Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Костровец Лариса Борисовна
Должность: директор
Дата подписания: 02.10.2026 00:12:43
Уникальный программный ключ:
6882606104c36dbde41c4ab93a65382136a292d6

Приложение 4
к образовательной программе

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.01.02.04 Программирование на Python
(индекс, наименование дисциплины в соответствии с учебным планом)

09.03.03 Прикладная информатика
(код, наименование направления подготовки/специальности)

Прикладная информатика в управлении корпоративными информационными системами
(наименование образовательной программы)

Очная форма обучения
(форма обучения)

Год набора – 2026
Донецк

Автор(ы)-составитель(и) РПД:

Лебезова Элла Михайловна, старший преподаватель кафедры информационных технологий

Заведующий кафедрой:

Брадул Наталья Валерьевна, канд. физ.-мат. наук, заведующий кафедрой информационных технологий

Рабочая программа дисциплины Б1.О.01.02.04 Программирование на Python одобрена на заседании кафедры информационных технологий администрирования факультета государственной службы и управления Донецкого филиала РАНХиГС.

Протокол № 7 от «05 » марта 2026 г.

СОДЕРЖАНИЕ

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

2. Объем и место дисциплины в структуре образовательной программы

3. Содержание и структура дисциплины

4. Типы оценочных материалов, показатели и критерии их оценивания

5. Формы аттестации, типовые оценочные материалы для текущего контроля успеваемости обучающихся, критерии и шкалы оценивания по контрольным точкам

6. Формы промежуточной аттестации, критерии и шкала оценивания, типовые оценочные материалы по дисциплине

7. Методические материалы по освоению дисциплины

8. Учебная литература и ресурсы информационно-телекоммуникационной сети «Интернет»

9. Материально-техническая база, информационные технологии, программное обеспечение и информационные справочные системы

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Дисциплина Б1.О.01.02.04 Программирование на Python обеспечивает формирование у обучающихся следующих общепрофессиональных компетенций*:

ОТФ /ТФ и реквизиты ПС *(при наличии)* **	**Код компетенции ****	**Наименование Компетенции ****	**Код индикатора достижения компетенций ****	**Наименование индикатора достижения компетенций ****	**Образовательный результат ****
-	ОПК-7.	Способен разрабатывать алгоритмы и программы, пригодные для практического применения	ОПК-7.4.	Разрабатывает продвинутые алгоритмы и программы на Python для применения в прикладных задачах	**Знает** базовые алгоритмы и структуры данных. **Умеет** разрабатывать алгоритмы и программы для решения типовых задач.

** Дисциплина может формировать компетенцию полностью или частично.*
***Должно соответствовать Приложению 1 к образовательной программе*

2. Объем и место дисциплины в структуре образовательной программы

Общий объем дисциплины:
3,00 з.е., 108 ак.час
Контактная работа обучающихся с преподавателем по видам учебных занятий: 59 ак. час на контактную работу с преподавателем, из них 24 ак. час на лекции и 24 ак. час на практические занятия. 22 ак. час на самостоятельную работу обучающихся.
Б1.О.01.02.04. Программирование на Python реализуется во 2-м семестре 1-го курса.

.

3. Содержание и структура дисциплины

3.1. Структура дисциплины

Очная форма обучения

№ п/п	Наименование тем и (или) разделов	ВСЕГО	Объем дисциплины, ак.час												Форма текущего контроля успеваемости, промежуточной аттестации
			Контактная работа обучающихся с преподавателем по видам учебных занятий									Самостоятельная работа			
			Период теоретического обучения						Период промежуточной аттестации (сессия)						
			Занятия лекционного типа		Занятия семинарского типа		ИК	КСР	КЭ	Каттэк	Контроль	СРкр	СРэк	СР	
			Л	ВЛ	ЛР	ПЗ									
РАЗДЕЛ 1. Процедурное программирование на Python															
Тема 1	Парадигмы программирования.	7	2	0	0	2	0	0	0	0	0	0	0	3	Контрольные вопросы, практические занятия, КР1
Тема 2	Функции Python и их применение. .	11	4	0	0	4	0	0	0	0	0	0	0	3	Контрольные вопросы, практические занятия, КР1
Тема 3	Алгоритмы поиска и	11	4	0	0	4	0	0	0	0	0	0	0	3	Контрольные вопросы,

	сортировки.														практические занятия, КР 1
Тема 4	Анализ сложности алгоритмов.	11	2	0	0	2	0	0	0	0	0	0	0	3	Контрольные вопросы, практические занятия, КР 1
РАЗДЕЛ 2. Структуры данных и файлов на Python															
Тема 5	Кортежи и множества в Python.	12	4	0	0	4	0	0	0	0	0	0	0	4	Контрольные вопросы, практические занятия, КР 2
Тема 6	Применение и работа со словарями Python.	12	4	0	0	4	0	0	0	0	0	0	0	4	Контрольные вопросы, практические занятия, КР 2
Тема 7	Работа с текстовыми и бинарными файлами в Python.	8	2	0	0	2	0	0	0	0	0	0	0	4	Контрольные вопросы, практические занятия, КР 2
Промежуточная аттестация		38	0	0	0	0	0	0	2	9	0	9	18	0	Экзамен
Итого		**108**	**24**	**0**	**0**	**24**	**0**	**0**	**2**	**9**	**0**	**9**	**18**	**22**	

Используемые сокращения:

Л – лекции - занятия, предусматривающие преимущественную передачу учебной информации обучающимся педагогическими работниками организации и (или) лицами, привлекаемыми организацией к реализации образовательных программ на иных условиях,).

ВЛ – видео лекции.

ЛР – лабораторные работы.

ПЗ – практические занятия (за исключением лабораторных работ).

ИК – индивидуальные консультации.

КСР – контроль самостоятельной работы

КЭ – консультации перед экзаменом

Каттэк – контактная работа на аттестацию в период экзаменационных сессий

Контроль - контактная работа на аттестацию в период экзаменационных сессий для заочной формы обучения
СРкр – самостоятельная работа на подготовку курсовой работы/ курсового проекта.
СРэк – самостоятельная работа на подготовку к экзамену.
СР – самостоятельная работа в семестре на подготовку к учебным занятиям.

3.2. Содержание дисциплины

РАЗДЕЛ 1. ПРОЦЕДУРНОЕ ПРОГРАММИРОВАНИЕ НА PYTHON

Тема 1. Парадигмы программирования. ОПК-7.4.

Понятие парадигмы программирования. Императивное программирование: особенности, базовые конструкции, примеры языков. Процедурное программирование: процедуры и функции, модульность, повторное использование кода. Объектно-ориентированное программирование: классы, объекты, инкапсуляция, наследование, полиморфизм. Функциональное программирование: чистые функции, неизменяемость данных, функции высшего порядка, рекурсия. Сравнительный анализ парадигм на примере Python как мультипарадигмального языка. Выбор парадигмы в зависимости от решаемой задачи. Принципы модульного программирования: декомпозиция, разделение интерфейса и реализации, управление зависимостями.

Тема 2. Функции Python и их применение. ОПК-7.4.

Определение и вызов функций в Python. Параметры и аргументы: позиционные, именованные, параметры со значениями по умолчанию, переменное количество аргументов (произвольное число позиционных и именованных аргументов). Возврат значений из функции, оператор возврата. Области видимости переменных: локальные, глобальные, нелокальные переменные. Рекурсивные функции: принципы организации, базовый случай и шаг рекурсии, преимущества и недостатки. Анонимные функции (лямбда-выражения): синтаксис, применение с функциями высшего порядка. Функции как объекты первого класса: передача функций в качестве аргументов, возврат функций из функций. Документирование функций (строки документации). Типовые задачи: создание библиотек вспомогательных функций, функциональная декомпозиция сложных алгоритмов.

Тема 3. Алгоритмы поиска и сортировки. ОПК-7.4.

Понятие алгоритма поиска. Линейный поиск: принцип работы, реализация, анализ эффективности. Бинарный поиск: требования к данным (отсортированность), принцип деления пополам, рекурсивная и итеративная реализации, сравнение с линейным поиском. Понятие алгоритма сортировки. Сортировка пузырьком: принцип последовательного сравнения и обмена соседних элементов, реализация, анализ временной сложности. Сортировка вставками: построение отсортированной части массива, реализация, преимущества на частично упорядоченных данных. Сортировка выбором: нахождение минимального элемента и перестановка, реализация. Быстрая сортировка: рекурсивный алгоритм, выбор опорного элемента, разделение массива, анализ эффективности. Сортировка слиянием: принцип «разделяй и властвуй», слияние отсортированных подмассивов. Встроенные функции сортировки в Python. Применение алгоритмов поиска и сортировки в прикладных задачах.

Тема 4. Анализ сложности алгоритмов. ОПК-7.4.

Понятие временной и пространственной сложности алгоритма. Необходимость анализа сложности для выбора оптимального решения. «O»-нотация (асимптотический анализ): определение, основные классы сложности (константная, логарифмическая, линейная, линейно-логарифмическая, квадратичная, экспоненциальная, факториальная). Правила подсчёта сложности для базовых конструкций: последовательное выполнение, условные операторы, циклы, вложенные циклы, рекурсивные вызовы. Анализ сложности изученных алгоритмов: линейный поиск, бинарный поиск, сортировка пузырьком, быстрая сортировка, сортировка слиянием. Понятие лучшего, среднего и худшего случая. Компромисс между временной и пространственной сложностью. Практическое применение анализа сложности при выборе алгоритмов и структур данных для реальных задач.

РАЗДЕЛ 2. СТРУКТУРЫ ДАННЫХ И ФАЙЛОВ НА PYTHON

Тема 5. Кортежи и множества в Python. ОПК-7.4.

Кортеж как неизменяемая упорядоченная последовательность: создание, индексация, срезы, операции. Отличия кортежа от списка: неизменяемость как преимущество (защита данных от случайного изменения, возможность использования в качестве ключей словаря). Методы кортежей (подсчёт вхождений, поиск индекса). Распаковка кортежей в переменные. Множество как неупорядоченная коллекция уникальных элементов: создание, добавление и удаление элементов. Операции над множествами: объединение, пересечение, разность, симметрическая разность. Проверка принадлежности элемента, сравнение множеств (подмножество, надмножество). Замороженное множество как неизменяемая версия множества. Применение множеств для удаления дубликатов из последовательностей, для проверки вхождения, для реализации операций над множествами (анализ пересекающихся данных). Типовые задачи: поиск общих элементов, удаление повторений, проверка уникальности данных.

Тема 6. Применение и работа со словарями Python. ОПК-7.4.

Словарь как структура данных «ключ-значение»: принципы организации (хэш-таблица), требования к ключам (неизменяемость, хэшируемость). Создание словарей: литерал, функция создания словаря, генераторы словарей. Доступ к значениям по ключу, методы получения значений (прямой доступ, метод получения с значением по умолчанию). Добавление, изменение и удаление пар ключ-значение. Итерация по словарю: по ключам, по значениям, по парам (ключ-значение). Методы словаря: получение всех ключей, всех значений, всех пар. Вложенные словари (словари словарей). Словари для агрегации данных, для организации переходов (замена многоуровневых условных конструкций), для кэширования результатов вычислений, для подсчёта частоты элементов. Генераторы словарей (dict comprehensions). Типовые задачи: построение частотного словаря, создание справочных структур, организация маршрутизации, группировка данных по ключам.

Тема 7. Работа с текстовыми и бинарными файлами в Python. ОПК-7.4.

Понятие файловой системы и файлового ввода-вывода. Открытие файлов: функция открытия, режимы доступа (чтение, запись, добавление, бинарный режим, текстовый режим). Контекстный менеджер для автоматического закрытия файлов. Чтение текстовых файлов: чтение всего содержимого, построчное чтение (в виде строки, в виде списка строк). Запись текстовых файлов: запись строки, запись нескольких строк, добавление данных. Кодировки символов (UTF-8, ASCII, CP1251 и другие), обработка проблем с кодировками. Работа с бинарными файлами: чтение и запись байтов, отличия от текстового режима. Управление позицией в файле: перемещение указателя, определение текущей позиции. Обработка исключений при работе с файлами (отсутствие файла, ошибки доступа, ошибки кодировки). Сериализация данных: модуль для сохранения и загрузки объектов Python (pickle). Форматы хранения данных (JSON, CSV): чтение и запись структурированных данных. Типовые задачи: сохранение результатов работы программы, загрузка конфигураций, обработка логов, импорт и экспорт данных между программами.

4. Типы оценочных материалов, показатели и критерии оценивания

4.1. Оценочные материалы по дисциплине Б1.О.01.02.04. Программирование на Python входят в состав оценочных материалов по образовательной программе. Совокупность оценочных материалов по всем дисциплинам (модулям) образовательной программы составляет фонд оценочных средств (далее – ФОС). ФОС используется при проведении текущего контроля успеваемости и промежуточной аттестации обучающихся с целью оценивания достижения обучающимися планируемых результатов обучения.

