Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Самарский государственный технический университет»
(ФГБОУ ВО «СамГТУ»)
Филиал ФГБОУ ВО «СамГТУ» в г. Белебее Республики Башкортостан

УТВЕРЖДАЮ
Директор филиала ФГБОУ ВО «СамГТУ»
в г. Белебее Республики Башкортостан
_____ Л.М. Инаходова
25.05.2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

Б1.О.02.05 «Языки и методы программирования»

Код и направление подготовки (специальность)	**09.03.02 Информационные системы и технологии**
Направленность (профиль)	**Информационные системы и технологии**
Квалификация	**Бакалавр**
Форма обучения	**Заочная**
Год начала подготовки	**2023**
Выпускающая кафедра	**Инженерные технологии**
Кафедра-разработчик	**Инженерные технологии**
Объем дисциплины, ч. / з.е.	**216 / 6**
Форма контроля (промежуточная аттестация)	**Экзамен, Экзамен**

Белебей 2023 г.

Рабочая программа дисциплины (далее – РПД) разработана в соответствии с требованиями ФГОС ВО по направлению подготовки (специальности) 09.03.02 «Информационные системы и технологии», утвержденного приказом Министерства образования и науки РФ от 19 сентября 2017 г. № 926, и соответствующего учебного плана.

Разработчик РПД:

доцент, к.т.н., доцент		З.Ф. Камальдинова
(должность, степень, ученое звание)	(подпись)	(ФИО)

РПД рассмотрена и одобрена на заседании кафедры 25.05.2023 г., протокол № 6.

Заведующий кафедрой	к.т.н., доцент	А.А. Цынаева
	(степень, ученое звание, подпись)	(ФИО)

СОГЛАСОВАНО:

Руководитель образовательной программы	доцент, к.т.н.	З.Ф. Камальдинова
	(степень, ученое звание, подпись)	(ФИО)

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программ

Универсальные компетенции

Таблица 1

Наименование категории (группы) компетенций	Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом				

Общепрофессиональные компетенции

Таблица 2

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	**ОПК-2.1** Использует и понимает принципы работы информационных технологий и программных средств при решении задач в сфере информационных систем и технологий	**З2 ОПК-2.1** Знать: теорию программно-аппаратного обеспечения информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы в организациях различных форм собственности с целью повышения эффективности деятельности организаций.
		ОПК-2.2 Применяет современные информационные технологии и программные средства отечественного производства при решении задач в сфере информационных систем и технологий	**У1 ОПК-2.2** Уметь: анализировать предметную область информационной системы и учитывать ее специфику для принятия проектных решений в процессе создания и использования; разрабатывать модели информационной системы; выполнять декомпозицию сложной информационной системы **В2 ОПК-2.2** Владеть: методами и средствами обеспечения работы программно-аппаратного обеспечения информационных систем, в том числе отечественного производства, при решении задач профессиональной деятельности
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;	**ОПК-6.1** Разрабатывает алгоритмы и программы, пригодные для практического применения	**З1 ОПК-6.1** Знать: основы алгоритмизации, языки программирования, методы реализации алгоритмов, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий
		ОПК-6.2 Осуществляет отладку и тестирование программного обеспечения	**В1 ОПК-6.2** Владеть: навыками разработки алгоритмов, программирования, отладки и тестирования прототипов программно-технических комплексов задач
		ОПК-6.3 Ведет и использует базы данных и информационные хранилища	**У1 ОПК-6.3** Уметь: применять языки программирования, работать с базами данных, разрабатывать информационные системы и технологии для автоматизации бизнес-процессов, решать прикладные задачи различных классов
ОПК-8	Способен применять математические модели, методы и средства проектирования информационных и автоматизированных систем.	**ОПК-8.1** Разрабатывает математические и имитационные модели процессов в сфере информационных систем и технологий	**З1 ОПК-8.1** Знать: принципы моделирования, классификацию способов представления моделей систем; приемы, методы, способы формализации объектов, процессов, явлений и реализацию их на компьютере; достоинства и недостатки различных способов представления моделей систем; разработку алгоритмов фиксации и обработки результатов моделирования систем; способы планирования машинных экспериментов с моделями

Профессиональные компетенции

Таблица 3

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом			

2. Место дисциплины (модуля) в структуре образовательной программы

Место дисциплины в структуре образовательной программы: обязательная часть.

Таблица 4

Код компетенции	Предшествующие дисциплины	Параллельно осваиваемые дисциплины	Последующие дисциплины
ОПК-2		Информационные технологии и программирование; Учебная практика: ознакомительная практика	Управление данными; Технологии программирования; Методы и средства проектирования информационных систем и технологий
ОПК-6			Системы искусственного интеллекта; Управление данными; Технологии программирования; Методы и средства проектирования информационных систем и технологий
ОПК-8			Теория информационных процессов и моделирование систем

3. Объем дисциплины в зачетных единицах с указанием количества академических часов, выделенных на контактную работу обучающихся с преподавателем (по видам учебных занятий) и на самостоятельную работу обучающихся

Таблица 5

Вид учебной работы	Всего часов	Курс 1
Аудиторная контактная работа (всего), в том числе:	**16**	**16**
лекционные занятия (ЛЗ)	8	8
лабораторные работы (ЛР)	8	8
практические занятия (ПЗ)	0	0
Внеаудиторная контактная работа, КСР	**6**	**6**
Самостоятельная работа (всего), в том числе:	**176**	**176**
самостоятельное изучение материала	88	88
подготовка к лабораторным работам, выполнение соответствующих заданий	88	88
Формы текущего контроля успеваемости	Тестирование	Тестирование
Формы промежуточной аттестации	**экзамен, экзамен**	**экзамен, экзамен**
Контроль	**18**	**18**
ИТОГО: час.	**216**	**216**
ИТОГО: з.е.	**6**	**6**

4. Содержание дисциплины, структурированное по темам (разделам), с указанием отведенного на них количества академических часов и видов учебных занятий

Таблица 6

№ раздела	Наименование раздела дисциплины	Виды учебной нагрузки и их трудоемкость, часы						
		ЛЗ	ЛР	ПЗ	СРС	КСР	Конт-роль	Всего часов
1	Вводный	2	-	-	22	1	3	28
2	Основы алгоритмизации. Методы реализации алгоритмов.	2	2	-	52	2	3	61
3	Основы программирования на языке высокого уровня	4	6	-	102	3	12	127
Итого:		**8**	**8**	**0**	**176**	**6**	**18**	**216**

4.1. Содержание лекционных занятий

Таблица 7

№ ЛЗ	Наименование раздела	Тема лекции	Содержание лекции (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
Курс 1				
1	Вводный	Вводная лекция	Краткий обзор языков программирования. Понятие	2

			алгоритма, исполнителя и программы. Краткий обзор современных средств разработки. Понятие алгоритмического, процедурного и объектно-ориентированного программирования.	
2	Основы алгоритмизации. Методы реализации алгоритмов	Структурное программирование. Модели трансляции	Понятие структурного программирования. Обзор основных типовых алгоритмов (линейных, циклических, разветвленных). Процесс создания выполняемого программного модуля. Понятие и особенности работы транслятора, компилятора, интерпретатора.	2
3	Основы программирования на языке высокого уровня	Основные элементы языка	Типы переменных и констант и способы их объявления. Простые операции с данными. Операции ввода и вывода данных. Математическая библиотека языка. Особенности тестирования вычислительных программ.	2
4	Основы программирования на языке высокого уровня	Обработка данных Массивы	Понятие статического массива. Объявление статического массива, организация ввода с клавиатуры и вывода на экран. Заполнение случайными числами.	2
			Итого за курс:	**8**
			Итого:	**8**

4.2. Содержание лабораторных занятий

Таблица 8

№ ЛР	Наименование раздела	Наименование лабораторной работы	Содержание лабораторной работы (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
			Курс 1	
1	Основы алгоритмизации. Методы реализации алгоритмов	Базовые алгоритмы	Разработка типовых (линейного, циклического и разветвленного) алгоритмов.	2
2	Основы программирования на языке высокого уровня	Циклические программы. Работа с примитивами	Разработка циклической программы. Оператор FOR с разными вариантами определения количества повторений в фиксированном и произвольном диапазоне аргумента. Разработка программы для создания простого графического объекта по выбору пользователя.	2
3	Основы программирования на языке высокого уровня	Матрицы (двухмерные массивы)	Создание и инициализация матрицы. Заполнение в режиме ввода с клавиатуры и заполнение случайными числами. Перебор элементов матрицы, работа с отдельными элементами, перестановка строк и столбцов, преобразование в одномерный массив. Запись матрицы в файл и чтение из файла.	2
4	Основы программирования на языке высокого уровня	Своя библиотека функций. Рекурсивные алгоритмы	Формирование набора ранее разработанных функций и процедур обработки данных. Разработка программы рекурсивного вычисления факториала числа N. Анализ эффективности. Сравнение с алгоритмом вычисления в цикле.	2
			Итого за курс:	**8**
			Итого:	**8**

4.3. Содержание практических занятий

Таблица 9

№ ПЗ	Наименование раздела	Тема практического занятия	Содержание практического занятия (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
не предусмотрены учебным планом				

4.4. Содержание самостоятельной работы

Таблица 10

№ п/п	Наименование раздела	Вид самостоятельной работы	Содержание самостоятельной работы (перечень дидактических единиц: рассматриваемых подтем, вопросов)	Кол-во часов
			Курс 1	
1	Вводный	самостоятельное изучение материала	История развития вычислительной техники языков программирования, история развития и классификация языков программирования.	22
2	Основы	подготовка к лабораторным	Изучение лекций в качестве подготовки к	52

	алгоритмизации. Методы реализации алгоритмов	работам, выполнение соответствующих заданий	лабораторным работам.	
3	Основы программирования на языке высокого уровня	подготовка к лабораторным работам, выполнение соответствующих заданий	Изучение лекций в качестве подготовки к лабораторным работам.	102
			Итого за курс:	**176**
			Итого:	**176**

5. Методические указания для обучающихся по освоению дисциплины (модуля)

Методические указания при работе на лекции

До лекции обучающийся должен просмотреть учебно-методическую и научную литературу по теме лекции для того, чтобы иметь представление о проблемах, которые будут подняты в лекции.

Перед началом лекции обучающимся сообщается тема лекции, план, вопросы, подлежащие рассмотрению, доводятся основные литературные источники. Весь учебный материал, сообщаемый преподавателем, должен не просто прослушиваться. Он должен быть активно воспринят, т.е. услышан, осмыслен, понят, зафиксирован на бумаге и закреплен в памяти. Приступая к слушанию нового учебного материала, полезно мысленно установить его связь с ранее изученным. Следя за техникой чтения лекции (акцент на существенном, повышение тона, изменение ритма, пауза и т.п.), необходимо вслед за преподавателем уметь выделять основные категории, законы и определять их содержание, проблемы, предполагать их возможные решения, доказательства и выводы. Осуществляя такую работу, можно значительно облегчить себе понимание учебного материала, его конспектирование и дальнейшее изучение.

