

Федеральное государственное образовательное бюджетное учреждение
высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)**

**Кафедра информационных технологий
Факультет информационных технологий и анализа больших данных**

Документ подписан усиленной неквалифицированной электронной подписью
Организация: Финансовый университет при Правительстве РФ
Утверждено: Проректор по учебной и методической работе Е.А. Каменева
Сертификат: Qqkr1D8EA9KSzrbodp4NRvEf/CYIR1xq
Дата: 07.10.2025 г.

Е. П. Догадина

Алгоритмы и структуры данных в языке Python

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки

09.03.04 - Программная инженерия

Образовательная программа:

Инженер-разработчик программного обеспечения

Профиль

Инженер-разработчик программного обеспечения

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол №2 от 18.11.2025)*

Одобрено

*Кафедра информационных технологий
(протокол №7 от 27.10.2025)*

Москва 2025

Содержание

Наименование разделов РПД		Стр.
1.	Наименование дисциплины	5
2.	Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	5
3.	Место дисциплины в структуре образовательной программы	8
4.	Объем дисциплины(модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	8
5.	Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий	8
5.1.	Содержание дисциплины	9
5.2.	Учебно-тематический план	13
5.3.	Содержание семинаров, практических занятий	14
6.	Перечень учебно-методического обеспечения для	18

	самостоятельной работы обучающихся по дисциплине	
6.1.	Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	19
6.2.	Перечень вопросов, заданий, тем для подготовки к текущему контролю	20
7.	Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	27
8.	Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	37
9.	Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	39
10.	Методические указания для обучающихся по освоению дисциплины	39
11.	Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных	40

	справочных систем	
12.	Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	40

1. Наименование дисциплины

«Алгоритмы и структуры данных в языке Python».

2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине

Код компетенции	Наименование компетенции	Индикатор достижения компетенции	Результаты обучения
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов	Разрабатывает алгоритмы решения простых информационных задач и выражает их на языке программирования	Знать: основные алгоритмы и базовые структуры данных Уметь: разрабатывать алгоритмы для работы со структурами данных
		Анализирует алгоритмы в части производительности, оптимальности, вырабатывает рекомендации для оптимизации алгоритмов программ	Знать: структурные, функциональные и объектно-ориентированные парадигмы программирования на языке Python, сложность алгоритмов Уметь: определять на уровне знания синтаксис и семантику, стандартные библиотеки языка Python, необходимые для

			оптимизации, оценивать сложность алгоритмов
		Проводит ручное и автоматизированное тестирование программных продуктов по методам черного и белого ящика, составляет набор тестовых случаев	Знать: основные принципы и подходы к тестированию программных продуктов Уметь: реализовывать программные продукты с помощью библиотек Python и проводить тестирование программных продуктов
ОПК-7	Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой	Демонстрирует знания основ теории информации и алгоритмов, основных элементарных алгоритмов и структуры данных	Знать: способы представления данных структур данных в памяти компьютера и принципы их обработки; базовые алгоритмические конструкции Уметь: выбирать способы представления информации, соответствующие решаемой задаче
		Применяет простые алгоритмы и структуры данных к решению поставленной задачи, проводит выбор наиболее	Знать: основные типы структур данных, алгоритмы сортировки и поиска Уметь: выбирать подходящие

		оптимальных методов	структуры данных для решения конкретных задач, оценивать эффективность алгоритмов, производя анализ временной и пространственной сложности
		Проводит подробный количественный анализ реализованной программной системы с точки зрения оптимальности применяемых алгоритмических решений	Знать: основные понятия анализа алгоритмов, методы оценки производительности алгоритмов, метрики для оценки алгоритмов Уметь: выполнять количественный анализ программной системы с использованием различных методов и подходов, исследовать и сравнивать различные алгоритмы и структуры данных на основе выбранных метрик, проводить тестирование производительности программных решений,

			включая написание тестов и использование инструментов для сбора метрик
--	--	--	--

3. Место дисциплины в структуре образовательной программы

Дисциплина «Алгоритмы и структуры данных в языке Python» относится к «Общепрофессиональному циклу»

4. Объем дисциплины(модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся

очно-заочная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 2 (в часах)	Семестр 3 (в часах)
Общая трудоемкость дисциплины	8/288	144	144
Контактная работа - Аудиторные занятия	42	22	20
<i>Лекции</i>	6	4	2
<i>Семинары, практические занятия</i>	36	18	18
Самостоятельная работа	246	122	124
Вид текущего контроля	Контрольная работа; Контрольная работа;	Контрольная работа	Контрольная работа
Вид промежуточной аттестации	Экзамен; Экзамен;	Экзамен	Экзамен

5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий

5.1. Содержание дисциплины

Тема 1. Введение в программирование на Python

Общая информация о языке Python . История языка программирования, его связь с другими языками программирования, распространенность Python и основные сферы его применения. Знакомство с первыми примерами кода на Python. Философия Python.

Базовая информация о языке Python . Основные типы данных. Основные числовые типы данных и операции над ними. Математические операции над числовыми типами данных. Преобразование типов данных. Переменные и специфика их объявления. Статическая и динамическая типизация. Работа с переменными. Управление памятью и сборка мусора в Python . Именование переменных.

Работа со строками: создание строк, специальные символы. Индексирование строк, получение срезов строк. Основные функции для работы со строками. Вывод на экран (работа с функцией print) и форматирование строк. Различные подходы к форматированию строк, форматирование с помощью f-строк. Расширенное форматирование в Python.

Тема 2. Управляющие конструкции, списки и кортежи

Управляющие конструкции в Python . Булев тип: объявление и операции. Операции сравнения в Python . Условные операторы в Python . Реализация задачи case в Python .

Циклы в Python: while, for. Специфика циклов в Python . Функции range и enumerate и их использование в циклах.

Списки и кортежи в Python . Специфика списков и их отличие от массивов. Создание списка, оперирование вложенными списками, копирование списков, операции над списками: индексация и срезы; изменение списка; поиск, сортировка и обход; изменение списка. Кортежи в Python : синтаксис, специфика использования.