4.2. ФОС разработан как комплекс проверочных заданий различного типа и уровня сложности, включает критерии и шкалы оценивания, а также «ключи» правильных ответов. ФОС формируется как отдельный документ и хранится в электронном виде, доступ к ФОС предоставлен ограниченному кругу лиц.

4.3. Для самостоятельной работы обучающихся при подготовке к текущему контролю успеваемости и промежуточной аттестации в рабочих программах дисциплин размещены типовые проверочные задания, которые можно условно разделить на задания закрытого, комбинированного и открытого типов.

Задания закрытого типа – это тестовые задания, в которых каждый вопрос сопровождается готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных.

Задания комбинированного типа – это тестовые задания, в которых каждый вопрос сопровождается готовыми вариантами ответов, из которых необходимо выбрать один или несколько правильных и обосновать свой выбор.

Задания открытого типа – это задания, в которых на каждый вопрос должен быть предложен развернутый обоснованный ответ.

В зависимости от типа задания рекомендованы определенная последовательность выполнения и система оценивания выполнения заданий.

4.4. Типы заданий, сценарии выполнения, критерии оценивания

ТИП ЗАДАНИЯ	ИНСТРУКЦИЯ	СЦЕНАРИИ ВЫПОЛНЕНИЯ	КРИТЕРИИ ОЦЕНИВАНИЯ
Задание закрытого типа с выбором одного правильного ответа из нескольких вариантов предложенных	Прочитайте текст, выберите правильный ответ	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается только один из предложенных вариантов. 2. Внимательно прочитать предложенные вариант-ты ответа. 3. Выбрать один верный ответ. 4. Записать только номер (или букву) выбранного варианта ответа (например, 3 или В).	Ответ считается верным, если правильно указана цифра или буква
Задание закрытого типа на установление соответствия	Прочитайте текст и установите соответствие	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидаются пары элементов. 2. Внимательно прочитать оба списка: список 1 – вопросы, утверждения, факты, понятия и т.д.; список 2 – утверждения, свойства объектов и т.д. 3. Сопоставить элементы списка 1 с элементами списка 2, сформировать пары элементов. 4. Записать попарно буквы и цифры (в зависимости от задания) вариантов ответа (например, А1 или Б4).	Ответ считается верным, если правильно указаны цифры или буквы

Задание закрытого типа с выбором нескольких правильных ответов из нескольких вариантов предложенных	Прочитайте текст, выберите правильные ответы	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается несколько правильных ответов из предложенных вариантов. 2. Внимательно прочитать предложенные вариант-ты ответа. 3. Выбрать несколько правильных ответов. 4. Записать только номера (или буквы) выбранного варианта ответа (например, 1 4 или А Г).	Ответ считается верным, если правильно установлены все соответствия (позиции из одного столбца верно сопоставлены с позициями другого)
Задание закрытого типа на установление последовательнос ти	Прочитайте текст и установите последовательность	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается последовательность элементов. 2. Внимательно прочитать предложенные варианты ответа. 3. Построить верную последовательность из предложенных элементов. 4. Записать буквы/цифры (в зависимости от задания) вариантов ответа в нужной последовательности (например, БВА или 135).	Ответ считается верным, если правильно указана вся последовательность цифр

Задание комбинированного типа с выбором одного правильного ответа из предложенных и обоснованием выбора	Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа	1. Внимательно прочитать текст задания и понять, что в качестве ответа ожидается только один из предложенных вариантов. 2. Внимательно прочитать предложенные варианты ответа. 3. Выбрать один верный ответ. 4. Записать только номер (или букву) выбранного варианта ответа. 5. Записать аргументы, обосновывающие выбор ответа (например, 4 текст обоснования).	Ответ считается верным, если правильно указана цифра или буква и приведены корректные аргументы, используемые при выборе ответа
Задание открытого типа с развернутым ответом	Прочитайте текст и запишите развернутый обоснованный ответ	1. Внимательно прочитать текст задания и понять суть вопроса. 2. Продумать логику и полноту ответа. 3. Записать ответ, используя четкие компактные формулировки. 4. В случае расчетной задачи, записать решение и ответ	Ответ считается верным: 1. Отсутствие фактических ошибок. 2. Раскрытие объема используемых понятий (полнота ответа). 3. Обоснованность ответа (наличие аргументов). 4. Логическая последовательность излагаемого материала.

4.5. Общая шкала оценивания результатов текущего контроля успеваемости и промежуточной аттестации обучающихся с применением БРС

Оценка по шкале ECTS	**Сумма баллов за все виды учебной деятельности**	**Оценка по государственной шкале**	**Определение**
A	90 – 100	«Отлично»	отличное выполнение с незначительным количеством неточностей
B	80 – 89	«Хорошо»	в целом правильно выполненная работа с незначительным количеством ошибок (до 10%)
C	75 – 79		в целом правильно выполненная работа с незначительным количеством ошибок (до 15%)
D	70 – 74	«Удовлетворительно»	неплохо, но со значительным количеством недостатков
E	60 – 69		выполнение удовлетворяет минимальные критерии
FX	35 – 59	«Не удовлетворительно»	с возможностью повторной сдачи
F	0 – 34		с обязательным повторным изучением дисциплины (выставляется комиссией)

Соотношение баллов за текущий контроль успеваемости и промежуточную аттестацию, а также повторную промежуточную аттестацию:

Максимальная сумма баллов за текущий контроль успеваемости	Максимальная сумма баллов за промежуточную аттестацию	Максимальная итоговая балльная оценка	Максимальная сумма баллов за повторную промежуточную аттестацию
100 баллов	100 баллов	100 баллов	100 баллов

5. Формы аттестации, типовые оценочные материалы для текущего контроля успеваемости обучающихся, критерии и шкалы оценивания по контрольным точкам

5.1. В ходе реализации дисциплины Б1.О.01.02.04 Программирование на Python используются следующие формы текущего

контроля успеваемости обучающихся (в том числе, задания к контрольным точкам):

Контрольные вопросы для проведения опроса, задания открытого типа на практических занятиях, контрольные задания

Таблица 5.1.

Распределение баллов по видам учебной деятельности (БРС)

Раздел/Темы	Формы текущего контроля		КЗР
	УО	ПЗ	
Р-1. / Т-1	3	7	15
Р-1. / Т-2	3	7	
Р-1. / Т-3	3	7	
Р-1. / Т-4	3	7	15
Р-2. / Т-5	3	7	
Р-2. / Т-6	3	7	
Р-2. / Т-7	3	7	
Итого: 100 б	18	7	30

УО – устный опрос;
ТЗ – тестовое задание;
КЗ – контрольные задания;
ПЗ – практическое занятие;
Д – доклад;
КЗР – контрольные работы по разделу.

Критерии оценивания опроса:

Балы	Описание критерия
3	Обучающийся полно излагает материал (отвечает на вопрос), дает правильное определение основных понятий; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только из учебника, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка.
2	Обучающийся дает ответ, удовлетворяющий тем же требованиям, что и для оценки «отлично», но допускает 1–2 ошибки, которые сам же исправляет, и 1–2 недочета в последовательности и языковом оформлении излагаемого.
1	Обучающийся обнаруживает знание и понимание основных положений данной темы, но излагает материал неполно и допускает неточности в определении понятий или формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого.
0	Обучающийся обнаруживает незнание вопроса, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал.

0* - в журнал академической группы не выставляется

Критерии оценивания практических занятий:

Балы	Описание критерия	
3	Свыше 90% правильных ответов.	Обучающийся демонстрирует глубокое познание в освоенном материале.
2	Свыше 70% правильных ответов.	Обучающимся материал освоен полностью, без существенных ошибок.
1	Реализовано более 50% поставленных задач	Обучающимся материал освоен не полностью, имеются значительные пробелы в знаниях.
0	Реализовано менее 30% поставленных задач.	Обучающимся материал не освоен, знания обучающегося ниже базового уровня.

0* - в журнал академической группы не выставляется

Критерии оценивания контрольных заданий:

Балы	Описание критерия
12-15	Обучающимся задание выполнено без ошибок и в полном объеме.
8-11	Обучающимся в целом задание выполнено, имеются отдельные неточности или недостаточно полные ответы, не содержащие ошибок.
5-7	Обучающимся допущены отдельные ошибки при выполнении задания
0-4	У обучающегося отсутствуют ответы на большинство вопросов задачи, задание не выполнено или выполнено не верно.

0* - в журнал академической группы не выставляется

5.2. Типовые оценочные материалы для текущего контроля успеваемости обучающихся (вне контрольных работ):

РАЗДЕЛ 1. ПРОЦЕДУРНОЕ ПРОГРАММИРОВАНИЕ НА PYTHON

Тема 1. Парадигмы программирования.

Контрольные вопросы:

1. Что такое парадигма программирования? Приведите примеры языков, которые являются строго «чистыми» представителями одной парадигмы.
2. Назовите основные черты императивного программирования. Чем оно отличается от декларативного? Приведите пример задачи, которую естественно решать императивно, и пример задачи, где декларативный подход был бы более выразительным.
3. В чём заключается суть процедурного программирования? Какие преимущества даёт разделение кода на процедуры и функции перед линейным (монолитным) кодом?
4. Перечислите основные принципы объектно-ориентированного программирования. Раскройте каждый принцип на конкретном примере из реальной практики программирования.

5. Чем функциональное программирование принципиально отличается от императивного? Что такое «чистая функция», и почему отсутствие побочных эффектов считается преимуществом при написании надёжного кода?

6. Что такое функции высшего порядка? Приведите примеры встроенных функций Python, которые являются функциями высшего порядка, и объясните, как они работают.

7. Почему Python называют мультипарадигмальным языком? В каких ситуациях программист на Python может выбрать объектно-ориентированный подход, в каких — функциональный, а в каких — процедурный? Приведите примеры.

8. Что такое инкапсуляция с точки зрения защиты данных? Как в Python реализуется сокрытие деталей реализации? В чём отличие подхода Python от языков вроде Java или C++?

9. Объясните понятие «рекурсия» как элемента функционального программирования. В чём заключаются преимущества и недостатки рекурсивного подхода по сравнению с итеративным? Приведите пример задачи, где рекурсия является наиболее естественным способом решения.

10. В каких случаях оправдано смешение парадигм в рамках одного проекта? Приведите пример, когда использование императивного подхода внутри объектно-ориентированной программы является разумным, а когда — свидетельством плохого стиля кодирования.

Практические занятия:

Задание 1. Вам предоставлены три фрагмента кода на Python, решающих одну и ту же задачу - вычисление суммы элементов списка. Определите, какая парадигма программирования преимущественно использована в каждом фрагменте. Аргументируйте свой ответ, указав конкретные признаки парадигмы.

Фрагмент А:

```
numbers = [1, 2, 3, 4, 5]
summa = 0
for n in numbers:
    summa = summa + n
print(summa)
```

Фрагмент Б:

```
def calculate_sum(numbers):
    total = 0
    for n in numbers:
        total += n
    return total
```

```
numbers = [1, 2, 3, 4, 5]
print(calculate_sum(numbers))
```

Фрагмент B:

```
from functools import reduce
numbers = [1, 2, 3, 4, 5]
print(reduce(lambda x, y: x + y, numbers))
```

Задание 2. Возьмите задачу: «Дан список целых чисел. Удвоить каждый элемент, который больше 10, и заменить его на 0, если элемент меньше или равен 10. Вывести результирующий список». Реализуйте решение этой задачи тремя способами, каждый из которых демонстрирует определённую парадигму:

- императивный стиль;
- процедурный;
- функциональный стиль.

Сравните полученные решения по читаемости и объёму кода. Напишите краткий вывод (3–5 предложений), какой стиль вам кажется наиболее понятным и почему.

Задание 3. Дан императивный фрагмент кода, который фильтрует и преобразует список чисел:

```
result = []
for num in range(1, 21):
    if num % 2 == 0:
        result.append(num ** 2)
```

Перепишите этот код в функциональном стиле, используя функции высшего порядка и/или генераторы списков. Сравните оба варианта. Какие преимущества даёт функциональный подход в данном конкретном случае?

Задание 4. Напишите программу, содержащую:

- глобальную переменную счётчика;
- функцию, которая изменяет эту глобальную переменную (побочный эффект);
- другую функцию, которая получает тот же результат через передачу параметра и возврат значения.

Продемонстрируйте работу обеих функций. Объясните, почему вторая функция соответствует принципам функционального программирования, а первая – нет. В каких ситуациях побочные эффекты неизбежны или даже удобны?

