нельзя параметризовать (имена таблиц, столбцов, LIMIT/OFFSET) и как быть в этих случаях (валидация по белому списку, sql.Identifier). 11. ORM: SQLAlchemy Core vs ORM, сессии, модели, миграции (Alembic) 12. SQLAlchemy 2.0: новый стиль (select(), Session), сравнение с Entity Framework из C#. 13. Паттерны: Repository, Unit of Work в контексте Python. Repository: абстракция над слоем хранения данных (интерфейс зависит от домена, реализация от БД). Unit of Work: атомарность операций над несколькими репозиториями. 14. Асинхронная работа с БД: asyncpg, SQLAlchemy async 15. NoSQL кратко: Redis (кэширование), MongoDB	ОПК-3 ОПК-7

		(PyMongo) – когда выбирать 16. Миграции БД: Alembic, автогенерация миграций 17. Pandas + SQL: <code>pd.read_sql()</code>, <code>to_sql()</code> – мост между анализом данных и БД 18. Тестирование взаимодействия с БД: фикстуры с тестовой базой, миграции в тестах, изоляция тестов, транзакционный откат. 19. Безопасное управление подключениями: хранение connection strings в GitHub Secrets, интеграция с Vault, аудит безопасности конфигураций.	
Тема 9	HTTP, REST API и веб-скрапинг	1. HTTP, REST API и веб-скрапинг 2. HTTP-протокол: методы, статус-коды, заголовки (повторение с привязкой к Python) 3. Как сделать HTTP-запрос в Python? 4. Библиотека requests: GET, POST, PUT, DELETE, сессии, таймауты, retry. Примеры использования retry с exponential backoff. 5. httpx: async HTTP-клиент, HTTP/2 6. Аутентификация: API keys, Bearer tokens, OAuth2 (концептуально) 7. REST API: потребление внешних API, пагинация, rate limiting 8. Веб-скрапинг: правовые и этические аспекты, robots.txt, статический vs динамический скрапинг 9. BeautifulSoup: парсинг HTML, CSS-селекторы, навигация по DOM 10. Selenium/Playwright: скрапинг SPA, headless-браузеры 11. Обработка ошибок при сетевых запросах: классификация ошибок, таймауты, retry, circuit breaker 12. Работа с API: GraphQL-клиенты, gRPC (Google Remote Procedure Call) 13. GraphQL-безопасность: предотвращение инъекций и ограничение глубины запросов. 14. Безопасность при работе с API и веб-скрапинге	ОПК-3 ОПК-7
Тема 10	Создание REST API с FastAPI	1. Фреймворки Python для веба: Flask, Django, FastAPI – сравнение, когда что выбирать 2. FastAPI: почему стал стандартом для новых проектов 3. Pydantic/Pydantic v2. Валидация данных, модели, сериализация 4. Маршрутизация, path/query параметры, тело запроса 5. Dependency Injection в FastAPI (аналогия с DI в ASP.NET) 6. Middleware: CORS, логирование, обработка ошибок 7. Аутентификация и авторизация: JWT-токены, OAuth2, RBAC 8. Обработка ошибок и логирование 9. Автодокументация: Swagger/OpenAPI (встроена в FastAPI) 10. Архитектура и структура проекта: роутеры, слои (controller/service/repository) 11. Тестирование API: TestClient, фикстуры с тестовой БД, изоляция через переопределение зависимостей: TestClient, pytest + fixtures 12. Мониторинг и логирование 13. Безопасность: хранение секретов, CORS, HTTPS, валидация входных данных, Helmet. 14. Масштабирование. Деплой	ОПК-3 ОПК-7

Тема 11	Асинхронное программирование: asyncio и конкурентность	1. Асинхронное программирование: asyncio и конкурентность 2. Синхронность, многопоточность, асинхронность – три модели (сравнение с async/await в C#) 3. Event Loop: как работает под капотом 4. Корутины: async def, await, отличия от генераторов 5. asyncio: gather, wait, create_task, таймауты 6. Асинхронные контекстные менеджеры и итераторы 7. aiohttp, httpx async – асинхронные HTTP-запросы 8. aiofiles – асинхронная работа с файлами 9. Многопоточность: threading, GIL и его влияние (IO-bound задачи) 10. Многопроцессность: multiprocessing, ProcessPoolExecutor (CPU-bound задачи) 11. concurrent.futures: ThreadPoolExecutor, ProcessPoolExecutor – унифицированный интерфейс 12. Когда что использовать: выбор (async vs threads vs processes)	ОПК-3 ОПК-7
Тема 12	Создание ботов: Telegram и другие мессенджеры, и системы	1. Архитектура мессенджер-ботов 2. Что такое бот: webhook vs polling, жизненный цикл сообщения 3. Платформы и их API: Telegram Bot API, Discord API (discord.py), WhatsApp Business API, VK API, Slack 4. Почему Telegram – доминирующая платформа для ботов в СНГ 5. Telegram Bot API: концепция update, chat_id, типы сообщений, команды (/start, /help). Настройка webhook с проверкой токена и подлинности запросов. 6. Telegram Bot на aiogram 7. FSM (Finite State Machine): машина состояний для многошаговых сценариев (регистрация, опрос, оформление заказа) 8. Клавиатуры: ReplyKeyboardMarkup (постоянные кнопки) и InlineKeyboardMarkup (callback-кнопки под сообщением) 9. Работа с медиа: отправка/получение фото, документов, обработка файлов 10. Интеграция с внешними сервисами: бот как UI для вашего FastAPI (лекция 10) 11. Хранение состояния и данных: подключение PostgreSQL через SQLAlchemy, Redis для кэша и FSM 12. Безопасность: хранение BOT_TOKEN в переменных окружения, защита от спама 13. Деплой бота: Docker, docker-compose, облачные платформы 14. Альтернативы: python-telegram-bot, discord.py, slack-bolt (краткий обзор) 15. Другие платформы для создания ботов 16. Discord.py: боты для Discord, slash-commands, interactions 17. VK API (vk_api/vkbottle): боты для ВКонтакте 18. Кросс-платформенные подходы: абстракция над несколькими мессенджерами 19. Тестирование ботов: эмуляция сообщений, проверка переходов FSM, моки API мессенджера. 20. Практика непрерывного обновления: работа с Git в ветках, pull request, автоматический деплой из main-ветки с помощью GitHub Actions.	ОПК-3 ОПК-7

