

**Федеральное государственное образовательное бюджетное
учреждение высшего образования
«Финансовый университет при Правительстве Российской Федерации»
(Финуниверситет)**

Владикавказский филиал Финуниверситета

Кафедра «Корпоративные инфокоммуникационные системы»

УТВЕРЖДАЮ

Директор филиала

Т.А. Хубаев

2026 г.



С.Б. Волошин

Практикум по программированию

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки

09.03.04 Программная инженерия,

ОП «Технологии разработки программного обеспечения»

*Рекомендовано Ученым советом Владикавказского филиала
Финансового университета*

(протокол от « 15 » апреля 2026 г. № 30)

*Одобрено на заседании кафедры «Корпоративные инфокоммуникационные
системы»*

(протокол от « 10 » апреля 2026 г. № 8)

Владикавказ 2026

Содержание

1. Наименование дисциплины	3
2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине.....	3
3. Место дисциплины в структуре образовательной программы	4
4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	4
5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий.....	4
5.1. Содержание дисциплины	4
5.2. Учебно-тематический план	8
5.3. Содержание семинаров, практических занятий.....	9
6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине.....	12
6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	12
6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю	14
7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине.....	34
8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	42
9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	43
10. Методические указания для обучающихся по освоению дисциплины.....	44
11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем	50
11.1 Комплект лицензионного программного обеспечения	50
11.2 Современные профессиональные базы данных, и информационные справочные системы	51
11.3 Сертифицированные программные и аппаратные средства защиты информации	51
12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине.....	51

1. Наименование дисциплины

Дисциплина «Практикум по программированию».

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

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания) соотнесенные с индикаторами достижения компетенции
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.	Знать: объектно-ориентированный язык программирования Python на уровне знания синтаксиса и семантики, основ стандартной библиотеки. Уметь: определять на уровне знания синтаксиса и семантику, стандартные библиотеки языка Python, необходимые для решения прикладных задач.
		2. Анализирует алгоритмы в части производительности, оптимальности, вырабатывает рекомендации для оптимизации алгоритмов программ.	Знать: технологии прикладного программирования, инструментальные средства программирования. Уметь: разрабатывать программы решения задач с использованием прикладного программирования, включая среды высокоуровневого программирования.
		3. Проводит ручное и автоматизированное тестирование программных продуктов по методам черного и белого ящика, составляет набор тестовых случаев.	Знать: особенности создания программного кода, основы проектирования различных видов интерфейса программной системы. Уметь: разрабатывать программный код, ориентироваться в существующем коде, применять знание общепринятых соглашений и политик в области оформления кода, разрабатывать текстовый, программный или графический интерфейс

			программной системы исходя из ее назначения.
--	--	--	--

3. Место дисциплины в структуре образовательной программы

Дисциплина «Практикум по программированию» является дисциплиной Общепрофессионального цикла обязательной части учебного плана образовательной программы «Технологии разработки программного обеспечения» по направлению подготовки 09.03.04 Программная инженерия, профиль «Технологии разработки программного обеспечения».

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

Вид учебной работы по дисциплине	Всего (в з/е и	Семестр 1 (в часах)	Семестр 2 (в часах)	Семестр 3 (в часах)	Семестр 4 (в часах)
Общая трудоемкость дисциплины	8 /288	72	72	72	72
Контактная работа-Аудиторные занятия	64	16	16	16	16
<i>Лекции</i>		-	-	-	-
<i>Семинары, практические</i>	64	16	16	16	16
Самостоятельная работа	224	56	56	56	56
Вид текущего контроля	Четыре проектные работы	Проектная работа	Проектная работа	Проектная работа	Проектная работа
Вид промежуточной аттестации	зачет	зачет	зачет	зачет	зачет

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. Основы языка Python

Обработка числовой информации. Встроенные функции и модули для работы с числами. Реализация числовых алгоритмов с использованием инструкции ветвления и циклов. Обработка текстовой информации.

Функции и методы для работы со строками. Регулярные выражения. Списки. Использование списков для хранения информации. Методы списков. Многомерные списки. Кортежи. Словари. Типовые случаи использования словарей в программах. Методы для работы со словарями. Множества.

Тема 2. Функции и модули

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. Глобальные и локальные переменные. Вложенные функции. Функции высшего порядка. Создание и использование модулей и пакетов. Модули стандартной библиотеки.

Тема 3. Обработка исключений. Работа с файлами средствами языка Python

Понятие исключения. Инструкция try ... except ... else ... finally. Классы встроенных исключений. Инструкции raise и assert. Инструкция with ... as. 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. Использование текстовых файлов в программе. Двоичные файлы. Сохранение объектов в файл. Работа с файлами различных форматов: csv, xlsv.