Задание 5. Реализуйте вычисление числа Фибоначчи двумя способами:

- итеративно;
- рекурсивно.

Для каждого способа вычислите значение для n = 10, 20, 30. Замерьте время выполнения или оцените количество рекурсивных вызовов. Объясните, почему рекурсивная реализация классического Фибоначчи неэффективна, и как эта проблема связана с понятием «пространственная сложность». Предложите способ улучшения рекурсивного решения.

Задание 6. Напишите функцию высшего порядка apply_to_all, которая принимает два аргумента:

- список чисел;
- функцию, преобразующую число в число.

Функция apply_to_all должна возвращать новый список, полученный путём применения переданной функции к каждому элементу исходного списка. Продемонстрируйте работу apply_to_all, передав ей в качестве второго аргумента:

- лямбда-функцию, возводящую число в квадрат;
- лямбда-функцию, проверяющую, является ли число чётным (возвращает логическое значение);
- любую другую функцию преобразования, придуманную вами.

Объясните, почему возможность передавать функции как аргументы является важной чертой функционального программирования и как это влияет на переиспользование кода.

Задание 7. Создайте программу, которая предлагает пользователю выбрать способ вычисления площади геометрической фигуры (круг, прямоугольник, треугольник). Реализуйте два варианта организации кода:

1. С использованием многоуровневых условных конструкций (if-elif-else) – императивный подход.
2. С использованием словаря, где ключами являются названия фигур, а значениями – соответствующие функции вычисления площади (объектно-ориентированный/функциональный гибрид).

Тема 2. Функции Python и их применение

Контрольные вопросы:

1. Как в Python определяется функция? Из каких обязательных и необязательных частей состоит определение функции? Что произойдёт, если функция не содержит оператора возврата?
2. В чём разница между позиционными и именованными аргументами при вызове функции? Приведите пример, когда использование именованных аргументов делает вызов функции более читаемым и менее подверженным ошибкам.

3. Что такое параметры функции со значениями по умолчанию? Какое важное ограничение связано с использованием изменяемых объектов (например, списков или словарей) в качестве значений по умолчанию? Приведите пример проблемы и способ её решения.

4. Как в функции обработать произвольное количество позиционных аргументов? А произвольное количество именованных аргументов? Приведите примеры объявления таких функций и их вызова.

5. Что такое область видимости переменной в Python? Объясните различие между локальными, глобальными и нелокальными переменными. В каких ситуациях возникает необходимость использовать ключевое слово global или nonlocal?

6. Что такое рекурсивная функция? Из каких двух обязательных компонентов состоит любая корректная рекурсивная функция? Что произойдёт, если не предусмотреть базовый случай? Приведите пример практической задачи, где рекурсия оправдана.

7. Что такое анонимные функции (лямбда-выражения)? Какова их синтаксическая структура, и в каких ограничениях по сравнению с обычными функциями они существуют? Приведите примеры ситуаций, где использование лямбда-функций делает код более лаконичным.

8. Что означает утверждение, что в Python «функции являются объектами первого класса»? Какие возможности это даёт программисту (присваивание переменным, передача в качестве аргументов, возврат из функций)? Приведите пример кода, демонстрирующий эти возможности.

9. Для чего используется строка документации в начале тела функции? Как к ней можно обратиться из программы, и почему написание документации считается хорошей практикой? Приведите пример правильного оформления строки документации.

10. Объясните механизм замыкания на примере функции, возвращающей другую функцию. Какие переменные и откуда «запоминает» возвращаемая функция? В каких практических задачах замыкания оказываются полезнее, чем создание классов?

Практические занятия:

Задание 1. Напишите функцию draw_triangle(fill, base), которая принимает два параметра:

- fill – символ заполнитель;
- base – величина основания равнобедренного треугольника;

Затем она должна вывести его.

Примечание. Гарантируется, что основание треугольника – нечётное число.

Пример ввода:

```
*
9
```

Пример вывода:

```
*
**
***
****
*****
****
***
**
*
```

Задание 2. Напишите функцию print_fio(name, surname, patronymic), которая принимает три параметра:

- name – имя человека;
- surname – фамилия человека;
- patronymic – отчество человека;

Затем она выводит на печать ФИО человека.

Примечание. Предусмотрите тот факт, что все три буквы в ФИО должны иметь верхний регистр.

Пример ввода:
Александр
Пушкин
Сергеевич

Пример вывода:
ПАС

Задание 3. Напишите функцию print_digit_sum(), которая принимает одно натуральное число num и выводит на печать сумму его цифр.

Пример ввода:
12345

Пример вывода:
15

Задание 4. Напишите функцию convert_to_miles(km), которая принимает в качестве аргумента расстояние в километрах и возвращает расстояние в милях. Формула для преобразования: мили = километры * 0.6214.

Примечание. Приведённый ниже код:

```
print(convert_to_miles(1))
print(convert_to_miles(5))
print(convert_to_miles(10))
```

должен выводить:

0.6214
3.107
6.214

Задание 5. Напишите функцию get_days(month), которая принимает в качестве аргумента номер месяца и возвращает количество дней в данном месяце.

Примечание 1. Гарантируется, что передаваемый аргумент находится в диапазоне от 1 до 12 (включительно).

Примечание 2. Считайте, что год является невисокосным.

Примечание 3. Приведённый ниже код:

```
print(get_days(1))
print(get_days(2))
print(get_days(9))
```

должен выводить:

```
31
28
30
```

Задание 6. Напишите функцию get_factors(num), принимающую в качестве аргумента натуральное число и возвращающую список всех делителей данного числа.

Примечание. Приведённый ниже код:

```
print(get_factors(1))
print(get_factors(5))
print(get_factors(10))
```

должен выводить:

```
[1]
[1, 5]
[1, 2, 5, 10]
```

Задание 7. Напишите функцию number_of_factors(num), принимающую в качестве аргумента число и возвращающую количество делителей данного числа.

Примечание 1. Используйте уже реализованную функцию get_factors(num) из предыдущей задачи.

Примечание 2. Приведённый ниже код:

```
print(number_of_factors(1))
print(number_of_factors(5))
print(number_of_factors(10))
```

должен выводить:

```
1
2
```

Задание 8. Напомним, что строковый метод find('a') возвращает местоположение первого вхождения символа a в строке. Проблема заключается в том, что данный метод не находит местоположение всех символов a.

Напишите функцию с именем find_all(target, symbol), которая принимает два аргумента: строку target и символ symbol и возвращает список, содержащий все местоположения этого символа в строке.

Примечание 1. Если указанный символ не встречается в строке, то следует вернуть пустой список.

Примечание 2. Приведённый ниже код:

```
print(find_all('abcdabcaaa', 'a'))
print(find_all('abcadbcaaa', 'e'))
print(find_all('abcadbcaaa', 'd'))
```

должен выводить:

```
[0, 4, 7, 8, 9] [] [4]
```

Задание 9. Напишите функцию merge(list1, list2), которая принимает в качестве аргументов два отсортированных по возрастанию списка, состоящих из целых чисел, и объединяет их в один отсортированный список.

Примечание 1. Списки list1 и list2 могут иметь разную длину.

Примечание 2. Можно использовать списочный метод sort(), а можно обойтись и без него.

Примечание 3. Приведённый ниже код:

```
print(merge([1, 2, 3], [5, 6, 7, 8]))
print(merge([1, 7, 10, 16], [5, 6, 13, 20]))
```

должен выводить:

```
[1, 2, 3, 5, 6, 7, 8]
[1, 5, 6, 7, 10, 13, 16, 20]
```

Задание 10. Напишите функцию is_valid_triangle(side1, side2, side3), которая принимает в качестве аргументов три натуральных числа, и возвращает значение True, если существует невырожденный треугольник со сторонами side1, side2, side3, или False в противном случае.

Примечание. Приведённый ниже код:

```
print(is_valid_triangle(2, 2, 2))
print(is_valid_triangle(2, 3, 10))
print(is_valid_triangle(3, 4, 5))
```

должен выводить:

```
True False True
```

Задание 11. Напишите функцию is_prime(num), которая принимает в качестве аргумента натуральное число и возвращает значение True, если число является простым, или False в противном случае.

Примечание. Приведённый ниже код:

```
print(is_prime(1))
print(is_prime(10))
print(is_prime(17))
```

должен выводить:

```
False
False
True
```

Задание 12. Напишите функцию get_next_prime(num), которая принимает в качестве аргумента натуральное число num и возвращает первое простое число, большее числа num.

Примечание 1. Используйте функцию is_prime() из предыдущей задачи.

Примечание 2. Приведённый ниже код:

```
print(get_next_prime(6))
print(get_next_prime(7))
print(get_next_prime(14))
```

должен выводить:

```
7
11
17
```

Задание 13. Напишите функцию is_password_good(password), которая принимает в качестве аргумента строковое значение пароля password и возвращает значение True, если пароль является надёжным, или False в противном случае.

Пароль является надёжным, если:

- его длина не менее 8 символов;
- он содержит как минимум одну заглавную букву (верхний регистр);
- он содержит как минимум одну строчную букву (нижний регистр);
- он содержит хотя бы одну цифру.

Примечание. Приведённый ниже код:

```
print(is_password_good('aabbCC11OP'))
print(is_password_good('abC1pu'))
```

должен выводить:

```
True
False
```

Задание 14. Напишите функцию get_middle_point(x1, y1, x2, y2), которая принимает в качестве аргументов координаты концов отрезка (x 1 ;

y 1) и (x 2 ; y 2) и возвращает координаты точки являющейся серединой данного отрезка.

Примечание 1. Координаты середины отрезка с концами в точках (x 1 ; y 1) и (x 2 ; y 2) вычисляются по формуле: ((x 1 + x 2)/ 2 ; (y 1 + y 2)/ 2)

Примечание 2. Приведённый ниже код:

```
print(get_middle_point(0, 0, 10, 0))
print(get_middle_point(1, 5, 8, 3))
```

должен выводить:

```
5.0 0.0
        4.54.0
```

Задание 15. Напишите функцию get_circle(radius), которая принимает в качестве аргумента радиус окружности и возвращает два значения: длину окружности и площадь круга, ограниченного данной окружностью.

Примечание 1. Длина окружности и площадь круга радиуса r вычисляются по формулам:

$C = 2\pi r$

$S = \pi r^2$

Примечание 2. Для числа π используйте глобальную константу из модуля math.

Примечание 3. Приведённый ниже код:

```
print(get_circle(1))
print(get_circle(1.5))
```

должен выводить:

```
6.283185307179586 3.141592653589793
9.42477796076938 7.0685834705770345
```

Задание 16. Напишите функцию solve(a, b, c), которая принимает в качестве аргументов три целых числа a, b, c – коэффициенты квадратного уравнения $ax^2 + bx + c = 0$ и возвращает его корни в порядке возрастания.

Примечание 1. Гарантируется, что квадратное уравнение имеет хотя бы один корень.

Примечание 2. Приведённый ниже код:

```
print(solve(1, -4, -5))
print(solve(-2, 7, -5))
print(solve(1, 2, 1))
```

должен выводить:

```
-1.0 5.0
1.0 2.5
-1.0 -1.0
```

Тема 3. Алгоритмы поиска и сортировки

Контрольные вопросы:

1. Сформулируйте задачу поиска элемента в неупорядоченном массиве. Опишите алгоритм линейного поиска. Какова его временная сложность в лучшем, среднем и худшем случаях? При каких условиях линейный поиск может быть предпочтительнее более быстрых алгоритмов?
2. Опишите алгоритм бинарного поиска. Какое ключевое требование предъявляется к данным для применения этого алгоритма? Какова его временная сложность, и почему бинарный поиск значительно эффективнее линейного на больших объёмах данных?
3. Приведите рекурсивную и итеративную реализации бинарного поиска. Какая из них, на ваш взгляд, более понятна для начинающих, а какая — более эффективна с точки зрения использования памяти? Аргументируйте ответ.
4. Опишите алгоритм сортировки пузырьком. В чём заключается принцип его работы? Почему алгоритм получил такое название? Какова временная сложность пузырьковой сортировки в лучшем, среднем и худшем случаях?
5. Какие оптимизации можно применить к классической сортировке пузырьком? Объясните, как работает оптимизация с досрочным завершением при отсутствии перестановок и оптимизация с уменьшением проверяемой области после каждого прохода.
6. Опишите алгоритм сортировки вставками. В чём заключается его сходство с тем, как человек упорядочивает игральные карты в руке? Для каких типов входных данных этот алгоритм показывает наилучшую производительность?
7. В чём заключается идея быстрой сортировки? Объясните роль опорного элемента и процесса разделения массива. Почему выбор опорного элемента критически важен для производительности алгоритма?
8. Каковы преимущества и недостатки быстрой сортировки по сравнению с сортировкой пузырьком и сортировкой вставками? Почему на практике быстрая сортировка часто оказывается эффективнее даже алгоритмов с теоретически лучшей асимптотикой?
9. Объясните принцип работы сортировки слиянием. Какая стратегия лежит в основе этого алгоритма? Какова его временная сложность в лучшем, среднем и худшем случаях, и каков основной недостаток этого алгоритма (по памяти)?
10. Какие алгоритмы сортировки являются устойчивыми, а какие — нет? Что означает свойство устойчивости, и в каких практических задачах оно может оказаться важным? Приведите примеры устойчивых и неустойчивых алгоритмов из числа изученных.