		21. CI/CD: сборка Docker-образа бота, запуск интеграционных тестов, деплой на staging-окружение. 22. Деплой: polling (для разработки) vs webhook (для продакшена) 23. Мониторинг: логирование, метрики, алертинг	
Тема 13	Тестирование и качество кода	1. Тестирование и качество кода 2. Специализированное ПО и библиотеки для тестирования и анализа кода 3. Пирамида тестирования: unit → integration → e2e 4. pytest: фикстуры, параметризация, маркеры, плагины 5. Мокирование: unittest.mock, patch, MagicMock (аналогия с Moq в C#) 6. Покрытие кода: pytest-cov, что считается хорошим покрытием 7. TDD: Red-Green-Refactor на примере 8. Безопасное тестирование (fuzzing) 9. Property-based тестирование: Hypothesis для поиска краевых случаев и потенциальных уязвимостей. 10. Линтеры и формatters: ruff (замена flake8, isort, black), mypy для статической типизации 11. Pre-commit hooks: автоматизация проверок 12. CI/CD для Python-проекта: GitHub Actions (базовый pipeline) 13. Настройка GitHub Actions: пайплайн lint → type check → test → build, блокировка pull request при ошибках. 14. Контейнеризация тестовой среды: запуск тестов в Docker с production-подобной конфигурацией. 15. Автоматическая публикация отчётов о покрытии и документации в GitHub Pages. 16. Документирование кода: docstrings (Google/NumPy стили), Sphinx	ОПК-3 ОПК-7
Тема 14	Компьютерное зрение	1. Введение в компьютерное зрение: задачи, области применения и тренды 2. Представление изображений: пиксели, каналы (BGR, RGB, Grayscale, HSV), глубина цвета, изображение как NumPy ndarray 3. Библиотеки компьютерного зрения: OpenCV, scikit-image, Pillow, YOLO, MediaPipe, Detectron2 и другие 4. OpenCV: установка, базовая работа с изображениями (imread, imshow, imwrite, работа с пикселями и регионами интереса – ROI) 5. Цветовые пространства и пороговая обработка (cvtColor, threshold, adaptiveThreshold, Otsu, интерактивные трекбары для подбора параметров) 6. Геометрические трансформации (resize, crop, rotate, flip, affine и perspective transform – выравнивание документов) 7. Фильтрация и сглаживание (blur, GaussianBlur, medianBlur, bilateral filter – шумоподавление с сохранением краёв) 8. Морфологические операции (erosion, dilation, opening, closing, морфологические градиенты – очистка бинарных масок) 9. Обнаружение краёв (Sobel, Laplacian, Canny – подготовка к поиску контуров) 10. Поиск и анализ контуров (findContours, drawContours, площадь, периметр, аппроксимация, bounding box, фильтрация по	ОПК-3 ОПК-7

		свойствам) 11. Классические методы детекции объектов (каскады Хаара для лиц, HOG для пешеходов, template matching – когда ML-модель не нужна) 12. Разметка данных и датасеты (форматы COCO, YOLO, Pascal VOC, инструменты – CVAT, Labelling, Roboflow, загрузка готовых датасетов) 13. Работа с видео: VideoCapture (файлы, веб-камера), покадровая обработка, VideoWriter, запись результата 14. Современная детекция объектов: YOLO (Ultralytics) – детекция в реальном времени на изображениях и видео 15. Распознавание жестов, поз и лиц: MediaPipe – быстрый старт и практическое применение 16. OCR: оптическое распознавание текста – Tesseract, pytesseract, предобработка изображений для улучшения распознавания	
Тема 15	Интеграция с LLM: API, prompt engineering и RAG-системы	1. LLM API 2. Ландшафт LLM: OpenAI, Anthropic, Google, DeepSeek, xAI (Grok), open-source и локальные модели (Ollama, LM Studio и др.) 3. Anthropic Claude, Google Gemini, DeepSeek и другие – сравнение API и цен 4. Агрегаторы LLM: OpenRouter, Together AI, Groq – единый доступ, маршрутизация, сравнение стоимости 5. Клиентские библиотеки: openai, anthropic, litellm – унифицированный интерфейс 6. OpenAI API: Chat Completions, модели, токены, pricing 7. Anthropic API (Claude), Google Gemini API – основные отличия 8. Оценка стоимости токенов: input/output pricing, tiktoken, бюджетирование запросов и контроль затрат, автоматическое переключение на более дешёвые модели при превышении лимита, кэширование ответов 9. Параметры генерации: temperature, top_p, max_tokens, stop sequences 10. Streaming ответов: Server-Sent Events, потоковая генерация 11. Обработка ошибок: rate limits, retry, fallback между провайдерами 12. Prompt Engineering: Системные, пользовательские и ассистентские сообщения. Техники: few-shot, chain-of-thought, structured output (JSON mode) 13. Pydantic + LLM: гарантированная структура ответа (Instructor, function calling) 14. Шаблоны промптов: Jinja2, LangChain prompt templates 15. RAG: архитектура (загрузка → чанкинг → эмбединг → векторная БД → retrieval → генерация) 16. Загрузка, чанкинг и эмбединги: PDF/DOCX/HTML/ Markdown, стратегии разбиения, модели эмбедингов 17. Векторные БД и retrieval: ChromaDB, pgvector, Qdrant, hybrid search, re-ranking, LangChain/LlamaIndex 18. Цепочки RAG: формирование контекста, промпт с контекстом, оценка качества 19. Безопасность: защита от prompt injection, безопасное хранение API-ключей в GitHub	ОПК-3 ОПК-7