Тема 4. Объектно-ориентированное программирование на Python

Python как объектно-ориентированный язык программирования. Базовые возможности ООП в Python: создание 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 5. Функциональное программирование на Python

Элементы функционального программирования в Python: функции "граждане первого класса", функции высшего порядка, замыкания, функции без побочных эффектов, рекурсия. неизменяемые структуры данных. Идиомы, распространенные в функциональных языках программирования: итераторы, последовательности, ленивые вычисления. 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 6. Алгоритмы и структуры данных

Динамические массивы. Стеки, очереди, деки. Связные списки. Реализация связных списков на языке Python. Бинарные деревья. Использование бинарных деревьев в прикладных задачах. Двоичное дерево поиска. Асимптотическая оценка сложности алгоритма. Алгоритмы сортировки и поиска. Бинарный поиск. Простые методы сортировки: обменные сортировки, сортировка выбором, сортировка вставками. Сортировка Шелла. Быстрая сортировка. Сортировка слиянием.

Тема 7. Паттерны проектирования

Основные принципы объектно-ориентированного проектирования приложений. Принцип абстракции. Уменьшение зависимости. Основные паттерны проектирования: интерфейс, делегирование. Порождающие шаблоны: фабричный метод, абстрактная фабрика, строитель. Структурные шаблоны: адаптер, мост, декоратор, фасад. Поведенческие шаблоны: цепочка обязанностей, команда, посредник, наблюдатель, состояние, стратегия, посетитель, шаблонный метод.

Тема 8. Программирование графических интерфейсов

Событийно-ориентированное программирование. События и обработчики событий. События мыши и клавиатуры. Основные библиотеки графических интерфейсов в Python: tkinter, PyQt, PyGTK. Главный цикл программы. Основные элементы управления: кнопки, ползунки, поля ввода. Работа с графикой.

Тема 9. Системное программирование на Python

Чтение и запись файлов. Работа с путями в Windows и Linux. Обход папок. Работа с архивами. Работа с офисными форматами. Работа со структурированными данными. Основные форматы хранения данных: CSV, XML, JSON.

Тема 10. Сетевое программирование на Python

Получение и разбор HTML-страниц. Библиотеки HTML-парсинга. Сокеты. Клиент-серверные приложения. Обращение к внешним API. Многопоточность. Библиотеки многопоточности и многопроцессности. Отправка и получение электронных писем.

Тема 11. Тестирование программ на Python

Основные виды тестирования программного обеспечения. Модульное и интеграционное тестирование. Понятие регрессии. Фиксирование и формализация требований. Библиотеки автоматизированного тестирования. Написание автоматических модульных тестов по техническому заданию. Разработка через тестирование. Проектирование через тестирование. Методология TDD.

Тема 12. Документирование и развертывание программ на Python

Основные приемы документирования программного кода. Соглашения о стиле документирования кода. Написание программной документации в формате docstring. Языки форматирования документации: ReST, Markdown. Автоматическая сборка документации с использованием Sphinx. Экспорт документации в популярные форматы.

5.2. Учебно-тематический план

№ п/ п	Наименование тем (разделов) дисциплины	Трудоемкость в часах					Формы текущего контроля успеваемости
		Всего	Контактная работа - Аудиторная работа			Самосто ятельна я работа	
			Общая, в т.ч.:	Лекци и	Семин ары, практи		
1	Тема 1. Основы языка Python	24	4	-	4	20	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
2	Тема 2. Функции и модули	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
3	Тема 3. Обработка исключений. Работа с файлами средствами языка Python	24	4	-	4	20	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
4	Тема 4. Объектно- ориентированное программирован ие на Python	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование), защита проектной работы
5	Тема 5. Функциональное программирован ие на Python	24	4	-	4	20	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
6	Тема 6. Функциональное программирован ие на Python	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
7	Тема 7. Паттерны проектирования	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач

							(программирование), защита проектной работы
8	Тема 8. Программирование графических интерфейсов	24	4	-	4	20	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
9	Тема 9. Системное программирование на Python	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
10	Тема 10. Сетевое программирование на Python	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование), защита проектной работы
11	Тема 11. Тестирование программ на Python	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование)
12	Тема 12. Документирование и развертывание программ на Python	24	6	-	6	18	Опрос, собеседование по домашним заданиям самостоятельной работы, решение задач (программирование), защита проектной работы
В целом по дисциплине		288	64	-	64	224	Согласно учебному плану: четыре проектные работы
Итого в %		100	22	-	100	78	

