МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Астраханский государственный университет имени В. Н. Татищева»
(Астраханский государственный университет им. В. Н. Татищева)

СОГЛАСОВАНО
Руководитель ОПОП
_____ Е.Ю. Степанович

«13» июня 2024 г.

УТВЕРЖДАЮ
Заведующий кафедрой ИТ
_____ А. Н. Марьенков

«13» июня 2024 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Программирование на языке высокого уровня

Составитель(и)	**Синельщиков А.В., доцент кафедры ИТ**
Направление подготовки / специальность	**15.03.06 Мехатроника и робототехника**
Направленность (профиль) ОПОП	**Промышленная робототехника**
Квалификация (степень)	**Бакалавр**
Форма обучения	**очная**
Год приёма	**2023**
Курс	**2**
Семестр(ы)	**3**

Астрахань – 2024

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

1.1. Целями освоения дисциплины (модуля) «Программирование на языке высокого уровня» являются формирование теоретических знаний и практических навыков по использованию современных персональных компьютеров и программных средств для решения широкого спектра задач в различных областях.

1.2. Задачи освоения дисциплины (модуля): «Программирование на языке высокого уровня»

- ознакомить студентов с основами теории программирования;
- привить навыки работы с различными языками программирования для создания прикладных программ;
- изложить основные принципы организации современного программного обеспечения.

2. МЕСТО ДИСЦИПЛИНЫ (МОДУЛЯ) В СТРУКТУРЕ ОПОП

2.1. Учебная дисциплина (модуль) «Программирование на языке высокого уровня» относится к части, формируемой участниками образовательных отношений и осваивается в 3 семестре.

2.2. Для изучения данной учебной дисциплины (модуля) необходимы следующие знания, умения, навыки, формируемые предшествующими учебными дисциплинами (модулями):

- цифровая грамотность;
- введение в информационные технологии;
- высшая математика.

Знания:

- знание основных структур данных, понимание того, как организовывать и хранить данные (массивы, списки, словари, деревья и т.д.) для эффективной обработки;
- принципов объектно-ориентированного программирования (ООП)

Умения:

- способность разбить задачу на последовательность логических шагов, которые компьютер может выполнить.
- использовать готовые функции и модули для решения распространенных задач
- строить логические цепочки и анализировать сложные ситуации.

Навыки:

- поиска и исправления ошибок в коде, а также проверка его работоспособности в различных ситуациях
- работы с инструментами разработки, использование IDE (интегрированных сред разработки)

2.3. Последующие учебные дисциплины (модули) и (или) практики, для которых необходимы знания, умения, навыки, формируемые данной учебной дисциплиной (модулем):

- Программное обеспечение мехатронных и робототехнических систем;
- Информационные технологии в профессиональной деятельности;
- Микропроцессорная техника в мехатронике и робототехнике;
- Управление роботами и робототехническими системами.

3. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Процесс освоения дисциплины (модуля) направлен на формирование элементов следующей(их) компетенции(ий) в соответствии с ФГОС ВО и ОПОП ВО по данному направлению подготовки / специальности:

а) универсальной(ых) (УК);

УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач.

УК-2. Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений.

Таблица 1 – Декомпозиция результатов обучения

Код и наименование компетенции	Планируемые результаты обучения по дисциплине (модулю)		
	Знать (1)	Уметь (2)	Владеть (3)
УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	ИУК-1.1.1 - знать методики поиска, сбора и обработки информации ИУК-1.1.2 - знать актуальные российские и зарубежные источники информации в сфере профессиональной деятельности ИУК-1.1.3 - знать метод системного анализа	УК-1.2.1 - уметь применять методики поиска, сбора и обработки информации УК-1.2.2 - уметь осуществлять критический анализ и синтез информации, полученной из разных источников УК-1.2.3 - уметь применять системный подход для решения поставленных задач	УК-1.3.1 - владеть методами поиска, сбора и обработки, критического анализа и синтеза информации УК-1.3.2 - владеть методикой системного подхода для решения поставленных задач
УК-2. Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений	УК-2.1.1 - знать виды ресурсов и ограничений для решения профессиональных задач УК-2.1.2 - знать основные методы оценки разных способов решения задач УК-2.1.3 - знать действующее законодательство и правовые нормы, регулирующие профессиональную деятельность	УК-2.2.1 - уметь проводить анализ поставленной цели и формулировать задачи, которые необходимо решить для ее достижения УК-2.2.2 - анализировать альтернативные варианты для достижения намеченных результатов УК-2.2.3 - уметь использовать нормативно-правовую документацию в сфере	УК-2.3.1 - владеть методиками разработки цели и задач проекта УК-2.3.2 - владеть методами оценки потребности в ресурсах, продолжительности и стоимости проекта УК-2.3.3 - владеть навыками работы с нормативно-правовой документацией

Код и наименование компетенции	Планируемые результаты обучения по дисциплине (модулю)		
	Знать (1)	Уметь (2)	Владеть (3)
		профессиональной деятельности.	

4. СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Объём дисциплины (модуля) составляет 4 зачётных единиц, в том числе 144 часов, выделенных на контактную работу обучающихся с преподавателем (из них 18 часов – лекции, 18 часов – лабораторные работы), и 108 часов – на самостоятельную работу обучающихся.

Таблица 2 – Структура и содержание дисциплины (модуля)

Раздел, тема дисциплины (модуля)	Семестр	Контактная работа (в часах)			Самост. Работа		Форма текущего контроля успеваемости, форма промежуточной аттестации
		Л	ПЗ	ЛР	КР	СР	
Тема 1: Основы программирования на Python	3	6		6		28	Выполнение лабораторных работ, вопросы к диф.зачету
Тема 2: Структурирование и организация кода		4		4		26	Выполнение лабораторных работ, вопросы к диф.зачету
Тема 3: Объектно-ориентированное программирование		4		4		28	Выполнение лабораторных работ, вопросы к диф.зачету
Тема 4: Прикладное программирование в мехатронике и робототехнике		4		4		26	Выполнение лабораторных работ, вопросы к диф.зачету
Итого		**18**		**18**		**108**	**Диф. зачёт (зачёт с оценкой)**

Примечание: Л – лекция; ПЗ – практическое занятие, семинар; ЛР – лабораторная работа; КР – курсовая работа; СР – самостоятельная работа.