		Secrets, ротация ключей. Фильтрация пользовательского ввода перед передачей в модель.	
		20. Безопасность и мониторинг AI-приложений	
Тема 16	Паттерны проектирования, автоматизация и DevOps-практики	1. Паттерны проектирования, автоматизация и DevOps-практики 2. SOLID в динамическом языке: особенности применения в Python 3. Порождающие паттерны: Singleton (через модуль), Factory, Builder 4. Структурные: Adapter, Decorator (через Python-декораторы), Facade 5. Поведенческие: Strategy (через функции первого класса), Observer, Command 6. Чистая архитектура: слои, Dependency Injection, структура проекта (src layout, pyproject.toml) 7. Автоматизация и DevOps 8. CLI-утилиты: argparse, click, typer 9. Планировщики: schedule, APScheduler 10. Контейнеризация и оркестрация: Docker, docker-compose, многоступенчатые сборки, health checks. 11. CI/CD: полный пайплайн в GitHub Actions (lint → type check → test → build → deploy), управление релизами и версиями (semver). 12. Автоматизация рутины: Makefile/Invoke для единых команд, pre-commit. 13. Промышленный Git-flow: стратегия ветвления, code review, защита main-ветки. 14. 12-factor app: переменные окружения (pydantic-settings), логи, конфигурация 15. Мониторинг: структурированные логи, health checks. 16. Документирование инфраструктуры и практика «Infrastructure as Code».	ОПК-3 ОПК-7

2.2 Учебно-методическая карта дисциплины

4 семестр

№ недели	Лекции (наименование тем)	Часы	Лабораторные занятия	Часы	Самостоятельная работа, часы	Форма контроля знаний	Баллы (max)
Модуль 1							
1	Тема 1 Введение в Python	2	Лабораторная работа № 1 Настройка окружения и реализация простых программ в Python	2	2	ЗЛР	3
2			Лабораторная работа № 1 Настройка окружения и реализация простых программ в Python	2	1	ЗЛР	3
3	Тема 2 Типы данных, переменные, коллекции и управляющие конструкции	2	Лабораторная работа № 2 Типы данных, коллекции и управляющие конструкции	2	2	ЗЛР	3
4			Лабораторная работа № 2 Типы данных, коллекции и управляющие конструкции	2	1	ЗЛР	3
5	Тема 3 Функции, замыкания, декораторы и генераторы	2	Лабораторная работа № 3 Функции, замыкания, декораторы и генераторы	2	2	ЗЛР	3

6			Лабораторная работа № 3 Функции, замыкания, декораторы и генераторы	2	1	ЗЛР	3
7	Тема 4 ООП в Python	2	Лабораторная работа № 4 Объектно-ориентированное программирование	2	2	ЗЛР	3
8			Лабораторная работа № 4 Объектно-ориентированное программирование	2	1	ЗЛР ТЗ ТКУ	3 6 30
Модуль 2							
9	Тема 5 Обработка ошибок, тестирование и отладка	2	Лабораторная работа № 5 Обработка ошибок, тестирование и отладка	2	2	ЗЛР	3
10			Лабораторная работа № 5 Обработка ошибок, тестирование и отладка	2	1	ЗЛР	3
11	Тема 6 Работа с файлами, форматами данных и сериализация	2	Лабораторная работа № 6 Работа с файлами и форматами данных	2	1	ЗЛР	3
12			Лабораторная работа № 6 Работа с файлами и форматами данных	2	1	ЗЛР	3
13	Тема 7 Обработка данных и визуализация: NumPy, Pandas, Matplotlib/ Seaborn	2	Лабораторная работа № 7 Обработка данных и визуализация	2	1	ЗЛР	3
14			Лабораторная работа № 7 Обработка данных и визуализация	2	1	ЗЛР	3
15	Тема 8 Работа с базами данных	2	Лабораторная работа № 8 Работа с базами данных	2	1	ЗЛР	3
16			Лабораторная работа № 8 Работа с базами данных	2	1	ЗЛР	3
17			Лабораторная работа № 8 Работа с базами данных	2	1	ЗЛР ЗИЗ ТКУ	3 3 30
17						ПА (зачет)	40
	Итого	16		34	22		100

5 семестр

№ недели	Лекции (наименование тем)	Часы	Лабораторные занятия	Часы	Самостоятельная работа, часы	Форма контроля знаний	Баллы (max)
Модуль 1							
1	Тема 9 HTTP, REST API и веб-скрапинг	2	Лабораторная работа № 9 HTTP-клиенты, REST API и веб-скрапинг	2	2	ЗЛР	3
2			Лабораторная работа № 9 HTTP-клиенты, REST API и веб-скрапинг	2	1	ЗЛР	3
3	Тема 10 Создание REST API с FastAPI	2	Лабораторная работа № 10 Создание REST API с FastAPI	2	2	ЗЛР	3
4			Лабораторная работа № 10 Создание REST API с FastAPI	2	1	ЗЛР	3
5	Тема 11 Асинхронное программирование: asyncio и конкурентность	2	Лабораторная работа № 11 Асинхронное программирование	2	2	ЗЛР	3
6			Лабораторная работа № 11 Асинхронное программирование	2	1	ЗЛР	3
7	Тема 12 Создание ботов: Telegram и другие мессенджеры, и системы	2	Лабораторная работа № 12 Разработка ботов	2	2	ЗЛР	3