Методические указания при работе на лабораторном занятии

Проведение лабораторной работы делится на две условные части: теоретическую и практическую.

Необходимыми структурными элементами занятия являются проведение лабораторной работы, проверка усвоенного материала, включающая обсуждение теоретических основ выполняемой работы.

Перед лабораторной работой, как правило, проводится технико-теоретический инструктаж по использованию необходимого оборудования. Преподаватель корректирует деятельность обучающегося в процессе выполнения работы (при необходимости). После завершения лабораторной работы подводятся итоги, обсуждаются результаты деятельности.

Возможны следующие формы организации лабораторных работ: фронтальная, групповая и индивидуальная. При фронтальной форме однотипная работа выполняется всеми обучающимися одновременно. При групповой форме работа выполняется группой (командой). При индивидуальной форме обучающимися выполняются индивидуальные работы.

По каждой лабораторной работе имеются методические указания по их выполнению, включающие необходимый теоретический и практический материал, содержащие элементы и последовательную инструкцию по проведению выбранной работы, индивидуальные варианты заданий, требования и форму отчетности по данной работе.

Методические указания по самостоятельной работе

Организация самостоятельной работы обучающихся ориентируется на активные методы овладения знаниями, развитие творческих способностей, переход от поточного к индивидуализированному обучению с учетом потребностей и возможностей обучающегося.

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. Все новые понятия по изучаемой теме необходимо выучить наизусть.

Самостоятельная работа реализуется:

- непосредственно в процессе аудиторных занятий;
- на лекциях, практических занятиях;
- в контакте с преподавателем вне рамок расписания;
- на консультациях по учебным вопросам, в ходе творческих контактов, при ликвидации задолженностей, при выполнении индивидуальных заданий и т. д.;
- в методическом кабинете, дома, на кафедре при выполнении обучающимся учебных и практических задач.

Эффективным средством осуществления обучающимся самостоятельной работы является электронная информационно-образовательная среда университета, которая обеспечивает доступ к учебным планам, рабочим программам дисциплин (модулей), практик, к изданиям электронных библиотечных систем.

Методические указания по подготовке к тестированию

Тестовые задания – система стандартизированных заданий, позволяющая автоматизировать процедуру измерения уровня знаний и умений обучающегося.

Успешное выполнение тестовых заданий является необходимым условием итоговой положительной оценки. Выполнение тестовых заданий предоставляет обучающимся возможность самостоятельно контролировать уровень своих знаний, обнаруживать пробелы в знаниях и принимать меры по их ликвидации. Форма изложения тестовых заданий позволяет закрепить и восстановить в памяти пройденный материал. Тестовые задания охватывают основные вопросы по изучаемой теме. Для формирования заданий использована как закрытая, так и открытая форма. У обучающегося есть возможность выбора правильного ответа или нескольких правильных ответов из числа предложенных вариантов. Для выполнения тестовых заданий обучающиеся должны изучить лекционный материал по теме, соответствующие разделы литературы по дисциплине. Контрольный тест выполняется обучающимся самостоятельно во время практических занятий.

6. Перечень учебной литературы и учебно-методического обеспечения для самостоятельной работы

Таблица 11

№ п/п	Автор(ы), наименование, место, год издания (если есть, указать «гриф»)	Книжный фонд (КФ) или электрон. ресурс (ЭР)	Литература	
			учебная	для самост. работы
1.	Алгоритмизация и программирование: учебное пособие / Тюльпинова Н. В., Вузовское образование: 2019. - Режим доступа: https://www.iprbookshop.ru/80539.html	ЭР	+	+
2.	Золин, А.Г. Программирование на языках высокого уровня : лаб.практикум / А. Г. Золин; Самар.гос.техн.ун-т, Информационные технологии.- Самара, 2007.- 86 с..- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||elib||785	ЭР	+	+
3.	Широков, А. И. Информатика: разработка программ на языке программирования Питон: базовые языковые конструкции: учебник / А. И. Широков, М. О. Пышняк. — Москва: Издательский Дом МИСиС, 2020. — 142 с. — ISBN 978-5-907226-76-0. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: https://www.iprbookshop.ru/106713.html	ЭР	+	
4.	Программирование на языке высокого уровня: учебно-методическое пособие / Фарафонов А.С., Липецкий государственный технический университет, ЭБС АСВ: 2013.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||22912	ЭР		+
5.	Маляров, А.Н. Объектно-ориентированное программирование : учеб. для техн. вузов / А. Н. Маляров; Самар.гос.техн.ун-т, Высшая математика и прикладная информатика.- Самара, 2017.- 332 с..- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||elib||2665	ЭР		+
6.	Программирование на языке Си: учебное пособие / Шишкин А.Д., Российский государственный гидрометеорологический университет: 2013.- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||iprbooks||17959	ЭР		+
7.	Гутман, Г.Н. Языки программирования: Python 3.1: учеб. пособие / Г. Н. Гутман; Самар.гос.техн.ун-т, Прикладная математика и информатика.- Самара, 2011.- 129 с..- Режим доступа: https://elib.samgtu.ru/getinfo?uid=els_samgtu||elib||2692	ЭР		+

Доступ обучающихся к ЭР НТБ СамГТУ (elib.samgtu.ru) осуществляется посредством электронной информационной образовательной среды университета и сайта НТБ СамГТУ по логину и паролю.

7. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень программного обеспечения

При проведении лекционных занятий используется мультимедийное оборудование. Организовано взаимодействие обучающегося и преподавателя с использованием электронной информационной образовательной среды университета.

Программное обеспечение

Таблица 12

№ п/п	Название	Способ распространения (лицензионное или свободно распространяемое)	Правообладатель (производитель)	Страна происхождения (иностранное или отечественное)
1.	Пакет офисных программ LibreOffice	свободно распространяемое	The Document Foundation	иностранное

2.	Пакет офисных программ Microsoft Office	лицензионное	Microsoft	иностранное
3.	Adobe Reader	свободно распространяемое	Adobe Systems Incorporated	иностранное
4.	Справочно-правовая система «Консультант Плюс»	лицензионное	НПО «ВМИ»	отечественное
5.	Антивирус Касперского	лицензионное	Лаборатория Касперского	отечественное
6.	Операционная система Microsoft Windows	лицензионное	Microsoft	иностранное
7.	Операционная система семейства Unix	свободно распространяемое	The Linux Foundation	иностранное
8.	Яндекс.Браузер	свободно распространяемое	Яндекс	отечественное
9.	Архиватор 7-Zip	свободно распространяемое	Igor Pavlov	иностранное

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», профессиональных баз данных, информационно-справочных систем

Таблица 13

№ п/п	Наименование	Краткое описание	Режим доступа
1.	Электронно-библиотечная система IPRbooks	Электронно-библиотечная система	http://www.iprbookshop.ru/
2.	Электронно-библиотечная система СамГТУ	Электронная библиотека СамГТУ	https://elib.samgtu.ru/
3.	eLIBRARY.RU	Научная электронная библиотека	http://www.elibrary.ru/

9. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Лекционные занятия

Аудитории для лекционных занятий укомплектованы мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории (наборы демонстрационного оборудования (проектор, экран, компьютер/ноутбук).

Лабораторные занятия

Аудитории для лабораторных работ (компьютерные классы) укомплектованы мебелью и персональными компьютерами с установленным на них необходимым программным обеспечением.

Самостоятельная работа

Помещения для самостоятельной работы оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и доступом к электронной информационно-образовательной среде СамГТУ:

методический кабинет (ауд. 9);

компьютерные классы (ауд. 6, 15).

10. Фонд оценочных средств по дисциплине

Фонд оценочных средств для проведения текущего контроля успеваемости и промежуточной аттестации представлен в Приложении 1.

Полный комплект контрольных заданий или иных материалов, необходимых для оценивания результатов обучения по дисциплине, практике хранится на кафедре-разработчике в бумажном и электронном виде.

Приложение 1

Фонд оценочных средств для проведения текущего контроля успеваемости и промежуточной аттестации

по дисциплине

Б1.О.02.05 «Языки и методы программирования»

Код и направление подготовки (специальность)	**09.03.02 Информационные системы и технологии**
Направленность (профиль)	**Информационные системы и технологии**
Квалификация	**бакалавр**
Форма обучения	**заочная**
Год начала подготовки	**2023**
Выпускающая кафедра	**Инженерные технологии**
Кафедра-разработчик	**Инженерные технологии**
Объем дисциплины, ч. / з.е.	**216 / 6**
Форма контроля (промежуточная аттестация)	**экзамен, экзамен**

1. Перечень компетенций, индикаторов достижения компетенций и признаков проявления компетенций (дескрипторов), которыми должен овладеть обучающийся в ходе освоения образовательной программы

Универсальные компетенции

Таблица 1

Наименование категории (группы) компетенций	Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом				

Общепрофессиональные компетенции

Таблица 2

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности	**ОПК-2.1** Использует и понимает принципы работы информационных технологий и программных средств при решении задач в сфере информационных систем и технологий	**З2 ОПК-2.1** Знать: теорию программно-аппаратного обеспечения информационных систем, автоматизирующих задачи организационного управления и бизнес-процессы в организациях различных форм собственности с целью повышения эффективности деятельности организаций.
		ОПК-2.2 Применяет современные информационные технологии и программные средства отечественного производства при решении задач в сфере информационных систем и технологий	**У1 ОПК-2.2** Уметь: анализировать предметную область информационной системы и учитывать ее специфику для принятия проектных решений в процессе создания и использования; разрабатывать модели информационной системы; выполнять декомпозицию сложной информационной системы **В2 ОПК-2.2** Владеть: методами и средствами обеспечения работы программно-аппаратного обеспечения информационных систем, в том числе отечественного производства, при решении задач профессиональной деятельности
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;	**ОПК-6.1** Разрабатывает алгоритмы и программы, пригодные для практического применения	**З1 ОПК-6.1** Знать: основы алгоритмизации, языки программирования, методы реализации алгоритмов, операционные системы и оболочки, современные программные среды разработки информационных систем и технологий
		ОПК-6.2 Осуществляет отладку и тестирование программного обеспечения	**В1 ОПК-6.2** Владеть: навыками разработки алгоритмов, программирования, отладки и тестирования прототипов программно-технических комплексов задач
		ОПК-6.3 Ведет и использует базы данных и информационные хранилища	**У1 ОПК-6.3** Уметь: применять языки программирования, работать с базами данных, разрабатывать информационные системы и технологии для автоматизации бизнес-процессов, решать прикладные задачи различных классов
ОПК-8	Способен применять математические модели, методы и средства проектирования информационных и автоматизированных систем.	**ОПК-8.1** Разрабатывает математические и имитационные модели процессов в сфере информационных систем и технологий	**З1 ОПК-8.1** Знать: принципы моделирования, классификацию способов представления моделей систем; приемы, методы, способы формализации объектов, процессов, явлений и реализацию их на компьютере; достоинства и недостатки различных способов представления моделей систем; разработку алгоритмов фиксации и обработки результатов моделирования систем; способы планирования машинных экспериментов с моделями