Краткое содержание каждой темы дисциплины (модуля)

Тема 1: Основы программирования на Python

Введение в программирование и язык Python, Понятие программирования и его применение в мехатронике и робототехнике, Обзор языка Python и его особенностей, Базовые типы данных и операторы, Числовые, строковые и логические типы данных, Арифметические, логические и операторы сравнения, Управление потоком выполнения, Условные операторы (if-else, elif), Циклы (for, while)

Тема 2: Структурирование и организация кода

Функции и модули, Создание и вызов функций, Использование модулей для организации кода, Работа с файлами, Чтение и запись данных в файлы, Использование библиотек для обработки файлов

Тема 3: Объектно-ориентированное программирование

Объектно-ориентированное программирование, Классы, объекты и методы, Наследование и полиморфизм

Тема 4: Прикладное программирование в мехатронике и робототехнике

Прикладное программирование в мехатронике и робототехнике, Управление датчиками и исполнительными механизмами, Обработка сигналов и управление данными, Разработка программ для промышленных роботов

5. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРЕПОДАВАНИЮ И ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЯ)

5.1. Указания для преподавателей по организации и проведению учебных занятий по дисциплине (модулю)

Учебная деятельность студента в процессе изучения строится из контактных форм работы с преподавателем (аудиторные занятия, зачет) и самостоятельной работы.

Для успешного освоения дисциплины является обязательным посещение всех занятий, выполнение самостоятельной работы, которая назначается преподавателем.

Методическая поддержка дисциплины обеспечивается использованием дистанционных технологий. Студентам предлагается информационный ресурс, расположенный по адресу: http://moodle.asu.edu.ru, на сервере дистанционного обучения АГУ. На сервере размещен методический материал по данной дисциплине, в содержание которого входит:

- теоретический материал;
- задания и указания по выполнению лабораторных работ;
- вопросы к дифференциальному зачету.

Аудиторные занятия проводятся на основе теоретического материала, опубликованного на образовательном портале, это позволяет студентам изучить пропущенный материал или самостоятельно разобраться с темой, не освоенной на занятии.

5.2. Указания для обучающихся по освоению дисциплины (модулю)

Самостоятельная работа студентов по дисциплине "Архитектура информационных систем" предполагает:

- изучение обязательных литературных источников;
- поиск и обзор дополнительных источников литературы и электронных источников информации по программе курса;
- самоконтроль изученного учебного материала в виде тестирования;
- подготовка рефератов и их презентаций;
- выполнение лабораторных работ;
- подготовка к дифференциальному зачету.

Таблица 4 – Содержание самостоятельной работы обучающихся

Вопросы, выносимые на самостоятельное изучение	Кол-во часов	Форма работы
Тема 1: Основы программирования на Python 1. Что такое язык программирования Python и каковы его особенности? 2. Как установить и настроить среду разработки Python? 3. Каковы основные типы данных в Python и как их использовать?	28	Устный опрос

Вопросы, выносимые на самостоятельное изучение	Кол-во часов	Форма работы
4. Как использовать операторы в Python для выполнения различных операций? 5. Как управлять потоком выполнения в Python с помощью условных операторов и циклов? 6. Как использовать функции в Python для организации и повторного использования кода? 7. Как работать с файлами в Python для чтения, записи и обработки данных? 8. Как использовать модули в Python для расширения функциональности программ? 9. Каковы основные принципы объектно-ориентированного программирования в Python? **10.** Как использовать библиотеки Python для решения задач в различных областях, таких как обработка данных, машинное обучение и веб-разработка?		
Тема 2: Структурирование и организация кода 1. **Модули и пакеты:** В чем разница между модулем и пакетом в Python? Как импортировать модули и пакеты различными способами (например, import module, from module import name, import package.module)? Приведите примеры. 2. **Пространства имен:** Что такое пространства имен (namespaces) в Python? Какие типы пространств имен существуют (локальное, глобальное, встроенное)? Как они влияют на доступность переменных? 3. **Функции:** Каковы основные принципы создания хороших функций? Что такое аргументы (позиционные, именованные, по умолчанию, *args, **kwargs)? Как использовать аннотации типов для функций? 4. **Область видимости переменных:** Что такое область видимости (scope) переменной? Как переменные, объявленные внутри функции, отличаются от переменных, объявленных вне функции? Что такое ключевое слово global и когда его следует использовать? 5. **Классы и объекты:** Что такое класс и объект в Python? Как создавать классы и объекты? Что такое атрибуты и методы? Как работает self?	26	Устный опрос
Тема 3: Объектно-ориентированное программирование	28	Устный опрос

Вопросы, выносимые на самостоятельное изучение	Кол-во часов	Форма работы
1. **Классы и объекты:** • В чем разница между классом и объектом? • Как создать класс в Python? • Как создать экземпляр (объект) класса? • Что такое конструктор (__init__) и как его использовать? • Как обращаться к атрибутам и методам объекта? 2. **Атрибуты и методы:** • Какие типы атрибутов существуют (атрибуты экземпляра, атрибуты класса)? • Как определить метод экземпляра? • Что такое метод класса (@classmethod) и как его использовать? • Что такое статический метод (@staticmethod) и как его использовать? • В чем разница между методами экземпляра, класса и статическими методами? 3. **Инкапсуляция:** • Что такое инкапсуляция и зачем она нужна? • Как реализовать инкапсуляцию в Python (используя соглашения об именовании _ и __)? • Что такое геттеры и сеттеры (свойства) и как их использовать?		
Тема 4: Прикладное программирование в мехатронике и робототехнике 1. **Взаимодействие с оборудованием:** Какие библиотеки Python используются для взаимодействия с аппаратным обеспечением (например, микроконтроллерами, сенсорами, моторами)? Приведите примеры и опишите их назначение (например, pyserial, RPi.GPIO, smbus). 2. **Работа с сенсорами:** Как считывать данные с различных типов сенсоров (например, датчиков расстояния, температуры, освещенности) с помощью Python? Как обрабатывать и калибровать полученные данные? 3. **Управление моторами:** Как управлять различными типами моторов (например,	26	Устный опрос

Вопросы, выносимые на самостоятельное изучение	Кол-во часов	Форма работы
DC-моторами, сервоприводами, шаговыми двигателями) с помощью Python? Какие методы управления используются (например, ШИМ, управление скоростью и положением)? 4. **Коммуникация:** Какие протоколы коммуникации используются в мехатронике и робототехнике (например, UART, I2C, SPI, CAN)? Как реализовать обмен данными по этим протоколам с помощью Python? 5. **Управление в реальном времени:** Как обеспечить управление в реальном времени с помощью Python? Какие проблемы могут возникнуть и как их решать (например, задержки, точность)?		

5.3. Виды и формы письменных работ, предусмотренных при освоении дисциплины (модуля), выполняемые обучающимися самостоятельно

Отчет по лабораторным работам.