8			Лабораторная работа № 12 Разработка ботов	2	1	ЗЛР ТЗ ТКУ	3 6 30
Модуль 2							
9	Тема 13 Тестирование и качество кода	2	Лабораторная работа № 13 Компьютерное зрение	2	2	ЗЛР	3
10			Лабораторная работа № 13 Компьютерное зрение	2	1	ЗЛР	3
11	Тема 14 Компьютерное зрение	2	Лабораторная работа № 13 Компьютерное зрение	2	1	ЗЛР	3
12			Лабораторная работа № 13 Компьютерное зрение	2	1	ЗЛР	3
13	Тема 15 Интеграция с LLM: API, prompt engineering и RAG-системы	2	Лабораторная работа № 13 Компьютерное зрение	2	1	ЗЛР	3
14			Лабораторная работа № 14 Работа с LLM	2	1	ЗЛР	3
15	Тема 16 Паттерны проектирования, автоматизация и DevOps-практики	2	Лабораторная работа № 14 Работа с LLM	2	1	ЗЛР	3
16			Лабораторная работа № 14 Работа с LLM	2	1	ЗЛР	3
17			Лабораторная работа № 14 Работа с LLM	2	1	ЗЛР ТЗ ТКУ	3 3 30
1-17	Выполнение курсовой работы				36		
18-20					36	ПА (экзамен)	40
	Итого	16		34	94		100

Принятые обозначения:

ЗЛР – защита лабораторной работы;

ТКУ – текущий контроль успеваемости;

ТЗ – тестовые задания;

ЗИЗ – защита индивидуального задания;

ПА – промежуточная аттестация.

Итоговая оценка определяется как сумма текущего контроля и промежуточной аттестации и соответствует баллам:

Экзамен

Оценка	Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Баллы	87-100	65-86	51-64	0-50

Зачет

Оценка	Зачтено	Не зачтено
Баллы	51-100	0-50

2.3 Требования к курсовому проекту (курсовой работе)

Целями курсовой работы являются:

- углубление, закрепление и систематизация теоретических знаний и практических умений, полученных в процессе обучения;
- оценка степени усвоения студентами основных положений дисциплины;
- приобретение навыков самостоятельной разработки программного обеспечения;
- получение навыков оформления программной документации и использования справочной литературы, ГОСТов при создании программных продуктов.

Содержание курсовой работы включает две части:

- 1) теоретическая часть – обзор по теме проектирования, исследование актуальных вопросов в данной области, постановка задач, обоснование принятого решения;

2) практическая – реализация программной системы и ее отладка.

На выполнение курсовой работы отводится 36 часов.

Разбивка этапов курсовой работы, определение количества минимальных и максимальных баллов за каждый из них производится преподавателем. Примерный перечень этапов выполнения курсовой работы и количества баллов за каждый из них представлен в таблице.

№	Этап выполнения	Минимум	Максимум
1	Составление технического задания	6	10
2	Проектирование программного модуля	6	10
3	Разработка документации программного модуля	9	15
4	Разработка программного модуля	9	15
5	Тестирование программного модуля	6	10
	Итого за выполнение курсового проекта	36	60
	Защита курсового проекта	15	40

Примерная тематика курсовых проектов (работ) представлена в приложении хранится на кафедре.

Итоговая оценка курсовой работы представляет собой сумму баллов за его выполнение и защиту и выставляется в соответствии со шкалой:

Оценка	Отлично	Хорошо	Удовлетворительно	Неудовлетворительно
Баллы	87-100	65-86	51-64	0-50

3 ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ

При изучении дисциплины используется модульно-рейтинговая система оценки знаний студентов. Применение форм и методов проведения занятий при изучении различных тем курса представлено в таблице.

№ п/п	Форма проведения занятия	Вид аудиторных занятий		Всего часов
		Лекции	Лабораторные занятия	
1	Мультимедиа	Темы 1–16		32
2	С использованием ЭВМ		Лаб. 1–14	68
	ИТОГО	32	68	100

4 ОЦЕНОЧНЫЕ СРЕДСТВА

Используемые оценочные средства по дисциплине представлены в таблице и хранятся на кафедре.

№ п/п	Вид оценочных средств	Количество комплектов
1	Вопросы к экзамену	1
2	Экзаменационные билеты	1
3	Вопросы к зачету	1
4	Вопросы к лабораторным работам	14
5	Тематики курсовой работы	1
6	Тестовые задания	1
7	Индивидуальные задания	1

5 КРИТЕРИИ СФОРМИРОВАННОСТИ КОМПЕТЕНЦИЙ

5.1 Формируемые компетенции и индикаторы их достижения

ОПК-3 Способен решать стандартные задачи профессиональной деятельности на основе информационной и библиографической культуры с применением информационно-коммуникационных технологий и с учетом основных требований информационной безопасности
ИОПК-3.1. Способен применять принципы, методы и средства решения прикладных задач профессиональной деятельности с учетом основных требований информационной безопасности
Знает современные стандарты безопасного кодирования, методы защиты от типовых уязвимостей, принципы безопасного взаимодействия с API, библиотеки и механизмы защиты данных при передаче и хранении, а также нормативные требования и лучшие практики обеспечения информационной безопасности на всех этапах жизненного цикла программного продукта – от написания кода до деплоя и мониторинга.
Умеет применять статический анализ и ручное ревью для выявления потенциальных уязвимостей в коде на Python; реализовывать принцип минимальных привилегий, безопасное хранение секретов, документировать найденные уязвимости и предлагать способы их устранения; настраивать межсетевые экраны приложений, валидацию и санитизацию входных данных, а также корректно обрабатывать ошибки без раскрытия чувствительной информации.
Владеет навыками безопасного использования сторонних зависимостей, настройки защищённого сетевого взаимодействия и контейнеризации с учётом требований ИБ; методикой внедрения security-чеклистов в CI/CD-пайплайны и автоматизированного тестирования безопасности; опытом разработки и развёртывания приложений, устойчивых к типовым атакам, с системой логирования и алертинга инцидентов.
ОПК-7 Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой
ИОПК-7.2. Способен применять в практической деятельности основные концепции и принципы написания программного кода
Знает фундаментальные концепции Python (объектно-ориентированную и функциональную парадигмы, динамическую типизацию, управление памятью, модель исключений, декораторы и контекстные менеджеры), основные структуры данных и алгоритмы, их алгоритмическую сложность, библиотеки для работы с коллекциями, итераторами и генераторами, а также паттерны проектирования (Singleton, Factory, Strategy, Repository и др.), стандарты оформления кода (PEP 8) и современные инструменты поддержки качества.
Умеет выбирать оптимальные структуры данных и алгоритмические решения для конкретной прикладной задачи, рефакторить код в соответствии с принципами чистого кода и SOLID; проектировать и реализовывать многослойные программные системы с использованием Dependency Injection, ORM, асинхронных и многопоточных подходов; писать модульные, интеграционные и end-to-end тесты (pytest), измерять покрытие кода и интерпретировать его результаты.
Владеет инструментарием промышленной разработки, навыками написания технической документации к программным продуктам и ведения проектной документации в соответствии с ГОСТ, а также методикой непрерывной интеграции и развёртывания (GitHub Actions), контейнеризации (Docker) и автоматизации рутинных операций разработки.