Практические занятия:

Задание 1. Напишите функцию linear_search_all, которая принимает два аргумента: список и искомое значение. Функция должна возвращать список индексов всех элементов, равных искомому значению (в порядке возрастания). Если искомое значение не найдено, функция должна возвращать пустой список. Продемонстрируйте работу функции на следующих примерах:

- поиск числа 5 в списке [1, 5, 3, 5, 2, 5, 4];
- поиск буквы «а» в строке «программирование»;
- поиск значения 100 в списке [10, 20, 30].

Объясните, какова временная сложность реализованной функции в лучшем, среднем и худшем случаях.

Задание 2. Реализуйте две версии бинарного поиска в отсортированном списке:

- итеративную;
- рекурсивную.

Обе функции должны принимать отсортированный список и искомое значение, а возвращать индекс элемента (если элемент найден) или значение -1 (если не найден). Продемонстрируйте работу обеих функций на одном и том же наборе тестов:

- поиск существующего элемента (например, 42 в середине списка);
- поиск элемента, находящегося в начале списка;
- поиск элемента, находящегося в конце списка;
- поиск отсутствующего элемента.

Сравните код итеративной и рекурсивной версий: какая из них, на ваш взгляд, более читаема и почему?

Задание 3. Реализуйте алгоритм сортировки пузырьком для сортировки списка целых чисел по возрастанию. Базовая версия должна проходить по списку и менять соседние элементы, если они стоят в неправильном порядке. Добавьте в реализацию две оптимизации:

1. Если за весь проход по списку не было произведено ни одной перестановки, список уже отсортирован – досрочно завершите работу алгоритма.
2. После каждого прохода самый большой элемент «всплывает» в конец списка, поэтому на следующем проходе можно не проверять последние N элементов (где N – номер прохода).

Продемонстрируйте работу оптимизированной сортировки пузырьком на примере списков:

- [5, 1, 4, 2, 8] (обычный случай);
- [1, 2, 3, 4, 5] (уже отсортированный — проверка досрочного завершения);
- [9, 8, 7, 6, 5] (обратный порядок — максимальное количество перестановок).

Выведите на экран состояние списка после каждого прохода для наглядности.

Задание 4. Напишите функцию сортировки вставками, которая сортирует список по возрастанию. Алгоритм должен работать следующим образом: последовательно извлекается каждый элемент (начиная со второго) и вставляется на правильную позицию в уже отсортированную левую часть списка. После каждой вставки программа должна выводить текущее состояние списка и сообщение: «Вставлен элемент [значение] на позицию [индекс]». Продемонстрируйте работу на примере списка [12, 11, 13, 5, 6]. Объясните, почему сортировка вставками эффективна для частично отсортированных данных и для маленьких массивов.

Задание 5. Сгенерируйте три списка случайных целых чисел размером 1000, 5000 и 10000 элементов (диапазон значений от 1 до 100000). Реализуйте три алгоритма сортировки: пузырьком (оптимизированная версия), вставками и быструю сортировку (можно использовать встроенную функцию для быстрой сортировки или реализовать самостоятельно). Для каждого алгоритма и каждого размера списка замерьте время выполнения (используйте модуль time). Результаты оформите в виде таблицы:

Размер списка	Пузырьок (сек)	Вставками (сек)	Быстрая сортировка (сек)
1000	...	...	...
5000	...	...	...
10000	...	...	...

Напишите вывод: какой алгоритм оказался самым быстрым и почему.

Задание 6. Модифицируйте реализацию линейного поиска и бинарного поиска так, чтобы каждая из функций подсчитывала и возвращала количество выполненных операций сравнения элементов. Для каждого алгоритма:

- создайте отсортированный список из 100 целых чисел (от 1 до 100);
- выполните поиск элемента 50 (существующий);
- выполните поиск элемента 101 (несуществующий);
- выполните поиск элемента 1 (первый в списке);
- выполните поиск элемента 100 (последний в списке).

Для каждого случая выведите количество сравнений. Объясните, почему бинарный поиск даже для первого элемента может выполнить несколько сравнений, в то время как линейный поиск найдёт его с первой же попытки. Какой алгоритм в среднем эффективнее и почему?

Задание 7. Реализуйте алгоритм быстрой сортировки (рекурсивно). Предусмотрите возможность выбора опорного элемента (pivot) тремя разными способами:

1. Опорный элемент — всегда последний элемент в текущем подмассиве.

2. Опорный элемент — средний по индексу элемент (медиана трёх: первый, средний, последний).

3. Опорный элемент — случайный элемент из текущего подмассива.

Для каждого способа реализуйте отдельную функцию быстрой сортировки (или передавайте стратегию выбора как параметр). Протестируйте все три версии на следующих списках:

- [3, 7, 8, 5, 2, 1, 9, 6, 4] (обычный случай);
- [1, 2, 3, 4, 5, 6, 7, 8, 9] (уже отсортированный);
- [9, 8, 7, 6, 5, 4, 3, 2, 1] (обратный порядок).

Сравните время выполнения (или количество рекурсивных вызовов) для каждого способа на уже отсортированных данных. Какой выбор опорного элемента делает алгоритм устойчивым к плохим входным данным?

Тема 4. Анализ сложности алгоритмов

Контрольные вопросы:

1. Что такое временная сложность алгоритма, и для чего необходимо проводить её анализ? Чем анализ сложности отличается от простого измерения времени выполнения на конкретных входных данных?
2. Что такое «О-нотация»? Объясните математический смысл записи «f(n) = O(g(n))». Почему при анализе сложности мы отбрасываем константы и младшие члены?
3. Перечислите основные классы сложности от самой эффективной к самой неэффективной. Для каждого класса приведите пример алгоритма, обладающего такой сложностью.
4. Как оценивается сложность алгоритма, содержащего последовательные блоки кода? Как оценивается сложность условного оператора? Как оценивается сложность вложенных циклов? Сформулируйте общие правила.
5. Что такое пространственная сложность алгоритма? В чём отличие вспомогательной памяти от памяти под входные данные? Приведите примеры алгоритмов с малой временной, но большой пространственной сложностью, и наоборот.
6. Объясните понятия «лучший случай», «средний случай» и «худший случай» на примере алгоритма линейного поиска. Почему для анализа

алгоритмов часто рассматривают именно худший случай? В каких ситуациях средний случай может быть информативнее худшего?

7. Как оценить временную сложность рекурсивного алгоритма? Что такое рекуррентное соотношение, и как его можно решить (например, методом подстановки или с помощью основной теоремы о рекуррентных соотношениях)?

8. Сравните временную сложность линейного поиска. Насколько быстрее работает бинарный поиск при n = 1 000 000? Какие накладные расходы могут сделать бинарный поиск менее выгодным на практике?

9. Почему алгоритмы с квадратичной сложностью считаются неприемлемыми для обработки больших объёмов данных? Приведите конкретные оценки: во сколько раз алгоритм $O(n^2)$ медленнее алгоритма $O(n \cdot \log n)$ при n = 10 000 и при n = 1 000 000?

10. Что такое компромисс между временной и пространственной сложностью? Приведите пример ситуации, когда выгодно пожертвовать дополнительной памятью (увеличить пространственную сложность) ради выигрыша во времени выполнения, и наоборот.

Практические занятия:

Задание 1. Для каждого из приведённых ниже фрагментов кода определите временную сложность в «O»-нотации (лучший, средний и худший случай, если они различаются). Ответ обоснуйте, объяснив, сколько раз выполняется каждая операция и почему.

Фрагмент 1:

```
def example1(arr):
    n = len(arr)
    total = 0
    for i in range(n):
        total += arr[i]
    return total
```

Фрагмент 2:

```
def example2(n):
    result = 0
    for i in range(n):
        for j in range(n):
            result += i * j
    return result
```

Фрагмент 3:

```
def example3(arr):
```

```
    n = len(arr)
    for i in range(n):
        for j in range(i, n):
            if arr[i] > arr[j]:
                arr[i], arr[j] = arr[j], arr[i]
    return arr
```

Фрагмент 4:

```
def example4(n):
    i = 1
    count = 0
    while i < n:
        i = i * 2
        count += 1
    return count
```

Фрагмент 5:

```
def example5(n):
    if n <= 1:
        return n
    return example5(n-1) + example5(n-2)
```

Задание 2. Расположите следующие функции в порядке возрастания скорости роста (от самой медленно растущей до самой быстро растущей). Для каждой функции укажите соответствующий класс сложности в «O»-нотации (например, константная, логарифмическая, линейная и т.д.).

- $f1(n) = 1000$
- $f2(n) = \log_2(n)$
- $f3(n) = n$
- $f4(n) = n \cdot \log_2(n)$
- $f5(n) = n^2$
- $f6(n) = 2^n$
- $f7(n) = n!$
- $f8(n) = 10 \cdot n + 5$
- $f9(n) = n^3$
- $f10(n) = \sqrt{n}$

Для каждой пары соседних функций в полученном порядке приведите пример значения n, при котором разница во времени выполнения становится заметной.

Задание 3. Возьмите реализации трёх алгоритмов сортировки из предыдущих тем: пузырьком (оптимизированная версия), вставками и быструю сортировку. Для каждого алгоритма:

1. Определите временную сложность в лучшем, среднем и худшем случае (запишите в «O»-нотации).

2. Опишите, при каких входных данных достигается каждый из этих случаев.
3. Определите пространственную сложность для каждого алгоритма.
4. Заполните итоговую таблицу:

Алгоритм	Лучший случай	Средний случай	Худший случай	Доп. память
Пузырьком	...	...	...	...
Вставками	...	...	...	...
Быстрая	...	...	...	...

Задание 4. Реализуйте функцию, которая вычисляет сумму всех элементов квадратной матрицы (двумерного списка) размером N×N тремя разными способами:

- Способ А: двойной вложенный цикл по строкам и столбцам;
- Способ Б: однократный проход по всем элементам;
- Способ В: рекурсивное суммирование.

Для каждого способа:

1. Теоретически определите временную сложность в «O»-нотации.
2. Проведите замеры времени для N = 10, 50, 100, 200, 500 (матрицы заполните случайными числами).
3. Постройте график зависимости времени от N (оси: N — горизонтальная, время — вертикальная).
4. Сравните экспериментальные результаты с теоретическими ожиданиями. Если наблюдаются расхождения, объясните возможные причины (например, влияние накладных расходов на вызовы функций).

Задание 5. Для каждого из следующих фрагментов кода определите, сколько раз выполняется самая внутренняя операция (например, result += 1) как функция от n. Ответ представьте в виде формулы (в замкнутой форме, без знаков суммирования) и в виде асимптотической оценки в «O»-нотации.

Фрагмент А:

```
result = 0
for i in range(n):
    for j in range(n):
        result += 1
```

Фрагмент Б:

```
result = 0
for i in range(n):
    for j in range(i):
        result += 1
```

Фрагмент В:

```
result = 0
for i in range(n):
    for j in range(n):
        for k in range(n):
            result += 1
```

Фрагмент Г:

```
result = 0
for i in range(n):
    for j in range(n):
        if i == j:
            for k in range(n):
                result += 1
```

Фрагмент Д:

```
result = 0
i = 1
while i < n:
    for j in range(n):
        result += 1
    i = i * 2
```

Задание 6. Для линейного и бинарного поиска выполните следующие вычисления:

1. Пусть операция сравнения в линейном поиске занимает 1 условную единицу времени. Сколько сравнений выполнит линейный поиск в худшем случае для массива размером N?
2. Сколько сравнений выполнит бинарный поиск в худшем случае для массива размером N?
3. Заполните таблицу для N = 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072:

N	Линейный поиск	Бинарный поиск	Во сколько раз бинарный быстрее
8	...	...	...

4. При каком N преимущество бинарного поиска становится стократным? При каком N – тысячекратным?
5. Какие накладные расходы не учитываются в этом сравнении, но важны для практического применения?

Задание 7. Для каждого из приведённых рекурсивных алгоритмов:

1. Составьте рекуррентное соотношение, описывающее временную сложность T(n).

2. Решите рекуррентное соотношение (методом подстановки, методом итераций или с помощью основной теоремы о рекуррентных соотношениях).