Результатом работы, выполняемой студентами, является электронный отчет по выполнению лабораторно-практической работы, тематика которых представлена в таблице 4,

Электронный отчет представляет собой файл формата Word (PDF, ODT), содержащий диаграммы, схемы и текстовые пояснения. Файл передается на проверку преподавателю путем загрузки на ресурс http://moodle.asu.edu.ru в соответствующий заданию раздел.

Задания к лабораторно-практическим занятиям размещены на образовательном портале http://moodle.asu.edu.ru. Рекомендуется заранее ознакомиться с темой, основными вопросами, рекомендациями, требованиями к представлению отчета и критериями оценивания заданий.

В процессе подготовки к лабораторно-практическим занятиям, необходимо обратить особое внимание на самостоятельное изучение рекомендованной литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной литературой, материалами периодических изданий и Интернета является наиболее эффективным методом получения дополнительных знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала.

6. ОБРАЗОВАТЕЛЬНЫЕ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

6.1. Образовательные технологии

Цели дисциплины достигаются путем сочетания контактной и самостоятельной работы студентов: проведение практических занятий, проведения лабораторных занятий на ПК и организации самостоятельной работы студентов.

Для самостоятельного изучения теоретического материала дисциплины рекомендуется использовать интернет-ресурсы, информационные базы, методические разработки, специальную учебную и научную литературу.

В рамках организации самостоятельной работы студентам рекомендуется:

– дополнительная подготовка к занятиям
– подготовка к текущей и промежуточной аттестации (дифференциальному зачету).

Таблица 5 – Образовательные технологии, используемые при реализации учебных занятий

Раздел, тема дисциплины (модуля)	Форма учебного занятия		
	Лекция	Практическое занятие, семинар	Лабораторная работа
Тема 1: Основы программирования на Python	Классическая лекция	Не предусмотрено	Лабораторная работа 1, 2
Тема 2: Структурирование и организация кода	Классическая лекция	Не предусмотрено	Лабораторная работа 3, 4
Тема 3: Объектно-ориентированное программирование	Классическая лекция	Не предусмотрено	Лабораторная работа 4, 6
Тема 4: Прикладное программирование в мехатронике и робототехнике	Классическая лекция	Не предусмотрено	Лабораторная работа 7, 8

6.2. Информационные технологии

При реализации различных видов учебной и внеучебной работы используются следующие информационные технологии:

- использование образовательного сайта http://moodle.asu.edu.ru, как элемента взаимодействия участников образовательного процесса (технологии дистанционного обучения);
- использование электронных учебников электронных библиотечных систем, доступ к которым предоставляется университетом;
- использование как источников информации сайтов, находящихся в Интернете в открытом доступе (электронные библиотеки, журналы, книги, психологические тесты);
- использование возможностей корпоративной электронной почты (рассылка заданий, материалов, ответы на вопросы).

6.3. Программное обеспечение, современные профессиональные базы данных и информационные справочные системы

6.3.1. Программное обеспечение

Наименование программного обеспечения	**Назначение**
Adobe Reader	Программа для просмотра электронных документов
Платформа дистанционного обучения LMS Moodle	Виртуальная обучающая среда
Mozilla FireFox	Браузер
Google Chrome	Браузер
VirtualBox	Программный продукт виртуализации операционных систем
VMware (Player)	Программный продукт виртуализации операционных систем

6.3.2. Современные профессиональные базы данных и информационные справочные системы

Наименование программного обеспечения	Назначение
PostgreSQL	PostgreSQL Это система управления объектно-реляционными базами данных, то есть можно создавать таблицы, соответствующие принципам объектно-ориентированного программирования (классы, наследование и т. д).

7. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

7.1. Паспорт фонда оценочных средств

При проведении текущего контроля и промежуточной аттестации по дисциплине (модулю) «Программирование на языке высокого уровня» проверяется сформированность у обучающихся компетенций, указанных в разделе 3 настоящей программы. Этапность формирования данных компетенций в процессе освоения образовательной программы определяется последовательным освоением дисциплин (модулей) и прохождением практик, а в процессе освоения дисциплины (модуля) – последовательным достижением результатов освоения содержательно связанных между собой разделов, тем.

Таблица 6 – Соответствие разделов, тем дисциплины (модуля), результатов обучения по дисциплине (модулю) и оценочных средств

Контролируемый раздел, тема дисциплины (модуля)	Код контролируемой компетенции	Наименование оценочного средства
Тема 1: Основы программирования на Python	УК-1	Отчет по ЛПР; вопросы к дифференциальному зачету.
Тема 2: Структурирование и организация кода	УК-1	Отчет по ЛПР; вопросы к дифференциальному зачету.
Тема 3: Объектно-ориентированное программирование	УК-1, УК-2	Отчет по ЛПР; вопросы к дифференциальному зачету.
Тема 4: Прикладное программирование в мехатронике и робототехнике	УК-1, УК-2	Отчет по ЛПР; вопросы к дифференциальному зачету.

Для оценивания результатов обучения в виде **знаний** используются следующие типы контроля:

- индивидуальное собеседование (устный опрос).
- письменные работы (отчеты по ЛПР).

Тестовые задания охватывают содержание всего пройденного материала.
Индивидуальное собеседование ё проводится по разработанным вопросам к диф.зачету.

7.2. Описание показателей и критериев оценивания компетенций, описание шкал оценивания

В системе Moodle балл за выполнение лабораторно-практической работы выставляется в 100-балльной шкале комплексно с учетом степени подготовки студента к выполнению работы, объема выполненной работы на занятии и оформлении отчета в соответствии с перечисленными критериями. В зависимости от выставленного

максимального балла перерасчет за каждый отчет ЛР начисляемых баллов производится автоматически. Итоговый балл за отчеты по лабораторным работам является числом от 0 до 50 баллов.

Таблица 7 – Показатели оценивания результатов обучения в виде знаний

Шкала оценивания	Критерии оценивания
5 «отлично»	демонстрирует глубокое знание теоретического материала, умение обоснованно излагать свои мысли по обсуждаемым вопросам, способность полно, правильно и аргументированно отвечать на вопросы, приводить примеры
4 «хорошо»	демонстрирует знание теоретического материала, его последовательное изложение, способность приводить примеры, допускает единичные ошибки, исправляемые после замечания преподавателя
3 «удовлетвори тельно»	демонстрирует неполное, фрагментарное знание теоретического материала, требующее наводящих вопросов преподавателя, допускает существенные ошибки в его изложении, затрудняется в приведении примеров и формулировке выводов
2 «неудовлетво рительно»	демонстрирует существенные пробелы в знании теоретического материала, не способен его изложить и ответить на наводящие вопросы преподавателя, не может привести примеры

