

УТВЕРЖДАЮ
Проректор по учебной работе и
качеству образования

_____ И. А. Долгова

30 сентября 2026

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Б1.О.04 Язык программирования Python

Направление подготовки:	45.04.04 Интеллектуальные системы в гуманитарной среде
Профиль подготовки:	Цифровая лингвистика
Квалификация:	магистр
Форма обучения:	очная
Год начала подготовки:	2027

Рабочая программа составлена в соответствии с:

- федеральным государственным образовательным стандартом высшего образования – магистратура по направлению подготовки 45.04.04 Интеллектуальные системы в гуманитарной среде, утверждённым приказом Министерства науки и высшего образования Российской Федерации от 13 августа 2020 года № 1003.

Разработчики программы: Макаров А. А., доктор технических наук, профессор; Кириллов А. Г., кандидат филологических наук, доцент

Рабочая программа согласована с руководителем образовательной программы 45.04.04 Интеллектуальные системы в гуманитарной среде. Рабочая программа рассмотрена и одобрена на заседании кафедры информационных систем и компьютерных технологий 10 сентября 2026, протокол № 2.

1. ОБЩАЯ ХАРАКТЕРИСТИКА

1.1. Цель и задачи дисциплины

Цель изучения дисциплины — формирование у обучающихся системы фундаментальных знаний о принципах алгоритмизации и объектно-ориентированного программирования на языке Python, а также развитие практических навыков проектирования, отладки и документирования программного кода для автоматизации обработки текстовых данных.

Задачи изучения дисциплины:

- изучение синтаксиса, управляющих конструкций, базовых и структурированных типов данных языка программирования Python;
- освоение международных стандартов оформления программного кода (PEP 8) и методов документирования скриптов (docstrings, комментарии);
- формирование практических навыков разработки, тестирования и отладки алгоритмов обработки списков, текстовых строк, кортежей, словарей и множеств;
- развитие умений автоматизировать процессы управления файловой системой и потоковой обработки текстовых и структурированных файлов различных форматов (TXT, CSV, JSON) для решения прикладных лингвистических задач.

1.2. Требования подготовленности обучающегося к освоению содержания дисциплины

Освоение дисциплины «Язык программирования Python» во 2 семестре опирается на математическую культуру, логику и первичные навыки структурирования информации, полученные магистрантами в 1 семестре в рамках курсов «Математика в гуманитарных науках» и «Научная деятельность в лингвистике».

Обучающийся перед началом освоения дисциплины должен:

Знать:

- базовые понятия математической логики, булевы операции (И, ИЛИ, НЕ) и таблицы истинности;
- основы теории множеств и базовые принципы построения алгоритмических блок-схем;
- системно-структурную организацию уровней языка (морфология, синтаксис).

Уметь:

- логически структурировать и классифицировать разнородную гуманитарную информацию;
- использовать электронные таблицы (Excel / Google Таблицы) для проведения простейших расчетных и сортировочных операций.

Владеть:

- навыками уверенного пользователя персонального компьютера в современных операционных системах;
- базовым понятийным аппаратом формального описания текстовых данных.

1.3. Планируемые результаты обучения по дисциплине

Код и наименование компетенции	Код и наименование индикатора достижения компетенции	Результаты обучения по дисциплине
ОПК-6. Способен осваивать, применять и разрабатывать документацию к программным системам в области программирования и информационных систем	ОПК-6.1. Осваивает техническую документацию, стандарты кодирования и требования к архитектуре современных информационных и программных систем	ОПК-6.1.1. Знает базовый синтаксис языка Python, типы данных, управляющие конструкции и международный стандарт оформления кода PEP 8
	ОПК-6.2. Применяет языки программирования и инструменты веб-технологий для разработки элементов информационных систем в профессиональной сфере	ОПК-6.2.1. Умеет писать, отлаживать и тестировать программный код на Python для автоматизации простейших операций с текстовыми файлами
	ОПК-6.3. Разрабатывает сопутствующую техническую документацию, спецификации, руководства пользователя и комментарии к программному коду	ОПК-6.3.1. Владеет навыками документирования разработанного кода (написание docstrings, комментариев) и создания базовых файлов описания проекта (README)

2. ОБЪЁМ, СТРУКТУРА И СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

2.1. Объём дисциплины

Семестр	Контактная работа			СР	Форма ПА	Итоговый объём, часов/з. е.
	Л	П	ПА			
2		72	9	279	зачёт/оценка	360/10
Итого:		72	9	279		360/10

Л — лекции, П — все виды занятий семинарского типа, ПА — промежуточная аттестация, СР — самостоятельная работа обучающегося.

2.2. Структура дисциплины

Наименование тематического раздела дисциплины	Количество часов		
	Л	П	СР
Введение в программирование на Python и базовые управляющие конструкции		6	24
Списки и матрицы: структуры и базовые алгоритмы обработки		8	32
Функции и встроенные методы работы со списками		6	24
Текстовые строки и основы регулярных выражений		12	46
Кортежи и словари как модели представления данных		8	30
Функциональный стиль и проектирование пользовательских функций		8	30
Управление файловой системой и рекурсивный обход директорий		6	24
Потоковая обработка текстовых файлов и структурированных форматов данных		12	46
Множества и операции над ними		6	23
Всего:	—	72	279

2.3. Содержание тематических разделов дисциплины

Раздел 1. Введение в программирование на Python и базовые управляющие конструкции
Общие сведения о языке Python, его история и место в компьютерной лингвистике. Подготовка и настройка легковесных сред разработки (VS Code, Jupyter Notebook). Запуск программ в терминале, интерактивный режим и пользовательский интерфейс IDLE. Структура программы, правила размещения операторов. Международный стандарт оформления кода PEP 8. Переменные, динамическая типизация, встроенные функции и базовые арифметические операции. Логические операторы и ветвление алгоритмов. Особенности реализации циклов (while и for), функция range(). Операторы управления циклами, множественный выбор и функция enumerate().
Раздел 2. Списки и матрицы: структуры и базовые алгоритмы обработки

Общие сведения о списках, их фундаментальные свойства и отличие от массивов в других языках. Базовые алгоритмы обработки списков: инициализация (в том числе генерация случайных значений), индексация и использование отрицательных индексов. Понятие матрицы как двумерного списка, алгоритмы их обхода и инициализации. Понятие и синтаксис срезов (slices), примеры практического применения. Исключения при работе со структурами данных, перехват и обработка ошибок с помощью конструкции `try-except-else-finally`.

Раздел 3. Функции и встроенные методы работы со списками

Специфика выполнения операции «+» и сравнения списков. Встроенные методы модификации списков: добавление элементов (`append`), расширение (`extend`), методы `count()`, `insert()`, `remove()`. Извлечение и очистка данных: `pop()`, `clear()`, `copy()`. Использование ключевого слова `del` и его отличие от метода `pop()`. Поиск индекса элемента по его значению. Практикум по реализации комплексных программ на использование списков и матриц.

Раздел 4. Текстовые строки и основы регулярных выражений

Формирование текстовых переменных и специфика работы с многострочными переменными в Python. Базовые строковые операции: конкатенация, дублирование. Форматирование строк: f-строки и r-строки. Соотношение структуры строки и списка символов: сходства и различия. Специфика работы со структурой «список строк». Встроенные функции и методы обработки текста: выделение подстроки, поиск вхождения, замена, удаление, очистка (`strip`), изменение регистра, разбиение строк (`split`) и слияние (`join`). Введение в регулярные выражения (`re`): синтаксис, метасимволы и примеры извлечения текстовых паттернов.