Тема 3. Словари, множества и выражения-генераторы

Словари Python . Словари: семантика, синтаксис создания, операции над словарями, перебор элементов словаря.

Множества в Python . Множества: семантика, синтаксис создания, операции над словарями, перебор элементов словаря. Специфика операций с множествами в Python .

Выражения-генераторы в Python . Выражения-генераторы для списков: семантика и синтаксис. Пример: задача приведения списка к "плоскому" виду. Выражения-генераторы для множеств и словарей. Кейсы использования и производительность решений с использованием выражений-генераторов.

Тема 4. Функции

Функции в Python : общая семантика. Создание функции и ее вызов. Расположение определений функций. Анонимные функции в Python . Необязательные параметры функций и сопоставление по ключам. Возвращение нескольких значений из функции. Распаковка и запаковка параметров функции. Аннотации и документирование функций. Глобальные и локальные переменные.

Тема 5. Работа с файлами и обработка исключительных ситуаций

Обработка исключений в Python : кейсы для использования. Инструкция `try ... except ... else ... finally`. Классы встроенных исключений. Создание пользовательских исключений. Инструкция `assert`.

Работа с файлами в Python . Концепция файла в современных ОС и языках программирования. Операции с файлами: открытие/закрытие файла, чтение и записи и другие методы для работы с файлами. Инструкция `with ... as` и ее использование для файлов.

Сохранение объектов в файл с помощью модуля `pickle` . Модуль `CSV` .

Тема 6. Модули и пакеты

Модули и пакеты в Python : подход к структурированию программного кода с помощью модулей и пакетов. Синтаксис

импортирования в Python . Создание и работа с пакетами в Python . Повторная загрузка модулей.

Написание и запуск скриптов на Python. Установка модулей из глобального репозитория.

Тема 7. Продвинутое коллекции

Модули стандартной библиотеки Python , предоставляющие дополнительный функционал по работе с коллекциями объектов. Класс Frozen set. Модуль collections. Класс Counter , класс defaultdict , класс OrderedDict , класс namedtuple . Модуль enum .

Тема 8. Объектно-ориентированное программирование в Python

Предпосылки и история появления ООП. Объекты и классы в ООП. Принципы и основные механизмы ООП. Логика работы абстракции, инкапсуляции, наследования и полиморфизма.

Python как объектно-ориентированный язык программирования. Базовые возможности ООП в Python : с оздание классов и объектов; наследование и полиморфизм; функция super(); проверка принадлежности к классу. Базовые типы в Python.

Методы классов и статические переменные и методы в Python . Управление доступом к атрибутам класса в Python . Динамические операции с атрибутами и интроспекция в Python . Использование специальных методов для расширенного функционала пользовательских классов. Кейс построения иерархии классов.

Тема 9. Функциональное программирование в Python

Парадигмы и идиомы программирования, общая концепция функционального программирования. Функциональные языки программирования.

Функции "граждане первого класса", функции высшего порядка, замыкания, функции без побочных эффектов, рекурсия, хвостовая рекурсия. Неизменяемые структуры данных. Идиомы, распространенные в функциональных языках программирования: итераторы, последовательности, ленивые вычисления, сопоставление с образцом, монады.

Элементы функционального программирования в Python : функции – граждане первого класса; глобальные и локальные переменные в Python ; вложенные функции и замыкания в Python .

Декораторы в Python : использование и создание собственных декораторов.

Подход : map, filter, reduce. Реализация функций map, filter, reduce в Python.

Итераторы в Python , итерируемый тип данных. Модуль itertools.

Функции-генераторы и выражения-генераторы в Python .

Тема 10. Структуры данных: массивы, стеки, очереди, списки

Введение в анализ сложности алгоритмов.

Массивы и их отличие от списков в Python . Динамические массивы, сложность операций работы с динамическими массивами.

Стек, операции со стеком. Реализации стека. Очередь, операции с очередью. Реализация очереди. Связные списки, варианты связанных списков.

Тема 11. Алгоритмы поиска и сортировки

Поиск в списках/массивах, бинарный поиск. Сортировка и ее использование в прикладных задачах.

Простые методы сортировки: обменные сортировки (с различными вариациями); сортировка выбором (извлечением); сортировка включением (вставками).

Эффективные методы сортировки: быстрая сортировка; сортировка Шелла; сортировка слиянием. Сравнение различных сортировок.

Тема 12. Структуры данных: деревья

Деревья, бинарные деревья. Использование бинарных деревьев в прикладных задачах: представление выражения (предложения) в виде дерева. Обход бинарных деревьев.

Двоичное дерево поиска. Двоичные кучи и очереди с приоритетом.

Тема 13. Хеш-таблицы

Абстрактный тип данных – ассоциативный массив. Таблица с прямой адресацией. Хеш-таблица, хеш-функция: метод деления; метод MAD. Полиномиальная хеш-функция.

Функция hash в Python. Методы разрешения коллизий.

5.2. Учебно-тематический план

очно-заочная форма обучения

№ п/п	Наименование тем(разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа			Самостоятельная работа
			Общая, в т.ч.:	Лекции	Семинары, практические занятия	
1	Введение в программирование на Python	22	2	1	1	20
2	Управляющие конструкции, списки и кортежи	25	5	1	4	20
3	Словари, множества и выражения-генераторы	24	4	0	4	20
4	Функции	26	6	2	4	20
5	Работа с файлами и обработка исключ	24	4	0	4	20

	ительны х ситуаци й					
6	Модули и пакеты	12	2	0	2	10
7	Продви нутые коллекц ии	12	2	0	2	10
8	Объектн о- ориенти рованно е програм мирован ие в Python	26	6	2	4	20
9	Функцио нальное програм мирование в Python	24	4	0	4	20
10	Структу ры данных: массивы , стеки, очереди, списки	22	2	0	2	20
11	Алгорит мы поиска и сортиро вки	22	2	0	2	20
12	Структу ры данных: деревья	32	2	0	2	30
13	Хеш- таблицы	17	1	0	1	16
	Итого	288	42	6	36	246