5.2 Критерии оценки лабораторных работ

Студент обязан самостоятельно в полном объеме выполнить лабораторные работы согласно рабочей программе.

Задание на работы выдает ведущий занятия преподаватель.

По результатам выполнения работ студент обязан оформить отчет по лабораторной работе в соответствии с действующими в Университете требованиями по оформлению отчета.

Отсутствие отчета является причиной недопуска к сдаче лабораторной работы.

Защита отчета проводится устно, путем ответов на контрольные вопросы к работе и демонстрации навыков, полученных при выполнении работы. Вопросы к лабораторным работам и критерии их оценки представлены в фонде оценочных средств по данной дисциплине.

При защите лабораторной работы студент имеет право пользоваться самостоятельно подготовленным отчетом.

Каждая выполненная и защищенная лабораторная работа оценивается 3 балла. При этом 1 балла начисляется за выполнение работы и от 1 до 2 баллов за оформление отчета и защиту работы в зависимости от качества оформления и уровня знаний студента по тематике работы.

При отсутствии ответов на заданные преподавателем вопросы отчет не засчитывается, и баллы не выставляются.

5.3 Критерии оценки зачета

Для оценки зачета используется следующее оценочное средство: выполнение индивидуального задания.

Индивидуальное задание выдает ведущий занятия преподаватель.

Студент обязан самостоятельно в полном объеме выполнять индивидуальное задание.

По результатам выполнения задания студент обязан оформить отчет в соответствии с действующими в Университете требованиями по оформлению отчета.

Отсутствие отчета является причиной недопуска к сдаче зачета.

Защита индивидуального задания проводится устно. При защите индивидуального задания студент имеет право пользоваться самостоятельно подготовленным отчетом и программой.

Выполненное и защищенное индивидуальное задание оценивается в 3 балла. При этом 1 балл начисляется за выполнение работы и от 1 до 2 баллов за оформление отчета и защиту работы в зависимости от качества оформления и уровня знаний студента по тематике работы.

Подробно средства оценивания приведены в фонде оценочных средств по данной дисциплине.

5.4 Критерии оценки тестовых заданий

Выполнение тестовых заданий оценивается в диапазоне до 6 баллов в зависимости от уровня знаний студентов по тематике тестовых заданий. Если по окончании модуля тестовые задания не выполнены, то баллы по ним не начисляются, и она попадает в разряд задолженности.

5.5 Критерии оценки экзамена

Для оценки экзамена используются следующие оценочные средства:

- Вопросы к экзамену
- Экзаменационные билеты

Экзаменационный билет включает три теоретических вопроса. Содержание вопросов, выносимых на экзамен, соответствуют тематикам, рассмотренным на лекциях и тематикам лабораторных работ, выполненных в течение семестра.

К сдаче экзамена допускаются студенты, получив за семестр в сумме не менее 36 баллов. На экзамене студент может набрать от 0 до 40 баллов. Студенты сдают экзамен в устной форме. Количество баллов, набранных студентом, рассчитывается как сумма баллов, полученных за ответы на три вопроса экзаменационного билета. Ответ на первый теоретический вопрос (от 0 до 10 баллов); на второй теоретический вопрос (от 0 до 10 баллов); на третий теоретический вопрос (от 0 до 10 баллов); ответы на дополнительные вопросы в рамках трех теоретических вопросов (от 0 до 10 баллов).

Оценка ответа на теоретический вопрос

10 баллов:

- систематизированные, глубокие и полные знания по всем разделам учебной программы, а также по основным вопросам, выходящим за ее пределы;
- точное использование научной терминологии (в том числе – на иностранном языке), стилистически грамотное, логически правильное изложение ответа на вопросы;

- безупречное владение инструментарием учебной дисциплины, умение его эффективно использовать в постановке и решении научных и профессиональных задач;
- выраженная способность самостоятельно и творчески решать сложные проблемы в нестандартной ситуации;
- полное и глубокое усвоение основной и дополнительной литературы, рекомендованной учебной программой дисциплины;
- умение ориентироваться в теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку, использовать научные достижения других дисциплин.

9 баллов:

- систематизированные, глубокие и полные знания по всем разделам учебной программы;
- точное использование научной терминологии (в том числе на иностранном языке), стилистически грамотное, логически правильное изложение ответа на вопросы;
- владение инструментарием учебной дисциплины, умение его эффективно использовать в постановке и решении научных и профессиональных задач;
- способность самостоятельно и творчески решать сложные проблемы в нестандартной ситуации в рамках учебной программы;
- полное усвоение основной и дополнительной литературы, рекомендованной учебной программой дисциплины;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку.

8 баллов:

- систематизированные, глубокие и полные знания по всем поставленным вопросам в объеме учебной программы;
- использование научной терминологии, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;
- владение инструментарием учебной дисциплины (методами комплексного анализа, техникой информационных технологий), умение его использовать в постановке и решении научных и профессиональных задач;
- способность самостоятельно решать сложные проблемы в рамках учебной программы;
- усвоение основной и дополнительной литературы, рекомендованной учебной программой;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку.