5.3. Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях,	Формы проведения занятия
Тема 1. Основы языка Python	Решение задач с использованием инструкции	Интерактивная форма: опрос, собеседование по домашним

	ветвления. Решение задач с использованием циклов. Создание и обработка списков. Многомерные списки. Обработка текстовой информации. Использование словарей и множеств. Совместное использование различных типов данных.	заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 2. Функции и модули	Создание и использование функций. Анонимные функции. Функции высшего порядка.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 3. Обработка исключений. Работа с файлами средствами языка Python	Работа с текстовыми файлами. Работа с двоичными файлами. Сохранение объектов в файл.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 4. Объектно-ориентированное программирование на Python	Создание классов и объектов. Наследование и полиморфизм. Специальные методы классов.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов, защита проектной работы
Тема 5. Функциональное программирование на Python	Функции map, filter, reduce, any, all. Декораторы. Итераторы. Функции генераторы.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 6. Функциональное программирование на Python	Создание связанных списков. Использование стеков и очередей при решении задач. Алгоритмы сортировки и поиска.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов

Тема 7. Паттерны проектирования	Создание примеров программных реализаций для распространенных шаблонов проектирования	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов, защита проектной работы
Тема 8. Программирование графических интерфейсов	Создание графического приложения с использованием встроенных возможностей языка и библиотек.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 9. Системное программирование на Python	Автоматизация рутинных административных задач. Создание Web-парсера.	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 10. Сетевое программирование на Python	Создание многопоточного многопользовательского многофункционального сервера на сокетах	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов, защита проектной работы
Тема 11. Тестирование программ на Python	Написание модульных тестов по описанию. Разработка программы по тестам	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов
Тема 12. Документирование и развертывание программ на Python	Документирование программы по описанию и исходному коду. Экспорт документации в формате сайта	Интерактивная форма: опрос, собеседование по домашним заданиям самостоятельной работы, практикум по решению задач по тематике занятия индивидуально с последующим коллективным обсуждением их результатов, защита проектной работы

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
Тема 1. Основы языка Python	Функции модуля math, random, copy. Регулярные выражения. Кортежи. Множества.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook.
Тема 2. Функции и модули	Создание и использование модулей и пакетов.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook
Тема 3. Обработка исключений. Работа с файлами средствами языка Python	Работа с файлами различных форматов: csv, docx, xlsx.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook
Тема 4. Объектно-ориентированное программирование на Python	Специальные методы классов.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook, выполнение проектной работы
Тема 5.	Функции any, all, zip.	Изучение материалов лекций и

Функциональное программирование на Python		литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook
Тема 6. Функциональное программирование на Python	Бинарные деревья. Использование бинарных деревьев в прикладных задачах. Двоичное дерево поиска.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook
Тема 7. Паттерны проектирования	Внедрение зависимости, инверсия управления.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook, выполнение проектной работы
Тема 8. Программирование графических интерфейсов	Создание игр. Библиотека PyGame.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook
Тема 9. Системное программирование на Python	Создание административных скриптов.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook

Тема 10. Сетевое программирование на Python	Создание веб-сервиса.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook, выполнение проектной работы
Тема 11. Тестирование программ на Python	Экстремальное программирование.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook
Тема 12. Документирование и развертывание программ на Python	Непрерывная интеграция.	Изучение материалов лекций и литературы, предложенной преподавателем, поиск и анализ информации, содержащейся в Интернет-ресурсах. Разбор вопросов, отводимых на самостоятельное освоение, выполнение домашних заданий самостоятельной работы, в том числе заданий с использованием Jupyter Notebook, выполнение проектной работы

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

Примерный перечень вопросов для подготовки к опросу

1. Обработка числовой информации.
2. Встроенные функции и модули для работы с числами.
3. Реализация числовых алгоритмов с использованием инструкции ветвления и циклов.
4. Обработка текстовой информации.
5. Функции и методы для работы со строками.
6. Регулярные выражения.
7. Списки. Использование списков для хранения информации.

Методы списков. Многомерные списки. Кортежи.