5.3. Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
Введение в программирование на Python	Установка Python, установка дистрибутива Anaconda. Работа в интерактивном режиме интерпретатора. Интерактивная оболочка IPython notebook: принципы работы и применения.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Управляющие конструкции, списки и кортежи	Базовые числовые типы, строки, списки, словари, переменные, базовые операторы.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Словари, множества и выражения-генераторы	Множества в Python. Множества: семантика, синтаксис. Словари в Python: создание, операции над словарями, перебор элементов словаря. Специфика операций с множествами и словарями в Python. Выражения-генераторы в Python. Выражения-генераторы для списков: семантика и синтаксис.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Функции	Создание функций, область видимости переменной, передача аргументов в функцию. Лямбда-функции.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)

Работа с файлами и обработка исключительных ситуаций	Исключения. Инструкция try...except...else...finally. Классы встроенных исключений. Создание пользовательских исключений. Инструкция assert. Работа с файлами в Python, операции с файлами: открытие/закрытие файла, чтение и записи и другие методы для работы с файлами. Инструкция with ... as и ее использование для файлов.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Модули и пакеты	Устройство модулей и пакетов, инструкции import и from. Создание собственных модулей и пакетов.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Продвинутые коллекции	Модули стандартной библиотеки Python, предоставляющие дополнительный функционал по работе с коллекциями объектов. Класс Frozen set. Модуль collections.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Объектно-ориентированное программирование в Python	Парадигмы программирования. Специализация языков программирования. Место Python в современном ландшафте языков программирования. Характеристики языков программирования: способ передачи параметров в функцию; способ	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)

	управления динамической памятью. Базовые возможности ООП в Python: создание классов и объектов; наследование и полиморфизм; функция super(). Методы классов и статические переменные и методы в Python. Управление доступом к атрибутам класса в Python. Динамические операции с атрибутами и интроспекция в Python. Использование специальных методов для расширенного функционала пользовательских классов. Кейс построения иерархии классов.	
Функциональное программирование в Python	Элементы функционального программирования в Python: функции – граждане первого класса; глобальные и локальные переменные в Python; вложенные функции и замыкания в Python. Декораторы в Python: использование и создание собственных декораторов. Реализация функций map, filter, reduce в Python. Итераторы в Python, итерируемый тип данных. Модуль itertools.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)

Структуры данных: массивы, стеки, очереди, списки	Массивы и их отличие от списков в Python. Динамические массивы, сложность операций работы с динамическими массивами. Стек, операции со стеком. Реализации стека. Очередь, операции с очередью. Реализация очереди. Связные списки, варианты связанных списков.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Алгоритмы поиска и сортировки	Реализация на Python простых методов сортировки: обменные сортировки (с различными вариациями); сортировка выбором (извлечением); сортировка включением (вставками). Реализация на Python эффективных методов сортировки: быстрая сортировка; сортировка Шелла; сортировка слиянием.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Структуры данных: деревья	Реализация на Python деревьев, бинарных деревьев. Использование бинарных деревьев в прикладных задачах: представление выражения (предложения) в виде дерева.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)
Хеш-таблицы	Работа с функцией hash в Python. Использование специализированных библиотек для работы с хэш-функциями.	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (не менее 30% времени на интерактивные технологии)

6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Введение в программирование на Python	Среда программирования. Использование документации.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Управляющие конструкции, списки и кортежи	Оперирование вложенными списками, копирование списков, некоторые операции над списками.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Словари, множества и выражения-генераторы	Выражения-генераторы в Python. Выражения-генераторы для словарей и множеств: семантика, синтаксис и практическое использование.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Функции	Возвращение нескольких значений из функции. Распаковка и запаковка параметров функции.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Работа с файлами и обработка исключительных ситуаций	Сохранение объектов в файл с помощью модуля pickle и shelve . Модуль CSV .	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Модули и пакеты	Написание и запуск скриптов на Python. Установка модулей из глобального репозитория.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Продвинутое	Класс Counter , класс	Индивидуальное

коллекции	defaultdict , класс OrderedDict , класс namedtuple . Модуль enum .	выполнение заданий с использованием Jupyter Notebook .
Объектно-ориентированное программирование в Python	Виды типизации переменных в различных современных языках программирования. Базовые типы в Python: взгляд с точки зрения ООП. Методы базовых типов. Проверка принадлежности к классу и интроспекция в Python.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Функциональное программирование в Python	Реализация декораторов с параметрами в Python. Функции-генераторы и выражения-генераторы в Python.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Структуры данных: массивы, стеки, очереди, списки	Реализация различных вариантов связанных списков на Python.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Алгоритмы поиска и сортировки	Сравнение различных сортировок (простых и эффективных) с использованием их реализаций на Python.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Структуры данных: деревья	Реализация обхода бинарных деревьев.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .
Хеш-таблицы	Использование специализированных библиотек для работы с хэш-функциями: проверка хэш-конфликтов для различных наборов данных.	Индивидуальное выполнение заданий с использованием Jupyter Notebook .

6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю

Примерные вопросы к контрольной работе 2 семестра

1. Операции над основными числовыми типами данных.
2. Операции над булевыми переменными.
3. Динамическая типизация в Python.
4. Преобразование типов в Python.
5. Создание строк в Python.
6. Организация и пример цикла while в Python.
7. Организация и пример цикла for в Python.
8. Операции над словарями в Python.
9. Выражения-генераторы для списков в Python.
10. Необязательные параметры функций в Python.
11. Заpackовка и распаковка параметров в Python.
12. Аннотации и документирование функций.
13. Создание объектов в Python.
14. Управление доступом к атрибутам класса в Python.
15. Выполнение интроспекции в Python.
16. Замыкания в Python.
17. Использование декораторов в Python.
18. Создание собственных декораторов в Python.
19. Компилятор и интерпретатор. Достоинства и недостатки.
20. Назовите и дайте краткую характеристику основных классов языков программирования.
21. Встроенные числовые типы языка Python.
22. Списки. Создание, основные операции.
23. Основные методы списка.
24. Кортежи. Создание, основные методы и операции.