Профессиональные компетенции

Таблица 3

Код компетенции	Наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения
не предусмотрены учебным планом			

Матрица соответствия оценочных средств запланированным результатам обучения

Таблица 4

Код и индикатор достижения компетенции	Оценочные средства		
	Раздел 1, 2	**Раздел 3**	**Промежуточная аттестация**
	Вводный. Основы алгоритмизации. Методы реализации алгоритмов.	Основы программирования на языке высокого уровня	
	Тестирование		Вопросы к экзамену
ОПК-2.1	32 ОПК-2.1	32 ОПК-2.1	32 ОПК-2.1
ОПК-2.2	У1 ОПК-2.2 В2 ОПК-2.2	У1 ОПК-2.2 В2 ОПК-2.2	У1 ОПК-2.2 В2 ОПК-2.2
ОПК-6.1	З1 ОПК-6.1	З1 ОПК-6.1	З1 ОПК-6.1
ОПК-6.2	В1 ОПК-6.2	В1 ОПК-6.2	В1 ОПК-6.2
ОПК-6.3	У1 ОПК-6.3	У1 ОПК-6.3	У1 ОПК-6.3
ОПК-8.1	З1 ОПК-8.1	З1 ОПК-8.1	З1 ОПК-8.1

2. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующие процесс формирования компетенций в ходе освоения образовательной программы

2.1. Формы текущего контроля успеваемости

Промежуточная аттестация проводится в виде письменного/устного опроса, тестирования и представляет собой ответы на 2 вопроса и выполнение тестовых заданий.

Примерные перечень вопросов для тестирования

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Б	Как обозначается комментарий в Python? а) // б) # в) /! г) /*	ОПК-6	2
2.	В	Как объявить целочисленную переменную в Python? а) int x = 5 б) x = int(5) в) x = 5 г) var x = 5	ОПК-6	2
3.	В	Какая функция используется для нахождения длины списка в Python? а) length(x) б) size(x) в) len(x) г) count(x)	ОПК-6	2
4.	А	Какой оператор используется для объединения двух списков в Python? а) + б) * в) - г) /	ОПК-6	2
5.	А	Какой оператор используется для проверки равенства в Python? а) == б) = в) === г) !=	ОПК-6	2
6.	Б	Какая команда используется для вывода данных на экран в Python? а) display()	ОПК-6	2

		б) print() в) write() г) echo()		
7.	В	Как объявить строковую переменную в Python? а) string x = "hello" б) x = str("hello") в) x = "hello" г) var x = "hello"	ОПК-6	2
8.	В	Как объявить список (array) в Python? а) list x = [1, 2, 3] б) x = array(1, 2, 3) в) x = [1, 2, 3] г) var x = {1, 2, 3}	ОПК-6	2
9.	А	Что будет выведено на экран? print(1^4) а) 5 б) 3 в) 7 г) 0	ОПК-6	2
10.	Б	Какой результат выражения 'Python'[:3] * 2? а) 'thonthon' б) 'PytPyt' в) 'Python' г) 'onon'	ОПК-6	2

2.2. Формы промежуточной аттестации

Промежуточная аттестация проводится в виде письменного/устного опроса, тестирования и представляет собой ответы на 2 вопроса и выполнение тестовых заданий.

1 семестр

Примерный перечень вопросов к экзамену

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Алгоритм — это последовательность шагов или инструкций, которые решают конкретную задачу или проблему. Алгоритм должен быть четко определен, точен, полон и понятен человеку. Алгоритмический язык — это язык программирования, специально разработанный для записи алгоритмов. Он обычно содержит команды, операторы и функции, которые позволяют программисту создавать и исполнять алгоритмы.	Дать определения следующих понятий: алгоритм, алгоритмический язык	ОПК-2	2
2.	Язык программирования — это формальный язык, который предназначен для записи компьютерных программ. Языки программирования могут быть общего назначения (как, например, Java, Python, C++), специализированные (например, SQL для работы с базами данных) или даже разработанные для конкретных задач (например, язык R для анализа данных).	Дать определение понятия язык программирования	ОПК-2	2
3.	Машинный код — это набор инструкций, которые компьютер может понимать и исполнять. Машинный код записывается в двоичной системе и состоит из набора инструкций, содержащих определенные битовые значения, которые представляют операции процессора, память и ввод-вывод устройств. Машинный код непосредственно исполняется процессором.	Дать определение понятия машинного кода.	ОПК-2	2
4.	Структурное программирование — это методология программирования, которая подразумевает разработку программы через последовательность логических шагов. Она основывается на подходе сверху вниз, когда программа организуется в виде модулей, каждый из которых соответствует определенной части алгоритма.	Методы программирования. Структурное программирование это…	ОПК-2	2
5.	Объектно-ориентированное программирование — это подход к программированию, который основывается на использовании объектов, которые управляются с помощью классов. Классы содержат описание переменных и методов объекта, а объекты создаются на основе этих классов. ООП позволяет разрабатывать более гибкие и расширяемые приложения.	Методы программирования. Объектно-ориентированное программирование это…	ОПК-2	2

6.	Функциональное программирование — это подход к программированию, основанный на использовании функций как основного строительного блока программы. Функции обрабатывают данные и возвращают результаты, при этом они не изменяют состояние программы и не взаимодействуют с другими функциями.	Методы программирования. Функциональное программирование это …	ОПК-2	2
7.	Декларативное программирование — это методика описания задачи на естественном языке, где алгоритмы выражаются в виде правил и ограничений, а не последовательности операций. Этот подход упрощает разработку и позволяет программистам решать задачи с помощью описания их сущностей, не задумываясь о том, как достигнуть результата.	Методы программирования. Декларативное программирование это …	ОПК-2	2
8.	Особенности ООП: Модульность - в ООП программа разбивается на независимые модули, каждый из которых может быть разработан и тестирован отдельно. Гибкость - объекты могут быть легко модифицированы и обновлены с минимальными затратами на изменение других частей программы. Повторное использование кода - объекты могут быть повторно использованы в разных проектах и приложениях. Поддержка параллельного программирования - ООП позволяет создавать легко масштабируемые и параллельные системы. В целом, использование ООП позволяет программистам создавать более эффективные и легко понятные программы. ООП также способствует повышению производительности и снижению стоимости разработки.	Перечислите особенности объектно-ориентированного программирования.	ОПК-2	2
9.	Инкапсуляция — это способность объекта скрыть свои данные от других объектов. Объект может предоставлять доступ к своим данным только через определенные методы. Наследование — это способность нового объекта или класса наследовать свойства и методы от уже существующего объекта или класса. Полиморфизм — это способность программного кода реагировать по-разному на вызовы одного и того же метода в зависимости от объекта, на котором он вызывается.	Три основных принципа объектно-ориентированного программирования	ОПК-2	
10.	Программирование на языке высокого уровня (Higher-level programming) — это подход к написанию программного кода, который позволяет разработчикам использовать абстрактные концепции и инструменты для написания программ, не углубляясь в низкоуровневые детали. Это означает, что программистам не нужно беспокоиться о работе с памятью напрямую или другими низкоуровневыми аспектами аппаратного обеспечения, такими как ввод-вывод. Особенности программирования на языке высокого уровня включают в себя: 1. Абстракции: языки высокого уровня предоставляют высокоуровневые конструкции, которые позволяют проще и быстрее описывать и реализовывать сложные алгоритмы и структуры данных. 2. Переносимость: программы, написанные на языке высокого уровня, могут быть запущены на различных платформах и операционных системах, не требуя переписывания кода для каждой платформы. 3. Удобство: языки высокого уровня предоставляют богатые библиотеки, которые содержат уже готовые реализации множества различных функций, что сокращает время разработки и упрощает процесс написания кода. 4. Уровень абстракции: языки высокого уровня скрывают от программиста детали взаимодействия с аппаратным обеспечением, такие как управление ресурсами и вызовы системных функций. 5. Оптимизация: компиляторы языков высокого уровня позволяют автоматически оптимизировать код для улучшения производительности и сокращения использования ресурсов. Программирование на языке высокого уровня имеет и некоторые недостатки. Например, высокоуровневые языки обычно более медленные, чем низкоуровневые языки.	Особенности программирования на языке высокого уровня	ОПК-2	2
11.	Синтаксическое правило - это правило или набор правил, которые определяют, каким образом должен быть оформлен текст программы на данном языке программирования. Синтаксические правила определяют правильную структуру программы, устанавливают порядок операций, определяют правила комментирования и форматирования, и т.д.	Что такое: синтаксическое правило в программировании?	ОПК-2	2
12.	Семантика - в программировании семантика относится к значению,	Что такое семантика	ОПК-2	2

	которое должна осуществлять программа. Семантика определяет, что должна делать программа и какие результаты она должна получать.	в программировании?		
13.	Идентификатор - это имя, которое используется для идентификации переменных, функций и других элементов программы. Идентификаторы должны быть уникальными и соответствовать правилам языка программирования. Идентификаторы могут состоять из букв, цифр, нижнего подчеркивания и других символов (в зависимости от языка программирования).	Идентификатор это …	ОПК-2	2
14.	Алфавит языка — это набор символов, которые могут использоваться для написания текста на данном языке. Алфавит языка программирования обычно состоит из символов латинского алфавита (A-Z, a-z), цифр (0-9) и специальных символов (например, знаков препинания, математических операций, скобок и т.д.). Алфавит языка определяет в каком виде и какие символы могут использоваться в идентификаторах, строках, комментариях и других элементах программы.	Алфавит языка это …	ОПК-2	2
15.	Компилятор и интерпретатор - это два различных способа преобразования исходного кода программы в машинный код, который может быть исполнен процессором. Однако у них есть ряд отличий: Принцип работы: Компилятор преобразует весь исходный код программы в машинный код одним проходом и сохраняет полученный код в файле или памяти. Интерпретатор же анализирует и исполняет код построчно, каждый раз переводя строку кода в машинный код во время выполнения программы. Время компиляции и исполнения: Компилятор требует времени для того, чтобы преобразовать всю программу в машинный код, а затем может мгновенно выполнять скомпилированный код. Интерпретатор не требует времени на компиляцию, но каждый раз, когда программа запускается, он анализирует и исполняет каждую строку кода заново, что может привести к значительному замедлению работы программы. Платформенная зависимость: Компилятор порождает машинный код определенной платформы, который может быть запущен только на этой платформе. Интерпретатор кросс-платформенный и может работать на различных платформах, так как он анализирует и исполняет код построчно. Анализ кода: Компилятор более тщательно анализирует код на этапе компиляции и может обнаруживать большое количество ошибок и предостерегать от них исполнение программы. Интерпретатор обычно анализирует код на этапе исполнения, так что ошибки могут проявиться в процессе работы программы, что делает интерпретацию менее безопасной. Оптимизации: Компилятор может применять более сложные оптимизации кода, так как он анализирует все программу целиком, в то время как интерпретатор может выполнять только простые оптимизации кода. В целом, компиляторы часто используются для создания скоростных приложений, так как скомпилированный код обычно работает быстрее чем интерпретированный. Интерпретаторы же используются для интерактивных приложений, таких как интерактивные обучающие программы и отладчики, где важно иметь быструю обратную связь на каждое действие пользователя.	В чем отличие компилятора от интерпретатора?	ОПК-2	2
16.	Язык программирования (ЯП) высокого уровня — это ЯП, который воспринимается человеком и предназначен для написания более абстрактных и высокоуровневых программ, чтобы упростить написание кода. Они позволяют разработчикам описывать сложные структуры данных и алгоритмы более естественным языком и с более высоким уровнем абстракции. Примеры ЯП высокого уровня: • Python • Ruby • Java • C# • PHP.	Дать определение языка программирования высокого уровня и привести примеры	ОПК-2	2
17.	Язык программирования (ЯП) низкого уровня - это ЯП, который более прямо связан с аппаратным обеспечением и имеет более низкий уровень абстракции. Они предназначены для создания более эффективных программ, манипулирующих непосредственно с оперативной памятью, процессором и устройствами ввода/вывода. Примеры ЯП низкого уровня:	Дать определение языка программирования низкого уровня и привести примеры	ОПК-2	2