8. Словари. Типовые случаи использования словарей в программах.
9. Методы для работы со словарями. Множества.
10. Создание и обработка списков.
11. Многомерные списки.
12. Обработка текстовой информации.
13. Использование словарей и множеств.
14. Совместное использование различных типов данных.
15. Создание и использование функций.
16. Анонимные функции.
17. Функции высшего порядка.
18. Работа с текстовыми файлами.
19. Работа с двоичными файлами.
20. Сохранение объектов в файл.
21. Создание классов и объектов.
22. Наследование и полиморфизм.
23. Специальные методы классов.
24. Функции map, filter, reduce, any, all.
25. Декораторы. Итераторы. Функции генераторы.
26. Создание связанных списков.
27. Использование стеков и очередей при решении задач.
28. Алгоритмы сортировки и поиска.
29. Создание примеров программных реализаций для
распространенных шаблонов проектирования.
30. Создание графического приложения с
31. использованием встроенных возможностей языка и библиотек
32. Автоматизация рутинных административных задач.
33. Создание Web-парсера.
34. Создание многопоточного многопользовательского
многофункционального сервера на сокетах.

- 35. Написание модульных тестов по описанию.
- 36. Разработка программы по тестам.
- 37. Документирование программы по описанию и исходному коду.
- 38. Экспорт документации в формате сайта.

Примеры задач

Задача 1.

Условие. Написать программу, которая конвертирует температуру между шкалами: Цельсия ($^{\circ}\text{C}$), Фаренгейта ($^{\circ}\text{F}$), Кельвина (K).

Требования

- Пользователь вводит значение и исходную шкалу (например, 25 C).
- Программа выводит эквиваленты в двух других шкалах.
- Формулы:
 - $^{\circ}\text{F} = (^{\circ}\text{C} \times 9/5) + 32$;
 - $\text{K} = ^{\circ}\text{C} + 273.15$.
- Обработать некорректный ввод (неверные обозначения шкал, буквы вместо чисел).
- Циклический ввод до команды exit.

Задача 2.

Условие. Реализовать алгоритм поиска всех простых чисел в диапазоне от 1 до N (вводится пользователем).

Требования

- Использовать решето Эратосфена или проверку делителей.
- Вывести список простых чисел и их количество.
- Оптимизировать код (например, проверять делители до \sqrt{N}).
- Обработать $N < 2$ (выводить сообщение).
- Время выполнения для $N = 10\,000$ не должно превышать 5 секунд.

Задача 3.

Условие. Создать программу, которая генерирует случайный пароль заданной длины из букв (верхний/нижний регистр), цифр и специальных символов.

Требования

- Запросить у пользователя длину пароля (минимум 6 символов).
- Включить хотя бы один символ из каждой категории:
 - буквы (A-Z, a-z);
 - цифры (0-9);
 - спецсимволы (!@#\$%^&*).
- Вывести сгенерированный пароль.
- Предоставить опцию генерации нескольких паролей (например, 3 пароля за раз).

Задача 4.

Условие. Написать скрипт, который читает CSV-файл с данными о студентах (столбцы: id, name, grade) и выводит:

- среднее значение оценок;
- список студентов с оценкой выше средней;
- количество студентов в каждой оценочной категории (например, 5, 4, 3, 2).

Требования

- Использовать модуль csv или pandas (на выбор).
- Проверить существование файла и корректность формата.
- Обработать пустые значения в столбце grade.
- Результаты вывести в консоль в читаемом виде.
- Добавить возможность указать путь к файлу через аргумент командной строки (например, `python parser.py data.csv`).

Примеры домашних заданий самостоятельной работы

Задание 1.

Используя программу Jupyter Notebook составьте код на Python, определяющий количество точек на плоскости, находящихся на заданном расстоянии от начала координат. Используйте библиотеку math.

Задание 2.

Установите пакет Anaconda запустите программу Jupyter Notebook составьте тестовый код на Python. Выберите библиотеку Python для разработки системы управления складом малого предприятия.

Реализуйте программу на Python, которая реализует систему управления складом малого предприятия.

Примерные задания проектной работы (семестр 1)

Задание

1. Реализовать программу, с которой можно играть в логическую игру «Быки и коровы» (описание правил игры: <http://робомозг.рф/Articles/BullsAndCowsRules>). Программа загадывает число, пользователь вводит очередной вариант отгадываемого числа, программа возвращает количество быков и коров и в случае выигрыша игрока сообщает о победе и завершается. Сама программа НЕ ходит, т.е. не пытается отгадать число загаданное игроком.

Взаимодействие с программой производится через консоль, при запросе данных от пользователя программа сообщает, что ожидает от пользователя и проверяет корректность ввода.

2. Реализовать программу, при помощи которой 2 игрока могут играть в «Крестики-нолики» на поле 3 на 3. Взаимодействие с программой производится через консоль. Игровое поле изображается в виде трех текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты новой отметки на поле) и проверяет корректность ввода. Программа должна уметь автоматически определять, что

партия окончена, и сообщать о победе одного из игроков или о ничьей. Сама программа НЕ ходит, т.е. не пытается ставить крестики и нолики с целью заполнить линию.