25. Словари. Создание, основные операции.
26. Методы для работы со словарями.
27. Множества. Создание, основные методы и операции.
28. Переменные. Правила именования переменных.
29. Динамическая типизация.
30. Операторы сравнения и логические операторы.
31. Инструкция if...else.
32. Инструкция цикла while.
33. Инструкция цикла for.
34. Создание и вызов функции.
35. Передача аргументов функцию.
36. Функции-генераторы.
37. Лямбда-функции.
38. Модули. Инструкции import и from.

Примерные задания контрольной работы 2 семестра

1. В строке содержащей последовательность слов, разделенных запятыми удалить все нечетные слова. Ответ представить в виде строки. Пример: строка 'SIX,SEVEN,EIGHT,NINE,TEN' будет преобразована в: 'SIX,EIGHT,TEN'.
2. Из списка списков элементами которого являются текстовые символы собрать строку, в которой вложенные списки объединены в слова, а слова через запятую объединены в строку. Пример список вида [['E', 'e', 'n', 'y'], ['m', 'e', 'e', 'n', 'y'], ['m', 'i', 'n', 'e', 'y'], ['m', 'o', 'e']] будет преобразован в строку 'Eeny,meeny,miney,moe'.
3. Используя генератор словарей (и не используя код вне него) инвертировать словарь, т.е. сделать ключи словаря, его значениями и наоборот. Значения, которые в исходном словаре повторяются не добавлять в итоговый словарь. Пример: {'a':1, 'b':3, 'c':4, 'd':3} -> {1:'a', 4:'c'}

4. Задана строка, в которой через запятую перечислены имена людей (с заглавной буквы) и их текущие занятия (со строчной буквы) в произвольном порядке (например, "Иван ест, поет Оля" и т.д.). С помощью генераторов создать словарь, в котором ключами будут имена, а значениями – занятия. Решить задачу в одну строку. Например: "Маша гуляет, Коля работает, дома Ваня, закупается Женя" представить в виде {'Ваня': 'дома', 'Женя': 'закупается', 'Коля': 'работает', 'Маша': 'гуляет'}.

5. Написать скрипт, который заменяет в текстовом файле все слова из определенного перечня на определенные для этих слов слова-заменители. Скрипт принимает через командную строку 2 параметра:

- первый параметр - имя файла из которого берется текст (файл имеет кодировку UTF8; нет переносов слов на другую строку);
- второй параметр - имя файла, являющегося файлом CSV (с разделителем ;) содержащего перечень слов для замены и соответствующих им заменителей (файл не имеет заголовка столбцов; в первом столбце слова подлежащие замене, во втором слова-заменители) (файл имеет кодировку UTF8).

Результат замены слов сохраняется в новом файле с именем совпадающем с исходным файлом с текстом, но имеющим префикс подчеркивание (_). Скрипт должен иметь вид `replace_Ivanov.py` (вместо `Ivanov` - ваша фамилия). Функции связанные с обработкой файлов и заменой в тексте должны находиться в отдельном модуле `modul_Ivanov.py` (вместо `Ivanov` - ваша фамилия).

6. Реализовать функцию `summmulate` для расчета накопленных двойных сумм (квадратов произведений). Функция принимает одно или более числовое значение (количество параметров заранее не определено). На основе этих значений рассчитываются накопленные суммы, которые сохраняются в списке, список возвращается как результат функции. Необязательный булевский параметр `mul` должен позволять заменять суммирование умножением.

Примерные вопросы к контрольной работе 3 семестра

1. Базовые принципы объектно-ориентированного программирования.
2. Класс, метод класса, атрибут класса. Определение класса и создание экземпляра класса.
3. Конструктор и деструктор.
4. Наследование.
5. Абстрактные методы класса.
6. Статические методы класса.
7. Свойства класса.
8. Исключения. Обработка исключений.
9. Пользовательские исключения.
10. Вложенные функции и замыкания, специфика реализации в Python.
11. Функции высшего порядка и декораторы в Python.
12. Реализация map/filter/reduce в Python и пример их использования.
13. Итераторы в Python.
14. Встроенные функции для работы с итераторами и возможности модуля itertools.
15. Специфика массивов, как структур данных.
16. Абстрактная структура данных стек и очередь: базовые и расширенные операции, их сложность.
17. Реализация основных операций в очереди на базе массива и связанного списка.
18. Связанные списки: однонаправленные и двунаправленные – принцип реализации.
19. Алгоритм обменной сортировки.
20. Алгоритм сортировки выбором.

21. Алгоритм сортировки вставками.
22. Алгоритм быстрого поиска в отсортированном массиве.
23. Алгоритм сортировки Шелла.
24. Алгоритм быстрой сортировки.
25. Алгоритм сортировки слиянием.
26. Структуры данных: деревья
27. Реализация двоичных деревьев в виде связанных объектов. Различные реализации рекурсивного обхода двоичных деревьев.
28. Двоичная куча, реализации основных операций.
29. Абстрактный тип данных - ассоциативный массив и принцип его реализации на основе хэш-таблиц и хэш-функций.
30. Хэш - функции multiply-add-and-divide. Принцип работы хэш - функции multiply-add-and-divide.
31. Полиномиальная хэш-функция.

Примерные задания контрольной работы 3 семестра

1. Создать иерархию классов для фруктов, продающихся в магазине. Иерархия должна содержать не менее 3 классов. Объекты должны содержать не менее 2-х атрибутов и 2-х методов. Реализовать механизм автоматического подсчета количества всех созданных фруктов и автоматического присвоения каждому фрукту уникального идентификатора. Необходимо заполнить список представителями всех классов (всего не менее 10 объектов) и продемонстрировать работу созданного механизма.