	• Ассемблер • C • C++.			
18.	Языки программирования (ЯП) высокого и низкого уровня отличаются не только синтаксисом, но и тем, насколько близко они связаны с аппаратным обеспечением компьютера. ЯП высокого уровня обычно предоставляют интерфейсы, которые упрощают работу с оперативной памятью, управление потоками, обработку ошибок и взаимодействие с устройствами ввода/вывода, что упрощает разработку, но может привести к более низкой производительности программы. ЯП низкого уровня, с другой стороны, обычно позволяют разработчикам непосредственно управлять программой, достигая более высокой производительности, но требуя также более сложной работы по управлению памятью и низко уровневым кодом.	Отличия языков высокого и низкого уровня	ОПК-2	2
19.	Функции и процедуры — это ключевые элементы структурного программирования, позволяющие переиспользовать код и повышать читаемость программы. Процедура - это логически связанная последовательность действий, выполняющая определенную задачу. Она может использоваться в различных частях программы, что делает код более модульным. Процедуры предназначены для выполнения определенных действий, может иметь параметры для приема входных значений и может возвращать результаты. Функция - это специальный тип процедуры, который возвращает значение и обычно используется для выполнения математических вычислений или проверки условий. Функция принимает параметры для приема входных значений, а затем возвращает результат в вызывающую программу или процедуру.	Функции и процедуры в структурном программировании	ОПК-2	2
20.	Преимущества использования функций и процедур: • Повторное использование кода: функции и процедуры позволяют избежать дублирования кода, упрощая программирование и облегчая поддержку кода. • Модульность: функции и процедуры могут быть использованы как самостоятельные модули, что упрощает создание более сложной логики с использованием меньших комбинаций кода. • Легкость чтения и поддержки: функции и процедуры позволяют создавать более простые и понятные программы с легко читаемым и понятным кодом. • Надежность: функции и процедуры делают программы более надежными, поскольку каждая функция и процедура может быть протестирована и проверена отдельно.	Преимущества использования функций и процедур	ОПК-2	2
21.	Интегрированная среда разработки (Integrated Development Environment или IDE) - это программное обеспечение, которое предоставляет инструменты для развития, тестирования и отладки программного обеспечения в единой среде. Они обычно располагают в себе множество инструментов, таких как редактор кода, отладчик, компилятор, средства автодополнения, система контроля версий и другие. Структура IDE включает следующие основные компоненты: 1. Редактор кода - это основной компонент IDE, который используется для создания и редактирования исходного кода, а также связанных файлов, таких как HTML, CSS, XML, PHP и т.д. 2. Компилятор - инструмент, который преобразует программный код из исходного кода в бинарный код, понятный компьютеру. В некоторых IDE компилятор может работать автоматически, при сохранении изменений в исходном коде. 3. Отладчик - позволяет программистам отслеживать работу программы на этапе выполнения, что помогает устранять ошибки и дефекты в коде, сокращать время разработки. 4. Система контроля версий - обеспечивает возможность регистрации копий версий файлов, отслеживание их изменений и их восстановления. Средства авто-дополнения - позволяют программистам быстро завершать код, подбирая ключевые слова и методы из предоставляемой системой библиотеки и функций.	Понятие и структура интегрированной среды разработки.	ОПК-2	2
22.	Для каждого языка программирования существует своя IDE, но сегодня многие IDE предоставляют возможность разработка на нескольких языках. Примеры некоторых IDE: • Microsoft Visual Studio - широко используемая IDE для Windows,	Приведите примеры IDE	ОПК-2	2

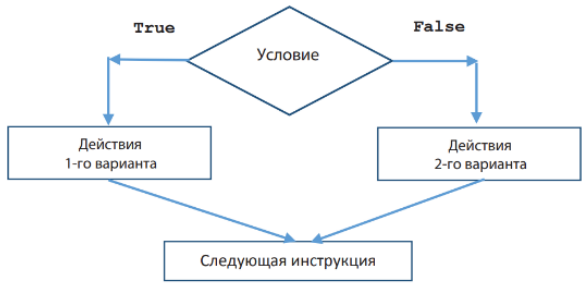
	поддерживающая множество языков программирования. • Eclipse - распространенная IDE, поддерживающая языки Java, C/C++, Python и другие. • NetBeans - популярная IDE для языка Java, предоставляющая функциональность для создания приложений на платформе Java SE, Java EE и HTML5. • PyCharm - IDE для языка Python, поддерживающая различные фреймворки, такие как Django и Flask. • Xcode - IDE для разработки приложений для операционных систем iOS и macOS, поддерживает языки Objective-C, Swift, C++, C и другие			
23.	Этапы решения задачи: 1. Понимание задачи: в этом этапе нужно убедиться, что вы правильно поняли поставленную задачу. Необходимо проанализировать проблему и уточнить дополнительные детали, если это необходимо. 2. Планирование: на этом этапе вы должны разработать план решения задачи. Вам нужно представить общий подход, создать список методов и инструментов, которые вы можете использовать для выполнения задачи. 3. Разработка: этот этап включает создание кода, пишущегося для решения поставленной задачи. Это может включать разработку алгоритмов, написание кода, интеграцию различных компонентов и т.д. 4. Тестирование: после написания кода нужно протестировать его, чтобы убедиться, что он работает правильно и соответствует требованиям. Если ошибки или недоработки были обнаружены, то их нужно исправить. 5. Поддержка: после того, как задача решена и проверена, нужно заботиться о поддержке приложения и дополнительной отладке в случае наличия ошибок или проблем.	Этапы решения задачи.	ОПК-2	2
24.	Виды ошибок: 1. Синтаксические ошибки: это ошибки, которые возникают, когда код нарушает правила языка программирования. Они часто возникают из-за неправильной синтаксиса конструкций языка, пропущенных скобок, забытых точек с запятой и других подобных ошибок. 2. Логические ошибки: это ошибки, которые возникают, когда программист не учел все возможные пути, по которым может протекать управление приложением. Логические ошибки могут произойти, если программист пропустит важную деталь. 3. Расходящиеся ошибки: это ошибки, которые возникают в результате случайного изменения значения переменной или данных. Они могут возникать всех типов, включая установку неправильных параметров, работу со старыми или нерабочими данными, используя неправильные алгоритмы, решения и т.д. 4. Локальные ошибки: это ошибки, которые возникают только в определенных условиях и не влияют на другие части программы. Например, локальной ошибкой может быть неправильное формирования запроса к базе данных на сервере. 5. Ошибки при выполнении: это ошибки, которые возникают во время выполнения приложения. Они могут возникнуть из-за неправильной конфигурации окружения, неизвестного пользователю ввода данных, нехватки памяти и других причин.	Виды ошибок в программе	ОПК-2	2
25.	Тестирование — это процесс проверки программного обеспечения, выполняемый с целью выявления ошибок, дефектов и недостатков в приложении. Процесс тестирования включает в себя шаги: 1. Планирование тестирования: это планирование, включающее цели тестирования, временной график, бюджет, предполагаемые результаты тестирования и другие планировочные моменты. 2. Обзор приложения: это этап, на котором тестирующее лицо изучает приложение и фиксирует возможные ошибки, дефекты и недостатки. 3. Написание тестов: это этап, на котором на основе обзора приложения и планирования тестирования разрабатываются тестовые кейсы, для выявления наиболее очевидных и важных ошибок. 4. Выполнение тестов: это этап, на котором тестировщик выполняет все написанные тесты и записывает результаты тестирования. 5. Запись отчета: на этом этапе, тестировщик записывает отчет о результатах тестирования вместе с выявленными ошибками,	Тестирование. Какие шаги выполняются в процессе тестирования программы?	ОПК-2	2

	дефектами и отклонениями. 6. Исправление ошибок: после тестирования необходимо исправить все выявленные ошибки и дефекты в приложении. 7. Повторный тест: после исправления ошибок необходимо провести повторное тестирование, чтобы убедиться в успешности исправлений и отсутствии новых ошибок. Осуществляется проверка, чтобы убедиться в корректности функционирования программы и работоспособности.			
26.	Жизненный цикл программного обеспечения включает в себя все этапы его развития: от возникновения потребности в нем до полного прекращения его использования вследствие морального старения или потери необходимости решения соответствующих задач	Жизненный цикл программного обеспечения	ОПК-2	2