Таблица 8 – Показатели оценивания результатов обучения в виде умений и владений

Шкала оценивания	Критерии оценивания
5 «отлично»	демонстрирует способность применять знание теоретического материала при выполнении заданий, последовательно и правильно выполняет задания, умеет обоснованно излагать свои мысли и делать необходимые выводы
4 «хорошо»	демонстрирует способность применять знание теоретического материала при выполнении заданий, последовательно и правильно выполняет задания, умеет обоснованно излагать свои мысли и делать необходимые выводы, допускает единичные ошибки, исправляемые после замечания преподавателя
3 «удовлетвори тельно»	демонстрирует отдельные, несистематизированные навыки, испытывает затруднения и допускает ошибки при выполнении заданий, выполняет задание по подсказке преподавателя, затрудняется в формулировке выводов
2 «неудовлетво рительно»	не способен правильно выполнить задания

Грубыми считаются ошибки, свидетельствующие о том, что студент:

- не овладел основным материалом дисциплины
- не может применять на практике полученные знания
- не знает формул, графиков, схем и т.п.
- не знает приемов решения задач, аналогичных ранее решенным. Не грубыми ошибками являются
- неточность чертежа, графика, схемы и т.п.
- неточно сформулированный вопрос или пояснение при решении задачи
- отдельные ошибки вычислительного характера Недочетами считаются

- отдельные погрешности в формулировке вопроса или ответа
- отдельные ошибки вычислительного характера
- небрежное выполнение записей, чертежей, схем, графиков и т.п.

7.3. Контрольные задания и иные материалы, необходимые для оценки результатов обучения по дисциплине (модулю)

Тема 1: Основы программирования на Python

Лабораторная работа 1: Введение в Python
Цель: Ознакомиться с основами языка программирования Python и научиться создавать простые программы.
Задачи:

- Установить и настроить среду разработки Python (например, IDLE или Jupyter Notebook).
- Создать простую программу для вывода текста на экран.
- Создать простую программу для ввода данных с клавиатуры.
- Написать программу для вычисления площади прямоугольника, запрашивая длину и ширину с клавиатуры.

Контрольные вопросы:

1. Как установить и настроить среду разработки Python?
2. Как создать новую программу на Python?
3. Как вывести текст на экран в программе на Python?
4. Как получить ввод с клавиатуры в программе на Python?
5. Как преобразовать строку в число в Python?
6. Как вычислить площадь прямоугольника, зная его длину и ширину?
7. Как использовать оператор print() для вывода результатов в Python?
8. Как использовать оператор input() для получения ввода с клавиатуры в Python?
9. Как использовать оператор int() для преобразования строки в число в Python?
10. Как использовать оператор * для умножения чисел в Python?

Лабораторная работа 2: Работа с типами данных и операторами
Задание:
Разработать программу на Python, которая выполняет следующие действия:

1. **Ввод данных:** Запросить у пользователя ввод двух целых чисел и одного вещественного числа.
2. **Математические операции:** Выполнить следующие математические операции:
    - Сумму двух целых чисел.
    - Разность первого целого числа и второго целого числа.
    - Произведение двух целых чисел.
    - Частное от деления первого целого числа на второе целое число (с проверкой на деление на ноль).
    - Сумму всех трех чисел (целых и вещественного).
    - Возведение первого целого числа в степень второго целого числа.
3. **Логические операции:**
    - Сравнить первое целое число со вторым целым числом (больше, меньше, равно).
    - Проверить, является ли первое целое число четным.
    - Проверить, является ли второе целое число положительным.
    - Проверить, выполняется ли условие: первое целое число больше 10 И второе целое число меньше 20.

4. **Вывод результатов:** Вывести на экран результаты всех выполненных математических и логических операций в понятном и читаемом формате.
5. **Преобразование типов:** Преобразовать вещественное число в целое число и вывести результат.

Контрольные вопросы:

1. **Типы данных:** Какие типы данных вы использовали в программе? Почему вы выбрали именно эти типы?
2. **Операторы:** Какие арифметические операторы вы использовали? Какие логические операторы вы использовали?
3. **Ввод данных:** Как вы получали ввод данных от пользователя? Какая функция для этого используется?
4. **Деление на ноль:** Как вы обработали ситуацию деления на ноль? Почему это важно?
5. **Форматированный вывод:** Как вы форматировали вывод результатов? Какие методы форматирования строк вы использовали?
6. **Преобразование типов:** Как вы преобразовали вещественное число в целое число? Какая функция для этого используется?
7. **Логические выражения:** Как вы составляли логические выражения для проверки условий?
8. **Приоритет операторов:** Как приоритет операторов влияет на порядок выполнения операций?
9. **Ошибки:** Какие ошибки могут возникнуть при выполнении программы? Как их можно избежать?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она работала с тремя целыми числами и двумя вещественными числами?

Тема 2: Структурирование и организация кода

Лабораторная работа 3: Управление потоком выполнения

Задание:

Разработать программу на Python, которая выполняет следующие действия:

1. **Ввод данных:** Запросить у пользователя ввод целого числа.
2. **Проверка числа:**
   - Используя условный оператор if-elif-else, проверить, является ли введенное число:
     - Положительным, отрицательным или нулем.
     - Четным или нечетным.
     - Делится ли на 3 без остатка.
   - Вывести соответствующие сообщения на экран.
3. **Цикл for:**
   - Используя цикл for, вывести все числа от 1 до введенного числа (включительно).
   - Если введенное число меньше 1, вывести сообщение об ошибке.
4. **Цикл while:**
   - Запросить у пользователя ввод чисел до тех пор, пока не будет введено отрицательное число.
   - Для каждого введенного числа (кроме отрицательного) вычислять его квадрат и выводить на экран.
5. **Комбинация циклов и условий:**
   - Запросить у пользователя ввод двух целых чисел (начало и конец диапазона).
   - Используя цикл for и условный оператор if, вывести все четные числа в заданном диапазоне.
   - Если начало диапазона больше конца, вывести сообщение об ошибке.
6. **Вывод результатов:** Вывести все результаты в понятном и читаемом формате.

Контрольные вопросы:

1. **Условные операторы:** Как работает оператор if-elif-else? В чем разница между if, elif и else?
2. **Цикл for:** Как работает цикл for? Как использовать range() для генерации последовательности чисел?
3. **Цикл while:** Как работает цикл while? В чем разница между циклами for и while?
4. **Условия в циклах:** Как использовать условные операторы внутри циклов?
5. **Бесконечный цикл:** Как можно создать бесконечный цикл? Как его прервать?
6. **Вложенные циклы:** Как работают вложенные циклы? Приведите пример использования вложенных циклов.
7. **Операторы break и continue:** Что делают операторы break и continue? Как они влияют на выполнение циклов?
8. **Обработка ошибок:** Как вы обрабатывали ошибки ввода данных (например, ввод нечислового значения)?
9. **Логические выражения:** Как вы составляли логические выражения для проверки условий в циклах и условных операторах?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она выводила все простые числа в заданном диапазоне?