2. Создайте класс Length (Длина), имеющий свойства:

- value (значение),
- unit (единица измерения).

При изменении единицы измерения значение должно соответственно меняться. Например, при переходе от сантиметров к метрам значение должно уменьшаться в 100 раз. Допустимые

значения свойства unit: 'см', 'м', 'км'. Организуйте эту проверку. Продемонстрируйте работу с классом.

3. Создайте класс Заказ(Order), у которого есть свойства код_товара(code), цена(price), количество(count) и методы __init__ и __str__. Создайте 2 класса-потомка: Опт(Opt) и Розница(Retail). В этих классах создайте методы __init__, __str__ и сумма_заказа(summa), позволяющий узнать стоимость заказа. Для опта стоимость единицы товара составляет 95% от цены, а при покупке более 500 штук – 90% цены. В розницу стоимость единицы товара составляет 100% цены. Стоимость заказа равна произведению цены на количество. Продемонстрируйте работу с классами, создав необходимые объекты и обратившись к их свойствам и методам.

4. Написать функцию-генератор my_func_2(lst), которая принимает объект, поддерживающий итерации с произвольным уровнем вложенности, и возвращает все элементы по одному.

5. С помощью механизма map/filter/reduce (хотя бы одна из этих функций должна быть использована в решении) посчитать в тексте количество слов, состоящих не менее, чем из 3-х букв. Слова в тексте разделены пробелами. Написать реализацию в одну строку. Оформить решение в виде функции my_func_3(text), т.е. шаблон таков: строка с import, если необходимо def my_func_3(text): return # однострочная реализация задания.

6. Написать декоратор с параметром my_decorator(n). Декоратор превращает функцию, возвращающую поддерживающий итерации объект (далее "последовательность"), в функцию-генератор. Если декорируемая функция возвращает что-то другое, а не последовательность, то декоратор должен вернуть этот результат вызова функции без изменений. Проверку объекта можно организовать при помощи условия import collections if isinstance (item, collections.Iterable). Параметром декоратора может быть целое положительное число n, тогда получившаяся функция-декоратор должна генерировать по одному значению из последовательности, повторенной n раз. Также параметр может принимать строковое значение 'inf', тогда функция-декоратор генерирует по одному значению из последовательности, повторенной бесконечное число раз (за циклирует генерирование результата). Подсказка: сначала

реализовать случай со значением аргумента 'inf', а затем модифицировать для целочисленного значения аргумента.

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях Кафедры информационных технологий.

7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

Перечень компетенций с указанием индикаторов их достижения в процессе освоения образовательной программы содержится в разделе 2. *Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине.*

Типовые контрольные задания или иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний

ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов

1) Разрабатывает алгоритмы решения простых информационных задач и выражает их на языке программирования

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: основные алгоритмы и базовые структуры данных

Уметь: разрабатывать алгоритмы для работы со структурами данных

Типовые контрольные задания

Выполните программный код вычисления арифметического выражения с помощью двоичного дерева.

Выберите библиотеку Python для работы с массивами. Выполните программный код бинарного поиска данных.

2) Анализирует алгоритмы в части производительности, оптимальности, вырабатывает рекомендации для оптимизации алгоритмов программ

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: структурные, функциональные и объектно-ориентированные парадигмы программирования на языке Python, сложность алгоритмов

Уметь: определять на уровне знания синтаксис и семантику, стандартные библиотеки языка Python, необходимые для оптимизации, оценивать сложность алгоритмов

Типовые контрольные задания

Создайте 2 класса-потомка: `Опт(Opt)` и `Розница(Retail)`. В этих классах создайте методы `__init__`, `__str__` и `сумма заказа (summa)`, позволяющие узнать стоимость заказа.

Реализуйте программу на Python которой в качестве аргумента командной строки передается имя CSV-файла, в первом столбце находятся числа, которые необходимо отсортировать. Программа создает новый файл, в котором первый столбец отсортирован.

3) Проводит ручное и автоматизированное тестирование программных продуктов по методам черного и белого ящика, составляет набор тестовых случаев

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: основные принципы и подходы к тестированию программных продуктов

Уметь: реализовывать программные продукты с помощью библиотек Python и проводить тестирование программных продуктов

Типовые контрольные задания

Выполните обработку и тестирование текстового документа, в котором зафиксированы факты продаж, необходимого для подсчета суммарных продаж по различным географическим

подразделениям фирмы. На основе полученных данных составьте CSV файл, хранящий полученные результаты.

Определите закономерности в синтаксических конструкциях Python для визуализации данных с применением графической библиотеки.

ОПК-7 Способен применять в практической деятельности основные концепции, принципы, теории и факты, связанные с информатикой

1) Демонстрирует знания основ теории информации и алгоритмов, основных элементарных алгоритмов и структуры данных

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: способы представления данных структур данных в памяти компьютера и принципы их обработки; базовые алгоритмические конструкции

Уметь: выбирать способы представления информации, соответствующие решаемой задаче

Типовые контрольные задания

В рамках проекта по равномерному распределению ресурсов необходимо преобразовать словарь так, чтобы ключами стали средние значения каждого кортежа; например, из словаря $\{(2, 4): 'a', (1, 1, 1): 'б', (2, 3): 'в'\}$ получается $\{3.0: 'a', 1.0: 'б', 2.5: 'в'\}$.

Создайте метод шифрования текстовых сообщений, перемещая символы на четных позициях в начало строки, а символы на нечетных позициях — в конец строки в обратном порядке, для повышения безопасности передачи данных.

2) Применяет простые алгоритмы и структуры данных к решению поставленной задачи, проводит выбор наиболее оптимальных методов

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: основные типы структур данных, алгоритмы сортировки и поиска

Уметь: выбирать подходящие структуры данных для решения конкретных задач, оценивать эффективность алгоритмов, производя анализ временной и пространственной сложности

Типовые контрольные задания

Решите задачу нахождения максимального элемента в массиве, и реализуйте несколько подходов (простой проход по массиву, разделение массива на части). Оцените время выполнения каждого метода на больших объемах данных и сделайте вывод о том, какой метод дает наилучшие результаты.