3. Реализовать программу, при помощи которой 2 игрока могут играть в игру «Супер ним». Правила игры следующие. На шахматной доске в некоторых клетках случайно разбросаны фишки или пуговицы. Игроки ходят по очереди. За один ход можно снять все фишки с какой-либо горизонтали или вертикали, на которой они есть. Выигрывает тот, кто заберет последние фишки. (описание правил игры: <https://www.iqfun.ru/articles/super-nim.shtml>)

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты новой отметки на поле) и проверяет корректность ввода. Программа должна уметь автоматически определять, что партия окончена, и сообщать о победе одного из игроков. Сама программа НЕ ходит, т.е. не пытается выбирать строки или столбцы с целью победить в игре.

4. Реализовать программу, с которой можно играть в игру «19». Правила игры следующие. Нужно выписать подряд числа от 1 до 19: в строчку до 9, а потом начать следующую строку, в каждой клетке по 1 цифре (не числу (см пример по ссылке)). Затем игроку необходимо вычеркнуть парные цифры или дающие в сумме 10. Условие - пары должны находиться рядом или через зачеркнутые цифры по горизонтали или по вертикали. После того как все возможные пары вычеркнуты, оставшиеся цифры переписываются в конец таблицы. Цель - полностью вычеркнуть все цифры. (описание правил игры: <http://podelki-fox.ru/igry-dlya-detey-na-bumage-s-chislami/>)

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде трех текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя

программа сообщает, что ожидает от пользователя (в частности, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять, что нужно выписать новые строки с цифрами и то, что партия окончена. Сама программа НЕ ходит, т.е. не пытается выбирать пары цифр с целью окончить игру.

5. Реализовать программу, при помощи которой 3 игрока могут играть в игру «Лоскутное одеяло». Правила игры следующие. На поле, имеющем размер 4 на 5 клеток за один ход каждый игрок должен заполнить одну клетку своим символом. Игрок старается, чтобы его символы были как можно дальше друг от друга. В ходе игры ведется подсчет очков: за каждое соседство клеток с одинаковыми символами игроку, владельцу символа добавляется одно штрафное очко. Соседними считаются клетки, имеющие общую сторону или расположенные наискосок друг от друга. Выигрывает тот, у кого в конце игры меньше всего штрафных очков.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 4 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять количество штрафных очков и окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается заполнять клетки символами с целью выиграть игру.

6. Реализовать программу, при помощи которой 2 игрока могут играть в игру «Клондайк». Правила игры следующие. Игра ведётся на игровом поле размером 10 на 10 клеток. Игроки по очереди выставляют в любую свободную клетку по отметке, и тот игрок, после чьего хода получилась цепочка длиной хотя бы в 3 отметке, проигрывает. При этом в цепочке считаются как свои отметки, так и отметки соперника, у игровых фишек как бы нет хозяина. Цепочка - это ряд фишек, следующая фишка в

котором примыкает к предыдущей с любого из 8-ми направлений. (описание правил игры: <https://www.iqfun.ru/printable-puzzles/klondike-igra.shtml>)

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 10 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается ставить в клетки отметки с целью выиграть игру.

7. Реализовать программу, при помощи которой 2 игрока могут играть в игру «Максит». Правила игры следующие. В клетках квадрата 3 на 3 пишутся случайные числа из диапазона от 1 до 9. Начинаящий выбирает любое понравившееся ему число и вычеркивает его, прибавляя к своей сумме. Второй игрок может выбрать любое из оставшихся чисел того столбца, в котором первый игрок делал свой предыдущий ход. Он тоже вычеркивает выбранное число, прибавляя его к своей сумме. Первый игрок далее поступает аналогично, выбирая число-кандидата из той строки, в которой второй игрок ходил перед этим. Может так случиться, что у какого-то игрока не будет хода. Тогда его соперник продолжает игру, делая ход в той же строке (для первого игрока) или в том же столбце (для второго игрока), что и до этого. Игра заканчивается, когда оба играющих не имеют ходов. Результат определяется по набранным суммам, у кого она больше, тот и выиграл. При равенстве сумм фиксируется ничья. (описание правил игры: <https://www.iqfun.ru/articles/maxit.shtml>).

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 3 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь

автоматически определять сумму очков каждого из игроков и окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается вычеркивать числа с целью выиграть игру.