Лабораторная работа 4: Функции и модули

Задание:

Разработать программу на Python, которая выполняет следующие действия:

1. **Создание функций:**
   - Создать функцию calculate_area(length, width), которая принимает длину и ширину прямоугольника в качестве аргументов и возвращает его площадь.
   - Создать функцию calculate_perimeter(length, width), которая принимает длину и ширину прямоугольника в качестве аргументов и возвращает его периметр.
   - Создать функцию is_even(number), которая принимает целое число в качестве аргумента и возвращает True, если число четное, и False в противном случае.
   - Создать функцию factorial(number), которая принимает целое неотрицательное число в качестве аргумента и возвращает его факториал.
2. **Использование функций:**
   - Запросить у пользователя ввод длины и ширины прямоугольника.
   - Вызвать функции calculate_area и calculate_perimeter с введенными значениями и вывести результаты на экран.
   - Запросить у пользователя ввод целого числа.
   - Вызвать функцию is_even с введенным числом и вывести сообщение о том, является ли число четным или нечетным.
   - Запросить у пользователя ввод целого неотрицательного числа.
   - Вызвать функцию factorial с введенным числом и вывести результат на экран.
3. **Создание модуля:**
   - Создать отдельный файл geometry.py и поместить в него функции calculate_area и calculate_perimeter.
   - В основной программе импортировать функции calculate_area и calculate_perimeter из модуля geometry.py.
4. **Вывод результатов:** Вывести все результаты в понятном и читаемом формате.

Контрольные вопросы:

1. **Функции:** Что такое функция? Зачем нужны функции?
2. **Аргументы и параметры:** В чем разница между аргументами и параметрами функции?
3. **Возвращаемое значение:** Как функция возвращает значение? Что происходит, если функция не возвращает значение?

4. **Область видимости:** Что такое область видимости переменных? Как переменные, объявленные внутри функции, отличаются от переменных, объявленных вне функции?
5. **Модули:** Что такое модуль? Зачем нужны модули?
6. **Импорт модулей:** Как импортировать модули в Python? Какие существуют способы импорта (например, import module, from module import name)?
7. **Создание модулей:** Как создать свой собственный модуль?
8. **Рекурсия:** Что такое рекурсия? Как можно реализовать функцию factorial с помощью рекурсии?
9. **Аннотации типов:** Как можно использовать аннотации типов для функций?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она работала с кругом (вычисляла площадь и длину окружности) и использовала функции из модуля math?

Тема 3: Объектно-ориентированное программирование

Лабораторная работа 5: Работа с файлами
Задание:
Разработать программу на Python, которая выполняет следующие действия:

1. **Создание файла:**
   - Создать текстовый файл с именем data.txt.
   - Записать в него несколько строк текста (не менее 5 строк), каждая строка должна содержать информацию о каком-либо объекте (например, имя, возраст, город).
2. **Чтение файла:**
   - Прочитать содержимое файла data.txt построчно.
   - Вывести каждую строку на экран.
3. **Обработка данных:**
   - Прочитать содержимое файла data.txt и сохранить каждую строку в список.
   - Создать новый файл processed_data.txt.
   - Для каждой строки из списка:
     - Разделить строку на отдельные элементы (например, по запятой или пробелу).
     - Преобразовать числовые элементы в числа (если они есть).
     - Записать обработанную строку в файл processed_data.txt в формате, удобном для дальнейшей обработки (например, в формате CSV).
4. **Работа с CSV:**
   - Использовать библиотеку csv для чтения данных из файла processed_data.txt.
   - Вывести данные на экран в виде таблицы.
5. **Обработка ошибок:**
   - Предусмотреть обработку ошибок, которые могут возникнуть при работе с файлами (например, файл не найден, ошибка записи).
6. **Вывод результатов:** Вывести все результаты в понятном и читаемом формате.

Контрольные вопросы:

1. **Открытие файла:** Как открыть файл для чтения? Как открыть файл для записи? Какие режимы открытия файлов существуют?
2. **Закрытие файла:** Почему важно закрывать файлы после работы с ними? Как это сделать?
3. **Чтение файла:** Как прочитать содержимое файла построчно? Как прочитать весь файл целиком?
4. **Запись в файл:** Как записать данные в файл? Как записать данные в файл построчно?
5. **Работа со списками:** Как сохранить строки из файла в список? Как обрабатывать элементы списка?
6. **Разделение строк:** Как разделить строку на отдельные элементы? Какие методы для этого используются (например, split())?

7. **Преобразование типов:** Как преобразовать строковые значения в числовые значения?
8. **Библиотека csv:** Как использовать библиотеку csv для чтения и записи данных в формате CSV?
9. **Обработка ошибок:** Как обрабатывать ошибки, которые могут возникнуть при работе с файлами (например, FileNotFoundError, IOError)?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она работала с бинарными файлами (например, изображениями) и выполняла их копирование?

Лабораторная работа 6: Объектно-ориентированное программирование
Задание:
Разработать программу на Python, которая моделирует работу с геометрическими фигурами, используя принципы объектно-ориентированного программирования (ООП):

1. **Создание базового класса:**
    - Создать базовый класс Shape с атрибутами color (цвет фигуры) и методами:
        - __init__(self, color): конструктор, устанавливающий цвет фигуры.
        - get_color(self): возвращает цвет фигуры.
        - area(self): возвращает площадь фигуры (по умолчанию возвращает 0).
        - perimeter(self): возвращает периметр фигуры (по умолчанию возвращает 0).
        - __str__(self): возвращает строковое представление фигуры (например, "Фигура цвета: красный").
2. **Создание производных классов:**
    - Создать класс Rectangle, наследующий от класса Shape, с атрибутами length (длина) и width (ширина).
        - Переопределить методы __init__, area и perimeter для прямоугольника.
    - Создать класс Circle, наследующий от класса Shape, с атрибутом radius (радиус).
        - Переопределить методы __init__, area и perimeter для круга (используйте math.pi).
    - Создать класс Square, наследующий от класса Rectangle, с атрибутом side (сторона).
        - Переопределить метод __init__ для квадрата.
3. **Создание объектов:**
    - Создать несколько объектов разных классов (например, прямоугольник, круг, квадрат).
    - Установить для них различные значения атрибутов.
4. **Использование полиморфизма:**
    - Создать список объектов разных классов.
    - Используя цикл for, вызвать методы area и perimeter для каждого объекта и вывести результаты на экран.
    - Используя цикл for, вызвать метод __str__ для каждого объекта и вывести результаты на экран.
5. **Вывод результатов:** Вывести все результаты в понятном и читаемом формате.