3. Запишите итоговую асимптотическую оценку в «O»-нотации.

Алгоритм 1:

```
def factorial(n):
    if n <= 1:
        return 1
    return n * factorial(n - 1)
```

Алгоритм 2:

```
def binary_search(arr, target, left, right):
    if left > right:
        return -1
    mid = (left + right) // 2
    if arr[mid] == target:
        return mid
    elif arr[mid] > target:
        return binary_search(arr, target, left, mid - 1)
    else:
        return binary_search(arr, target, mid + 1, right)
```

Алгоритм 3:

```
def sum_recursive(arr, left, right):
    if left == right:
        return arr[left]
    mid = (left + right) // 2
    return sum_recursive(arr, left, mid) + sum_recursive(arr, mid + 1, right)
```

Алгоритм 4:

```
def power(base, exp):
    if exp == 0:
        return 1
    if exp % 2 == 0:
        half = power(base, exp // 2)
        return half * half
    else:
        return base * power(base, exp - 1)
```

Задание 8. Вам необходимо обработать массив данных размером N (от 10^6 до 10^8 элементов). Для решения задачи подходят четыре алгоритма со

следующими характеристиками (время в условных единицах, где 1 единица — 1 наносекунда на элементарную операцию):

Алгоритм	Формула времени T(N)
A1	$T_1(N) = 5 \cdot 10^{-6} \cdot N$
A2	$T_2(N) = 10^{-3} \cdot N \cdot \log_2(N)$
A3	$T_3(N) = 10^{-9} \cdot N^2$
A4	$T_4(N) = 10^{-7} \cdot N \cdot \log_2(N)$

Для каждого N из набора: N = 1000, 10000, 100000, 1 000 000, 10 000 000, 100 000 000 вычислите ожидаемое время выполнения (в секундах, наносекундах — любой наглядной единице) для каждого алгоритма. Результаты оформите в виде таблицы. Определите:

1. Какой алгоритм лучше всего подходит для N = 1000?
2. Какой алгоритм лучше всего подходит для $N = 10^8$?
3. Существует ли значение N, при котором происходит «перелом» — более сложный алгоритм с лучшей асимптотикой начинает обгонять более простой алгоритм с худшей асимптотикой?
4. Какой вывод о выборе алгоритмов вы можете сделать на основе этого упражнения (связь между константами и асимптотикой)?

РАЗДЕЛ 2. СТРУКТУРЫ ДАННЫХ И ФАЙЛОВ НА PYTHON

Тема 5. Кортежи и множества в Python

Контрольные вопросы:

1. Что такое кортеж в Python? Чем кортеж принципиально отличается от списка с точки зрения изменяемости? Какие преимущества даёт неизменяемость кортежа по сравнению со списком?
2. В каких ситуациях использование кортежа предпочтительнее использования списка? Приведите не менее трёх примеров задач, где кортеж был бы более подходящей структурой данных, чем список.
3. Что такое распаковка кортежа? Приведите пример, как с помощью распаковки можно обменять значения двух переменных без использования временной переменной. Как распаковка работает с кортежами разной длины?
4. Почему кортеж может использоваться в качестве ключа словаря, а список — нет? Какое требование предъявляется к объектам, используемым в качестве ключей словаря, и как неизменяемость кортежа обеспечивает выполнение этого требования?
5. Что такое множество в Python? Какими свойствами обладает множество? Как создать пустое множество, и почему нельзя использовать для этого фигурные скобки?

6. Перечислите основные операции над множествами: объединение, пересечение, разность, симметрическая разность. Для каждой операции приведите пример и объясните, какой результат она возвращает.

7. Как проверить, является ли одно множество подмножеством или надмножеством другого? Какие операторы и методы для этого существуют? Приведите примеры.

8. Какие практические задачи удобно решать с помощью множеств? Опишите, как с помощью множества можно удалить дубликаты из списка, найти общие элементы двух коллекций, проверить наличие элемента в большом объёме данных.

9. Что такое замороженное множество? Чем оно отличается от обычного множества? В каких ситуациях возникает необходимость использовать именно frozenset, а не set?

10. Сравните множества и кортежи с точки зрения производительности операций: проверка принадлежности элемента, добавление элемента, удаление элемента. Для каких операций множества дают значительный выигрыш и почему?

Практические занятия:

Задание 1. Дополните приведенный код, используя индексацию кортежа, чтобы переменная last, содержала последний элемент кортежа countries.

```
countries = ('Russia', 'Argentina', 'Spain', 'Slovakia', 'Canada', 'Slovenia', 'Italy')
last =
print(last)
```

Задание 2. Дополните приведенный код, используя срезы, так чтобы он вывел первые 6 элементов кортежа primes.

Примечание. Результатом вывода должна быть строка:

(2, 3, 5, 7, 11, 13).

```
primes = (2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71)
print( )
```

Задание 3. Дополните приведенный код, используя срезы, так чтобы он вывел элементы кортежа countries кроме первых двух.

Примечание. Результатом вывода должна быть строка:

('Slovakia', 'Canada', 'Slovenia', 'Italy', 'Spain', 'Ukraine', 'Chile', 'Cameroon')

```
countries = ('Russia', 'Argentina', 'Slovakia', 'Canada', 'Slovenia', 'Italy', 'Spain', 'Ukraine', 'Chile', 'Cameroon')
print( )
```

Задание 4. Дополните приведенный код, используя срезы, чтобы он вывел все элементы кортежа countries, кроме последних трех.

countries = ('Russia', 'Argentina', 'Slovakia', 'Canada', 'Slovenia', 'Italy', 'Spain', 'Ukraine', 'Chile', 'Cameroon')
print( )

Задание 5. Дополните приведенный код, используя срезы, чтобы он вывел все элементы кортежа countries, кроме двух последних и трех первых.

countries = ('Russia', 'Argentina', 'Slovakia', 'Canada', 'Slovenia', 'Italy', 'Spain', 'Ukraine', 'Chile', 'Cameroon')
print( )

Задание 6. Дополните приведенный код так, чтобы переменная number содержала количество элементов кортежа countries.

countries = ('Romania', 'Poland', 'Estonia', 'Bulgaria', 'Slovakia', 'Slovenia', 'Hungary')
number =
print(number)

Задание 7. Дополните приведенный код так, чтобы он вывел сумму минимального и максимального элементов кортежа numbers.

numbers = (12.5, 3.1415, 2.718, 9.8, 1.414, 1.1618, 1.324)
print( )

Задание 8. Дополните приведенный код так, чтобы переменная index содержала индекс элемента «Slovenia» в кортеже countries.

countries = ('Russia', 'Argentina', 'Spain', 'Slovakia', 'Canada', 'Slovenia', 'Italy')
index =
print(index)

Задание 9. Дополните приведенный код так, чтобы переменная number, содержала количество вхождений «Spain» в кортеж countries.

countries = ('Russia', 'Argentina', 'Spain', 'Slovakia', 'Canada', 'Slovenia', 'Italy', 'Spain', 'Ukraine', 'Chile', 'Spain', 'Cameroon')
number =
print(number)

Задание 10. Дополните приведенный код, используя операторы конкатенации (+) и умножения кортежа на число (*), чтобы он вывел кортеж: (1, 2, 3, 1, 2, 3, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 8, 9, 10, 11, 12, 13).

numbers1 = (1, 2, 3)
numbers2 = (6,)
numbers3 = (7, 8, 9, 10, 11, 12, 13)
...
print( )

Задание 11. В переменную city_name вводится название города (например, Москва), а в переменную city_year – год его основания (например,

1147). Заполните пропущенную строку таким образом, чтобы в переменной city оказался кортеж из значений этих двух переменных (сначала название города, затем год основания).

```
city_name = input()
city_year = int(input())
...
print(city)
```

Задание 12. Дополните приведенный код, так чтобы получить список, содержащий только непустые кортежи исходного списка tuples, не меняя порядка их следования.

Примечание. Используйте списочное выражение.

```
tuples = [(), (), ('',), ('a', 'b'), (), ('a', 'b', 'c'), (1,), (), (), ('d',), ('', ''), ()]
non_empty_tuples =
print(non_empty_tuples)
```

Задание 13. Дополните приведенный код так, чтобы переменная new_tuples содержала список кортежей на основе списка tuples с последним элементом каждого кортежа, замененным на численное значение 100.

Примечание. Используйте списочное выражение.

```
tuples = [(10, 20, 40), (40, 50, 60), (70, 80, 90), (10, 90), (1, 2, 3, 4), (5, 6, 10, 2, 1, 77)]
new_tuples =
print(new_tuples)
```

Задание 14. Множества A и B содержат 5 и 6 элементов соответственно, а множество A∩B – 2 элемента. Сколько элементов в множестве A∪B?

Задание 15. Каждый ученик в классе изучает английский или французский язык. Английский язык изучает 25 человек, французский – 27, а оба языка – 18. Сколько учащихся в классе?

Задание 16. В одном из классов онлайн-школы учится 67 человек. Из них 47 умеют решать задачи с параметрами, 35 –экономические задачи, а 23 – и те и другие. Сколько человек в классе не умеют решать ни экономические задачи, ни задачи с параметрами?

Задание 17. В классе учатся 30 учеников. Среди них 17 отличников по математике, 10 отличников по физике и 13 — по информатике. Трое — отличники по всем предметам, пятеро — по математике и физике, четверо — по физике и информатике, а 6 человек — по математике и информатике. Сколько учеников не являются отличниками ни по одному из этих предметов?

Задание 18. В классе 36 учеников, из которых двое не знают иностранных языков. Английским языком владеют 25 учеников, немецким — 11, французским — 17 человек. Известно, что и английским, и немецким языком владеют 6 учеников, и английским, и французским — 10 учеников, и немецким, и французским — 4 ученика. Сколько учеников владеют всеми тремя языками?

Задание 19. На летних каникулах Тимур и ученики онлайн-школы решили отдохнуть. В результате n учеников школы поехали отдыхать на море, m учеников съездили в деревню, а k учеников сходили в горы. Оказалось, что и в деревне, и на море были x учеников, а в деревне и в горах — y учеников. Побывать и в горах, и на море не удалось никому. Напишите программу для определения количества учеников в школе, если никто не смог посетить все три места сразу, а z учеников писали ДВИ по математике для поступления в МГУ, и никуда не ездили.

- Формат входных данных. На вход программе подаются числа n,m,k,x,y,z, каждое на отдельной строке.
- Формат выходных данных. Программа должна вывести одно число в соответствии с условием задачи.

Тема 6. Применение и работа со словарями Python

Контрольные вопросы:

1. Что такое словарь в Python? Опишите его структуру. Какие требования предъявляются к ключам словаря? Почему в качестве ключа нельзя использовать список или другое множество?
2. Какими способами можно создать словарь в Python? Перечислите не менее четырёх способов. Приведите пример для каждого способа.
3. Как выполняется доступ к значению по ключу в словаре? Чем отличается прямой доступ через квадратные скобки от использования метода получения значения с значением по умолчанию? В каких случаях предпочтительнее каждый из способов?
4. Как добавить новую пару ключ-значение в словарь? Как изменить значение существующего ключа? Как удалить пару по ключу? Какие исключения могут возникнуть при этих операциях?
5. Что такое генератор словаря? Приведите пример создания словаря, где ключами являются числа от 1 до 10, а значениями — их квадраты. Чем генератор словаря отличается от генератора списка?
6. Какие методы словаря позволяют получить все ключи, все значения и все пары (ключ, значение) в виде итерируемых объектов? Как выполнить итерацию по словарю в цикле «для»? Что именно перебирается по умолчанию?
7. Объясните принцип работы словаря как хэш-таблицы. Почему операции доступа, добавления и удаления элемента в среднем выполняются

за O(1)? При каких условиях производительность словаря может ухудшиться до O(n)?

8. Как объединить два словаря в Python? Назовите не менее двух способов (оператор объединения, метод обновления, распаковка). В чём разница между этими способами, и что происходит при совпадении ключей?

9. Какие практические задачи удобно решать с помощью словарей? Приведите примеры: подсчёт частоты элементов, реализация switch/case, кэширование результатов, группировка данных по ключу.

10. Сравните словарь и список по производительности для следующих операций: доступ по индексу/ключу, проверка наличия элемента, добавление элемента, удаление элемента. Для каких задач словарь является оптимальным выбором, а для каких — список?

Практические занятия:

Задание 1. Школьное расписание хранится в виде словаря schedule, где ключ — день недели, а значение — список предметов.

Датасет:

```
schedule = {
   "Понедельник": ["Математика", "Физика", "История"],
   "Вторник": ["Биология", "Английский язык", "Информатика"]
}
```

Задачи:

1. Во вторник после "Информатики" добавьте "Физкультуру".
2. Добавьте новый день — "Среда", с предметами: "Химия", "Литература", "Обществознание".
3. Выведите расписание по дням.