Раздел 5. Кортежи и словари как модели представления данных

Понятие кортежа (`tuple`) как неизменяемого списка. Сравнение кортежа со списком, доступ к элементам, преобразование типов. Понятие о словарях (`dict`) как ассоциативных массивах. Создание словаря, добавление, обновление и удаление пар ключ-значение. Базовые операции: перебор элементов, копирование, безопасное получение значений по ключу. Методы `keys()`, `values()`, `items()`. Работа с несуществующими ключами (`setdefault`). Примеры лингвистической обработки данных на основе списков словарей (парсинг метаданных).

Раздел 6. Функциональный стиль и проектирование пользовательских функций

Создание пользовательских функций и их назначение. Возврат множественных значений (функции, возвращающие кортежи и списки). Документирование функций: создание docstrings. Изменяемые и неизменяемые типы данных, особенности их передачи в качестве параметров. Позиционные и именованные аргументы, функции с произвольным количеством аргументов (`*args`, `**kwargs`). Элементы функционального программирования в Python. Встроенные функции высшего порядка: `lambda`, `map`, `filter`, `reduce`, оператор `yield`. Области видимости переменных, вложенные функции, понятие о замыканиях и декораторах.

Раздел 7. Управление файловой системой и рекурсивный обход директорий

Работа с операционной системой посредством встроенного модуля `os`. Получение текущей рабочей директории, смена папок, создание, переименование, перемещение и удаление файлов. Использование относительных и абсолютных путей. Алгоритмы рекурсивного обхода вложенных папок и каталогов (`os.walk`) для пакетной обработки текстовых файлов. Получение метаданных о файлах системы.

Раздел 8. Потокковая обработка текстовых файлов и структурированных форматов данных

Различия между текстовыми и бинарными файлами. Конструкция безопасного управления ресурсами: оператор `with` (менеджеры контекста). Способы считывания информации: методы `read()`, `readline()`, `readlines()`. Запись данных в файл. Режимы чтения, записи и добавления (`r`, `w`, `a`). Создание пользовательских модулей и пакетов программ. Практикум по

чтению и записи структурированных данных распространенных форматов: CSV, JSON, а также основы импорта/экспорта данных в форматы XML и Excel с использованием открытых библиотек.

Раздел 9. Множества и операции над ними

Особенности и свойства типа данных «множество» (`set`). Создание множеств и преобразование типов (список во множество и обратно). Базовые операции: перебор элементов, добавление, удаление, использование генераторов множеств (`set comprehensions`). Теоретико-множественные операции в Python: пересечение, объединение, вычитание, симметричная разность, проверка подмножеств. Практикум по применению множеств для автоматического удаления дубликатов слов и подсчета лексического разнообразия в текстовом файле.

2.4. Организация учебных занятий

Дисциплина может реализовываться с применением электронного обучения и дистанционных образовательных технологий.

3. ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

3.1. Учебная литература

3.1.1. Основная

1. Калитвин, В. А. Введение в программирование на Python : учебное пособие : [16+] / В. А. Калитвин ; Липецкий государственный педагогический университет им. П. П. Семенова-Тян-Шанского. – Липецк : Липецкий государственный педагогический университет им. П.П. Семенова-Тян-Шанского, 2023. – 85 с. : ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=714538>.
2. Карякин, М. И. Технологии программирования и компьютерный практикум на языке Python : учебное пособие : [16+] / М. И. Карякин, К. А. Ватульян, Р. М. Мнухин ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2022. – 244 с. : ил., табл. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=698687>.

3.1.2. Дополнительная

1. Воробьев, Г. А. Основы программирования на Python : учебно-методическое пособие : [16+] / Г. А. Воробьев ; Липецкий государственный педагогический университет им. П. П. Семенова-Тян-Шанского. – Липецк : Липецкий государственный педагогический университет им. П.П. Семенова-Тян-Шанского, 2022. – 89 с. : ил., табл. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=700515>.
2. Северенс, Ч. Введение в программирование на Python : учебное пособие / Ч. Северенс. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 231 с. : схем., ил. – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=429184>.
3. Сузи, Р. А. Язык программирования Python : учебное пособие : [16+] / Р. А. Сузи. – 2-е изд., испр. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ) : Бином. Лаборатория знаний, 2007. – 327 с. – (Основы информационных технологий). – Режим доступа: по подписке. – URL: <https://biblioclub.ru/index.php?page=book&id=233288>.

3.2. Профессиональные базы данных и информационные справочные системы

Наименование ресурса	Адрес	Тип доступа
Электронно-библиотечные системы		
ЭБС «Университетская библиотека онлайн»	https://biblioclub.ru/	Индивидуальный неограниченный доступ после регистрации
Профессиональные базы данных и информационные справочные системы		
Документация по языку программирования Python	https://pydocs.ru/home/	Открытый ресурс
Python 3 Documentation	https://docs.python.org/3/	
Стиль оформления кода PEP 8 (Официальное руководство)	https://peps.python.org/pep-0008/	

3.3. Сетевые ресурсы

Наименование ресурса	Адрес
Основы Python	https://education.yandex.ru/handbook/python
A Byte of Python	https://python.swaroopch.com/
A Byte of Python (Russian), Версия 2.01	https://myrobot.ru/downloads/books/A_Byte_of_Python_Rus_2.01.pdf
Python	https://www.python.org/
The Python Tutorial	https://python.land/python-tutorial
PyPI (Python Package Index)	https://pypi.org
Интерактивные учебные модули Jupyter	https://jupyter.org
GitHub (раздел открытого кода)	https://github.com

3.4. Методическое обеспечение дисциплины

3.4.1. Методические указания для преподавателя

Изучение дисциплины проводится в форме лекций, практических занятий, организации самостоятельной работы студентов, консультаций.

Лекции реализуются через изложение учебного материала с возможным мультимедийным сопровождением. Основными целями лекции являются системное освещение ключевых понятий и положений по соответствующей теме, обзор и оценка существующей проблематики, её методологических и социокультурных оснований, возможных вариантов решения, предложение методических рекомендаций для дальнейшего изучения курса, в том числе литературы и источников. Лектор должен стимулировать обучающихся к участию в обсуждении вопросов лекционного занятия, к высказыванию собственной точки зрения по обсуждаемой проблеме. Главное назначение лекции — обеспечить теоретическую основу обучения, развить интерес к учебной деятельности и конкретной учебной дисциплине, сформировать у обучающихся ориентиры для самостоятельной работы над курсом.

Практические занятия направлены на развитие самостоятельности обучающихся в исследовании изучаемых вопросов и приобретение умений и навыков. Практические занятия традиционно проводятся в форме обсуждения проблемных вопросов в группе при активном участии обучающихся. Они способствуют углубленному изучению наиболее фундаментальных и сложных проблем курса, служат важной формой анализа и синтеза исследуемого материала, а также подведения итогов самостоятельной работы обучающихся, стимулируя развитие профессиональной компетентности, навыков и умений. На практических занятиях обучающиеся учатся работать с научной литературой, чётко и доходчиво излагать проблемы и предлагать варианты их решения, аргументировать свою позицию, оценивать и критиковать позиции других, свободно публично высказывать свои мысли и суждения, грамотно вести полемику и представлять результаты собственных исследований. Практические занятия проводятся в форме устных и письменных опросов, диспута, тестирования, выполнения заданий, обсуждения докладов, выполнения контрольных заданий и пр.