Контрольные вопросы:

1. **Классы и объекты:** Что такое класс? Что такое объект? Как создать класс и объект в Python?
2. **Конструктор:** Что такое конструктор (__init__)? Зачем он нужен?
3. **Атрибуты и методы:** Что такое атрибуты класса? Что такое методы класса? Как обращаться к атрибутам и методам объекта?
4. **Наследование:** Что такое наследование? Как создать подкласс (дочерний класс)?
5. **Переопределение методов:** Что такое переопределение методов? Как переопределить метод родительского класса в подклассе?
6. **Полиморфизм:** Что такое полиморфизм? Как полиморфизм проявляется в Python?

7. **Метод super():** Что такое super()? Как его использовать для вызова методов родительского класса?
8. **Метод __str__:** Что такое метод __str__? Зачем он нужен?
9. **Абстрактные классы:** Как можно сделать класс Shape абстрактным?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она работала с другими геометрическими фигурами (например, треугольником) и использовала абстрактные методы?

Тема 4: Прикладное программирование в мехатронике и робототехнике

Лабораторная работа 7: Прикладное программирование в мехатронике и робототехнике
Задание:
Разработать программу на Python, которая имитирует работу простой мехатронной системы, состоящей из датчика расстояния и сервопривода.

1. **Имитация датчика расстояния:**
   - Создать функцию simulate_distance_sensor(), которая имитирует работу датчика расстояния. Функция должна возвращать случайное значение расстояния в диапазоне от 10 до 100 (например, в сантиметрах).
2. **Управление сервоприводом:**
   - Создать функцию control_servo(angle), которая имитирует управление сервоприводом. Функция должна принимать угол поворота сервопривода в градусах (от 0 до 180).
   - Внутри функции вывести сообщение о том, на какой угол повернулся сервопривод.
3. **Обработка данных:**
   - Создать функцию process_data(distance), которая принимает значение расстояния от датчика.
   - Внутри функции:
     - Если расстояние меньше 30, установить угол поворота сервопривода на 0 градусов.
     - Если расстояние больше 70, установить угол поворота сервопривода на 180 градусов.
     - В противном случае, установить угол поворота сервопривода на 90 градусов.
     - Вызвать функцию control_servo с вычисленным углом поворота.
4. **Основная программа:**
   - В основной программе:
     - В цикле (например, 10 раз) вызвать функцию simulate_distance_sensor() для получения значения расстояния.
     - Вывести полученное значение расстояния на экран.
     - Вызвать функцию process_data() для обработки полученного значения расстояния и управления сервоприводом.
     - Добавить задержку между итерациями цикла (например, 1 секунда).
5. **Вывод результатов:** Вывести все результаты в понятном и читаемом формате.

Контрольные вопросы:

1. **Имитация датчика:** Как вы имитировали работу датчика расстояния? Какие библиотеки Python вы использовали для генерации случайных чисел?
2. **Управление сервоприводом:** Как вы имитировали управление сервоприводом? Почему вы использовали вывод сообщения вместо реального управления?
3. **Обработка данных:** Как вы обрабатывали данные от датчика расстояния? Какие условные операторы вы использовали?

4. **Функции:** Как вы использовали функции для организации кода? Зачем нужны функции?
5. **Циклы:** Как вы использовали циклы для повторения действий?
6. **Задержка:** Как вы реализовали задержку между итерациями цикла? Какие библиотеки Python вы использовали?
7. **Реальное управление:** Как бы вы модифицировали программу для управления реальным сервоприводом и датчиком расстояния? Какие библиотеки Python вам бы понадобились?
8. **Обработка ошибок:** Какие ошибки могут возникнуть при работе с реальным оборудованием? Как их можно обработать?
9. **Масштабирование:** Как бы вы масштабировали программу для управления несколькими датчиками и сервоприводами?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она использовала ПИД-регулятор для более точного управления сервоприводом?

Лабораторная работа 8: Разработка программ на для промышленных роботов
Задание:
Разработать программу на Python, которая имитирует управление промышленным роботом для выполнения простой задачи: перемещение объекта с одной точки на другую.
Примечание: Поскольку прямое управление реальным промышленным роботом может быть сложным и требовать специального оборудования, мы будем использовать имитацию робота и его движений.

1. **Имитация робота:**
    - Создать класс Robot с атрибутами:
        - position (текущая позиция робота в виде списка [x, y, z]).
        - gripper_state (состояние захвата: True - захват открыт, False - захват закрыт).
    - Создать методы:
        - __init__(self, start_position): конструктор, устанавливающий начальную позицию робота.
        - move_to(self, target_position): имитирует перемещение робота в заданную позицию. Метод должен выводить сообщение о перемещении и обновлять атрибут position.
        - open_gripper(self): имитирует открытие захвата. Метод должен выводить сообщение и обновлять атрибут gripper_state.
        - close_gripper(self): имитирует закрытие захвата. Метод должен выводить сообщение и обновлять атрибут gripper_state.
        - get_position(self): возвращает текущую позицию робота.
        - get_gripper_state(self): возвращает текущее состояние захвата.
2. **Имитация рабочего пространства:**
    - Задать начальную позицию объекта (object_start_position) и конечную позицию объекта (object_end_position).
3. **Программа управления роботом:**
    - Создать объект класса Robot с начальной позицией.
    - Разработать последовательность действий для робота, чтобы он:
        - Переместился к начальной позиции объекта.
        - Открыл захват.
        - Закрыл захват (имитируя захват объекта).
        - Переместился к конечной позиции объекта.
        - Открыл захват (имитируя освобождение объекта).
    - Выводить на экран текущую позицию робота и состояние захвата после каждого действия.

4. **Вывод результатов:** Вывести все результаты в понятном и читаемом формате.

Контрольные вопросы:

1. **Классы и объекты:** Как вы создали класс Robot? Что такое объект класса?
2. **Атрибуты и методы:** Какие атрибуты и методы вы определили для класса Robot?
3. **Имитация движения:** Как вы имитировали движение робота? Почему вы использовали вывод сообщения вместо реального управления?
4. **Управление захватом:** Как вы имитировали управление захватом робота?
5. **Последовательность действий:** Как вы организовали последовательность действий робота?
6. **Использование методов:** Как вы вызывали методы объекта Robot для выполнения действий?
7. **Реальное управление:** Как бы вы модифицировали программу для управления реальным промышленным роботом? Какие библиотеки Python вам бы понадобились?
8. **Обработка ошибок:** Какие ошибки могут возникнуть при работе с реальным роботом? Как их можно обработать?
9. **Масштабирование:** Как бы вы масштабировали программу для выполнения более сложных задач (например, сборка объекта из нескольких частей)?
10. **Модификация:** Как бы вы модифицировали программу, чтобы она использовала траекторию движения (например, линейное движение между точками) вместо прямого перемещения?