Реализовать несколько алгоритмов сортировки (например, пузырьковая сортировка и сортировка выбором). Сравнить производительность этих алгоритмов на одном и том же наборе данных (например, 1000 случайных чисел). Вывести на экран время выполнения каждого алгоритма.

3) Проводит подробный количественный анализ реализованной программной системы с точки зрения оптимальности применяемых алгоритмических решений

Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции

Знать: основные понятия анализа алгоритмов, методы оценки производительности алгоритмов, метрики для оценки алгоритмов

Уметь: выполнять количественный анализ программной системы с использованием различных методов и подходов, исследовать и сравнивать различные алгоритмы и структуры данных на основе выбранных метрик, проводить тестирование производительности программных решений, включая написание тестов и использование инструментов для сбора метрик

Типовые контрольные задания

Выбрать алгоритм, который требуется оптимизировать (например, поиск максимального элемента в массиве). Реализовать как текущую, так и оптимизированную версии алгоритма. Провести количественный анализ оптимизированного алгоритма относительно временной и пространственной сложности. Провести тестирование и сравнить эффективность существующего и

оптимизированного алгоритма, предоставив подробные результаты анализа.

Сравнить различные алгоритмы поиска по эффективности. Реализовать как линейный поиск, так и бинарный поиск в массиве (при условии, что массив отсортирован). Провести количественный анализ: измерить время выполнения каждого алгоритма при различных размерах массивов (например, 100, 1000, 10000 и 100000 элементов). Провести анализ ситуаций, в которых выбранные алгоритмы показывают наилучшую или наихудшую производительность.

Примеры практико-ориентированных заданий

1. Используя генератор словарей (и не используя код вне него) инвертировать словарь, т.е. сделать ключи словаря, его значениями и наоборот. Значения, которые в исходном словаре повторяются не добавлять в итоговый словарь. Пример: {'a':1, 'b':3, 'c':4, 'd':3} -> {1:'a', 4:'c'}
2. Задана строка, в которой через запятую перечислены имена людей (с заглавной буквы) и их текущие занятия (со строчной буквы) в произвольном порядке (например, "Иван ест, поет Оля" и т.д.). С помощью генераторов создать словарь, в котором ключами будут имена, а значениями – занятия. Решить задачу в одну строку. Например: "Маша гуляет, Коля работает, дома Ваня, закупается Женя" представить в виде {'Ваня': 'дома', 'Женя': 'закупается', 'Коля': 'работает', 'Маша': 'гуляет'}.
3. Написать скрипт, который заменяет в текстовом файле все слова из определенного перечня на определенные для этих слов слова-заменители. Скрипт принимает через командную строку 2 параметра:
 - первый параметр - имя файла из которого берется текст (файл имеет кодировку UTF8; нет переносов слов на другую строку);
 - второй параметр - имя файла, являющегося файлом CSV (с разделителем ;) содержащего перечень слов для замены и

соответствующих им заменителей (файл не имеет заголовка столбцов; в первом столбце слова подлежащие замене, во втором слова-заменители) (файл имеет кодировку UTF8).

Результат замены слов сохраняется в новом файле с именем совпадающем с исходным файлом с текстом, но имеющим префикс подчеркивание (_). Скрипт должен иметь вид `replace_Ivanov.py` (вместо `Ivanov` - ваша фамилия). Функции связанные с обработкой файлов и заменой в тексте должны находиться в отдельном модуле `modul_Ivanov.py` (вместо `Ivanov` - ваша фамилия).

4. Реализовать функцию `summmulate` для расчета накопленных двойных сумм (квадратов произведений). Функция принимает одно или более числовое значение (количество параметров заранее не определено). На основе этих значений рассчитываются накопленные суммы, которые сохраняются в списке, список возвращается как результат функции. Необязательный булевский параметр `mul` должен позволять заменять суммирование умножением.

5. Создайте класс `Заказ(Order)`, у которого есть свойства `код_товара(code)`, `цена(price)`, `количество(count)` и методы `__init__` и `__str__`. Создайте 2 класса-потомка: `Опт(Opt)` и `Розница(Retail)`. В этих классах создайте методы `__init__`, `__str__` и `сумма_заказа(summa)`, позволяющий узнать стоимость заказа. Для опта стоимость единицы товара составляет 95% от цены, а при покупке более 500 штук – 90% цены. В розницу стоимость единицы товара составляет 100% цены. Стоимость заказа равна произведению цены на количество. Продемонстрируйте работу с классами, создав необходимые объекты и обратившись к их свойствам и методам.

6. Написать функцию-генератор `my_func_2(lst)`, которая принимает объект, поддерживающий итерации с произвольным уровнем вложенности, и возвращает все элементы по одному.

7. С помощью механизма `map/filter/reduce` (хотя бы одна из этих функций должна быть использована в решении) посчитать в тексте количество слов, состоящих не менее, чем из 3-х букв. Слова в тексте разделены пробелами. Написать реализацию в одну строку. Оформить решение в виде функции `my_func_3(text)`, т.е. шаблон таков: строка с `import`, если необходимо `def my_func_3(text): return #` однострочная реализация задания.

8. Написать декоратор с параметром `my_decorator(n)`. Декоратор превращает функцию, возвращающую поддерживающий итерации объект (далее "последовательность"), в функцию-генератор. Если декорируемая функция возвращает что-то другое, а не последовательность, то декоратор должен вернуть этот результат вызова функции без изменений. Проверку объекта можно организовать при помощи условия `import collections if isinstance (item, collections.Iterable)`. Параметром декоратора может быть целое положительное число `n`, тогда получившаяся функция-декоратор должна генерировать по одному значению из последовательности, повторенной `n` раз. Также параметр может принимать строковое значение `'inf'`, тогда функция-декоратор генерирует по одному значению из последовательности, повторенной бесконечное число раз (зацикливает генерирование результата). Подсказка: сначала реализовать случай со значением аргумента `'inf'`, а затем модифицировать для целочисленного значения аргумента.