8. (*) Реализовать программу, при помощи которой 2 игрока могут играть в игру «Мостики». Правила игры следующие. В ходе игры каждый из игроков старается построить мост с одного своего берега на другой по камням, образующим массив 4 на 5 (4 камня вдоль берега игрока и 5 камней между берегами). У первого игрока - крестики в качестве камней и берега крестиков (левый и правый край поля), у второго игрока – нолики и берега ноликов (верхний и нижний край поля). Игру можно начинать в любой точке поля. За один ход игрок может соединить два своих соседних камня вертикальным или горизонтальным мостиком (обозначаются в текстовом режиме символами «-» и «|»). Мосты первого и второго игрока пересекаться не должны. Выигрывает тот, кто построит непрерывный мост с одного своего берега на другой. (описание правил игры: <https://www.7ya.ru/article/Chem-zanyat-rebenka-13-igr-na-liste-bumagi-so-slovami-kartinkami/>)

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 9 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна уметь автоматически определять недопустимые ходы (приводящие к пересечению мостов соперников) и окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается строить мосты с целью выиграть игру.

9. (*) Реализовать программу, с которой можно играть в игру «Морской бой». Программа автоматически случайно расставляет на поле размером 10 на 10 клеток: 4 1-палубных корабля, 3 2-палубных корабля, 2 3-палубных корабля и 1 4-х палубный. Между любыми двумя кораблями по

горизонтالي и вертикали должна быть как минимум 1 незанятая клетка. Программа позволяет игроку ходить, производя выстрелы. Сама программа НЕ ходит т.е. не пытается топить корабли расставленные игроком.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 10 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (в частности, координаты очередного «выстрела») и проверяет корректность ввода. Программа должна уметь автоматически определять потопление корабля и окончание партии и сообщать об этих событиях.

10. (*) Реализовать программу, с которой можно играть в игру «Пятнашки». Правила игры следующие.

Головоломка представляет собой 15 квадратных костяшек с числами от 1 до 15. Все костяшки заключены в квадратную коробку (поле) размером 4 на 4. При размещении костяшек в коробке остается одно пустое место, которое можно использовать для перемещения костяшек внутри коробки. Цель игры - упорядочить размещение чисел в коробке, разместив их по возрастанию слева направо и сверху вниз, начиная с костяшки с номером 1 в левом верхнем углу и заканчивая пустым местом в правом нижнем углу коробки.

Взаимодействие с программой производится через консоль. Игровое поле изображается в виде 4 текстовых строк и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например, координаты очередного хода) и проверяет корректность ввода. Программа должна считать количество сделанных ходов, уметь автоматически определять недопустимые ходы, окончание партии и ее победителя.

Сама программа НЕ ходит, т.е. не пытается упорядочить костяшки с целью выиграть игру.

Примерные задания проектной работы (семестр 2)

Задание

Базовая часть:

Написать калькулятор для строковых выражений вида '<число> <операция> <число>', где <число> - не отрицательное целое число меньшее 100, записанное словами, например "тридцать четыре", <арифметическая операция> - одна из операций "плюс", "минус", "умножить". Результат выполнения операции вернуть в виде текстового представления числа. Пример `calc("двадцать пять плюс тринадцать")` -> "тридцать восемь"

Оформить калькулятор в виде функции, которая принимает на вход строку и возвращает строку.

1. Реализовать поддержку операции деления и остатка от деления и работу с дробными числами (десятичными дробями). Пример: `calc("сорок один и тридцать одна сотая разделить на семнадцать")` -> "два и сорок три сотых". Обращать дробную часть до тысячных включительно, если при делении получаются числа с меньшей дробной частью выполнять округление до тысячных. *Сложность: 2*

2. Расширение задания 1. Реализовать поддержку десятичной дробной части до миллионных долей включительно. Реализовать корректный вывод информации о периодической десятичной дроби (период дроби вплоть до 4х десятичных знаков). Пример: `calc("девятнадцать и восемьдесят две сотых разделить на девяносто девять")` -> "ноль и двадцать сотых и ноль два в периоде ". *Сложность 3*

3. Реализовать текстовый калькулятор для выражения из произвольного количества операций с учетом приоритета операций. Пример: `calc("пять плюс два умножить на три минус один")` -> "десять". *Сложность 3*

4. Расширение задания 3. Добавить поддержку приоритета операций с помощью скобок. Пример: `calc("скобка открывается пять плюс два скобка закрывается умножить на три минус один")` -> "двадцать". *Сложность 3*

5. Добавить возможность использования отрицательных чисел. Пример: `calc("пять минус минус один")` -> "шесть". *Сложность 1*

6. Добавить возможность оперировать с дробями (правильными и смешанными). Реализовать поддержку сложения, вычитания и умножения, дробей. Результат операций не должен представлять неправильную дробь, такие результаты нужно превращать в смешанные дроби. Пример: `calc("один и четыре пятых плюс шесть седьмых ")` -> "два и двадцать три тридцать пятых". *Сложность 3*