Перечень вопросов и заданий, выносимых на дифференцированный зачёт

Основы Python:

1. Что такое Python и каковы его основные особенности?
2. Какие типы данных существуют в Python?
3. Что такое переменная и как ее объявить в Python?
4. Какие операторы используются в Python для арифметических операций?
5. Какие операторы используются в Python для логических операций?
6. Что такое условный оператор if-elif-else и как он работает?
7. Что такое циклы for и while и как они работают?
8. Как использовать функцию range() для генерации последовательности чисел?
9. Что такое список (list) в Python и как с ним работать?
10. Что такое кортеж (tuple) в Python и чем он отличается от списка?
11. Что такое словарь (dictionary) в Python и как с ним работать?
12. Что такое множество (set) в Python и как с ним работать?
13. Как получить ввод данных от пользователя в Python?
14. Как вывести данные на экран в Python?
15. Что такое форматирование строк и как его использовать?
16. Что такое комментарии и как их использовать в Python?
17. Что такое область видимости переменных?
18. Что такое глобальные и локальные переменные?
19. Что такое функция и как ее объявить в Python?
20. Что такое аргументы и параметры функции?
21. Как функция возвращает значение?
22. Что такое рекурсия?

Работа с файлами:

23. Как открыть файл для чтения в Python?
24. Как прочитать содержимое файла в Python?
25. Как открыть файл для записи в Python?
26. Как записать данные в файл в Python?

27. Как закрыть файл после работы с ним?
28. Что такое менеджер контекста with и как его использовать для работы с файлами?
29. Как обрабатывать ошибки при работе с файлами?
30. Как использовать библиотеку csv для работы с CSV-файлами?

Объектно-ориентированное программирование (ООП):

31. Что такое класс и объект в Python?
32. Что такое конструктор (__init__) и как его использовать?
33. Что такое атрибуты и методы класса?
34. Что такое self и как его использовать?
35. Что такое наследование и как его использовать?
36. Что такое переопределение методов?
37. Что такое полиморфизм и как он проявляется в Python?
38. Что такое метод super() и как его использовать?
39. Что такое инкапсуляция и как ее реализовать в Python?
40. Что такое абстрактные классы и интерфейсы?

Модули и библиотеки:

41. Что такое модуль и как его импортировать в Python?
42. Как создать свой собственный модуль?
43. Какие стандартные библиотеки Python вы знаете?
44. Как использовать библиотеку math?
45. Как использовать библиотеку random?
46. Как использовать библиотеку datetime?
47. Как использовать библиотеку os?
48. Как использовать библиотеку re для работы с регулярными выражениями?
49. Как использовать библиотеку json для работы с JSON-данными?
50. Как установить сторонние библиотеки с помощью pip?

Таблица 9 – Примеры оценочных средств с ключами правильных ответов

№ п/п	Тип задания	Формулировка задания	Правильный ответ	Время выполнения (в минутах)
УК-1. Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач				
1.	Задание закрытого типа	Какой тип данных используется для хранения целых чисел в Python? a) float b) str c) int d) bool	**Правильный ответ: c) int**	1
2.		Какой оператор используется для проверки равенства двух значений в Python? a) = b) == c) != d) is	**Правильный ответ: b) ==**	1
3.		Какой цикл используется для перебора элементов последовательности (например, списка) в Python? a) while b) if c) for d) switch	**Правильный ответ: c) for**	1
4.	Задание	Объясните разницу между списками (lists) и кортежами	Списки и кортежи - это последовательности в Python, но	5

№ п/п	Тип задания	Формулировка задания	Правильный ответ	Время выполнения (в минутах)
	открытого типа	(tuples) в Python. Приведите примеры, когда использование одного типа данных предпочтительнее другого.	они отличаются по своей изменяемости. Списки являются *изменяемыми* (mutable), то есть их можно модифицировать после создания: добавлять, удалять или изменять элементы. Кортежи же, напротив, *неизменяемы* (immutable), и после создания их нельзя изменить. Списки обычно используют, когда нужно динамически менять набор данных, например, при сборе результатов или хранении последовательности действий. Кортежи же подходят для представления фиксированных наборов данных, например, координат точки или записей в базе данных, где неизменность гарантирует целостность данных.	
5.		Опишите, что такое область видимости переменных в Python. Приведите примеры локальной и глобальной переменных и объясните, как они взаимодействуют.	Область видимости переменной определяет, в какой части программы она доступна. Локальные переменные объявляются внутри функции и доступны только внутри этой функции. Глобальные переменные объявляются вне функций и доступны во всей программе, включая функции. Если внутри функции объявляется переменная с тем же именем, что и глобальная, то она будет считаться локальной, и изменения не затронут глобальную переменную. Чтобы изменить глобальную переменную внутри функции, нужно использовать ключевое слово global. Понимание области видимости важно для предотвращения конфликтов имен и обеспечения правильной работы программы.	5
6.		Объясните, что такое наследование в объектно-ориентированном программировании (ООП) и как его реализовать в Python. Приведите пример использования наследования.	Наследование - это механизм ООП, позволяющий создавать новый класс (подкласс или дочерний класс) на основе существующего класса (суперкласс или родительский класс). Подкласс наследует все атрибуты и методы суперкласса, что позволяет повторно использовать код и создавать иерархию классов. В Python наследование реализуется путем указания имени суперкласса в скобках после имени подкласса. Например, можно создать класс Animal и затем классы Dog и Cat, наследующие от Animal, добавляя свои специфические атрибуты и методы,	5