7 баллов:

- систематизированные, глубокие и полные знания по всем разделам учебной программы;
- использование научной терминологии (в том числе на иностранном языке), лингвистически и логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;
- владение инструментарием учебной дисциплины, умение его использовать в постановке и решении научных и профессиональных задач;
- усвоение основной и дополнительной литературы, рекомендованной учебной программой дисциплины;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им критическую оценку.

6 баллов:

- достаточно полные и систематизированные знания в объеме учебной программы;
- использование необходимой научной терминологии, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать обоснованные выводы;

- владение инструментарием учебной дисциплины, умение его использовать в решении учебных и профессиональных задач;
- способность самостоятельно применять типовые решения в рамках учебной программы;
- усвоение основной литературы, рекомендованной учебной программой дисциплины;
- умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку.

5 баллов:

- достаточные знания в объеме учебной программы;
- использование научной терминологии, стилистически грамотное, логически правильное изложение ответа на вопросы, умение делать выводы;
- владение инструментарием учебной дисциплины, умение его использовать в решении учебных и профессиональных задач;
- способность самостоятельно применять типовые решения в рамках учебной программы;
- усвоение основной литературы, рекомендованной учебной программой дисциплины;
- умение ориентироваться в базовых теориях, концепциях и направлениях по изучаемой дисциплине и давать им сравнительную оценку.

4 балла:

- достаточный объем знаний в рамках образовательного стандарта;
- усвоение основной литературы, рекомендованной учебной программой дисциплины;
- использование научной терминологии, стилистическое и логическое изложение ответа на вопросы, умение делать выводы без существенных ошибок;
- владение инструментарием учебной дисциплины, умение его использовать в решении стандартных (типовых) задач;
- умение под руководством преподавателя решать стандартные (типовые) задачи;
- умение ориентироваться в основных теориях, концепциях и направлениях по изучаемой дисциплине и давать им оценку.

3 балла:

- недостаточно полный объем знаний в рамках образовательного стандарта;
- знание части основной литературы, рекомендованной учебной программой дисциплины;
- использование научной терминологии, изложение ответа на вопросы с существенными лингвистическими и логическими ошибками;
- слабое владение инструментарием учебной дисциплины, некомпетентность в решении стандартных (типовых) задач;
- неумение ориентироваться в основных теориях, концепциях и направлениях изучаемой дисциплины.

2 балла:

- фрагментарные знания в рамках образовательного стандарта;
- знания отдельных литературных источников, рекомендованных учебной программой дисциплины;
- неумение использовать научную терминологию дисциплины, наличие в ответе грубых стилистических и логических ошибок.

1 балл:

- Отсутствие знаний и компетенций в рамках образовательного стандарта или отказ от ответа.

6 МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ И ВЫПОЛНЕНИЮ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

Самостоятельная работа обучающихся (СР) направлена на закрепление и углубление освоения учебного материала, развитие практических умений. СР включает следующие виды:

- работа с материалами курса, вынесенными на самостоятельное изучение;
- работа со справочной литературой;

Перечень контрольных вопросов и заданий для СР приведен в приложении и хранится на кафедре.

Для СР рекомендуется использовать источники, приведенные в п. 7.

7 УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

7.1 Основная литература

№ п/п	Библиографическое описание	Гриф	Количество экземпляров/ URL ссылка
1	Чернышев, С. А. Основы программирования на Python : учебник для вузов / С. А. Чернышев. – 2-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2026. – 349 с.	Рекомендовано Учебно-методическим отделом высшего образования в качестве учебника для студентов высших учебных заведений, обучающихся по ИТ направлениям	-/ https://urait.ru/bcode/588667
2	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, 2026. – 241 с.	Рекомендовано Учебно-методическим отделом высшего образования в качестве учебника для студентов высших учебных заведений, обучающихся по инженерно-техническим направлениям	-/ https://urait.ru/bcode/583784

7.2 Дополнительная литература

№ п/п	Библиографическое описание	Гриф	Количество экземпляров/ URL ссылка
1	Сацюк, А. В. Компьютерное зрение и нейронные сети. Практика : учебное пособие / А. В. Сацюк. – Москва ; Вологда : Инфра-Инженерия, 2025. – 364 с. .	–	-/ https://znanium.ru/catalog/product/2225332
2	Федоров, Д. Ю. Программирование на Python : учебное пособие для вузов / Д. Ю. Федоров. – 6-е изд., перераб. и доп. – Москва : Издательство Юрайт, 2026. – 187 с.	Рекомендовано Учебно-методическим отделом высшего образования в качестве учебного пособия для студентов высших учебных заведений, обучающихся по инженерно-техническим направлениям	-/ https://urait.ru/bcode/585806

7.3 Перечень ресурсов сети Интернет по изучаемой дисциплине

<http://moodle.bru.by> – Образовательный портал Белорусско-Российского университета;

<http://e.biblio.bru.by/> – Электронная библиотека Белорусско-Российского университета;

<https://znanium.com/> – Электронно-библиотечная система Znanium;

<https://urait.ru> – Образовательная платформа Юрайт;

<https://stepik.org/catalog> – Российская образовательная платформа и конструктор бесплатных открытых онлайн-курсов и уроков;