Результаты работы на практических занятиях учитываются преподавателем при выставлении итоговой оценки по данной дисциплине. На усмотрение преподавателя обучающиеся, активно отвечающие на занятиях и выполняющие рекомендации преподавателя при подготовке к ним, могут получить повышающий балл к своей оценке в рамках промежуточной аттестации.

3.4.2. Методические рекомендации по самостоятельной работе обучающихся

Самостоятельная работа обучающихся выполняется по заданию и при методическом руководстве преподавателя, но без его непосредственного участия. Самостоятельная работа подразделяется на самостоятельную работу на аудиторных занятиях и на внеаудиторную самостоятельную работу. Самостоятельная работа обучающихся включает как полностью самостоятельное освоение отдельных тем (разделов) дисциплины, так и проработку тем (разделов), осваиваемых во время аудиторной работы. Во время самостоятельной работы обучающиеся читают и конспектируют учебную, научную и справочную литературу, выполняют задания, направленные на закрепление знаний и отработку умений и навыков, готовятся к текущему и промежуточному контролю по дисциплине.

Самостоятельная работа включается в общую трудоёмкость учебной нагрузки обучающегося.

Реализация поставленной цели предполагает решение следующих задач:

- освоение и расширение теоретических знаний по изучаемой дисциплине;
- систематизация и закрепление полученных теоретических знаний и практических навыков;
- формирование умений по поиску и использованию нормативной, справочной и специальной литературы, а также других источников информации;
- развитие познавательных способностей и активности, творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самообразованию, самосовершенствованию и самореализации;
- развитие научно-исследовательских навыков;
- формирование умения решать практические задачи (в профессиональной деятельности), используя приобретённые знания, способности и навыки.

Самостоятельная работа должна быть выполнена индивидуально или являться частью коллективной работы (в случае выполнения группового задания в работе делается соответствующая оговорка).

3.5. Материально-техническое обеспечение дисциплины

3.5.1. Аудитории для проведения занятий

Специальные помещения представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского типа, выполнения курсовых работ, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования.

Специальные помещения укомплектованы учебной мебелью, в том числе мебелью для преподавателя дисциплины, учебной доской.

Обучающиеся из числа лиц с ограниченными возможностями здоровья обеспечиваются электронными образовательными ресурсами в формах, адаптированных к ограничениям их здоровья.

Материально-техническое оснащение учебных аудиторий конкретизировано на официальном сайте Университета в подразделе «Материально-техническое обеспечение и оснащённость образовательного процесса. Доступная среда» раздела «Сведения об образовательной организации».

3.5.2. Оборудование и технические средства обучения

Специальные помещения укомплектованы демонстрационным оборудованием (мультимедийный проектор, экран, компьютер, звуковые колонки).

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети Интернет и обеспечением доступа в электронную информационно-образовательную среду Университета.

Электронно-библиотечная система (электронная библиотека) и электронная информационно-образовательная среда обеспечивают одновременный доступ не менее 25 процентов обучающихся.

3.5.3. Программное обеспечение

Наименование	Сведения о лицензии
Geany	GNU General Public Licence, свободно распространяемое с открытым исходным кодом
IDLE	Python Software Foundation License Version 2, свободно распространяемое с открытым исходным кодом
Moodle, среда дистанционного обучения	GNU General Public Licence, свободно распространяемое с открытым исходным кодом
PyCharm, Community Edition	Apache License 2.0, свободно распространяемое с открытым исходным кодом
Python (включая дистрибутив Anaconda с Jupyter Notebook)	Свободная (Python Software Foundation License — PSFL)
Visual Studio Code (VS Code)	Свободная (лицензия MIT / Microsoft Software License)
SQLite (встроенное расширение sqlite3)	Общественное достояние (Public Domain)

4. ОЦЕНОЧНЫЕ СРЕДСТВА

4.1. Методика проведения текущего контроля успеваемости и промежуточной аттестации

Оценивание уровня учебных достижений обучающихся по дисциплине осуществляется в виде текущего контроля успеваемости и промежуточной аттестации. Фонд оценочных средств по дисциплине включает оценочные материалы, направленные на проверку освоения компетенций, в том числе знаний, умений и навыков. Фонд оценочных средств включает оценочные средства текущего контроля и оценочные средства промежуточной аттестации.

4.1.1. Балльно-рейтинговая карта дисциплины

Виды работы	Виды контроля, критерии оценки и количество баллов
Текущий контроль	
Аудиторная работа (0-20 баллов)	Посещение занятий и участие в работе: 10 баллов — посещение не менее 50% занятий 15 баллов — посещение 60-80% занятий, участие в обсуждениях материала 20 баллов — посещение 90-100% занятий, активное участие в обсуждениях материала
Самостоятельная работа (0-40 баллов)	Выполнение и оформление доклада в соответствии с индивидуальной темой. Критерии оценивания: Степень раскрытия сущности проблемы — 15 баллов; Обоснованность выбора источников — 10 баллов; Наличие презентации/ демонстрационного материала — 10 баллов; Изложение — 5 баллов
Промежуточная аттестация	
Контрольное задание (0-40 баллов)	Выполнение письменного контрольного задания в СДО Moodle

4.1.2. Шкала перевода рейтинговых баллов в пятибалльную систему оценивания

Общее количество баллов	Оценка сформированности компетенций	Оценка результатов обучения по дисциплине	Оценка ECTS
0–34	Компетенции не сформированы.	неудовлетворительно (не зачтено)	F
Теоретическое содержание не освоено, практические навыки не сформированы, большинство учебных заданий не выполнено, качество их выполнения минимальное, все задания содержат грубые ошибки. Обучающийся не готов решать типовые профессиональные задачи.			
35–49	Уровень владения компетенциями недостаточный для их формирования в результате обучения по дисциплине.	неудовлетворительно (не зачтено)	FX

Общее количество баллов	Оценка сформированности компетенций	Оценка результатов обучения по дисциплине	Оценка ECTS
Теоретическое содержание освоено частично, практические навыки не сформированы, большинство учебных заданий не выполнено, качество их выполнения минимальное, большинство заданий выполнено с ошибками. Обучающийся не готов решать типовые профессиональные задачи.			
50–59	Уровень владения компетенциями посредственный для их формирования в результате обучения по дисциплине.	удовлетворительно (зачтено)	E
Теоретическое содержание освоено частично, практические навыки сформированы фрагментарно, многие учебные задания не выполнены, качество их выполнения минимальное, многие задания выполнены с ошибками. Обучающийся готов решать типовые профессиональные задачи.			
60–69	Уровень владения компетенциями удовлетворительный для их формирования в результате обучения по дисциплине.	удовлетворительно (зачтено)	D
Теоретическое содержание освоено частично, пробелы не носят существенного характера, практические навыки в основном сформированы, большинство учебных заданий выполнено, качество их выполнения удовлетворительное, некоторые задания выполнены с ошибками. Обучающийся готов решать типовые профессиональные задачи.			
70–89	Уровень владения компетенциями преимущественно высокий для их формирования в результате обучения по дисциплине.	хорошо (зачтено)	C
Теоретическое содержание освоено полностью, некоторые практические навыки сформированы недостаточно, все учебные задания выполнены, качество их выполнения высокое, некоторые задания выполнены с ошибками. Обучающийся готов решать типовые и ситуативные профессиональные задачи.			
90–94	Уровень владения компетенциями высокий для их формирования в результате обучения по дисциплине.	отлично (зачтено)	B
Теоретическое содержание освоено полностью, практические навыки сформированы, все учебные задания выполнены, качество их выполнения близко к максимальному, однако есть несколько незначительных ошибок. Обучающийся готов эффективно решать типовые и ситуативные профессиональные задачи, в том числе повышенного уровня сложности.			
95–100	Уровень владения компетенциями превосходный для их формирования в результате обучения по дисциплине.	отлично (зачтено)	A
Теоретическое содержание освоено полностью, практические навыки сформированы, все учебные задания выполнены, качество их выполнения близко к максимальному. Обучающийся готов эффективно решать типовые и ситуативные профессиональные задачи, в том числе повышенного уровня сложности, и способен разрабатывать новые решения.			