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Для начала работы с Python потребуется установить среду разработки и настроить необходимые параметры. Загрузить и установить Python можно с официального сайта Python, следуя инструкциям по установке для операционной системы. Можно выбрать среду разработки по своему усмотрению. Среды разработки для Python: IDLE: Интегрированная среда разработки, поставляемая вместе с Python, PyCharm: интегрированная среда разработки от JetBrains, Visual Studio Code: среда разработки от Microsoft, Jupyter Notebook: интерактивная среда для работы с блокнотами Python. Настройка пользовательских параметров зависит от выбранной среды разработки. Вот общие шаги: Установка виртуальной среды (рекомендуется): Создайте виртуальную среду для изоляции проектов Python с помощью venv или virtualenv. Это поможет избежать конфликтов зависимостей между проектами. Настройка параметров интерпретатора: необходимо указать путь к интерпретатору Python, который был установлен в первом шаге. Установите необходимые пакеты и плагины для работы в среде разработки. Создание исходного файла Python. В IDLE, выбрать "New File" из меню "File", затем ввести свой код и сохранить файл с расширением .py. В PyCharm или Visual Studio Code, создать новый файл, ввести свой код и сохранить его с расширением .py. Можно использовать командную строку. Можно использовать текстовый редактор, например, notepad, nano, vim или code (для VS Code), чтобы создать и редактировать файл с расширением .py. При использовании Jupyter Notebook: запустите Jupyter Notebook и создайте новый блокнот. В нем можно писать и выполнять код Python по ячейкам.	Инсталляция среды разработки Python. Интерфейс и настройка пользовательских параметров. Способы создания исходного файла в Python.	ОПК-6	2
2.	В скобках в функции print() в Python можно указывать: 1. Одно или несколько значений, которые нужно вывести на экран, разделенных запятой: print("Hello, world!") Выведет на экран строку "Hello, world!" 2. Значения переменных, разделенные запятой: x = 5 y = "five" print(x, y) Выведет на экран значение переменной x (5) и строку "five", разделенные пробелом. 3. Строку с использованием символа форматирования (фигурные скобки {}) и передать значения, которые будут подставлены внутрь этих скобок: name = "Alice" age = 25 print("My name is {} and I'm {} years old.".format(name, age)) Выведет на экран строку "My name is Alice and I'm 25 years old." 4. Символ переноса строки '\n' для переноса текста на новую строку:	Что можно указывать в скобках в инструкции print() в Python? Что будет выведено на экран в том или ином случае?	ОПК-6	2

	print("Hello,\nworld!") Выведет на экран две строки: "Hello," и "world!", каждая на отдельной строке. 5. Значение параметра end, которое задает символ, который будет добавлен в конец строки (по умолчанию это символ переноса строки '\n'): print("Hello,", end="") print("world!") Выведет на экран две строки: "Hello," и "world!", но они будут выведены на одной строке без пробела между ними. 6. Значение параметра sep, которое задает разделитель между значениями, переданными в print(): print("apple", "banana", "cherry", sep=", ") Выведет на экран строку "apple, banana, cherry".			
3.	а) 31 1577 б) 51 36 77 45 в) F(3) = 5	Определите, что будет выведено на экран в результате выполнения следующих инструкций: а) print(31, 15, end = ' ') print(77) б) print(51, 36) print(77, 45, end = ' ') в) a = 5; b = 3 print('F(', b, ') = ',a, sep = ' ')	ОПК-6	2
4.	а) -1/*x* **2 б) (-b + (b**2 – 4*a*c)**(1/2))/(2*a) в) a/(b*c) г) (-b+1/a)*c/2 или (-b+1/a)/(2/c) д) 1/(1+(a+b)/2)	Запишите в одну строку по правилам языка Python следующие арифметические выражения: а) $\frac{-1}{x^2}$; б) $\frac{-b+\sqrt{b^2-4ac}}{2a}$; в) $\frac{a}{bc}$; г) $\frac{-b+\frac{1}{a}}{\frac{2}{c}}$; д) $\frac{1}{1+\frac{a+b}{2}}$.	ОПК-6	2

5.	В случаях, когда в программе возможны два варианта действий необходимо использовать инструкцию if в следующей форме:
	if <условие>:
	<Действия 1-го варианта (1-я серия инструкций)>
	else:
	<Действия 2-го варианта (2-я серия инструкций)
	Приведенный вариант инструкции if называют «полный вариант».
		В каких случаях в программе используется полный вариант инструкции if? Как он оформляется?	ОПК-6	2

6.		Нарисуйте графическую схему выполнения полной инструкции if.	ОПК-6	2

7.	a=3, b=5	if a<0:
		a=-a
		if b<0:
		b=-b
		while a > b:
		a=a-b
		В результате выполнение данного алгоритма с начальными значениями a=-13; b=5 переменные примут следующие значения:
			ОПК-6	2

8.	n=4, s=10	В результате выполнения фрагмента программы:	ОПК-6	2

		s=1 n=1 for i in range(2, 5): n=n+1 s=s+i переменные n, s примут значения		
9.	n=6, s=5	В результате выполнения фрагмента программы: s=0 n=1 for i in range(2, 4): n=n*i s=s+i переменные n, s примут значения	ОПК-6	2
10.	n=2, s=2	В результате выполнения фрагмента программы: s=1 n=0 for i in range(1, 3): n = n +1 s=s*i переменные n, s примут значения	ОПК-6	2
11.	n=2, s=4	В результате выполнения фрагмента программы: s=0 n=0 for i in range(1, 4, 2): n=n+1 s=s+i переменные n, s примут значения	ОПК-6	2
12.	n=3, s=7	В результате выполнения фрагмента программы: s=1 n=1 for i in range(2, 6, 2): n=n+1 s=s+i переменные n, s примут значения	ОПК-6	2
13.	n=2, s=6	В результате выполнения фрагмента программы: s=0 n=0 for i in range(2, 6, 2): n=n+1 s=s+i переменные n, s примут значения	ОПК-6	2

14.	c=1, d=2	if a<b:
		c=b-a
		else:
		c=2*(b-a)
		d=0
		while c>d:
		d= d +1
		c=c-1
		В результате выполнение данного алгоритма с начальными значениями a=3; b=6 переменные c, d примут следующие
			ОПК-6	2

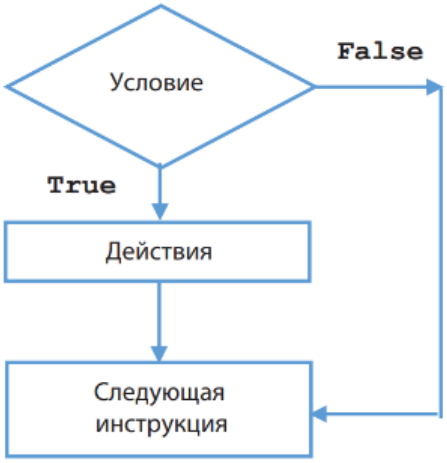
		значения:		
15.	Значение переменных z и s. Значение переменной z равно произведению значений переменной x и синуса от x, найденного в радианах. Значение переменной S равно сумме **двух** значений Z	Приведённый фрагмент программы: import math s=0 for i in range(2, 6, 2): z=x*math.sin(x) s=s+z находит для x, введенного ранее…	ОПК-6	2

16.	Значение переменных z и s. Значение переменной z равно произведению значений переменной x и синуса от x, найденного в радианах. Значение переменной S равно произведению двух значений Z	Приведённый фрагмент программы:
		import math
		s=1
		x=5
		for i in range(2, 6, 2):
		 z=x*math.sin(x)
		 if z!=0:
		 s=z*s
		находит:
			ОПК-6	2

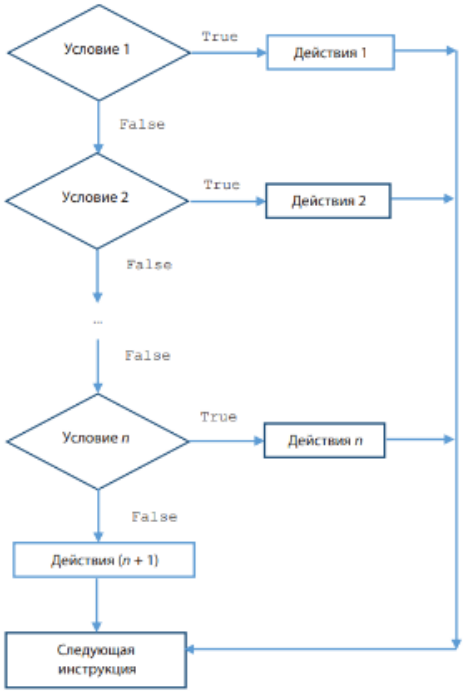
17.	В инструкции if возможно также использование так называемых «сложных условий» – состоящих из двух или нескольких простых условий, соединенных служебными словами (логическими операторами) and (И), or (ИЛИ) или not (НЕ)	Что такое сложное условие? Какие логические операции могут использоваться в нем?	ОПК-6	2
18.	При проверке сложного условия сначала выполняются операции сравнения (<, <=, >, >=, ==, !=), а затем – логические операции в таком порядке: сначала все операции с оператором not, затем – с оператором and, и в самом конце – с оператором or (во всех случаях – слева направо). Для изменения порядка действий используют круглые скобки.	Каков порядок (приоритет) выполнения логических операций? Как изменить этот порядок?	ОПК-6	2

19.	В качестве условия в инструкции if можно использовать также:
	1) логические функции, то есть функции, возвращающие результат логического типа:
	n = int(input('Введите целое число '))
	if Chet(n):
	 print('Это число четное')
	else:
	 print('Это число нечетное')
	где Chet() – функция, возвращающая результат True, если ее параметр является четным числом, и False – в противном случае;
	2) оператор in (оператор проверки принадлежности), который проверяет, принадлежит ли некоторый объект (число, символ, переменная и т. п.) набору значений (списку, строке, диапазону чисел и т. п.):
	a = 3
	if a in range(10): #Если значение переменной a входит
	 #в диапазон значений, возвращаемый
	 #функцией range()
	3) операторы is/is not (операторы проверки идентичности), которые определяют, ссылаются ли (или не ссылаются) две переменные на один и тот же объект.
		Что может быть использовано в инструкции if, кроме простых и сложных условий?	ОПК-6	2

20.	Когда какие-то действия в программе выполняются не всегда, а только при определенном условии) используется инструкция if в следующей форме: if <условие>: <Действия (серия инструкций)> Такой вариант инструкции if называют «неполный вариант».	В каких случаях в программе используется неполный вариант инструкции if? Как он оформляется?	ОПК-6	2

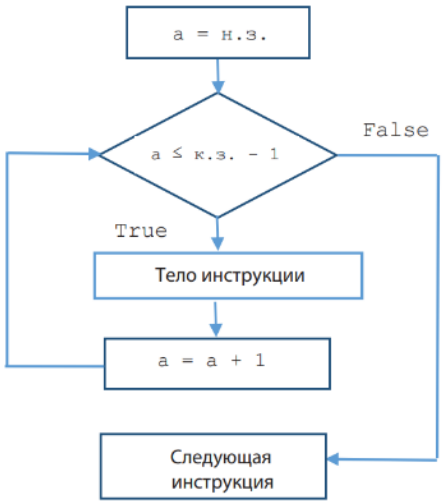
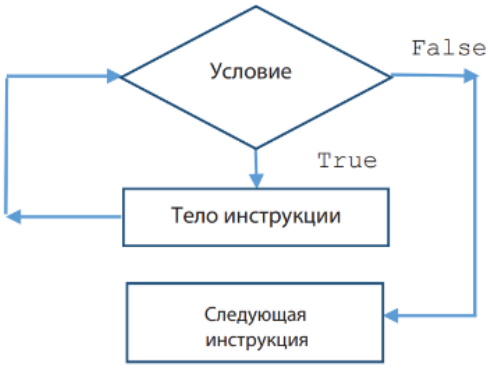
21.		Нарисуйте графическую схему выполнения неполного варианта инструкции if.	ОПК-6	2

22.	В общем случае вложенная инструкция if выглядит следующим образом:
	if <условие 1>:
	<действия 1>
	elif <условие 2>:
	<действия 2>
	elif ...:
	...
	elif <условие N>:
	<действия N>
	else:
	<действия (N+1)>
		В каких случаях в программе используется вложенная инструкция if? Как она оформляется?	ОПК-6	2