<https://openedu.ru> – Открытое образование. Курсы ведущих вузов России;
<https://habr.com/ru/> – Хабр. Публикации по ИТ тематикам;
<https://www.kaggle.com/> – онлайн-сообщество специалистов по обработке данных и практиков машинного обучения;
<https://pypi.org/> – репозиторий пакетов Python;
<https://github.com/> – платформа для хостинга кода и контроля версий;
<https://stackoverflow.com/> – сообщество разработчиков;
<https://pyfound.blogspot.com> – официальный блог Python Software Foundation: анонсы релизов, инициативы сообщества, гранты
<https://docs.opencv.org/> – документация OpenCV (алгоритмы обработки изображений, примеры кода на Python/C++);
<https://docs.ultralytics.com/> – документация Ultralytics YOLO (инструкции по обучению, инференсу и экспорту моделей);
<https://github.com/topics/computer-vision> – подборка открытых проектов по компьютерному зрению на GitHub;
<https://pyimagesearch.com/> – блог PyImageSearch (практические руководства по OpenCV, Deep Learning, YOLO);
<https://learnopencv.com/> – ресурс LearnOpenCV (тutorials, примеры кода, разбор архитектур нейросетей).
<https://trendshift.io> – аналитика трендов на GitHub: рост популярности репозитория, новые звёзды, активность контрибьюторов

7.4 Перечень наглядных и других пособий, методических рекомендаций по проведению учебных занятий, а также методических материалов к используемым в образовательном процессе техническим средствам

Методические рекомендации

Решение прикладных задач на языке Python. Методические рекомендации к лабораторным работам для студентов направления подготовки 09.03.04 «Программная инженерия» дневной формы обучения / В. В. Кутузов. – Могилев: Белорусско-Российский университет (электронный вариант)

Информационные технологии

Мультимедийные презентации по лекционному курсу.

Тема 1 Введение в Python

Тема 2 Типы данных, переменные, коллекции и управляющие конструкции

Тема 3 Функции, замыкания, декораторы и генераторы

Тема 4 ООП в Python

Тема 5 Обработка ошибок, тестирование и отладка

Тема 6 Работа с файлами, форматами данных и сериализация

Тема 7 Обработка данных и визуализация: NumPy, Pandas, Matplotlib/ Seaborn

Тема 8 Работа с базами данных

Тема 9 HTTP, REST API и веб-скрапинг

Тема 10 Создание REST API с FastAPI

Тема 11 Асинхронное программирование: asyncio и конкурентность

Тема 12 Создание ботов: Telegram и другие мессенджеры, и системы

Тема 13 Тестирование и качество кода

Тема 14 Компьютерное зрение

Тема 15 Интеграция с LLM: API, prompt engineering и RAG-системы

Тема 16 Паттерны проектирования, автоматизация и DevOps-практики

Перечень программного обеспечения, используемого в образовательном процессе

Категория	ПО	Описание	Лицензия / Цена
Язык и среда выполнения	Python 3.10–3.14	Язык программирования и стандартная среда выполнения.	Open Source (PSF)
Среды разработки и интерактивные инструменты	VS Code	Редактор кода с расширениями для Python, отладки и AI-ассистентов.	Open Source (MIT)
	PyCharm Community	IDE от JetBrains с глубокой поддержкой Python, отладкой и рефакторингом.	Open Source (Apache 2.0)
	Google Colab	Бесплатная облачная среда Jupyter Notebook с GPU для экспериментов.	Freemium
	Jupyter Notebook / JupyterLab	Интерактивная среда для анализа данных, визуализации и прототипирования.	Open Source (BSD)
Управление окружением и зависимостями	pyenv	Управление несколькими версиями Python на одном компьютере.	Open Source (MIT)
	Poetry	Инструмент для управления зависимостями, виртуальными окружениями и сборки пакетов.	Open Source (MIT)
	pipx	Установка и запуск Python-приложений в изолированных окружениях.	Open Source (MIT)
	venv / virtualenv	Виртуальные окружения для изоляции зависимостей	Python Software Foundation / MIT
Контроль версий и CI/CD	Git	Распределённая система контроля версий.	Open Source (GPLv2)
	GitHub Actions	Платформа CI/CD для автоматизации сборки, тестирования и деплоя.	Freemium
	Pre-commit	Фреймворк для управления git hooks и автоматизации проверок до коммита.	Open Source (MIT)
Веб-фреймворки и серверы	FastAPI	Современный высокопроизводительный веб-фреймворк для создания REST API.	Open Source (MIT)
	Uvicorn	ASGI-веб-сервер для запуска приложений на FastAPI с поддержкой асинхронности.	Open Source (BSD)
Данные и визуализация	NumPy	Фундаментальная библиотека для многомерных массивов и линейной алгебры.	Open Source (BSD)
	Pandas	Библиотека для обработки, анализа и визуализации табличных данных.	Open Source (BSD)
	Pydantic v2	Валидация, сериализация и описание моделей данных на основе type hints.	Open Source (MIT)
	Matplotlib	Базовая библиотека для построения графиков	PSF-compatible
	Seaborn	Статистическая визуализация на основе Matplotlib	BSD 3-Clause
	Plotly	Интерактивная визуализация и дашборды	MIT
Базы данных, кэширование и очереди	SQLite	Встраиваемая реляционная БД, не требующая отдельного сервера.	Открытое достояние (Public Domain)
	PostgreSQL	Мощная объектно-реляционная СУБД с открытым исходным кодом.	Open Source (PostgreSQL)
	psycopg2 / psycopg3	Драйвер для работы с PostgreSQL	LGPL / BSD
	pyodbc	Драйвер ODBC для MS SQL Server и других СУБД	MIT
	asyncpg	Асинхронный драйвер для PostgreSQL	Apache 2.0