4.2. Оценочные средства текущего контроля

4.2.1. Вопросы для подготовки к практическим занятиям

Раздел 1. Введение в программирование на Python и базовые управляющие конструкции

1. Какова история создания языка Python и почему он занял доминирующие позиции в сфере компьютерной лингвистики и анализа текстовых данных (NLP)?
2. Проведите сравнительный анализ сред разработки: в каких случаях для лингвистического исследования эффективнее использовать VS Code, а в каких — интерактивные блокноты Jupyter Notebook?
3. Что такое интерактивный режим (REPL) и IDLE? Как запустить скрипт обработки текста через системный терминал?
4. Опишите базовые правила оформления кода согласно международному стандарту PEP 8 (отступы, именование переменных, длина строк, пробелы).
5. Что означает концепция динамической типизации в Python? Как работают встроенные функции определения типа (`type()`) и приведения типов (`str()`, `int()`, `float()`)?
6. Дайте характеристику базовым арифметическим операциям. В чём лингвистический смысл применения операций деления нацело (`//`) и взятия остатка от деления (`%`) при постраничном выводе текста?
7. Как устроены логические операторы (`and`, `or`, `not`) и конструкция ветвления алгоритмов `if-elif-else`?
8. Опишите синтаксис и особенности применения циклов `while` и `for`. Как функция `range()` оптимизирует итерацию по текстовым индексам?
9. Каковы функции операторов управления циклами `break` и `continue`? Как избежать логической ошибки заикливания программ?
10. Что такое множественный выбор и как встроенная функция `enumerate()` позволяет одновременно получать индекс токена и сам токен при переборе текста?
11. Практическое задание: Напишите скрипт, который принимает от пользователя предложение, перебирает его посимвольно в цикле с использованием `enumerate()` и выводит на экран только гласные буквы вместе с их порядковыми индексами.
12. Практическое задание: Реализуйте алгоритм, имитирующий простейший лингвистический фильтр: программа в цикле запрашивает у пользователя слова. Если слово входит в условный «стоп-список», цикл прерывается с помощью `break`, иначе слово выводится на экран.

Раздел 2. Списки и матрицы: структуры и базовые алгоритмы обработки

1. Что такое список (`list`) в Python, каковы его фундаментальные свойства (изменяемость, упорядоченность) и в чём его отличие от классических массивов в других языках программирования?
2. Опишите способы инициализации списков. Как создать пустой список для последующего наполнения его словами из файла?
3. Как устроена индексация элементов списка? В чём удобство и специфика использования отрицательных индексов при поиске знаков препинания в конце предложения?
4. Раскройте понятие матрицы как двумерного списка (`list of lists`). Как двумерная структура используется для хранения лингвистических таблиц (например, списка предложений, где каждое предложение — список слов)?
5. Опишите алгоритмы обхода двумерных списков с помощью вложенных циклов `for`. Как извлечь конкретную строку или столбец из матрицы?
6. Что такое срезы (`slices`) списков? Объясните синтаксис `list[start:stop:step]`. Как с помощью среза получить первые 5 слов текста, последние 3 слова или развернуть список слов в обратном порядке?
7. С какими исключениями и ошибками чаще всего сталкивается программист при работе со списками (например, `IndexError`)?

8. Опишите синтаксис и логику работы конструкции `try-except-else-finally` для перехвата и безопасной обработки ошибок времени выполнения.
9. Зачем нужен блок `else` в конструкции `try-except`, и в каких случаях код, помещенный в блок `finally`, выполнится обязательно?
10. Каким образом генерация случайных значений (модуль `random`) может применяться для создания тестовых выборок лингвистических данных?
11. Практическое задание: Создайте двумерный список (матрицу), представляющий собой разметку 3 предложений по 4 слова в каждом (каждый элемент — кортеж (слово, часть_речи)). Напишите вложенный цикл, который выводит на экран только слова, являющиеся существительными.
12. Практическое задание: Напишите программу, которая запрашивает у пользователя индекс элемента списка слов. Оберните обращение к списку в блок `try-except`, чтобы корректно перехватывать ошибку `IndexError` и выводить понятное гуманитариям предупреждение вместо аварийного завершения скрипта.

Раздел 3. Функции и встроенные методы работы со списками

1. Какова специфика выполнения операции конкатенации (+) и операции дублирования (*) для списков? Чем операция `list1 + list2` отличается от метода `extend()`?
2. Опишите механизм сравнения списков в Python. На основе каких критериев интерпретатор определяет равенство или неравенство двух структур данных?
3. Подробно охарактеризуйте встроенные методы модификации списков: `append()`, `insert()`, `remove()`. Каковы их параметры и вычислительная сложность?
4. Как работает метод `count()`? Как с его помощью быстро подсчитать абсолютную частоту конкретной словоформы в списке токенов?
5. Опишите методы извлечения и очистки данных: `pop()`, `clear()`. В чём разница между удалением элемента по индексу через `pop()` и удалением по значению через `remove()`?
6. Как работает метод `copy()`? Что такое «поверхностное» и «глубокое» копирование списков и почему простое присваивание `list2 = list1` не создает копию данных?
7. В чём заключается разница между использованием ключевого слова `del` и вызовом метода списков `pop()`?
8. Как работает метод `index()`? Как безопасно найти индекс первого вхождения слова в тексте и избежать ошибки `ValueError`, если слова в списке нет?
9. Каковы особенности работы методов сортировки списков `sort()` и встроенной функции `sorted()`? Как отсортировать список слов по алфавиту в обратном порядке или по длине слов (параметр `key=len`)?
10. Как функции `min()`, `max()` и `sum()` работают со списками? Что вернет функция `max()`, если ей передать список строк на английском языке?
11. Практическое задание: Напишите программу, которая принимает список токенов предложения, находит индекс слова "not" (если оно есть), удаляет его с помощью `pop()`, а в конец списка добавляет маркер отрицания "[NEG]".
12. Практическое задание: Создайте список слов, содержащий дубликаты. Напишите алгоритм, который итерируется по списку и с помощью методов `count()` и `remove()` оставляет только уникальные вхождения слов, сохраняя их исходный порядок.

Раздел 4. Текстовые строки и основы регулярных выражений

1. Как формируются текстовые переменные (`str`) в Python? В чём специфика работы с многострочными переменными, оформленными тройными кавычками (`"""`)?
2. Опишите базовые строковые операции: конкатенация (+) и дублирование (*). Каковы ограничения совместного использования строк и чисел?