Примерные вопросы для подготовки к Экзамену, Экзамену

2 семестр

1. Присвоение по ссылке и по значению. Специфика создания объектов и присвоения в Python , особенности Python в связи с распространенностью использования неизменяемых типов.
2. Специфика типизации в языках программирования (различные аспекты типизации). Реализация типизации в Python .
3. Организация ветвлений в Python. Инструкция `if...else`. Множественный выбор.
4. Циклы в Python, работа и устройство цикла `for`, типичное применение `range` и `enumerate` в цикле `for`.
5. Циклы в Python, работа и устройство цикла `while`. Организация бесконечного цикла, аварийное завершение цикла.
6. Списки в Python. Обращение к элементам списка и создание срезов. Обход списка и поиск элементов в списке.
7. Ключевые операции, проводящие к изменению списка и порождающие измененные списки, копирование списков.

8. Словари в Python . Итерирование по словарям, преобразование между словарями и списками в Python. Операции с представлениями словарей.
9. Операции со словарями, учитывающие возможное отсутствие ключа. Операции многоэлементного изменения словарей. Операции поэлементного извлечения из словаря и их использование.
10. Множества в Python. Основные способы создания, получения и изменения значений. Обход множеств. Выполнение основных операций с парой множеств в Python.
11. Кортежи в Python. Отличия кортежей от списков. Распаковка и частичная распаковка кортежей.
12. Выражения генераторы и генераторы списков в Python. Использование условий в генераторах.
13. Функции стандартной библиотеки для работы с контейнерами.
14. Объявление и вызов функции в Python. Параметры функции со значением по умолчанию и комментирование функции.
15. Функция. Получение информации о функции. Способы передачи параметров при вызове функции.
16. Передача переменного количества параметров (именованных и не именованных) в функции Python.
17. Вызов функции с позиционными параметрами, находящимися в списке, и именованными параметрами, находящимися в словаре.
18. Вложенные функции и замыкания, специфика реализации в Python.
19. Функции высшего порядка и декораторы в Python.
20. Концепция map/filter/reduce. Реализация map/filter/reduce в Python и пример их использования.
21. Итераторы в Python: встроенные итераторы, создание собственных итераторов, типичные способы обхода итераторов и принцип их работы. Встроенные функции для работы с итераторами и возможности модуля itertools.

22. Функции генераторы и выражения генераторы: создание и применение в Python.
 23. Синтаксис и семантика обработки исключительных ситуаций в Python. Создание пользовательских исключений и инструкция assert.
 24. Базовые операции для работы с файлами в Python. Использование инструкции with ... as на примере работы с файлами.
 25. CSV файлы. Методы и функции работы с CSV файлами.
 26. Бинарные файлы. Методы и функции работы с бинарными файлами.
 27. Модули в Python и их отличие от скриптов Python. Варианты синтаксиса импорта модуля и объектов модуля.
 28. Применение импортированных объектов. Порядок поиска модулей и специфика их загрузки. Загрузка модулей из глобального репозитория.
- нее 10 объектов) и продемонстрировать созданную защиту. **(20 баллов)**

3 семестр

1. Концепция класса и объекта. Принципы и механизмы ООП.
2. Объявление класса, конструктор, создание объектов и одиночное наследование в Python. Управление доступом к атрибутам класса в Python.
3. Полиморфизм и утиная типизация и проверка принадлежности объекта к классу в языке Python.
4. Методы классов и статические переменные и методы в Python. Специальные методы для использования пользовательских классов со стандартными операторами и функциями.
5. Основные возможности, поддерживаемые функциональными языками программирования. Поддержка элементов функционального программирования в Python.
6. Концепция «функции – граждане первого класса» в языке программирования, поддержка этой концепции в Python. Специфика

лямбда-функций в Python их возможности и ограничения. Типичные сценарии использования лямбда-функций в Python.

7. Глобальные и локальные переменные в функциях на примере Python. Побочные эффекты вызова функций и их последствия.

8. Вложенные функции и замыкания, специфика реализации в Python.

9. Функции высшего порядка и декораторы в Python.

10. Концепция map/filter/reduce. Реализация map/filter/reduce в Python и пример их использования.

11. Итераторы в Python: встроенные итераторы, создание собственных итераторов, типичные способы обхода итераторов и принцип их работы. Встроенные функции для работы с итераторами и возможности модуля itertools.

12. Функции генераторы и выражения генераторы: создание и применение в Python.

13. Специфика массивов, как структур данных. Динамические массивы – специфика работы, сложность операций. Специфика работа с array в Python.

14. Абстрактная структура данных стек и очередь: базовые и расширенные операции, их сложность.

15. Специфика реализации и скорости основных операций в очереди на базе массива и связанного списка.

16. Связанные списки: однонаправленные и двунаправленные – принцип реализации. Сравнение скорости выполнения основных операций в связанных списках и в динамическом массиве.

17. Алгоритм обменной сортировки, сложность сортировки и возможности по ее улучшению.

18. Алгоритм сортировки выбором, сложность сортировки и возможности по ее улучшению.

19. Алгоритм сортировки вставками, его сложность. Алгоритм быстрого поиска в отсортированном массиве. Сложность поиска в отсортированном и не отсортированном массиве.