№ п/п	Тип задания	Формулировка задания	Правильный ответ	Время выполнения (в минутах)
			но при этом используя общие свойства Animal.	
УК-2. Способен определять круг задач в рамках поставленной цели и выбирать оптимальные способы их решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений				
7.	Задание закрытого типа	Какая функция используется для получения длины списка в Python? a) size() b) length() c) len() d) count()	**Правильный ответ: c) len()**	1
8.		Какой метод используется для добавления элемента в конец списка в Python? a) insert() b) add() c) append() d) push()	**Правильный ответ: c) append()**	1
9.		Какой оператор используется для импорта модуля в Python? a) include b) using c) import d) require	**Правильный ответ: c) import**	1
10.	Задание открытого типа	Опишите, как работает цикл for в Python. Приведите пример использования цикла for для перебора элементов списка и для генерации последовательности чисел с помощью range().	Цикл for в Python используется для итерации по элементам последовательности (например, списка, кортежа, строки) или по последовательности чисел, сгенерированной функцией range(). Для перебора элементов списка, цикл for последовательно присваивает каждому элементу списка переменную цикла. Например, for item in my_list: print(item). Функция range() генерирует последовательность чисел, и цикл for может использовать ее для повторения действий определенное количество раз. Например, for i in range(5): print(i) выведет числа от 0 до 4.	5
11.		Объясните, что такое обработка исключений в Python и как использовать блоки try-except для обработки ошибок. Приведите пример использования try-except для обработки ошибки деления на ноль.	Обработка исключений - это механизм для перехвата и обработки ошибок, возникающих во время выполнения программы. Блок try-except позволяет выполнить код, который может вызвать ошибку, и перехватить эту ошибку, если она возникнет. Код, который может вызвать ошибку, помещается в блок try, а код, который должен быть выполнен в случае ошибки, помещается в блок except. Например, для обработки ошибки деления на ноль можно использовать: try: result = 10 / 0;	5

№ п/п	Тип задания	Формулировка задания	Правильный ответ	Время выполнения (в минутах)
			except ZeroDivisionError: print("Деление на ноль!").	
12.		Опишите, как можно создать свой собственный модуль в Python и как импортировать его в другую программу. Приведите пример создания простого модуля и его использования.	Модуль в Python - это файл, содержащий определения функций, классов и переменных. Чтобы создать свой модуль, нужно просто сохранить файл с расширением .py. Например, файл my_module.py может содержать функцию def greet(name): print(f"Hello, {name}!"). Чтобы использовать этот модуль в другой программе, нужно импортировать его с помощью import my_module. Затем можно вызвать функцию из модуля: my_module.greet("Alice"). Можно также импортировать конкретные имена из модуля с помощью from my_module import greet.	5

Полный комплект оценочных материалов по дисциплине (модулю) (фонд оценочных средств) хранится в электронном виде на кафедре, утверждающей рабочую программу дисциплины (модуля), и в Центре мониторинга и аудита качества обучения.

7.4. Методические материалы, определяющие процедуры оценивания результатов обучения по дисциплине (модулю)

Таблица 10 – Технологическая карта рейтинговых баллов по дисциплине (модулю)

№ п/п	Контролируемые мероприятия	Количество мероприятий / баллы	Максимальное количество баллов	Срок представления
Основной блок				
1.	Ответ на занятии	6	6	По расписанию
2.	Выполнение лабораторной работы	7	84	По расписанию
Всего			**90**	-
Блок бонусов				
3.	Посещение занятий	1	1	По расписанию
4.	Своевременное выполнение всех заданий	9	9	По расписанию
Всего			**10**	-
ИТОГО			**100**	-

Таблица 11 – Система штрафов (для одного занятия)

Показатель	Балл
Опоздание (два и более)	-2
Не готов к практической части занятия	-3
Нарушение учебной дисциплины	-2

Пропуски лекций без уважительных причин (за одну лекцию)	-2
Пропуск занятий без уважительной причины (за одно занятие)	-2
Нарушение правил техники безопасности	-1
Отсутствие конспектов лекций, семинарских занятий, первоисточников при начислении баллов не учитываются	0

Таблица 12 – Шкала перевода рейтинговых баллов в итоговую оценку за семестр по дисциплине (модулю)

Сумма баллов	Оценка по 4-балльной шкале
90–100	5 (отлично)
85–89	4 (хорошо)
75–84	
70–74	
65–69	3 (удовлетворительно)
60–64	
Ниже 60	2 (неудовлетворительно)

При реализации дисциплины (модуля) в зависимости от уровня подготовленности обучающихся могут быть использованы иные формы, методы контроля и оценочные средства, исходя из конкретной ситуации.

8. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

8.1. Основная литература

1. Лутц, М. Изучаем Python / М. Лутц. — СПб.: Питер, 2022.
2. Свейгарт, А. Automate the Boring Stuff with Python. Практическое программирование для начинающих / А. Свейгарт. — СПб.: Питер, 2021.
3. Седжвик, Р., Уэйн, К. Алгоритмы на Python / Р. Седжвик, К. Уэйн. — М.: Диалектика, 2021.
4. Макки, Д. Python. Полный справочник / Д. Макки. — М.: БХВ-Петербург, 2020.
5. Грин, Д. Python. Разработка приложений / Д. Грин. — СПб.: Питер, 2019.
6. Марк Лутц. Программирование на Python. — СПб.: Символ-Плюс, 2021.
7. Биллар, Б. Python. Карманный справочник / Б. Биллар. — СПб.: Питер, 2020.
8. Холл, Н., Шварц, М. Python. Основы программирования / Н. Холл, М. Шварц. — М.: ДМК Пресс, 2021.
9. Дауни, А. Python для информатики: Анализ данных и вычисления / А. Дауни. — СПб.: Питер, 2020.
10. Рэмси, Б. Эффективное программирование на Python / Б. Рэмси. — СПб.: Питер, 2021.

8.2. Дополнительная литература

1. Фаулер, М. Шаблоны корпоративных приложений на Python / М. Фаулер. — СПб.: Питер, 2022.
2. Хетланд, М. Python. Глубокое программирование / М. Хетланд. — СПб.: Питер, 2020.
3. Ван Россум, Г., Дрейк, Ф. Python. Руководство по языку / Г. Ван Россум, Ф. Дрейк. — М.: Диалектика, 2021.
4. Шилдт, Г. Python. Самоучитель / Г. Шилдт. — СПб.: Питер, 2022.
5. Кеннеди, Д. Python для анализа данных и машинного обучения / Д. Кеннеди. — М.: БХВ-Петербург, 2021.

8.3. Интернет-ресурсы, необходимые для освоения дисциплины (модуля)

Электронно-библиотечная система (ЭБС) ООО «Политехресурс» «Консультант студента», http://www.studentlibrary.ru.

9. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ (МОДУЛЯ)

Учебные аудитории, библиотеки АГУ, центр мониторинга и аудита качества образования, компьютерные классы, мультимедийные аудитории.

Рабочая программа дисциплины (модуля) при необходимости может быть адаптирована для обучения (в том числе с применением дистанционных образовательных технологий) лиц с ограниченными возможностями здоровья, инвалидов. Для этого требуется заявление обучающихся, являющихся лицами с ограниченными возможностями здоровья, инвалидами, или их законных представителей и рекомендации психолого-медико-педагогической комиссии. Для инвалидов содержание рабочей программы дисциплины (модуля) может определяться также в соответствии с индивидуальной программой реабилитации инвалида (при наличии).