3. Что такое форматирование строк? Объясните синтаксис и преимущества современных f-строк перед старыми методами форматирования. Зачем нужны r-строки (raw strings) в лингвистических скриптах?
4. Каково соотношение структуры строки и списка символов? В чём сходства (индексация, срезы, итерация) и фундаментальные различия (строка — неизменяемый тип данных)?
5. Опишите встроенные методы очистки и модификации регистра текста: `strip()`, `lstrip()`, `rstrip()`, `lower()`, `upper()`, `title()`, `capitalize()`. Почему они возвращают новую строку, а не меняют старую?
6. Как работают методы поиска и замены подстрок: `find()`, `rfind()`, `replace()`, `count()`, `startswith()`, `endswith()`? Как с их помощью проверить, является ли слово формой на -ing или содержит ли префикс un-?
7. Раскройте синтаксис и колоссальное лингвистическое значение методов разбиения строк `split()` и слияния списков строк `join()`. Как преобразовать текст в список слов и обратно?
8. Что такое регулярные выражения (модуль `re`)? Сформулируйте назначение базовых метасимволов: `.`, `^`, `$`, `*`, `+`, `?`, `\d`, `\w`, `\s`, `[]`.
9. В чём разница между функциями модуля `re`: `re.search()`, `re.match()`, `re.findall()`, `re.sub()`, `re.split()`? Какая из них оптимальна для извлечения всех совпадений из текста?
10. Как устроены квантификаторы и группы захвата `()` в регулярных выражениях? Как извлечь фрагменты текста по шаблону?
11. Практическое задание: Напишите скрипт на Python, который берет сырую англоязычную строку текста, приводит её к нижнему регистру, удаляет пробелы по краям, заменяет все знаки препинания на пустые места, а затем с помощью `split()` превращает в чистый список токенов.
12. Практическое задание: Используя модуль `re` и функцию `re.findall()`, напишите регулярное выражение для автоматического извлечения из текста всех дат, записанных в формате ДД-ММ-ГГГГ или ГГГГ/ММ/ДД.

Раздел 5. Кортежи и словари как модели представления данных

1. Дайте определение кортежу (`tuple`) как неизменяемому списку в Python. Каковы его синтаксические особенности (круглые кавычки, запятая для синглтона)?
2. Проведите подробное сравнение кортежа со списком. Почему кортежи работают быстрее и в каких лингвистических задачах их неизменяемость выступает гарантией целостности данных (например, хранение пар (слово, теги))?
3. Опишите операции доступа к элементам кортежа, распаковку кортежей (`tuple unpacking`) и особенности взаимного преобразования типов `list()` и `tuple()`.
4. Что такое словарь (`dict`) в Python? Почему словари называют ассоциативными массивами или хэш-таблицами? Каковы требования к ключам словаря (хэшируемость, неизменяемость)?
5. Опишите синтаксис создания словаря, добавления, обновления и удаления пар «ключ-значение». Что произойдет, если присвоить значение по уже существующему ключу?
6. Какие встроенные методы используются для безопасного получения значений из словаря? Чем метод `get()` отличается от прямого обращения по ключу через квадратные скобки `dict[key]`?
7. Подробно опишите функции методов `keys()`, `values()`, `items()`. В каком виде они возвращают данные и как их итерировать в цикле `for`?
8. Как работает метод `setdefault()` и встроенный класс `defaultdict` из модуля `collections`? Как они автоматизируют задачу добавления новых слов в частотный словарь?

9. Что такое «список словарей» и как эта структура данных используется в компьютерной лингвистике для представления размеченных корпусов (где каждый элемент-слово — это словарь с ключами 'token', 'lemma', 'pos', 'index')?
10. Как работают операторы `in` и `not in` при проверке наличия элементов в словарях и кортежах? По какому компоненту словаря (ключу или значению) осуществляется проверка?
11. Практическое задание: Напишите программу, которая берет список слов, строит частотный словарь (ключ — слово, значение — абсолютная частота его встречаемости в тексте) с использованием метода `get()` для обработки новых ключей.
12. Практическое задание: Создайте словарь, где ключами являются английские глаголы в инфинитиве, а значениями — списки их трех форм. Напишите цикл, который перебирает словарь через `items()` и выводит только те глаголы, у которых вторая и третья формы совпадают.

Раздел 6. Функциональный стиль и проектирование пользовательских функций

1. Каковы цели, синтаксис и правила проектирования пользовательских функций (`def`) в Python? Что такое сигнатура функции и её тело?
2. Как реализовать возврат множественных значений из функции? Объясните механизм автоматической упаковки возвращаемых элементов в кортеж (`return a, b`).
3. Что такое строки документирования (`docstrings`)? Каковы стандарты их написания и как встроенная функция `help()` использует докстринги для генерации документации по коду?
4. Раскройте особенности передачи изменяемых (`list`, `dict`) и неизменяемых (`str`, `tuple`) типов данных в качестве параметров функции. Почему опасно использовать изменяемые типы в качестве дефолтных аргументов?
5. В чём разница между позиционными и именованными (ключевыми) аргументами функции? Каковы правила их совместного размещения в вызове функции?
6. Как проектируются функции с произвольным количеством аргументов? Опишите назначение и синтаксис операторов синтаксического разбора `*args` и `**kwargs`.
7. Раскройте суть функционального стиля программирования в Python. Что такое анонимные функции `lambda` и в каких случаях их применение лаконичнее создания полноценных функций?
8. Опишите принципы работы встроенных функций высшего порядка: `map()` и `filter()`. Как с помощью `filter()` и `lambda`-функции очистить список слов от пустых строк?
9. Каковы особенности функционирования функции `reduce()` из модуля `functools`? Приведите пример математического или логического свертывания данных.
10. Что такое генераторы и ключевое слово `yield`? В чём преимущество генераторов перед функциями, возвращающими готовые списки, при потоковой обработке сверхбольших текстовых файлов (экономия памяти)?
11. Опишите области видимости переменных в Python (локальная, глобальная, встроенная — правило LEGB). Что такое замыкание функции (`closure`) и как устроены декораторы кода?
12. Практическое задание: Напишите пользовательскую функцию `clean_text()`, которая принимает строку текста и необязательный именованный аргумент `stop_words` (по умолчанию пустой список). Функция должна возвращать список токенов, очищенный от знаков препинания и стоп-слов. Напишите для функции подробный `docstring`.

Раздел 7. Управление файловой системой и рекурсивный обход директорий

1. Каковы назначение и возможности встроенного системного модуля `os` при работе с файловой структурой операционной системы?

2. Опишите методы получения текущей рабочей директории (`os.getcwd()`) и смены папок (`os.chdir()`). Почему лингвистическому скрипту важно знать, из какого каталога он запущен?
3. Как с помощью модуля `os` выполнять базовые дисковые операции: создание папок (`os.mkdir()`, `os.makedirs()`), переименование (`os.rename()`) и удаление файлов/каталогов (`os.remove()`, `os.rmdir()`)?
4. Проведите строгое концептуальное разграничение между абсолютными и относительными путями к файлам. Почему в коде программ предпочтительнее использовать относительные пути?
5. Каким образом функции подмодуля `os.path` (например, `os.path.join()`, `os.path.exists()`, `os.path.split()`) обеспечивают кроссплатформенность кода при работе с путями в Windows, macOS и Linux?
6. Раскройте синтаксис и логику работы алгоритма рекурсивного обхода вложенных папок и каталогов с помощью функции `os.walk()`. Какие три элемента она возвращает на каждом шаге итерации?
7. Как генератор `os.walk()` используется в корпусной лингвистике для пакетной обработки сотен текстовых файлов, распределенных по тематическим подпапкам языкового корпуса?
8. Как получить метаданные о файлах системы (размер файла, дата создания и модификации) с помощью функции `os.stat()`? Как отфильтровать пустые текстовые файлы в каталоге?
9. Охарактеризуйте возможности современного встроенного модуля `pathlib` как объектно-ориентированной альтернативы модулю `os`. В чём удобство использования класса `Path`?
10. Как с помощью модуля `glob` осуществлять поиск файлов в директориях по текстовым шаблонам-маскам (например, выбрать все файлы с расширением `.txt`)?
11. Практическое задание: Напишите скрипт на Python, который рекурсивно обходит заданную папку с помощью `os.walk()`, находит все файлы с расширением `.txt`, подсчитывает их суммарный размер в килобайтах и выводит список путей к этим файлам.
12. Практическое задание: Используя модуль `pathlib`, напишите программу, которая проверяет существование подпапки `"/corpus"`. Если её нет, программа создаёт её, а если она есть — выводит список всех файлов внутри неё.