Задание 2. Словарь orders хранит информацию о заказах, где ключ — номер заказа, а значение — словарь с полями товары (список кортежей (название, количество)) и статус.

Датасет:

```
orders = {
   1001: {"товары": [("Мышь", 2), ("Клавиатура", 1)], "статус": "оформлен"},
   1002: {"товары": [("Монитор", 1)], "статус": "в обработке"}
}
```

Задачи:

1. Добавьте к заказу 1001 ещё 1 наушники.
2. Измените статус заказа 1002 на "доставлен".
3. Создайте новый заказ 1003: товары — "Флешка" (2 шт.), "Коврик" (1 шт.), статус — "оформлен".
4. Выведите все заказы с перечнем товаров и статусом.

Задание 3. Результаты тестов студентов хранятся в словаре results, где ключ — фамилия студента, а значение — словарь из предметов и оценок.

Датасет:

```
results = {
   "Иванов": {"Математика": 4, "Физика": 5},
   "Петрова": {"Математика": 3, "Информатика": 5}
}
```

Задачи:

1. Добавить Иванову оценку 4 по Информатике.
2. У Петровой оценку по Математике повысить на 1 балл.
3. Удалить у Иванова Физику.
4. Вывести всех студентов и их оценки.

Задание 4. Словарь income содержит месяцы как ключи и значения в виде кортежей: (зарплата, подработка, инвестиции).

Датасет:

```
income = {
   "Январь": (40000, 5000, 2000),
   "Февраль": (42000, 6000, 2500)
}
```

Задачи:

1. В марте доходы составили: зарплата — 43000, подработка — 5500, инвестиции — 3000. Добавьте данные.
2. В феврале зарплату пересчитали, она составила 45000. Обновите запись.
3. Подсчитайте и выведите общий доход за каждый месяц.

Задание 5. В словаре participants хранится информация об участниках соревнования: ключ — фамилия, значение — кортеж (возраст, [виды спорта]).

Датасет:

```
participants = {
   "Смирнов": (15, ["Бег", "Плавание"]),
   "Кузнецова": (16, ["Гимнастика"])
}
```

Задачи:

1. К Смирнову добавьте вид спорта "Фехтование".
2. У Кузнецовой возраст изменился на 17 лет.
3. Добавьте нового участника — "Иванов", 14 лет, виды спорта: "Бег", "Прыжки".
4. Выведите список участников в формате: Фамилия (возраст): вид1, вид2, ... видN

Задание 6. Покупки хранятся в словаре shopping_list, где ключ — категория (например, "Продукты", "Хозяйственное"), а значение — список товаров.

Датасет:

```
shopping_list = {
   "Продукты": ["Хлеб", "Молоко"],
   "Хозяйственное": ["Мыло", "Губки"]
}
```

Задачи:

1. Добавьте в категорию "Продукты" товар "Сыр".
2. Удалите "Губки" из "Хозяйственного".
3. Добавьте новую категорию "Электроника" с товарами "Батарейки", "Лампочка".
4. Выведите все категории и соответствующие им товары.

Тема 7. Работа с текстовыми и бинарными файлами в Python

Контрольные вопросы:

1. Какие режимы открытия файла существуют в Python? Опишите режимы чтения, записи, добавления, чтения и записи одновременно, а также бинарные версии этих режимов. Для чего нужен каждый из них?
2. Что такое контекстный менеджер при работе с файлами? В чём его преимущество перед ручным открытием и закрытием файла? Что произойдёт, если забыть закрыть файл?
3. Какие способы чтения данных из текстового файла существуют? Опишите методы: чтение всего файла целиком, построчное чтение в цикле, чтение всех строк в список. В каких случаях каждый способ предпочтителен?
4. Как выполняется запись данных в текстовый файл? В чём разница между методами записи строки и записи нескольких строк? Как добавить данные в конец существующего файла, не затирая его содержимое?
5. Что такое кодировка символов? Почему при работе с текстовыми файлами важно правильно указывать кодировку? Какие проблемы могут возникнуть при открытии файла с неверной кодировкой, и как их обработать?
6. Чем работа с бинарными файлами отличается от работы с текстовыми? В каких режимах открываются бинарные файлы? Какие типы данных можно сохранять в бинарные файлы и для чего это может потребоваться?
7. Что такое сериализация? Как с помощью модуля pickle сохранить объект Python в файл и восстановить его обратно? Какие ограничения и риски безопасности связаны с использованием pickle?

8. Как управлять позицией чтения/записи в файле? Что делают методы для получения текущей позиции и для перемещения указателя? Приведите пример, когда может потребоваться перемещение указателя.
9. Какие форматы структурированных данных широко используются для обмена данными между программами? Как в Python выполнить чтение и запись данных в формате JSON? В формате CSV? Какие встроенные модули для этого существуют?
10. Какие исключения могут возникнуть при работе с файлами? Перечислите не менее трёх типов ошибок. Как корректно обрабатывать такие ситуации с помощью конструкции «try-except»?

Практические занятия:

1. На вход программе подается строка с именем текстового файла. Напишите программу, которая выводит на экран его содержимое. (Считайте, что исполняемая программа и указанный файл находятся в одной папке)

2. На вход программе подается строка с именем текстового файла. Напишите программу, которая выводит на экран его предпоследнюю строку. (Считайте, что исполняемая программа и указанный файл находятся в одной папке)

3. Вам доступен текстовый файл lines.txt из нескольких строк. Напишите программу, которая выводит на экран случайную строку из этого файла.

4. Вам доступен текстовый файл numbers.txt из двух строк, на каждой из них записано целое число. Напишите программу, выводящую на экран сумму этих чисел.

5. Вам доступен текстовый файл nums.txt. В файле записано два целых числа, они могут быть разделены символами пробела и конца строки. Напишите программу, выводящую на экран произведение этих чисел. (Считайте, что исполняемая программа и указанный файл находятся в одной папке)

6. Вам доступен текстовый файл prices.txt с информацией о заказе из интернет магазина. В нем каждая строка с помощью символа табуляции (\t) разделена на три колонки:
   - наименование товара;
   - количество товара (целое число);
   - цена (в рублях) товара за 1 шт (целое число).

   Напишите программу, выводящую на экран общую стоимость заказа.

7. Вам доступен текстовый файл text.txt с одной строкой текста. Напишите программу, которая выводит на экран эту строку в обратном

порядке. (Считайте, что исполняемая программа и указанный файл находятся в одной папке)

8. Вам доступен текстовый файл data.txt, в котором записаны строки текста. Напишите программу, выводящую все строки данного файла сначала в прямом, потом в обратном порядке: сначала последнюю строку, затем предпоследнюю и т.д.

9. Вам доступен текстовый файл lines2.txt, в котором записаны строки текста. Напишите программу, которая выводит все строки наибольшей длины из файла, не меняя их порядок.

10. Вам доступен текстовый файл numbers.txt, каждая строка которого может содержать одно или несколько целых чисел, разделенных одним или несколькими пробелами. Напишите программу, которая вычисляет сумму чисел в каждой строке и выводит эту сумму на экран (для каждой строки выводится сумма чисел в этой строке).

11. Вам доступен текстовый файл file.txt, набранный латиницей. Напишите программу, которая выводит количество букв латинского алфавита, слов и строк.

12. Вам доступны два текстовых файла first_names.txt и last_names.txt, один с именами, другой с фамилиями. Напишите программу, которая с помощью модуля random создает 3 случайные пары имя + фамилия, а затем выводит их, каждую на отдельной строке.

13. Вам доступен текстовый файл population.txt с названиями стран и численностью их населения, разделенными символом табуляции '\t'. Напишите программу выводящую все страны, название которых начинается с буквы 'G', численность населения которых больше чем 500000 человек, не меняя их порядок.

5.3. Один или несколько тематических блоков дисциплины завершаются контрольной работой по разделу (далее – КР). Текущий контроль успеваемости по дисциплине предусматривает не менее 2 (двух) и не более 10 (десяти) КР в течение периода освоения дисциплины.

Максимальное количество баллов за любой тип работ в рамках КР составляет 100 (сто) баллов.

Распределение весовых коэффициентов по КР в рамках текущего контроля успеваемости по дисциплине и формулы расчета:

Наименование контрольной работы	Максимальное количество баллов за работу в рамках КР, которое может набрать студент	Коэффициент веса контрольной работы	Результат контрольной работы, участвующий в формировании итоговой балльной оценки по дисциплине
КР 1	100	0,15	15
КР 2	100	0,15	15
Итого:	x	0,30	30

Формула расчета результата контрольной работы:

Результат контрольной работы = Количество баллов за работу в рамках КР X Коэффициент веса контрольной работы.

5.4. Формы текущего контроля успеваемости обучающихся в рамках КР и типовые оценочные материалы:

КР-1

Раздел 1. Процедурное программирование на Python

Задание 1. Напишите программу, которая решает следующую задачу: «Дан список целых чисел. Необходимо отфильтровать только чётные числа, затем каждое из них возвести в квадрат, после чего вычислить сумму полученных значений». Реализуйте три варианта решения, каждый из которых демонстрирует определённую парадигму программирования:

- императивный подход (циклы и условные операторы, явное управление состоянием);
- процедурный подход (разбиение на отдельные функции для фильтрации, преобразования и суммирования);
- функциональный подход (с использованием функций высшего порядка filter, map и reduce или генераторов списков).

После реализации сравните все три подхода по критериям: читаемость, объём кода, возможность повторного использования компонентов.

Задание 2. Разработайте функцию create_power_function, которая принимает целочисленный показатель степени exp и возвращает новую функцию, возводящую переданное ей число в эту степень. Продемонстрируйте работу функции, создав с её помощью функции square (возведение в квадрат), cube (возведение в куб) и quad (возведение в четвёртую степень). Затем создайте список из чисел от 1 до 10 и для каждой из трёх полученных функций вычислите и выведите

результат применения этой функции ко всем элементам списка. Объясните механизм замыкания: какие переменные и откуда «запоминает» возвращаемая функция.

Задание 3. Реализуйте рекурсивную функцию is_palindrome(s), которая проверяет, является ли строка s палиндромом (читается одинаково слева направо и справа налево). Функция должна игнорировать пробелы и не учитывать регистр символов. Рекурсивный алгоритм должен сравнивать первый и последний символы строки, а затем рекурсивно проверять подстроку без этих символов. Обязательно предусмотрите базовые случаи (строка из одного символа или пустая строка). Продемонстрируйте работу функции на примерах: «А роза упала на лапу Азора», «Python», «12321». Проанализируйте глубину рекурсии и временную сложность реализованного алгоритма.

Задание 4. Вам дан отсортированный по возрастанию список целых чисел (размером 100 элементов, можно сгенерировать случайно и отсортировать). Реализуйте две функции:

- linear_search_count_comparisons(arr, target) – выполняет линейный поиск и возвращает количество выполненных операций сравнения элементов;
- binary_search_count_comparisons(arr, target) – выполняет бинарный поиск и возвращает количество выполненных операций сравнения элементов.

Выполните поиск для трёх значений: элемента, находящегося в середине списка; элемента, находящегося в начале списка; элемента, отсутствующего в списке. Для каждого случая выведите количество сравнений для обоих алгоритмов. Объясните, почему бинарный поиск даже для первого элемента может выполнить несколько сравнений, и в каком случае линейный поиск оказывается эффективнее бинарного.

Задание 5. Реализуйте алгоритм быстрой сортировки (quick sort) с выбором опорного элемента как «медианы трёх» (первый, средний, последний элементы текущего подмассива). Продемонстрируйте его работу на следующих трёх списках:

- список случайных целых чисел размером 20;
- список, уже отсортированный по возрастанию;
- список, отсортированный по убыванию.

Для каждого случая выведите исходный и отсортированный список. Также выведите количество рекурсивных вызовов, выполненных при сортировке каждого списка. Объясните, почему выбор опорного элемента как медианы трёх улучшает производительность алгоритма на уже отсортированных данных.

Задание 6. Для каждого из следующих фрагментов кода определите временную сложность в «О»-нотации (лучший, средний и худший случай,

если они различаются). Ответ обоснуйте, объяснив, сколько раз выполняется каждая операция как функция от n.