20. Алгоритм сортировки Шелла, сложность сортировки и возможности по ее улучшению.
 21. Алгоритм быстрой сортировки, сложность сортировки и возможности по ее улучшению.
 22. Алгоритм сортировки слиянием, сложность сортировки.
 23. Реализация двоичных деревьев в виде связанных объектов. Различные реализации рекурсивного обхода двоичных деревьев.
 24. Двоичное дерево поиска – принципы реализации и логика реализации основных операций.
 25. Двоичная куча – принципы реализации и логика реализации основных операций.
 26. Абстрактный тип данных - ассоциативный массив и принцип его реализации на основе хэш-таблиц и хэш-функций.
 27. Общая схема построения хэш-функции и возможная роль в этой схеме хэш-функции multiply-add-and-divide. Принцип работы хэш-функции multiply-add-and-divide.
 28. Полиномиальная хэш-функция – принцип работы, специфика эффективной реализации и специфика применения хэш-функции.
 29. Различные методы разрешения коллизий в хэш-таблицах.
- *Промежуточная аттестация обучающихся проводится в формате тестирования.*

8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

Нормативно-правовые акты:

- не предусмотрены

Основная литература:

1. Чернышев, С.А. Алгоритмы и структуры данных на Python : Учебное пособие / С.А. Чернышев Электрон. дан. Москва : КноРус, 2024 326 с. Режим доступа: book.ru Internet access <https://book.ru/book/949701> ISBN 978-5-406-11683-8. [БИК ID: RU\bookru\bibl\949701]
2. Апанасевич, С. А. Структуры и алгоритмы обработки данных. Линейные структуры [Электронный ресурс] : учебное пособие для вузов / Апанасевич С. А. 2-е изд., стер. Санкт-Петербург : Лань, 2025 136 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/489332> ISBN 978-5-507-53508-8. [БИК ID: RU-LAN-BOOK-489332]
3. Практикум по программированию на языке Python : учебное пособие для студентов, обучающихся по направлениям "Прикладная математика и информатика", "Прикладная информатика", "Информационная безопасность", "Бизнес-информатика" (программа подготовки бакалавра) / Р.И. Горохова, Е.П. Догадина, В.И. Долгов, В.И. Макрушин ; Финуниверситет, Департамент анализа данных и машинного обучения факультета информационных технологий и анализа больших данных Москва : Финуниверситет, 2023 1 файл (6,93 Мб) Свободный доступ из сети Интернет (чтение, печать, копирование) (дата обращения 09.10.2023) + Текст : электронный. [БИК ID: RU\FA\books\137296]

Дополнительная литература:

1. Титов, А. Н. Обработка данных в Python. Основы работы с библиотекой Pandas [Электронный ресурс] : учебно-методическое пособие / Титов А. Н.,Тазиева Р. Ф. Казань : КНИТУ, 2022 116 с. Книга из коллекции КНИТУ - Информатика <https://e.lanbook.com/book/331013> ISBN 978-5-7882-3164-8. [БИК ID: RU-LAN-BOOK-331013]
2. Базаркин, А. Ф. Программирование [Электронный ресурс] / Базаркин А. Ф.,Бакаева О. А.,Вознесенская Н. В.,Шалина О. Н. Саранск : МГПУ им. М. Е. Евсевьева, 2018 174 с. Книга из коллекции МГПУ им. М. Е. Евсевьева - Информатика <https://e.lanbook.com/book/163496>. [БИК ID: RU-LAN-BOOK-163496]

3. Мяготин, А. В. Алгоритмы, структуры данных и численные методы [Электронный ресурс] : учебное пособие / Мяготин А. В. Санкт-Петербург : СПбГУ ГА, 2015 116 с. Книга из коллекции СПбГУ ГА - Информатика <https://e.lanbook.com/book/145579>. [БИК ID: RU-LAN-BOOK-145579]

9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины

1. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/> (<http://library.fa.ru/files/elibfa.pdf>)
2. Электронно-библиотечная система BOOK.RU <http://www.book.ru>
3. Электронно-библиотечная система "Университетская библиотека ОНЛАЙН" <http://biblioclub.ru/>
4. Электронно-библиотечная система издательства "Лань" <https://e.lanbook.com/>
5. Научная электронная библиотека eLibrary.ru <http://elibrary.ru>
6. Информационно-образовательный портал Финуниверситета: <https://org.fa.ru>
7. •Электронные коллекции книг и журналов издательства Springer: <http://link.springer.com/>

10. Методические указания для обучающихся по освоению дисциплины

Лекционные занятия проводятся в соответствии с тематическим планом, при изложении материала рекомендуется использовать презентации в среде PowerPoint программный код из Jupyter Notebook и фрагменты печатных материалов по теме лекции.

В ходе интерактивных занятий следует проводить разбор конкретных примеров программного кода из Jupyter Notebook .

Проведение практических занятий осуществляется в компьютерных классах и включает в себя реализацию всех этапов проектирования и реализации алгоритмов.

Лекционные занятия проводятся в соответствии с тематическим планом, при изложении материала рекомендуется использовать презентации в среде PowerPoint программный код из Jupyter Notebook и фрагменты печатных материалов по теме лекции.

В ходе интерактивных занятий следует проводить разбор конкретных примеров программного кода из Jupyter Notebook.

Проведение практических занятий осуществляется в компьютерных классах и включает в себя реализацию всех этапов проектирования и реализации алгоритмов.

11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем

- Комплект лицензионного программного обеспечения:

1. Python 3.8
2. Jupyter Notebook (Windows 10)
3. Kaspersky
4. Пакет офисных программ

- Современные профессиональные базы данных и информационные справочные системы:

1. Язык программирования Python 3. <https://pythonworld.ru/>
2. Размещение ЭУК: <https://www.campus.fa.ru/>
3. VK-звонки
4. VK Mail

- Сертифицированные программные и аппаратные средства защиты информации

1. не предусмотрены

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

1. **Учебная аудитория** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенная оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), набор демонстрационного оборудования (проектор, экран)

2. **Помещение для самостоятельной работы** обучающихся оснащенное компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа к электронной информационно-образовательной среде Университета.

3. **Компьютерный класс** для проведения учебных занятий, предусмотренных программой, в том числе групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, оснащенный оборудованием и техническими средствами обучения: мебель аудиторная (столы, стулья, доска аудиторная), персональные компьютеры, набор демонстрационного оборудования (проектор, экран)