	Redis	Хранилище структур данных в памяти, используемое как БД, кэш и брокер сообщений.	Open Source (BSD)
	SQLAlchemy 2.0	ORM и инструмент для работы с реляционными БД из Python.	Open Source (MIT)
	PyMongo	Официальный драйвер для MongoDB	Apache 2.0
	Alembic	Фреймворк для миграций схем баз данных, интегрированный с SQLAlchemy.	Open Source (MIT)
	ChromaDB / Qdrant / pgvector	Векторные базы данных для RAG-приложений	Apache 2.0 / MIT
Тестирование и качество кода	pytest	Фреймворк для модульного, интеграционного и функционального тестирования.	Open Source (MIT)
	pytest-asyncio	Поддержка асинхронных тестов в pytest	Apache 2.0
	pytest-cov	Плагин для измерения покрытия кода тестами.	Open Source (MIT)
	unittest.mock / MagicMock	Мокирование и изоляция зависимостей в тестах	Python Software Foundation
	mypy	Статический анализатор типов для Python с поддержкой type hints.	Open Source (MIT)
	ruff	Быстрый линтер и форматер (замена flake8, isort, black)	Open Source (MIT)
	black	Бескомпромиссный автоформаттер кода Python, гарантирующий единообразие стиля.	Open Source (MIT)
	Hypothesis	Property-based тестирование	MPL 2.0
	isort	Сортировка импортов	MIT
	pre-commit	Фреймворк для управления git pre-commit хуками	MIT
Асинхронность, потоки и процессы	httpx	HTTP-клиент с поддержкой синхронного и асинхронного API и протокола HTTP/2.	Open Source (BSD)
	aiohttp	Асинхронный HTTP-клиент/сервер, построенный на asyncio.	Open Source (Apache 2.0)
	aiofiles	Асинхронная работа с файловой системой без блокировки event loop.	Open Source (Apache 2.0)
Веб-скрапинг и автоматизация браузеров	Beautiful Soup	Библиотека для извлечения данных из HTML и XML файлов.	Open Source (MIT)
	Playwright	Инструмент для автоматизации браузеров и скрапинга динамических сайтов.	Open Source (Apache 2.0)
	Selenium	Инструмент для автоматизации действий веб-браузера и тестирования.	Open Source (Apache 2.0)
Боты и мессенджеры	aiogram	Асинхронный фреймворк для Telegram-ботов	MIT
	python-telegram-bot	Альтернативная библиотека для Telegram	LGPLv3
	discord.py / vkbottle / slack-bolt	Библиотеки для Discord, VK, Slack соответственно	MIT / Apache 2.0
Компьютерное зрение и OCR	OpenCV	Библиотека алгоритмов компьютерного зрения, обработки изображений и видео.	Open Source (Apache 2.0)
	Pillow	Форк PIL для открытия, изменения и сохранения изображений в разных форматах.	Open Source (BSD)
	YOLO (Ultralytics)	Модели и фреймворк для детекции, сегментации и классификации объектов.	AGPL-3.0 / Enterprise

	Labelbox	Альтернативные платформы разметки	бесплатный образовательный тариф
	VGG Image Annotator (VIA)	VIA – легковесный браузерный инструмент	Apache 2.0
	MediaPipe	Фреймворк для построения ML-пайплайнов для работы с жестами, позами и лицом.	Open Source (Apache 2.0)
	Tesseract OCR	Движок оптического распознавания текста от Google.	Open Source (Apache 2.0)
	pytesseract	Python-обёртка для движка Tesseract OCR.	Open Source (Apache 2.0)
	CVAT / LabelImg / Roboflow	Инструменты разметки данных для CV	MIT / GPL-3.0 / Proprietary (Roboflow)
Интеграция с LLM и AI	OpenAI / Anthropic / Gemini SDK	Клиентские библиотеки для взаимодействия с API коммерческих LLM.	MIT / Бесплатно
	litellm	Унифицированный интерфейс для вызова 100+ LLM от разных провайдеров.	Open Source (MIT)
	Instructor	Библиотека для получения структурированных ответов (Pydantic-модели) от LLM.	Open Source (MIT)
	LangChain	Фреймворк для разработки приложений на базе LLM с цепочками и агентами.	Open Source (MIT)
	LlamaIndex	Фреймворк для построения RAG-систем с индексацией и поиском по данным.	Open Source (MIT)
	ChromaDB	Векторная база данных с открытым исходным кодом для построения AI-приложений.	Open Source (Apache 2.0)
	tiktoken	Токенизация для моделей OpenAI	MIT
	Jinja2	Шаблонизатор для промптов и конфигураций	BSD 3-Clause
Контейнеризация и деплой	Docker	Платформа для разработки и запуска приложений в изолированных контейнерах.	Open Source (Apache 2.0)
	Docker Compose	Инструмент для описания и запуска многоконтейнерных Docker-приложений.	Open Source (Apache 2.0)
	APScheduler / schedule	Планировщики задач для фоновых процессов	MIT
	typer / click / argparse	Создание CLI-интерфейсов	MIT / BSD / Python stdlib
	structlog	Структурированное логирование	MIT / Apache 2.0
Профилирование и отладка	cProfile	Модуль стандартной библиотеки для профилирования производительности кода.	Встроен в Python
	pdb	Интерактивный отладчик исходного кода Python.	Встроен в Python
Безопасность и аутентификация	python-jose / PyJWT	Работа с JWT-токенами	MIT / BSD
	passlib / bcrypt	Хэширование и проверка паролей	BSD / Apache 2.0
Утилиты и вспомогательные библиотеки	pathlib / os / sys / json / csv	Стандартная библиотека Python для работы с файлами и данными	PSF License
	collections / itertools / functools	Расширенные структуры данных и функциональные утилиты	PSF License
	contextlib / logging / pdb	Контекстные менеджеры, логирование, отладчик	PSF License
	cProfile / timeit / line_profiler	Профилирование производительности	PSF License / BSD
	openpyxl	Работа с файлами Excel (.xlsx)	MIT

	PyYAML / tomllib	Парсинг YAML и TOML-конфигураций	MIT / PSF (tomllib с 3.11)
--	------------------	----------------------------------	----------------------------

8 МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Материально-техническое обеспечение дисциплины содержится в паспорте лаборатории а. 517/2, рег. № паспорта лаборатории № ПУЛ - 4 517/2- 25; в паспорте лаборатории а. 518/2, рег. № паспорта лаборатории № ПУЛ - 4 518/2- 25; в паспорте лаборатории а. 502/2, рег. № паспорта лаборатории № ПУЛ - 4 502/2- 25.