7. Расширение задания 6. Добавить автоматическое сокращение дроби в ответе. Пример: `calc("одна шестая умножить на две третьих")` -> "одна девятая". *Сложность 1*

8. Расширение задания 1. Добавить операции возведения в степень и тригонометрические операции синус, косинус, тангенс и константу π . Допускается как минимум одна из этих функций в выражении с обычными операциями. Пример: `calc("два в степени четыре")` -> "шестнадцать". Пример: `calc("синус от пи разделить на четыре")` -> "ноли и семьсот семь тысячных". *Сложность 1 или 2*

9. Добавить комбинаторные операции перестановки, размещения и сочетания. Пример: `calc("размещений из трех по два")` -> "шесть". *Сложность 1 или 2*

10. Диагностировать ошибки: неправильную запись числа; неправильную последовательность чисел и операций; (задание 1) деление на ноль; (задание 3) неправильную последовательность чисел и операций; (задание 4) некорректный баланс и вложенность скобок; (задание 6) некорректную запись числа. *Сложность 1 или 2*

Примерные задания проектной работы (семестр 3)

Задание

Базовая часть (выполняется всеми самостоятельно!):

На базе модулей: `csv`, `pickle` и прямой работы с файлами реализовать следующий базовый функционал:

1. функций **load_table**, **save_table** по загрузке/сохранению табличных данных во внутреннее представление модуля/из внутреннего представления модуля:

- файла формата csv (отдельный модуль с **load_table**, **save_table** в рамках общего пакета)

- файла формата pickle (отдельный модуль с **load_table**, **save_table** в рамках общего пакета), модуль использует структуру данных для представления таблицы, удобную автору работы.

- текстового файла (только функция **save_table** сохраняющая в текстовом файле представление таблицы, аналогичное выводу на печать с помощью функции **print_table()**).

Примечание: внутреннее представление может базироваться на словаре, где по разным ключам хранятся ключевые «атрибуты» таблицы, а значения таблицы хранятся в виде вложенных списков. Студент может выбрать другое внутреннее представление таблицы (согласовав его с преподавателем), в том числе, студенты знакомые с ООП на Python, могут реализовать собственный класс для таблицы.

При определении api модулей максимально полно использовать возможности сигнатур функций на Python (значения по умолчанию, упаковка/распаковка, в т.ч. именованных параметров, возвращение множественных значений), интенсивно выполнять проверки и возбуждать исключительные ситуации.

2. модуля с базовыми операциями над таблицами:

- **get_rows_by_number(start, [stop], copy_table=False)** – получение таблицы из одной строки или из строк из интервала по номеру строки. Функция либо копирует исходные данные, либо создает новое представление таблицы, работающее с исходным набором данных (**copy_table=False**), таким образом изменения, внесенные через это представления будут наблюдаться и в исходной таблице.

– **get_rows_by_index(val1, ... , copy_table=False)** – получение новой таблицы из одной строки или из строк со значениями в первом столбце, совпадающими с переданными аргументами **val1, ... , valN**. Функция либо копирует исходные данные, либо создает новое представление таблицы, работающее с исходным набором данных (**copy_table=False**), таким образом изменения, внесенные через это представления будут наблюдаться и в исходной таблице.

– **get_column_types(by_number=True)** – получение словаря вида *столбец:тип_значений*. Тип значения: int, float, bool, str (по умолчанию для всех столбцов). Параметр **by_number** определяет вид значения столбец – целочисленный индекс столбца или его строковое представление.

– **set_column_types(types_dict, by_number=True)** – задание словаря вида *столбец:тип_значений*. Тип значения: int, float, bool, str (по умолчанию для всех столбцов). Параметр **by_number** определяет вид значения столбец – целочисленный индекс столбца или его строковое представление.

– **get_values(column=0)** – получение списка значений (типизированных согласно типу столбца) таблицы из столбца либо по номеру столбца (целое число, значение по умолчанию 0, либо по имени столбца)

– **get_value(column=0)** – аналог **get_values(column=0)** для представления таблицы с одной строкой, возвращает не список, а одно значение (типизированное согласно типу столбца).

– **set_values(values, column=0)** – задание списка значений **values** для столбца таблицы (типизированных согласно типу столбца) либо по номеру столбца (целое число, значение по умолчанию 0, либо по имени столбца).

– **set_value(column=0)** – аналог **set_values(value, column=0)** для представления таблицы с одной строкой, устанавливает не список значений, а одно значение (типизированное согласно типу столбца).