Раздел 8. Потокковая обработка текстовых файлов и структурированных форматов данных

1. В чём заключаются принципиальные различия между текстовыми (содержащими кодированную строку символов) и бинарными (содержащими байт-код) файлами? Зачем при открытии текстового файла на русском или английском языках важно явно указывать параметр `encoding='utf-8'`?
2. Раскройте назначение конструкции безопасного управления внешними ресурсами — оператора `with` (менеджеры контекста). Почему использование `with` полностью вытеснило старую схему явного вызова методов `open()` и `close()`?
3. Подробно сравните три встроенных метода считывания информации из файлов: `read()`, `readline()`, `readlines()`. Каковы их особенности, возвращаемые типы данных и ограничения по объёму оперативной памяти?
4. Как реализовать построчную итерацию по файлу наиболее эффективным способом (`for line in file_object`) без считывания всего файла в память устройства?
5. Опишите режимы работы с файлами: чтение (`'r'`), перезапись (`'w'`), добавление данных в конец файла (`'a'`). Что произойдет со старым контентом файла при открытии его в режиме `'w'`?

6. Каковы принципы создания собственных пользовательских модулей (файлов `.py`) и пакетов программ? Как работает инструкция `import` и зачем нужен блок `if __name__ == '__main__':`?
7. Охарактеризуйте структуру и текстовые свойства табличного формата CSV. Как встроенный модуль `csv` (классы `csv.reader()`, `csv.writer()`, `csv.DictReader()`) автоматизирует чтение лингвистических таблиц?
8. Раскройте сущность формата структурированных данных JSON. Опишите соответствие типов данных JSON структурам Python (объекты — в словари, массивы — в списки).
9. Как работают функции встроенного модуля `json`: `json.load()`, `json.loads()`, `json.dump()`, `json.dumps()`? В чём разница между функциями с буквой `s` на конце и без неё?
10. Каковы общие принципы импорта и экспорта данных в форматы XML (модуль `xml.etree.ElementTree`) и электронные таблицы Excel? Какие открытые сторонние библиотеки (например, `openpyxl`) используются в Python для генерации `xlsx`-файлов без установленного офисного пакета?
11. Практическое задание: Напишите программу, которая открывает текстовый файл корпуса, построчно считывает его, приводит к нижнему регистру и записывает очищенные строки в новый файл, добавляя к имени исходного файла префикс `_cleaned`.
12. Практическое задание: Создайте лингвистическую структуру данных (список словарей с разметкой токенов). Напишите код, который сериализует эту структуру и записывает её на диск в виде структурированного файла в формате `.json` с отступами для удобочитаемости (`indent=4`).

Раздел 9. Множества и операции над ними

1. Охарактеризуйте особенности, свойства и внутреннюю структуру типа данных «множество» (`set`). Что означает свойство уникальности и неупорядоченности элементов множества? Почему элементы множества должны быть неизменяемыми (`hashable`)?
2. Опишите синтаксис создания множеств. Как создать пустое множество, учитывая, что фигурные скобки `{}` по умолчанию зарезервированы под пустой словарь?
3. Как осуществляется взаимное преобразование типов данных из списков во множества и обратно (`set(list)` $\rightarrow$ `list(set)`)? Какой побочный эффект (изменение порядка элементов) при этом возникает?
4. Опишите базовые операции модификации множеств: добавление элементов (`add()`, `update()`) и удаление элементов (`remove()`, `discard()`, `pop()`, `clear()`). В чём важное различие между методами `remove()` и `discard()` при попытке удалить несуществующий элемент?
5. Что такое генераторы множеств (`set comprehensions`)? Каков их синтаксис и как с их помощью быстро отфильтровать элементы текстового списка?
6. Раскройте синтаксис и лингвистический смысл теоретико-множественной операции объединения (`union()` или оператор `|`). Как объединить словари терминов двух разных доменов?
7. Раскройте синтаксис и смысл операции пересечения множеств (`intersection()` или оператор `&`). Как с её помощью быстро найти общие слова в двух разных текстах (например, при сопоставительном анализе)?
8. Опишите операцию вычитания множеств (разности) (`difference()` или оператор `-`). Как вычитание множеств позволяет мгновенно удалить все стоп-слова из частотного списка текста?
9. Что такое симметричная разность множеств (`symmetric_difference()` или оператор `^`)? Какие уникальные языковые единицы она позволяет выделить при сравнении двух идиостилей?

10. Как работают логические методы проверки подмножеств и подмножественных связей: `issubset()`, `issuperset()`, `isdisjoint()`? Как проверить, входят ли все слова поискового запроса пользователя в лексический состав документа?
11. Практическое задание: Напишите программу, которая принимает на вход две строки текста (два предложения на английском языке), превращает их в два множества уникальных лексем (очищенных от регистра) и выводит на экран: а) список общих слов; б) список слов, которые есть в первом предложении, но отсутствуют во втором.
12. Практическое задание: Реализуйте алгоритм количественного анализа лексического разнообразия текста. Скрипт должен прочитать текстовый файл, рассчитать общее количество токенов (объем текста — Tokens), количество уникальных слов (размер словаря — Types) с помощью типа данных `set` и вычислить коэффициент TTR (Type-Token Ratio) по формуле $TTR = \frac{Types}{Tokens}$, выведя результат в процентах.

Критерии оценки работы на практическом занятии

Критерии	Максимальное количество баллов за занятие
Устный опрос, коллоквиум	
Основные теоретические положения по вопросу раскрыты. Имеются элементы обоснования выводов. Имеются элементы систематизации информации, факты применения профессиональной терминологии. Очевидно использование источников рекомендованной литературы.	5 баллов

4.2.2. Темы докладов

1. Место и роль языка программирования Python в экосистеме современной компьютерной и прикладной лингвистики.
2. Эволюция стандартов оформления программного кода в Python: влияние PEP 8 на культуру командной разработки ИТ-продуктов.
3. Сравнительно-сопоставительный анализ производительности и структуры списков, кортежей и массивов при обработке сверхбольших текстовых массивов.
4. Технологии безопасной обработки исключений и логгирования ошибок в автоматизированных лингвистических процессорах.
5. Строковые операции и f-строки как базовый синтаксический инструмент первичного текстового форматирования и генерации контента.
6. Регулярные выражения (модуль `re`) в задачах извлечения текстовых паттернов, парсинга именованных сущностей и очистки интернет-дискурса.
7. Ассоциативные массивы (словари) как оптимальная модель представления лексикографических данных и частотных словарей.
8. Элементы функционального программирования в Python: функции высшего порядка (`map`, `filter`, `reduce`) и `lambda`-выражения в NLP-пайплайнах.
9. Генераторы и оператор `yield` как инструмент потоковой обработки и оптимизации оперативной памяти при работе с терабайтными корпусами текстов.
10. Области видимости (LEGB) и концепция замыканий: проектирование декораторов кода для замера скорости работы лингвистических алгоритмов.
11. Кроссплатформенное управление файловыми системами: эволюция от модуля `os` к объектно-ориентированному подходу `pathlib`.
12. Пакетная автоматизация обработки данных: алгоритмы рекурсивного обхода директорий (`os.walk`) при формировании языковых корпусов.