23.		Нарисуйте схему вложенной конструкции if	ОПК-6	2

24.	Переменным величинам в программе дадим имена a, b (заданные числа) и Max (максимальное из них).
	Так как возможны два варианта ответа, то в программе следует использовать полный вариант инструкции if:
	...
	if a > b:
	Max = a
	else:
	Max = b
	...
	Или: в Python есть стандартная функция max(), которая возвращает максимальное значение среди указанных для нее аргументов. Она вызывается так:
	Max = max(a, b, ...)
		Определить максимальное значение из двух заданных различных вещественных чисел	ОПК-6	2

25.	Функция range() возвращает набор целых чисел, образующих арифметическую прогрессию. Например, вызов функции:	Опишите функцию range()	ОПК-6	2

	range(7) получит числа 0, 1, 2, …, 6. В общем случае – для получения n последовательных целых чисел, начиная с 0: range(n) Также можно задать первое число в последовательности и шаг. Например: range(a, b, -1)			
26.	а) девять раз; б) десять раз	Сколько раз будет выполняться тело инструкции for со следующим «заголовком»: а) for a in range(1, 10): б) for b in range(10, 20):	ОПК-6	2
27.	Инструкция (цикл) for используется в программе, когда известно количество повторений каких-то действий или известен набор значений, для которых должны выполняться повторяющиеся действия. Общий вид этой инструкции: for <параметр> in <набор значений параметра>: <тело инструкции> #Записывается с отступом где <тело инструкции> – действия, повторяющиеся при работе инструкции (это могут быть любые инструкции Python); <параметр> – имя величины, меняющейся при повторении действий; <набор значений параметра> – набор значений параметра инструкции, для которых проводится повторение.	В каких случаях используют инструкцию for? Каков ее общий вид?	ОПК-6	2
28.	 a – параметр, н.з. – начальное значение, к.з. – конечное значение параметра	Нарисуйте графическую схему выполнения инструкции for	ОПК-6	2
29.	В случаях, когда заранее не известно количество повторений применяется инструкция while. Общий вид: while <условие>: <тело инструкции> где <условие> – условие, при котором выполняется <тело инструкции>.	В каких случаях используется конструкция циклов while?	ОПК-6	2
30.		Нарисуйте графическую схему выполнения инструкции while	ОПК-6	2
31.	Инструкцию while называют «инструкцией цикла с предусловием», или просто «циклом с предусловием», потому что проверка условия проводится сначала, до выполнения тела цикла. Иными словами, действия повторяются, пока соблюдается заданное условие	Почему инструкцию while называют «циклом с предусловием»?	ОПК-6	2

32.	Инструкция break позволяет осуществить досрочный выход из цикла и организации цикла с постусловием. Например:
	while True:
	  print('Задайте значение коэффициента a уравнения:')
	  a = float(input())
	  if a != 0:
	  break
	Инструкция break может быть применена для досрочного выхода из цикла for. Это удобно делать, например, в задачах поиска некоторого заданного значения в наборе значений. В таких случаях инструкция for оформляется следующим образом:
	for <всех значений в наборе>:
	  if <условие поиска значения> истинно:
	  break
	...
		Для чего используется инструкция break?	ОПК-6	2

33.	Например, при замене в программе инструкции for на инструкцию while (или только оформляя тело инструкции while), часто забывают менять значение величины, от которой зависит заданное в инструкции условие. В таком случае инструкция будет выполняться бесконечно долго (условие, записанное в ней, никогда не станет ложным). В этом случае говорят, что «программа зациклилась».	Что такое «зацикливание программы»? В каком случае оно может произойти?	ОПК-6	2
34.	Циклом можно назвать инструкции, обеспечивающие в программе повторение одних и тех же или аналогичных действий. В теле каждой из них могут выполняться любые действия, в том числе повторяющиеся, то есть также может использоваться инструкция цикла. В этом случае получается программная конструкция, которую называют «цикл в цикле», или «вложенный цикл». Соответственно, для вложенного цикла определяют наружную и внутреннюю инструкции.	Какую программную конструкцию называют «цикл в цикле», или «вложенный цикл»? Как называют ее части?	ОПК-6	2
35.	Для получения случайных чисел в программах на Python используют функции: • random() – возвращает случайное вещественное число x – такое, что 0 ≤ x < 1; • randint(a, b) – возвращает случайное целое число из интервала [a, b] (a и b – целые числа, a > b). Чтобы их использовать, надо записать в программе: from random import random, randint	Случайные числа в Python	ОПК-6	2
36.	int() – переводит строку, указанную в скобках, в целое число; float() – переводит строку в вещественное число, str() – переводит целое или вещественное число в строку.	Опишите стандартные функции int(), float(), str()	ОПК-6	2
37.	Список — структура данных, предназначенная для хранения упорядоченных наборов элементов. Элементы списка индексированы, то есть имеют порядковый номер. Нумерация всегда начинается с нуля. Значения могут быть разных типов, например в одном списке может быть как и int(целое число) так и float(число с плавающей точкой) и string(строка, буквы). Все данные в списке упорядоченные индексацией. Нумерация начинается с нуля.	Что такое список в Python?	ОПК-6	2
38.	Создать список можно несколькими способами. Например, можно обработать любой итерируемый объект (например, строку) встроенной функцией list: list('список') ['с', 'п', 'и', 'с', 'о', 'к'] Либо так: my_list = [1, 12.5, ‘Hello’] Числа в список вносятся через запятую, без кавычек, а строки с одинарными либо двойными кавычками. И еще один способ создать список — это генераторы списков. Генератор списков - способ построить новый список, применяя выражение к каждому элементу последовательности a = [i for i in range(n)]	Как создать список?	ОПК-6	2
39.	Если в ходе выполнения программы необходимо добавлять в имеющийся список новые элементы, это можно сделать с помощью метода append() my_list.append(5) #Добавлен новый элемент списка my_list со значением 5	В каких случаях применяется метод append()?	ОПК-6	2
40.	Представление списков – лучший способ улучшить читаемость кода. Поэтому его стоит использовать всякий раз, когда у вас есть множество данных, которые нужно проверить с помощью одной и той же функции или логики. Списки лучше подходят для хранения данных разного типа нежели, массивы. С массивами ничего	Какие преимущества дает использование списка в Python?	ОПК-6	2

	подобного не получится. Однако массивы быстрее и лучше подходят для больших последовательностей данных. Большая гибкость списков позволяет модифицировать данные. Массивы более компактны в плане использования памяти по сравнению со списками. Но списки легко вывести без использования явного цикла. А чтобы вывести элементы массива, нужно пройтись по нему циклом.			

Примерный перечень тестовых заданий к промежуточной аттестации

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Б	Что такое Python? а) Язык программирования для создания веб-сайтов б) Язык программирования для научных вычислений в) Язык программирования для создания десктопных приложений г) Язык программирования для работы с базами данных	ОПК-2	2
2.	А	Кто является автором Python? а) Guido van Rossum б) Bill Gates в) Larry Page г) Mark Zuckerberg	ОПК-2	2
3.	А	Какой тип языка программирования относится Python? а) Скриптовый б) Компилируемый в) Ассемблерный г) Объектно-ориентированный	ОПК-2	2
4.	А	Как называется главный интерпретатор для Python? а) CPython б) Jython в) IronPython г) PyPy	ОПК-2	2
5.	А	Какой из вариантов подключения модуля правильный? а) import random б) from random import as rand в) import random from random г) random import	ОПК-2	2
6.	В	Как происходит процесс присваивания в Python? а) Данные перемещаются, не сохраняя предыдущую связку б) Данные копируются в) Данные связываются ссылками на объекты г) Данные перемещаются, сохраняя предыдущую связку	ОПК-2	2
7.	А	Какой из операторов производит немедленный выход из цикла? а) break б) continue в) pass г) the end	ОПК-2	2

2 семестр
Примерный перечень вопросов к экзамену

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Математическая модель — это абстрактное описание реальной системы или явления в терминах математики. Она может быть использована для анализа, прогнозирования или оптимизации деятельности системы. В проектировании информационных и автоматизированных систем математические модели могут	Что такое математическая модель, и как она используется в проектировании информационных и	ОПК-8	2

	помочь оценить эффективность предлагаемых решений и принять лучшие решения	автоматизированных систем?		
2.	Математическая модель должна удовлетворять по крайней мере двум требованиям: реалистичности и реализуемости. Под реалистичностью понимается правильное отражение моделью наиболее **существенных черт** исследуемого явления. Реализуемость достигается разумной абстракцией, отвлечением от второстепенных деталей, чтобы свести задачу к проблеме с известным решением. Условием реализуемости является возможность практического выполнения необходимых вычислений за отведенное время при доступных затратах требуемых ресурсов	Каким требованиям должна удовлетворять математическая модель?	ОПК-8	2
3.	Алгоритм — это определенным образом организованная последовательность действий, которая за конечное число шагов приводит к решению задачи.	Что такое «алгоритм решения задачи»?	ОПК-8	2
4.	Словесно - формульный; структурный или блок - схемный; с помощью графов - схем; на специальном алгоритмическом языке.	Какие способы записи алгоритма вы знаете?	ОПК-8	2
5.	Python — это высокоуровневый, интерпретируемый язык программирования, который используется для разработки программного обеспечения. Python имеет простой и понятный синтаксис, который делает его привлекательным для новичков в программировании.	Что такое Python?	ОПК-8	2
6.	Для установки Python можно загрузить инсталлятор с официального сайта Python (python.org) и следовать инструкциям. Также можно использовать менеджер пакетов для установки Python (например, в Linux - apt-get или yum).	Как установить Python?	ОПК-8	2
7.	Для новых проектов вам рекомендуется использовать версию Python 3, поскольку она имеет более мощные функции и улучшенную безопасность. Однако, некоторые старые программы и инструменты могут требовать Python 2.	Какая версия Python лучше использовать?	ОПК-8	2
8.	Чтобы сделать процесс разработки программ удобным для программистов, разработаны системы программирования, включающие транслятор, текстовый редактор (позволяющий копировать, удалять и перемещать фрагменты программ), справочную систему, средства отладки (поиска ошибок в программе) и др. Например для ЯП Python имеется несколько систем программирования. Например CPython, который распространяется по свободной лицензии Python Software Foundation License	Что включает в себя система программирования?	ОПК-8	2
9.	Чтобы объявить переменную в Python, вы можете присвоить ей значение с использованием знака «=». Например, x = 5. Т.е. в отличие от других языков программирования, Python не имеет команды для объявления переменной. Переменная создается тогда, когда вы назначили ей значение. Не нужно указывать конкретный тип переменной при объявлении. Можно даже изменять их тип после создания.	Как объявить переменную в Python?	ОПК-8	2
10.	Переменная может иметь краткое имя (например, x и y) или более содержательное имя (age, carname, total_volume). Правила для переменных в Python: Имя переменной должно начинаться с буквы или символа подчеркивания. Оно не может начинаться с числа. Имя переменной может содержать только буквенно-цифровые символы и символы подчеркивания (Az, 0-9 и _). Имена переменных чувствительны к регистру (age, Age и AGE — три разных переменные)	Имена переменных в Python	ОПК-8	2
11.	Операторы — это символы, которые позволяют проводить операции в Python. Некоторые примеры операторов в Python: арифметические операторы (+, -, *, /), операторы сравнения (==, !=, >, <), операторы присваивания (=), логические операторы (and, or, not) и др.	Что такое операторы в Python?	ОПК-8	2
12.	При определении порядка действий используется приоритет (старшинство) операций. Они выполняются в следующем порядке: • действия в скобках; • возведение в степень (**), справа налево (!); • умножение (*) и деление (/), слева направо; • сложение и вычитание, слева направо.	Что такое приоритет операций? Зачем он нужен? Перечислите арифметические операции в порядке уменьшения приоритета	ОПК-8	2
13.	Над величинами целого типа, кроме операций сложения, вычитания, умножения, деления и возведения в степень, можно выполнять также еще две операции:	Чем отличаются операции, знаки которых «/», «//» и «%»?	ОПК-8	2