Фрагмент А:

```
def func_a(n):
    s = 0
    for i in range(n):
        for j in range(n):
            s += i * j
    return s
```

Фрагмент Б:

```
def func_b(n):
    s = 0
    for i in range(n):
        for j in range(i, n):
            s += i * j
    return s
```

Фрагмент В:

```
def func_c(n):
    i = 1
    count = 0
    while i < n:
        for j in range(100):
            count += 1
        i *= 2
    return count
```

Фрагмент Г:

```
def func_d(arr):
    n = len(arr)
    for i in range(n):
        for j in range(n):
            if arr[i] < arr[j]:
                return i
    return -1
```

КР-2
Раздел 2. Структуры данных и файлов на Python

Задание 1. Дан список целых чисел, содержащий повторяющиеся элементы. Напишите программу, которая:

1. Создаёт из этого списка множество, тем самым удаляя дубликаты.
2. Находит и выводит элементы, которые встречаются в исходном списке ровно один раз (уникальные элементы).
3. Находит и выводит элементы, которые встречаются в исходном списке более одного раза (повторяющиеся элементы).
4. Выполняет операцию объединения полученного множества уникальных элементов с множеством {10, 20, 30} и выводит результат.
5. Определяет, является ли множество уникальных элементов подмножеством множества чисел от 1 до 100.

Все операции должны быть реализованы с использованием встроенных типов set и соответствующих методов. Продемонстрируйте работу на примере списка: [5, 2, 8, 2, 9, 1, 5, 5, 3, 8, 7, 10].

Задание 2. Опишите, в каких ситуациях использование кортежа предпочтительнее списка. Затем выполните практическое задание: дан список кортежей, где каждый кортеж содержит три элемента: название товара, цену и количество на складе (например, [("яблоки", 50, 100), ("бананы", 30, 150), ("апельсины", 70, 80)]). Напишите программу, которая:

1. Создаёт новый список, содержащий только названия товаров (используя распаковку кортежей).
2. Вычисляет общую стоимость всех товаров на складе (цена × количество) и выводит результат.
3. Добавляет в исходный список новый товар, введённый пользователем, с проверкой, что пользователь ввёл три значения.
4. Сортирует список товаров по цене (по второму элементу кортежа) и выводит отсортированный результат.

Объясните, почему для представления записи о товаре удобно использовать кортеж, а не список.

Задание 3. Напишите программу «Частотный анализатор текста». Программа запрашивает у пользователя строку текста (на русском или английском языке) и выполняет следующие действия:

1. Приводит весь текст к нижнему регистру и удаляет все знаки препинания (точки, запятые, восклицательные знаки, вопросительные знаки и т.д.).
2. Разбивает текст на отдельные слова (слова разделяются пробелами).
3. Создаёт словарь, где ключами являются слова, а значениями — количество их вхождений в текст.
4. Выводит на экран:
   - общее количество уникальных слов;
   - топ-5 самых частых слов и их частоту;
   - слово, которое встречается реже всего (если таких несколько — вывести любое).
5. Предоставляет пользователю возможность ввести слово и узнать, сколько раз оно встретилось в тексте (с учётом того, что слова могут быть введены в любом регистре).

Все операции должны быть реализованы с использованием словарей и их методов.

Задание 4. С помощью генератора словаря создайте и выведите на экран следующие словари:

1. Словарь, где ключами являются числа от 1 до 15, а значениями — их кубы (число, возведённое в третью степень).
2. Словарь, где ключами являются буквы строки «программирование», а значениями — количество вхождений каждой буквы в эту строку (без учёта регистра).
3. Словарь, где ключами являются числа от 1 до 20, а значениями — «чётное» для чётных чисел и «нечётное» для нечётных.
4. Словарь, полученный из исходного словаря {"a": 1, "b": 2, "c": 3, "d": 4}, где ключи и значения поменяны местами (значение становится ключом, а ключ — значением). Объясните, какое требование должно выполняться для значений исходного словаря, чтобы такая операция была корректной.

Задание 5. Создайте текстовый файл data.txt со следующим содержимым:

```
Иванов,Иван,85
Петрова,Мария,92
Сидоров,Алексей,78
Иванова,Анна,85
Козлов,Дмитрий,95
```

Петрова,Елена,88

Каждая строка содержит фамилию, имя и оценку, разделённые запятыми. Напишите программу, которая:

1. Считывает данные из файла data.txt.
2. Для каждой строки создаёт словарь с ключами "surname", "name", "score".
3. Сохраняет все словари в список.
4. Вычисляет и выводит:

- среднюю оценку по всем студентам;
- максимальную оценку и фамилию и имя студента, получившего её (если несколько — вывести всех);
- словарь, где ключ — фамилия, а значение — список оценок (с учётом того, что у одной фамилии может быть несколько записей с разными именами или повторными оценками).

5. Записывает в новый файл sorted_by_score.txt всех студентов, отсортированных по убыванию оценки (каждая строка в формате: «Иванов Иван: 85»).

Предусмотрите обработку ошибки, если исходный файл не найден.

Задание 6. Напишите программу, которая:

1. Создаёт список словарей (не менее 5 элементов) с произвольными данными о книгах: название, автор, год издания, цена.
2. Сохраняет этот список в бинарный файл с помощью модуля pickle.
3. Затем считывает данные из этого бинарного файла обратно.
4. Выводит на экран все считанные книги в отформатированном виде (каждая книга с новой строки).
5. Добавляет в список новую книгу (данные вводятся пользователем) и снова сохраняет обновлённый список в тот же файл (перезаписывая старый).
6. Реализует возможность поиска книги по автору: пользователь вводит фамилию автора, программа выводит все книги этого автора из загруженного списка.

В ответе объясните, в каких ситуациях сериализация через pickle удобна, а в каких лучше использовать текстовые форматы (JSON, CSV). Какие риски безопасности связаны с использованием pickle при загрузке данных из ненадёжных источников?

6. Формы промежуточной аттестации, критерии и шкала оценивания, типовые оценочные материалы по дисциплине

6.1. Промежуточная аттестация по дисциплине «Программирование на Python» проводится в форме экзамена во втором семестре в письменной форме. Обучающийся получает два теоретических вопроса и одно практическое задание.

Теоретические вопросы направлены на проверку:

- понимания основных парадигм программирования (императивная, процедурная, объектно-ориентированная, функциональная) и их особенностей применительно к Python;
- знания механизмов создания и использования функций (параметры, аргументы, области видимости, рекурсия, лямбда-выражения, замыкания, функции высшего порядка);
- понимания принципов работы алгоритмов поиска (линейный, бинарный) и сортировки (пузырьком, вставками, быстрая, слиянием), их сильных и слабых сторон;
- знания методов анализа сложности алгоритмов (временная и пространственная сложность, «O»-нотация, лучший/средний/худший случай);
- понимания структур данных Python (кортежи, множества, словари), их свойств, операций и областей применения;
- знания способов работы с текстовыми и бинарными файлами (режимы открытия, чтение/запись, сериализация, работа с JSON и CSV);
- умения выбирать оптимальные алгоритмы и структуры данных для решения прикладных задач.

Практическое задание направлено на проверку умений:

- разрабатывать алгоритмы и программы на Python с использованием различных парадигм программирования;
- создавать и использовать функции с различными способами передачи аргументов, включая функции высшего порядка и замыкания;
- реализовывать алгоритмы поиска и сортировки, анализировать их эффективность на реальных данных;
- применять структуры данных (кортежи, множества, словари) для эффективной обработки и хранения информации;
- выполнять чтение, обработку и запись данных из текстовых и бинарных файлов;
- проводить анализ сложности реализованных алгоритмов и обосновывать выбор конкретных решений;
- интегрировать все изученные темы в едином программном решении для прикладной задачи.

6.2. Типовые оценочные материалы промежуточной аттестации.

Вопросы к экзамену:

1. Дайте определение парадигмы программирования. Назовите основные парадигмы, поддерживаемые в Python, и кратко охарактеризуйте каждую.
2. В чём заключаются ключевые отличия императивного программирования от декларативного? Приведите примеры задач, где каждый подход проявляет свои преимущества.
3. Перечислите и раскройте три основных принципа объектно-ориентированного программирования. Как эти принципы реализуются в Python?
4. Что такое «чистая функция» в контексте функционального программирования? Почему отсутствие побочных эффектов упрощает тестирование и отладку кода?
5. Какие существуют способы передачи аргументов в функции Python? Чем отличаются позиционные, именованные аргументы и аргументы со значениями по умолчанию?
6. Как обработать произвольное количество позиционных аргументов в функции? А произвольное количество именованных аргументов? Приведите примеры.
7. Что такое область видимости переменной? Объясните различие между локальными, глобальными и нелокальными переменными, приведите примеры.
8. Что такое рекурсия? Из каких обязательных компонентов состоит корректная рекурсивная функция? В чём преимущества и недостатки рекурсии перед итерацией?
9. Что такое анонимные функции в Python? Каковы их синтаксис и ограничения по сравнению с обычными функциями?
10. Что означает утверждение, что функции в Python являются объектами первого класса? Какие возможности это даёт программисту?
11. Объясните механизм замыкания в Python. Приведите пример функции, возвращающей замыкание, и объясните, какие переменные и откуда «запоминаются».
12. Опишите алгоритм линейного поиска. Какова его временная сложность в лучшем, среднем и худшем случаях? В каких ситуациях он предпочтительнее бинарного поиска?
13. Опишите алгоритм бинарного поиска. Какое ключевое требование предъявляется к данным? Какова его временная сложность и почему он эффективнее линейного на больших объёмах данных?
14. Опишите алгоритм сортировки пузырьком. В чём заключается принцип его работы? Какова временная сложность и какие оптимизации можно применить?
15. Опишите алгоритм сортировки вставками. Для каких типов входных данных этот алгоритм показывает наилучшую производительность?

16. В чём заключается идея быстрой сортировки? Какова роль опорного элемента, и почему выбор опорного элемента критически важен для производительности?

17. Опишите алгоритм сортировки слиянием. Какая стратегия лежит в его основе? Каковы его временная и пространственная сложность?

18. Что такое устойчивость алгоритма сортировки? Какие из изученных алгоритмов являются устойчивыми, а какие нет и почему?

19. Что такое временная сложность алгоритма, и для чего необходимо её анализировать? Чем анализ сложности отличается от простого измерения времени выполнения?

20. Что такое «О-нотация»? Почему при анализе сложности мы отбрасываем константы и младшие члены?

21. Перечислите основные классы сложности от самой эффективной к самой неэффективной. Для каждого класса приведите пример алгоритма.

22. Как оценивается сложность последовательных блоков кода, условных операторов и вложенных циклов? Сформулируйте общие правила.

23. Что такое пространственная сложность алгоритма? Приведите примеры алгоритмов с малой временной, но большой пространственной сложностью, и наоборот.

24. Объясните понятия «лучший случай», «средний случай» и «худший случай» на конкретном примере алгоритма. Почему часто рассматривают именно худший случай?

25. Как оценить временную сложность рекурсивного алгоритма? Что такое рекуррентное соотношение?

26. Что такое кортеж в Python? Чем он принципиально отличается от списка? Какие преимущества даёт неизменяемость кортежа?

27. В каких ситуациях использование кортежа предпочтительнее использования списка? Приведите не менее трёх примеров.

28. Что такое распаковка кортежа? Как с её помощью можно обменять значения двух переменных без временной переменной?

29. Почему кортеж может использоваться в качестве ключа словаря, а список — нет? Какое требование предъявляется к ключам словаря?

30. Что такое множество в Python? Какими свойствами оно обладает? Как создать пустое множество?

31. Перечислите основные операции над множествами: объединение, пересечение, разность, симметрическая разность. Для каждой операции приведите пример.

32. Как проверить, является ли одно множество подмножеством другого? Какие операторы и методы для этого существуют?

33. Какие практические задачи удобно решать с помощью множеств? Опишите, как удалить дубликаты из списка или найти общие элементы двух коллекций.

34. Что такое замороженное множество? Чем оно отличается от обычного множества и в каких ситуациях используется?

35. Что такое словарь в Python? Опишите его структуру «ключ-значение». Какие требования предъявляются к ключам?

36. Какими способами можно создать словарь? Перечислите не менее четырёх способов, приведите примеры.

37. Как выполняется доступ к значению по ключу? Чем отличается прямой доступ от использования метода получения значения с значением по умолчанию?

38. Что такое генератор словаря? Приведите пример создания словаря с его помощью.

39. Какие методы словаря позволяют получить все ключи, все значения и все пары в виде итерируемых объектов?

40. Объясните принцип работы словаря как хэш-таблицы. Почему операции доступа в среднем выполняются за O(1)?

41. Какие практические задачи удобно решать с помощью словарей? Приведите примеры.

42. Какие режимы открытия файла существуют в Python? Опишите назначение каждого режима.

43. Что такое контекстный менеджер при работе с файлами? В чём его преимущество перед ручным управлением?

44. Какие способы чтения данных из текстового файла существуют? В каких случаях каждый способ предпочтителен?

45. Что такое кодировка символов? Почему важно правильно указывать кодировку при открытии файла? Какие проблемы могут возникнуть?

46. Чем работа с бинарными файлами отличается от работы с текстовыми? Приведите примеры задач, где требуется работа с бинарными файлами.