13. Менеджеры контекста (*with*) как гарант стабильности и безопасности ресурсов при потоковом чтении и записи внешних файлов.
14. Парсинг и сериализация структурированных данных: технологии обработки форматов JSON и CSV в прикладной лингвистике.
15. Интеграция Python с внешними офисными форматами: методы работы с XML и Excel (*openpyxl*) в лексикографических проектах.
16. Теоретико-множественные операции в Python: математические основы применения типа данных *set* в квантитативной лингвистике.
17. Использование множеств для автоматического вычисления лексического разнообразия (*TTR*, индекс Караулова) и фильтрации стоп-слов.
18. Проектирование пользовательских модулей и пакетов программ: архитектура чистого кода в распределенных междисциплинарных ИТ-командах.

Шкала и критерии оценки доклада

Критерии	Показатели	Баллы
1. Степень раскрытия сущности проблемы	– соответствие теме доклада; – полнота и глубина раскрытия основных понятий; – умение работать с литературой, систематизировать и структурировать материал; – умение обобщать, сопоставлять различные точки зрения по рассматриваемому вопросу, аргументировать основные положения и выводы.	70
2. Обоснованность выбора источников	– круг, полнота использования литературных источников по теме; – привлечение новейших работ (журнальные публикации, материалы сборников научных трудов и т.д.).	15
3. Изложение	– литературный стиль.	15

Доклад оценивается по 100 балльной шкале, баллы переводятся в оценки успеваемости следующим образом:

- 90 – 100 баллов – «отлично»;
 70 – 89 баллов – «хорошо»;
 50 – 69 баллов – «удовлетворительно»;
 менее 50 баллов – «неудовлетворительно».

4.2.3. Примерные темы для индивидуальных проектов

1. Регулярные выражения и парсинг текста: «Разработка Python-скрипта с использованием библиотеки *re* для автоматического извлечения именованных сущностей (имен, дат, геолокаций) из неструктурированных новостных сводок».
2. Поточковая обработка и структурированные форматы: «Скрипт автоматической конвертации сырых текстовых файлов (*.txt*) в валидный формат JSON/CSV с автоматическим расчетом лексического разнообразия (метрика *TTR*)».
3. Словари, хэш-таблицы и автоматизация: «Проектирование локальной базы данных синонимов и антонимов на Python на основе вложенных словарей со сверхбыстрым поиском».
4. Рекурсивный обход файловой системы: «Создание консольной утилиты для автоматического поиска, фильтрации по стоп-словам и архивации текстовых документов в глубоко разветвленных директориях».

4.3. Оценочные средства промежуточной аттестации

4.3.1. Контрольные задания

ОПК-3.1.1-1. Прочитайте текст и установите соответствие.

Установите соответствие между ключевыми словами (А-Г) и типами данных в Python (1-4):

А. list	1. Кортеж
Б. set	2. Словарь
В. tuple	3. Множество
Г. dict	4. Список

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

ОПК-3.1.1-2. Прочитайте текст и установите последовательность.

Установите правильную последовательность операторов для решения следующей задачи: ввести несколько слов, разделенных пробелами, вывести в консоль эти слова, отсортированные сначала по возрастанию, потом по убыванию:

- А. print(mbuf)
- Б. mbuf=buf.split()
- В. mbuf.sort(reverse=True)
- Г. buf=input()
- Д. mbuf.sort()

Запишите соответствующую последовательность букв слева направо:

--

ОПК-3.1.2-1. Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа.

Для отображения выборки из базы данных, лучше всего подходит следующая структура:

- А. Множество строк
- Б. Список строк
- В. Список словарей
- Г. Кортеж строк
- Д. Кортеж чисел

ОПК-3.1.2-2. Прочитайте текст и установите соответствие.

Установите соответствие между знаком (1-4) и действием в Python (А-Г):

А. #	1. Остаток от деления
Б. "	2. Комментарий
В. +	3. Начало многострочного комментария
Г. %	4. Конкатенация строки

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

ОПК-3.1.2-3. Прочитайте текст и установите последовательность.

Установите правильную последовательность операторов для выполнения следующих действий: создать кортеж, проинициализировать список из 5 элементов, вывести последний элемент списка, создать множество, получить индексы словаря:

- А. `b=a[-1:]`
- Б. `a=[i for i in range(5)]`
- В. `b=set()`
- Г. `d.keys()`
- Д. `a=tuple()`

Запишите соответствующую последовательность букв слева направо:

--

ОПК-3.1.3-1. Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа.

Сколько раз выполнится тело следующего цикла: **for i in range(5.5):**

- А. 5 раз
- Б. 6 раз
- В. Ни разу
- Г. Другой ответ

ОПК-3.1.3-2. Прочитайте текст и установите последовательность.

Установите правильную последовательность операторов для выполнения следующих действий: перевернуть `slovo`, выделить слово **"кара"** из `slovo= "карандаш"`, выйти из цикла **for**, выделить слово **"рак"** из `slovo= "карандаш"`, перейти к следующей итерации цикла **while**:

- А. `break`
- Б. `slovo[-6::-1]`
- В. `slovo[0:4]`
- Г. `continue`
- Д. `slovo[::1]`

Запишите соответствующую последовательность букв слева направо:

--

ОПК-3.1.3-3. Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа.

Имеется `stroka="КоШКа"`. Какой текст выведет следующая конструкция `print(stroka.reversed().capitalise())`:

- А. КОШКА
- Б. аКШоК
- В. Ничего
- Г. Акшок

ОПК-3.2.1-1. Прочитайте текст и установите соответствие.

Установите соответствие между функцией (1-4) и действием в Python (А-Г):

А. <code>capitalize()</code> Б. <code>swapcase()</code> В. <code>title()</code> Г. <code>upper()</code>	1. Преобразует первые буквы всех слов в заглавные 2. Перевод всех символов в верхний регистр 3. Перевод первой буквы в верхний регистр, остальных — в нижний 4. Замена регистра буквенных символов на противоположный
--	--

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

ОПК-3.2.1-2. Прочитайте текст и установите последовательность.

Установите правильную последовательность операторов для выполнения следующих действий: описать функцию с несколькими позиционными параметрами, описать функцию с несколькими именованными параметрами, описать функцию возвращающую список, описать функцию, возвращающую единичное значение:

- А. `map(lambda num: num**2, numbers)`
- Б. `def func(**kwargs):`
- В. `reduce(lambda x,y: x+y, items)`
- Г. `def func(*args):`

Запишите соответствующую последовательность букв слева направо:

--

ОПК-3.2.2-1. Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа.

Имеется `stroka="Двор"`. Какой текст выведет следующая конструкция `print(stroka.sort().swapcase())`:

- А. дВОР
- Б. ровД

- В. ВдОР
Г. Другой ответ

ОПК-3.2.2-2. Прочитайте текст и установите соответствие.

Дана строка **S**. Установите соответствие между ключевыми словами (1-4) и типами данных в Python (А-Г):

А. S.split()	1. Удаление спецсимволов и пробелов
Б. S.clear()	2. Разбивка строки
В. S.sort()	3. Не поддерживается
Г. S.strip()	4. Сортировка символов

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

ОПК-3.2.3-1. Прочитайте текст и установите последовательность.