	1) определение целой части частного от деления одного целого числа на другое – знак операции «//»; 2) определение остатка от деления одного целого числа на другое – знак операции «%».			
14.	Условный оператор в Python — это оператор, который позволяет проверять условие и выполнять определенные действия в зависимости от результата проверки. Например, if-else оператор проверяет условие и выполняет определенный блок кода, если условие возвращает True, и другой блок кода, если условие возвращает False.	Что такое условный оператор в Python?	ОПК-8	2
15.	Циклы в Python — это конструкции, которые позволяют повторять выполнение одного и того же блока кода. В Python есть два типа циклов: цикл for и цикл while	Что такое циклы в Python?	ОПК-8	2
16.	Функции в Python — это блоки кода, которые могут быть вызваны из других мест в программе. Они способствуют упрощению программы и уменьшают количество дублированных строк кода.	Что такое функции в Python?	ОПК-8	2
17.	Функции бывают: 1) стандартные («встроенные» в систему программирования и доступные без всяких условий). Например, стандартными являются функции print() (решает задачу вывода информации на экран), input() (возвращает в программу строку символов, введенных с клавиатуры), int() (преобразует строку символов-цифр в число), float() и др.; 2) нестандартные – созданные пользователем; 3) полустандартные – предусмотренные в системе программирования, но для вызова (использования) которых необходимо подключить к разрабатываемой программе модуль, в который входят такие функции. Например, функция sqrt() относится к последней группе. Для ее применения в программе следует подключить модуль math. Для этого в начале программы нужно записать: import math	Какие виды функций возможны в программах на Python? В чем особенность каждого вида?		
18.	Для работы с файлами в Python вы можете использовать модули os, shutil и pathlib. Модуль os содержит функции для работы с файловой системой, модуль shutil предоставляет функции для копирования, перемещения и удаления файлов и папок, а модуль pathlib позволяет взаимодействовать с путями файлов и каталогов.	Какие модули Python нужно использовать для работы с файлами?	ОПК-8	2
19.	Чтобы использовать Python в консоли, можно запустить интерпретатор Python в терминале или командной строке. После запуска интерпретатора вы можете вводить Python-код и нажимать Enter для его выполнения	Как можно использовать Python в консоли?	ОПК-8	2
20.	Импорт необходимых модулей и библиотек Определение констант и переменных Определение функций Основная логика программы Завершение программы	Структура программы на языке программирования Python	ОПК-8	2
21.	Этапы создания исполняемой программы: Постановка задачи и выбор языка программирования Разработка алгоритма программы Написание кода программы Тестирование программы Оптимизация и улучшение программы Компиляция или интерпретация кода программы, чтобы получить исполняемый файл Запуск и проверка исполняемого файла Модули и библиотеки (множество готовых функций и инструментов) Исключения и обработка ошибок	Этапы создания исполняемой программы	ОПК-8	2
22.	Константы - это значения, которые не могут быть изменены в ходе выполнения программы. В Python константы обычно записываются в верхнем регистре и определяются с помощью ключевого слова const. Например: const PI = 3.14159 Переменные - это именованные области памяти, которые могут содержать различные значения в ходе выполнения программы. В Python переменные определяются путем присваивания значений с использованием оператора =. Например: x = 5	Константы и переменные Python	ОПК-8	2
23.	Операторы (арифметические, сравнения, логические и т.д.) Конструкции (условные, циклы, функции и т.д.) Классы и объекты (объектно-ориентированное	Состав языка программирования Python	ОПК-8	2

	программирование) Модули и библиотеки (множество готовых функций и инструментов) Исключения и обработка ошибок Встроенные типы данных (числа, строки, списки, словари и т.д.)			
24.	Python является языком с динамической типизацией, что означает, что тип данных определяется автоматически по значению, которое присваивается переменной. В Python есть несколько встроенных типов данных: Числа (целые числа, вещественные числа, комплексные числа) Строки Списки Кортежи Множества Словари Булев тип	Типы данных в языке программирования Python.	ОПК-8	2
25.	Выражения в Python являются комбинацией операторов и операндов, которые могут быть вычислены до определенного значения. Операторы в Python могут быть арифметическими, сравнения, логическими или специальными. Скобки используются для управления порядком выполнения операций и явно указывание, какие выражения должны быть вычислены и сгруппированы вместе. При вычислении выражения Python следует следующему порядку: Скобки, в том числе скобки функций Унарный оператор "-" (минус) для чисел Операторы возведения в степень "**" Умножение "*", деление "/", целочисленное деление "//" и остаток от деления "%" Сложение "+" и вычитание "-"	Выражения. Знаки операций. Сводка операций: скобки, порядок вычислений	ОПК-8	2
26.	Арифметические операторы: + (сложение), - (вычитание), * (умножение), / (деление), // (целочисленное деление), % (остаток от деления), ** (возведение в степень). Операторы сравнения: == (равно), != (не равно), > (больше), < (меньше), >= (больше или равно), <= (меньше или равно).	Арифметические операторы и операторы сравнения	ОПК-8	2
27.	Логические операторы: and (логическое И), or (логическое ИЛИ), not (логическое НЕ). Специальные операторы: = (присваивание), += (оператор присваивания с добавлением), -= (оператор присваивания с вычитанием), *= (оператор присваивания с умножением) и т.д.	Логические и специальные операторы	ОПК-8	2
28.	Типы данных в Python могут быть преобразованы из одного типа в другой, называемые "приведение типов". Например, строка можно преобразовать в число с помощью функции int(), а число в строку с помощью функции str(). Приведение типов может быть неявным, когда Python автоматически преобразует один тип в другой внутри выражения.	Преобразование типа данных	ОПК-8	2
29.	В языке программирования Python есть основные типы операторов, которые используются для управления выполнением программы. Они включают в себя операторы присваивания, составные операторы, операторы выбора, операторы цикла и операторы перехода. Операторы присваивания Операторы присваивания используются для присвоения значений переменным. Общий синтаксис оператора присваивания выглядит как "переменная = значение". Пример: x = 5 y = "Hello, world!"	Основные операторы в языке программирования Python (присваивание, составные, выбора, циклов, перехода). Синтаксис, семантика, примеры	ОПК-8	2
30.	Составные операторы — это операторы, которые позволяют выполнять несколько операций одновременно. Примеры составных операторов: +=: добавляет значение к переменной и присваивает результат обратно переменной. Например: x += 2 эквивалентно x = x + 2. -=: вычитает значение из переменной и присваивает результат обратно переменной. Например: x -= 2 эквивалентно x = x - 2. *=: умножает переменную на значение и присваивает результат обратно переменной. Например: x *= 2 эквивалентно x = x * 2.	Составные операторы	ОПК-8	2

	/=: делит переменную на значение и присваивает результат обратно переменной. Например: x /= 2 эквивалентно x = x / 2. //=: производит целочисленное деление переменной на значение и присваивает результат обратно переменной. Например: x //= 2 эквивалентно x = x // 2. %=: находит остаток от деления переменной на значение и присваивает результат обратно переменной. Например: x %= 2 эквивалентно x = x % 2.			

31.	Операторы выбора используются для выполнения разных действий, в зависимости от условия. Самый распространенный оператор выбора в Python - это оператор if-else. Синтаксис оператора if-else выглядит следующим образом:
	if условие:
	блок кода, если условие истинно
	else:
	блок кода, если условие ложно
	Пример:
	x = 5
	if x > 10:
	print("x больше 10")
	else:
	print("x меньше или равен 10")
		Операторы выбора Python	ОПК-8	2

32.	Операторы цикла используются для многократного выполнения блока кода. В Python существуют два типа циклов: цикл for и цикл while.
	Цикл for используется для перебора элементов в итерируемом объекте. Синтаксис цикла for выглядит так:
	for элемент in итерируемый_объект:
	блок кода
	Пример:
	for i in range(5):
	print(i)
	Цикл while используется для выполнения блока кода, пока условие истинно. Синтаксис цикла while выглядит так:
	while условие:
	блок кода
	Пример:
	x = 0
	while x < 5:
	print(x)
	x += 1
		Операторы цикла	ОПК-8	2

33.	Операторы перехода используются для регулирования потока выполнения программы. Они позволяют выполнение перейти с одного места в коде на другое место. Операторы перехода в Python включают:
	break: используется для выхода из цикла.
	continue: используется для перехода к следующей итерации цикла.
	return: используется для возврата значения из функции.
	pass: используется для объявления пустого блока кода.
	Пример:
	for i in range(10):
	if i == 5:
	break
	print(i)
	В этом примере цикл for завершится, когда i станет равным 5, потому что оператор break выйдет из цикла.
		Операторы перехода	ОПК-8	2

34.	return n	Допишите функцию func, чтобы функция print могла вывести результат: def func(n): n = n + 1 ……. print(func(0))	ОПК-8	2
35.	return	Какой из операторов возвращает данные после выполнения функции?	ОПК-8	2
36.	ошибку	Что выведет следующий	ОПК-8	2

		код? var = 10 print(Var)		
37.	[1, 2, 2]	Что напечатает следующий код? a = [1, 2, 3] a[2] = 2 print(a)	ОПК-8	2
38.	25	Что будет выведено на экран? x = 25 def func(x): x += 50 func(x) print(x)	ОПК-8	2
39.	7	Что будет выведено на экран? print(1|3|4)	ОПК-8	2
40.	a+=str(i)	Допишите программу, чтобы она выводила результат: 01234 Запишите строчку без пробелов между символами. a = '' for i in range(0,5): …….. print(a)	ОПК-8	2