47. Что такое сериализация? Как с помощью модуля pickle сохранить объект Python в файл и восстановить его обратно? Какие риски безопасности связаны с pickle?

48. Как управлять позицией чтения/записи в файле? Какие методы для этого существуют и для чего они могут потребоваться?

49. Какие форматы структурированных данных широко используются для обмена данными? Как в Python выполнить чтение и запись в этих форматах?

50. Какие исключения могут возникнуть при работе с файлами? Как корректно обрабатывать эти ситуации с помощью try-except?

Пример практического задания

Необходимо разработать программу на языке Python, которая выполняет полный цикл обработки данных из файла, их анализ с использованием различных структур данных и алгоритмов, а также сохраняет результаты отчёта.

Исходный файл students_grades.csv содержит данные об успеваемости студентов в следующем формате (первая строка — заголовок):

```
student_id,name,group,grade,subject
1,Иванов Иван,ИН-21,85,Математика
2,Петрова Мария,ИН-21,92,Математика
3,Сидоров Алексей,ИН-22,78,Математика
4,Иванова Анна,ИН-21,85,Программирование
5,Козлов Дмитрий,ИН-22,95,Программирование
6,Петрова Елена,ИН-21,88,Программирование
7,Сидоров Алексей,ИН-22,82,Физика
8,Иванов Иван,ИН-21,79,Физика
9,Козлов Дмитрий,ИН-22,91,Физика
10,Петрова Мария,ИН-21,87,Физика
```

Требования к реализации:

1. Реализуйте функцию load_data(filename), которая:
   - считывает файл students_grades.csv;
   - обрабатывает возможную ошибку отсутствия файла (если файл отсутствует, программа должна создать его с данными из примера выше);
   - возвращает список словарей, где каждый словарь соответствует одной строке (ключи: student_id, name, group, grade, subject). Оценку (grade) преобразовать в целое число.
2. Реализуйте функцию get_unique_students(data), которая с использованием множества возвращает:
   - множество уникальных идентификаторов студентов;
   - множество уникальных фамилий и имён студентов (строки полностью);
   - количество уникальных студентов.
3. Реализуйте функцию calculate_average_grade_by_subject(data), которая возвращает словарь, где ключ — название предмета, значение — средняя оценка по этому предмету (с округлением до двух знаков после запятой).
4. Реализуйте функцию get_top_student_by_subject(data), которая для каждого предмета определяет студента (или нескольких студентов при равной максимальной оценке), получившего наивысший балл. Результат должен быть словарём вида: предмет → список словарей с информацией о студенте (имя, группа, оценка).
5. Реализуйте функцию get_student_statistics(data), которая возвращает словарь, где ключ — идентификатор студента, значение — словарь с полями: name, group, average_grade (средняя оценка студента по всем предметам), subjects_count (количество сданных предметов), best_grade (максимальная оценка).

6. Реализуйте функцию linear_search_by_name(data, name), которая выполняет линейный поиск всех записей, соответствующих заданному имени (полному или частичному совпадению с учётом регистра). Функция должна возвращать список найденных записей и количество выполненных сравнений.

7. Реализуйте функцию sort_students_by_average_grade(student_statistics), которая сортирует студентов по убыванию средней оценки. Используйте собственноручно реализованный алгоритм сортировки вставками или быстрой сортировки (не встроенную функцию sorted). Функция должна возвращать отсортированный список словарей (с полями: name, group, average_grade).

8. Реализуйте функцию analyze_complexity(data), которая:

- измеряет время выполнения линейного поиска по имени (на примере любого существующего и несуществующего имени);
- оценивает теоретическую временную сложность использованных алгоритмов (линейный поиск, сортировка вставками/быстрая, операции со словарями и множествами);
- выводит на экран таблицу с оценками сложности для каждой операции.

9. Создайте функцию save_report(filename, statistics), которая сохраняет отчёт в текстовый файл report.txt в следующем формате:

--- ОТЧЁТ ПО УСПЕВАЕМОСТИ СТУДЕНТОВ ---

1. Средние оценки по предметам:
Математика: 85.00
Программирование: 89.33
Физика: 84.75

2. Лучшие студенты по предметам:
Математика: Петрова Мария (ИН-21) - 92
Программирование: Козлов Дмитрий (ИН-22) - 95
Физика: Козлов Дмитрий (ИН-22) - 91

3. Топ студентов по средней оценке:
1. Козлов Дмитрий (ИН-22) - 93.00
2. Петрова Мария (ИН-21) - 89.50
3. Петрова Елена (ИН-21) - 88.00
...

4. Анализ сложности алгоритмов:
Линейный поиск: O(n) - выполнено сравнений: X
Сортировка вставками: O(n^2) в худшем случае
Операции со словарём: O(1) в среднем

6.3. Критерии и шкала оценивания на основе БРС.

Соответствие государственной шкалы оценивания академической успеваемости и шкалы ECTS при экзамене

Оценка по шкале ECTS	**Сумма баллов за все виды учебной деятельности**	**Оценка по государственной шкале**	**Определение**
A	90 – 100	«Отлично»	отличное выполнение с незначительным количеством неточностей
B	80 – 89	«Хорошо»	в целом правильно выполненная работа с незначительным количеством ошибок (до 10%)
C	75 – 79		в целом правильно выполненная работа с незначительным количеством ошибок (до 15%)
D	70 – 74	«Удовлетворительно»	неплохо, но со значительным количеством недостатков
E	60 – 69		выполнение удовлетворяет минимальные критерии
FX	35 – 59	«Не удовлетворительно»	с возможностью повторной сдачи
F	0 – 34		с обязательным повторным изучением дисциплины (выставляется комиссией)

6.4. Описание дополнительных материалов и оборудования, необходимых для выполнения проверочных заданий

Компьютер с операционной системой RedOS или Windows с устойчивым Интернет-соединением для работы с ноутбуками Google Colab, программные продукты с открытой лицензией: PyCharm Community Edition, Visual Studio Code, Jupyter Notebook.

7. Методические материалы по освоению дисциплины

Получение углубленных знаний по изучаемой дисциплине достигается за счет дополнительных часов к аудиторной работе самостоятельной работы студентов. Выделяемые часы целесообразно использовать для знакомства с дополнительной научной литературой по проблематике дисциплины, анализа научных концепций и современных подходов к осмыслению рассматриваемых проблем. К самостоятельному виду работы студентов относится работа в библиотеках, в электронных поисковых системах и т.п. по сбору материалов, необходимых для проведения практических занятий или выполнения конкретных заданий преподавателя по изучаемым темам. Студенты могут установить диалог с преподавателем, получать консультации по выполнению заданий. В качестве оценочных средств на протяжении семестра используются практические задания.

Обучение по дисциплине «Программирование на Python» предполагает изучение курса на аудиторных занятиях (лекции, практические занятия) и самостоятельную работу студентов. Практические занятия дисциплины предполагают их проведение в различных формах с целью выявления полученных знаний, умений, навыков и компетенций с проведением контрольных мероприятий. С целью обеспечения успешного обучения студент должен готовиться к лекции, поскольку она является важнейшей формой организации учебного процесса, поскольку:

– знакомит с новым учебным материалом;
– разъясняет учебные элементы, трудные для понимания;
– систематизирует учебный материал;
– ориентирует в учебном процессе.

Работа обучающегося на лекции:

Слушание и запись лекций – сложный вид вузовской аудиторной работы. Внимательное слушание и конспектирование лекций предполагает интенсивную умственную деятельность обучающегося. Краткие записи лекций, их конспектирование помогает усвоить учебный материал. Конспект является полезным тогда, когда записано самое существенное, основное и сделано это самим обучающимся.

Подготовка к практическим занятиям:

Подготовку к каждому практическому занятию каждый обучающийся должен начать с ознакомления с планом, который отражает содержание предложенной темы. Тщательное продумывание и изучение вопросов плана основывается на проработке текущего материала лекции, а затем изучения обязательной и дополнительной литературы, рекомендованную к данной теме. Если программой дисциплины предусмотрено выполнение практического задания, то его необходимо выполнить с учетом предложенной инструкции. Все новые понятия по изучаемой теме необходимо внести в глоссарий, который целесообразно вести с самого начала изучения курса. Результат такой работы должен проявиться в способности обучающегося свободно ответить на теоретические вопросы практического занятия, его выступлении и участии в коллективном обсуждении вопросов изучаемой темы, правильном выполнении практических заданий и контрольных работ.

Структура практического занятия:

В зависимости от содержания и количества отведенного времени на изучение каждой темы может практическое занятие состоять из четырех-пяти частей:

1. Устный опрос.

2. Обсуждение теоретических вопросов, определенных программой дисциплины.

3. Выполнение практических заданий с последующим разбором полученных результатов или обсуждение практического задания, выполненного дома.

4. Подведение итогов занятия.

Работа с литературными источниками:

В процессе подготовки к практическим занятиям, обучающимся необходимо обратить особое внимание на самостоятельное изучение рекомендованной учебно-методической (а также научной и популярной) литературы. Самостоятельная работа с учебниками, учебными пособиями, научной, справочной и популярной литературой, материалами периодических изданий и Интернета, статистическими данными является наиболее эффективным методом получения знаний, позволяет значительно активизировать процесс овладения информацией, способствует более глубокому усвоению изучаемого материала, формирует у обучающихся свое отношение к конкретной проблеме. Более глубокому раскрытию вопросов способствует знакомство с дополнительной литературой, рекомендованной преподавателем, что позволяет обучающимся проявить свою индивидуальность в рамках выступления на занятиях, выявить широкий спектр мнений по изучаемой проблеме.

8. Учебная литература и ресурсы информационно-телекоммуникационной сети Интернет

8.1. Основная литература

1. PYTHON-программирование : учебное пособие / И. И. Баглаев, Т. Ж. Базаржапова, Н. В. Очирова, Н. В. Юможапова. — Улан-Удэ : БГУ, 2026. — 118 с. — ISBN 978-5-9793-2087-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/519059 (дата обращения: 13.05.2026). — Режим доступа: для авториз. пользователей.

2. Сергеева, О. А. Программирование на Python : учебно-методическое пособие / О. А. Сергеева. — Кемерово : КемГУ, 2024. — 157 с. — ISBN 978-5-8353-3123-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/420758 (дата обращения: 13.05.2026). — Режим доступа: для авториз. пользователей.

8.2. Дополнительная литература

1. Лысаков, К. Ф. Практическое программирование на Python : учебное пособие / К. Ф. Лысаков. — Новосибирск : НГУ, 2023. — 76 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/388274 (дата обращения: 13.05.2026). — Режим доступа: для авториз. пользователей.

2. Василекина, О. М. Учебно-методическое пособие по дисциплине «Алгоритмизация и программирование»: Структурное и процедурное программирование на языке Python направление подготовки 09.03.03 Прикладная информатика профиль «Прикладная информатика в экономике» : учебно-методическое пособие / О. М. Василекина. — Великие Луки : Великолукская ГСХА, 2024. — 104 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/426992 (дата обращения: 13.05.2026). — Режим доступа: для авториз. пользователей.

8.3.Нормативные правовые документы и иная правовая информация

1. Конституция Российской Федерации. – Текст : электронный // Сайт Президента Российской Федерации. – URL: http://www.kremlin.ru/acts/constitution

8.4 Интернет-ресурсы

1. Информационно-правовой портал ГАРАНТ.РУ. – URL: https://www.garant.ru/

2. Научная электронная библиотека eLIBRARY.RU. – URL: https://elibrary.ru/

3. Научная электронная библиотека «КиберЛенинка». – URL: https://cyberleninka.ru

4. Электронно-библиотечная система «Лань». – URL: http://e.lanbook.com

5. Документация по Python – URL: https:// https://docs.python.org/3/

9. Материально-техническая база, информационные технологии, программное обеспечение и информационные справочные системы

Материально-техническое обеспечение дисциплины включает в себя:

- лекционные аудитории, оборудованные видеопроекционным оборудованием для презентаций, средствами звуковоспроизведения, экраном;
- помещения для проведения практических занятий, оборудованные учебной мебелью.

Дисциплина поддержана соответствующими программными продуктами с открытой лицензией: PyCharm Community Edition, Visual Studio Code, Jupyter Notebook.

Вуз обеспечивает каждого обучающегося рабочим местом в компьютерном классе в соответствии с объемом изучаемых дисциплин, обеспечивает выход в сеть Интернет.

Помещения для самостоятельной работы обучающихся включают следующую оснащенность: столы аудиторные, стулья, доски аудиторные, компьютеры с подключением к локальной сети института (для компьютерных аудиторий) и Интернет. Для изучения учебной дисциплины используются автоматизированная библиотечная информационная система и электронные библиотечные системы.