Установите правильную последовательность операторов для того, чтобы узнать: начинаются ли слова строки с заглавной буквы, состоит ли строка из букв и цифр, преобразовать строку для сравнения без учета регистра, определить суффикс у строки, дополнить строку заданным символом до нужной длины:

- А. center()
Б. endswith()
В. casefold()
Г. isalnum()
Д. istitle()

Запишите соответствующую последовательность букв слева направо:

--

ОПК-3.2.3-2. Прочитайте текст, выберите правильный ответ и запишите аргументы, обосновывающие выбор ответа.

Можно ли, для считывания строк из текстового файла использовать цикл **for**:

- А. Нет, т.к. этот цикл предназначен для обработки списков
Б. Нет, т.к. мы заранее не знаем количество строк в файле
В. Да, но в этом цикле нельзя автоматически считывать номер строки
Г. Да, и в этом цикле можно автоматически считывать номер строки
Д. Другой ответ

ОПК-3.2.3-3.Прочитайте текст и установите соответствие.

Установите соответствие между знаком (1-4) и действием в Python (А-Г):

А. read() Б. readlines() В. readswap() Г. readline()	1. Построчное считывание файла 2. Считывание строк файла в список 3. Считывание строк файла в одну строку 4. Не существует
---	---

Запишите выбранные цифры под соответствующими буквами:

А	Б	В	Г

4.3.2. Ключи к контрольным заданиям

ОПК-3.1.1-1	А4Б3В1Г2
ОПК-3.1.1-2	ГБДАВА
ОПК-3.1.2-1	В Список словарей по структуре повторяет выборку из записей базы данных.
ОПК-3.1.2-2	А2Б3В4Г1
ОПК-3.1.2-3	ДБАВГ
ОПК-3.1.3-1	В Аргументом функции range() может быть только целое число.
ОПК-3.1.3-2	ДВАБГ
ОПК-3.1.3-3	В К строке нельзя применить метод reverse().
ОПК-3.2.1-1	А3Б4В1Г2
ОПК-3.2.1-2	ГБАВ
ОПК-3.2.2-1	Г К строке нельзя применить метод sort().
ОПК-3.2.2-2	А2Б3В4Г1
ОПК-3.2.3-1	ДГВБА
ОПК-3.2.3-2	Г Строки из текстового файла можно считывать с помощью цикла for. Кроме того, можно считывать и номер строки с помощью функции enumerate().
ОПК-3.2.3-3	А3Б2В4Г1

Шкала и критерии оценки текущего тестирования

Число правильных ответов	Оценка
90-100% правильных ответов	Оценка «отлично»
70-89% правильных ответов	Оценка «хорошо»
50-69% правильных ответов	Оценка «удовлетворительно»
Менее 50% правильных ответов	Оценка «неудовлетворительно»

5. ОСОБЕННОСТИ РЕАЛИЗАЦИИ ДИСЦИПЛИНЫ В ОТНОШЕНИИ ЛИЦ ИЗ ЧИСЛА ИНВАЛИДОВ И ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

Профессорско-преподавательский состав знакомится с психолого-физиологическими особенностями обучающихся инвалидов и лиц с ограниченными возможностями здоровья, индивидуальными программами реабилитации инвалидов (при наличии). При необходимости осуществляется дополнительная поддержка преподавания психологами, социальными работниками, прошедшими подготовку ассистентами.

В соответствии с методическими рекомендациями Министерства науки и высшего образования Российской Федерации, в курсе предполагается использовать социально-активные и рефлексивные методы обучения, технологии социокультурной реабилитации с целью оказания помощи в установлении полноценных межличностных отношений с другими студентами, создания комфортного психологического климата в студенческой группе. Подбор и разработка учебных материалов производятся с учётом предоставления материала в различных формах: аудиальной, визуальной, с использованием специальных технических средств и информационных систем.

Согласно требованиям к порядку реализации образовательной деятельности в отношении инвалидов и лиц с ОВЗ, установленным Министерством науки и высшего образования Российской Федерации, необходимо иметь в виду, что:

1) инвалиды и лица с ОВЗ по зрению имеют право присутствовать на занятиях вместе с ассистентом, оказывающим обучающемуся необходимую помощь;

2) инвалиды и лица с ОВЗ по слуху имеют право на использование звукоусиливающей аппаратуры.

При проведении промежуточной аттестации по дисциплине обеспечивается соблюдение следующих общих требований:

- проведение аттестации для инвалидов в одной аудитории совместно с обучающимися, не являющимися инвалидами, если это не создает трудностей для инвалидов и иных обучающихся при промежуточной аттестации;

- присутствие в аудитории ассистента (ассистентов), оказывающего обучающимся инвалидам необходимую техническую помощь с учётом их индивидуальных особенностей (занять рабочее место, передвигаться, прочитать и оформить задание, общаться с экзаменатором);

- пользование необходимыми обучающимся инвалидам техническими средствами при прохождении промежуточной аттестации с учётом их индивидуальных особенностей;

- обеспечение возможности беспрепятственного доступа обучающихся инвалидов в аудитории, туалетные и другие помещения, а также их пребывания в указанных помещениях.

По письменному заявлению обучающегося инвалида продолжительность прохождения испытания промежуточной аттестации (зачёта, экзамена, и др.) обучающимся инвалидом может быть увеличена по отношению к установленной продолжительности его сдачи:

- продолжительность сдачи испытания, проводимого в письменной форме, — не более чем на 90 минут;

- продолжительность подготовки обучающегося к ответу, проводимому в устной форме, — не более чем на 20 минут.

В зависимости от индивидуальных особенностей обучающихся с ОВЗ Университет обеспечивает выполнение следующих требований при проведении аттестации:

а) для слепых:

- задания и иные материалы для прохождения промежуточной аттестации оформляются в виде электронного документа, доступного с помощью компьютера со специализированным программным обеспечением для слепых, либо зачитываются ассистентом;

- письменные задания выполняются обучающимися на бумаге рельефно-точечным шрифтом Брайля или на компьютере со специализированным программным обеспечением для слепых, либо надиктовываются ассистенту;

- при необходимости обучающимся предоставляется комплект письменных принадлежностей и бумага для письма рельефно-точечным шрифтом Брайля, компьютер со специализированным программным обеспечением для слепых;

б) для слабовидящих:

- задания и иные материалы для сдачи экзамена оформляются увеличенным шрифтом;

- обеспечивается индивидуальное равномерное освещение не менее 300 люкс;

- при необходимости обучающимся предоставляется увеличивающее устройство, допускается использование увеличивающих устройств, имеющихся у обучающихся;

в) для глухих и слабослышащих, с тяжелыми нарушениями речи:

- обеспечивается наличие звукоусиливающей аппаратуры коллективного пользования, при необходимости обучающимся предоставляется звукоусиливающая аппаратура индивидуального пользования;

- по их желанию испытания проводятся в письменной форме;

г) для лиц с нарушениями опорно-двигательного аппарата (тяжёлыми нарушениями двигательных функций верхних конечностей или отсутствием верхних конечностей):

- письменные задания выполняются обучающимися на компьютере со специализированным программным обеспечением или надиктовываются ассистенту;

- по их желанию испытания проводятся в устной форме.

О необходимости обеспечения специальных условий для проведения аттестации обучающийся должен сообщить письменно не позднее, чем за 10 дней до начала аттестации. К заявлению прилагаются документы, подтверждающие наличие у обучающегося индивидуальных особенностей (при отсутствии указанных документов в организации).

При необходимости для обучающихся с инвалидностью процедура оценивания результатов обучения может проводиться в несколько этапов.