Примерный перечень тестовых заданий к промежуточной аттестации

Номер задания	Правильный ответ	Содержание вопроса	Компетенция	Время выполнения задания, мин
1.	Б	Чем отличаются set и frozenset? а) set неизменяемые множества, frozenset – изменяемые б) set изменяемые множества, frozenset – неизменяемые в) ничем не отличаются г) set является подмножеством множества frozenset	ОПК-8	2
2.	А	В чем отличие списка от кортежа? а) Список - изменяем, кортеж - нет б) Кортеж изменяем, список - нет в) Нет отличий, кроме обозначения г) В Python нет понятия кортеж	ОПК-8	2

3.	Б	Что напечатает следующий код?
		for i in range(3):
		  if i < 1:
		  print(i)
		  else:
		  print(i)
		  break
		а) 0
		б) 0 1
		в) 0 1 2
		г) Ничего не напечатает
			ОПК-8	2

4.	Г	Какая из переменных в коде локальная, а какая глобальная?
		def square(a, b):
		  s = a*b
		  return s
		m = int(input('Введите число: '))
		n = int(input('Введите число: '))
		print(square(m,n))
		а) s - глобальная, m - локальная
		б) s - глобальная, n - локальная
		в) s, m - локальные, n - глобальная
		г) s - локальная, m, n - глобальные
			ОПК-8	2

5.	А	Что вернет срез ‘Python’[-2:]?	ОПК-8	2

		а) 'on' б) 'Py' в) 'th' г) ошибка		
6.	А	Какие из перечисленных выражений создадут список ровно из трех элементов? а) list(range(3)) б) 'asd'.split() в) a = 1, 2, 3 г) 'a b c'.split(' ')	ОПК-8	2

7.	А	Что необходимо добавить на месте пропущенной строки, учитывая, что функция ищет наибольшее число?
		def find_max(nums):
		max_num = float("-inf")
		for num in nums:
		if num > max_num:
		# пропущенная строка
		return max_num
		а) max_num = num
		б) num = max_num
		в) max_num += 1
		г) max_num += num
			ОПК-8	2

8.	Б	Какой будет результат выполнения кода?
		a = [1, 2, 3]
		if a[2] < 3:
		print(a[a[1]])
		else:
		print(a[1])
		а) 1
		б) 2
		в) 3
		г) 4
			ОПК-8	2

9.	Г	Что вернет срез 'Python'[:]? а) ' ' б) 'Pyth' в) 'Pytho' г) 'Python'	ОПК-8	2
10.	А	Что из нижеперечисленного относится к кортежам? а) (1, 2, 3) б) {'a':1, 'b': 2} в) [1, 2, 's'] г) 'str'	ОПК-8	2

Образец экзаменационного билета

САМАРСКИЙ ПОЛИТЕХ
Опорный университет

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Самарский государственный технический университет» (ФГБОУ ВО «СамГТУ»)

Филиал ФГБОУ ВО «СамГТУ» в г. Белебее Республики Башкортостан

Кафедра «Инженерные технологии»

ЭКЗАМЕНАЦИОННЫЙ БИЛЕТ № 1

по дисциплине (модулю): «Языки и методы программирования»
Код направления подготовки (специальности), направленность (профиль): 09.03.02 Информационные системы и технологии, Информационные системы и технологии
Курс 1

1. Операторы цикла
2. Операторы перехода

Составил:
доцент _____ З.Ф. Камальдинова
(подпись)
«____» _____ _____ г.

Утверждаю:
Зав.кафедрой _____ А.А. Цынаева
(подпись)
«____» _____ _____ г.

3. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующие процесс формирования компетенций

3.1. Характеристика процедуры текущей и промежуточной аттестации по дисциплине

Таблица 5

№ п/п	Наименование оценочного средства	Периодичность и способ проведения процедуры оценивания	Методы оценивания	Виды выставляемых оценок	Способ учета индивидуальных достижений, обучающихся
1.	Тестирование	систематически на всех видах занятий /письменно и устно	экспертный	По пятибалльной шкале	рабочая книжка преподавателя
2.	Промежуточная аттестация – вопросы экзаменационных билетов	по окончании изучения дисциплины/ устно и письменно	экспертный	По пятибалльной шкале	экзаменационная ведомость, зачетная книжка

3.2. Критерии и шкала оценивания результатов изучения дисциплины во время занятий (текущий контроль успеваемости)

Критерии оценки и шкала оценивания теста

Таблица 6

Шкала оценивания	Критерии оценки	Кол-во баллов
«Отлично»	86-100% правильных ответов	71-100 баллов
«Хорошо»	71-85% правильных ответов	41-70 баллов
«Удовлетворительно»	65-70% правильных ответов	21-40 баллов
«Неудовлетворительно»	менее 65% правильных ответов	0-20 баллов

Общие критерии и шкала оценивания результатов для допуска к промежуточной аттестации

Таблица 7

Наименование оценочного средства		Балльная шкала
1	Тестирование	0-100 баллов
	Итого:	100 баллов

Максимальное количество баллов за семестр – 100. Обучающийся допускается к экзамену при условии 51 и более набранных за семестр баллов.

3.3. Критерии и шкала оценивания результатов изучения дисциплины на промежуточной аттестации

Основанием для определения оценки на экзаменах служит уровень освоения обучающимися материала и формирования компетенций, предусмотренных программой учебной дисциплины.

Успеваемость определяется оценками: 5 «отлично»; 4 «хорошо»; 3 «удовлетворительно»; 2 «неудовлетворительно».

Оценку «отлично» получает обучающийся, освоивший компетенции дисциплины на всех этапах их формирования **на 85-100 %,** показавший всестороннее, систематическое и глубокое знание учебного материала, умение свободно выполнять задания, предусмотренные рабочей программой, усвоивший основную и ознакомленный с дополнительной литературой, рекомендованной программой. Как правило, оценка «отлично» выставляется обучающимся, усвоившим взаимосвязь основных положений учебной дисциплины, необходимых для приобретаемой профессии, проявившим творческие способности в понимании, изложении и использовании учебного материала.

Оценку «хорошо» заслуживает обучающийся, освоивший компетенции дисциплины на всех этапах их формирования **на 71-84 %,** обнаруживший полное знание учебного материала, успешно выполняющий предусмотренные рабочей программой задания, усвоивший основную литературу, рекомендованную в программе. Как правило, оценка «хорошо» выставляется обучающимся, продемонстрировавшим систематическое владение материалом дисциплины, способным к их самостоятельному пополнению и обновлению в ходе дальнейшей учебной работы и профессиональной деятельности, но допустившим несущественные неточности в ответе.

Оценку «удовлетворительно» получает обучающийся, освоивший компетенции дисциплины на всех этапах их формирования **на 51-70 %,** обнаруживший знание основного учебного материала в объёме, необходимом для дальнейшей учебы и предстоящей работы по профессии, справляющийся с выполнением заданий, предусмотренных рабочей программой, знакомый с основной литературой, рекомендованной программой. Как правило, оценка «удовлетворительно» выставляется обучающимся, допустившим погрешности в ответе на экзамене и при выполнении экзаменационных заданий, но

обладающим необходимыми знаниями для устранения под руководством преподавателя допущенных недочетов.

Оценка «неудовлетворительно» выставляется обучающемуся, освоившему компетенции дисциплины на всех этапах их формирования менее чем **на 51%,** обнаружившему пробелы в знаниях основного учебного материала, допустившему принципиальные ошибки в выполнении предусмотренных рабочей программой заданий.

Шкала оценивания результатов

Таблица 8

Процентная шкала (при ее использовании)	**Оценка в системе «неудовлетворительно – удовлетворительно – хорошо – отлично»**
0-50%	Неудовлетворительно
51-70%	Удовлетворительно
71-84%	Хорошо
85-100%	Отлично

Приложение 2

УТВЕРЖДАЮ
Директор филиала ФГБОУ ВО «СамГТУ»
в г. Белебее Республики Башкортостан

_____ Л.М. Инаходова
«____» _____ 20__ г.

Дополнения и изменения к рабочей программе дисциплины (модуля)

Б1.О.02.05 «Языки и методы программирования»

по направлению подготовки (специальности) 09.03.02 «Информационные системы и технологии» по направленности (профилю) подготовки «Информационные системы и технологии»

на 20__/20__ учебный год

В рабочую программу вносятся следующие изменения:

1) .....;
2) .....;

Разработчик дополнений и изменений:

_____	_____	_____
(должность, степень, ученое звание)	(подпись)	(ФИО)

Дополнения и изменения рассмотрены и одобрены на заседании кафедры «____» _____ 20___ г., протокол № ____.

Заведующий кафедрой

_____	_____
(степень, звание, подпись)	(ФИО)

Приложение 3

Аннотация рабочей программы дисциплины

Б1.О.02.05 «Языки и методы программирования»

Код и направление подготовки (специальность) **09.03.02 Информационные системы и технологии**

Направленность (профиль) **Информационные системы и технологии**

Квалификация **бакалавр**

Форма обучения **заочная**

Год начала подготовки **2023**

Выпускающая кафедра **Инженерные технологии**

Кафедра-разработчик **Инженерные технологии**

Объем дисциплины, ч. / з.е. **216 / 6**

Форма контроля (промежуточная аттестация) **экзамен, экзамен**

Курс	Час. / з.е.	Лек. зан., час.	Лаб. зан., час.	Практич. зан., час.	КСР	СРС	Контроль	Форма контроля
1	108 / 3	4	4	-	3	88	9	экзамен
2	108 / 3	4	4	-	3	88	9	экзамен
Итого	216 / 6	8	8	-	6	176	18	экзамен, экзамен

Универсальные компетенции:	
не предусмотрены учебным планом	
Общепрофессиональные компетенции:	
ОПК-2	Способен понимать принципы работы современных информационных технологий и программных средств, в том числе отечественного производства, и использовать их при решении задач профессиональной деятельности
ОПК-2.1	Использует и понимает принципы работы информационных технологий и программных средств при решении задач в сфере информационных систем и технологий
ОПК-2.2	Применяет современные информационные технологии и программные средства отечественного производства при решении задач в сфере информационных систем и технологий
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического применения в области информационных систем и технологий;
ОПК-6.1	Разрабатывает алгоритмы и программы, пригодные для практического применения
ОПК-6.2	Осуществляет отладку и тестирование программного обеспечения
ОПК-6.3	Ведет и использует базы данных и информационные хранилища
ОПК-8	Способен применять математические модели, методы и средства проектирования информационных и автоматизированных систем.
ОПК-8.1	Разрабатывает математические и имитационные модели процессов в сфере информационных систем и технологий
Профессиональные компетенции:	
не предусмотрены учебным планом	

Содержание дисциплины охватывает круг вопросов, связанных с формированием у студентов навыков программирования на языках высокого уровня для решения задач по разработке алгоритмов, баз данных и программ с последующей их отладкой.

Преподавание дисциплины предусматривает следующие формы организации учебного процесса: лекции, лабораторные занятия, самостоятельная работа студента.

Программой дисциплины предусмотрены следующие виды контроля: текущий контроль успеваемости в форме тестирование и промежуточный контроль в форме экзамена.