– **print_table()** – вывод таблицы на печать.

3. Для каждой функции должно быть реализована генерация не менее одного вида исключительных ситуаций.

Индивидуальные задания:

1. В `load_table` реализовать `load_table(file1, ...)` – поддержку загрузки таблицы, разбитой на несколько файлов (произвольное количество файлов) (для форматов `csv` и `pickle`). В случае несоответствия структуры столбцов файлов вызывать исключительную ситуацию. *Сложность 1*

2. *Расширение задания 1.* В `save_table` реализовать поддержку сохранения таблицы в разбитой на несколько файлов (произвольное количество файлов) по параметру `max_rows`, определяющему максимальное количество строк в файле. Файлы `csv` и `pickle`, полученные с помощью `save_table` должны быть совместимы с `load_table` из задания 1. *Сложность 1*

3. Реализовать функцию `concat(table1, table2)` и `split(row_number)` склеивающую две таблицы или разбивающую одну таблицу на 2 по номеру строки. *Сложность 1*

4. Реализовать автоматическое определение типа столбцов по хранящимся в таблице значениям. Оформить как отдельную функцию и встроить этот функционал как опцию работы функции `load_table`. *Сложность 1 или 2*

5. Реализовать поддержку дополнительного типа значений «дата и время» на основе модуля `datetime`. *Сложность 1 или 2*

6. Добавить набор функций `add`, `sub`, `mul`, `div`, которые обеспечат выполнение арифметических операций для столбцов типа `int`, `float`, `bool`. Продумать сигнатуру функций и изменения в другие функции, которые позволят удобно выполнять арифметические операции со столбцами и присваивать результаты вычислений. Реализовать реагирование на некорректные значения с помощью генерации исключительных ситуаций. *Сложность 2*

7. По аналогии с п. 6 реализовать функции `eq (==)`, `gr (>)`, `ls (<)`, `ge (>=)`, `le (<=)`, `ne (==)`, которые возвращают список булевских значений

длинной в количество строк сравниваемых столбцов. Реализовать функцию `filter_rows (bool_list, copy_table=False)` – получение новой таблицы из строк для которых в `bool_list` (длинной в количество строк в таблице) находится значение `True`. *Сложность 3*

8. Реализовать функцию `merge_tables(table1, table2, by_number=True)`: в результате слияния создается таблица с набором столбцов, соответствующих объединенному набору столбцов исходных таблиц. Соответствие строк ищется либо по их номеру (`by_number=True`) либо по значению индекса (1й столбец). При выполнении слияния возможно множество конфликтных ситуаций. Автор должен их описать и определить допустимый способ реакции на них (в т.ч. через дополнительные параметры функции и инициацию исключительных ситуаций). *Сложность 2*

9. Реализовать полноценную поддержку значения `None` в незаполненных ячейках таблицы. Должно работать при загрузке ячеек с пропусками значений, при операциях приводящих к появлению пустых ячеек, при работе с `get` и `set` операциями. *Сложность 1 или 2*

Примерные задания проектной работы (семестр 4)

Задание

Базовая часть (выполняется всеми самостоятельно!):

Реализовать программу, которая позволяет играть в шахматы на компьютере в консольном режиме.

Цель задания

Разработать консольную программу для игры в шахматы, обеспечивающую взаимодействие двух игроков через текстовый интерфейс. Программа должна корректно отображать игровое поле, принимать и проверять ходы, вести учёт количества сделанных ходов.

Базовая часть (обязательна для всех)

Реализуйте программу со следующими функциями:

1. Визуализация игрового поля

- Поле отображается в текстовом режиме как 8 строк (по количеству рядов шахматной доски).
- Сверху добавляются строки с буквенными обозначениями столбцов (a–h), снизу — с цифровыми обозначениями строк (1–8);
- После каждого хода поле перерисовывается с актуальным состоянием фигур.

Пример отображения поля (упрощённый), рисунок 1.



































	A	B	C	D	D	F	G	H	
8									8
7									7
6									6
5									5
4									4
3									3
2									2
1									1
	A	B	C	D	D	F	G	H	

Рис. 1 Пример изображения шахматного поля

(Здесь ,  и т. д. — символы фигур; · — пустая клетка; буквы и цифры — обозначения столбцов и строк.)

Требования к реализации

- Язык программирования: Python (или иной по согласованию с преподавателем).
- Интерфейс: консольный (без графического окна).
- Код должен быть структурирован (функции/классы для логики игры, отображения, проверки ходов).
- Обработайте ошибки ввода (некорректные координаты, попытки сходить не своей фигурой и т. п.) с понятными сообщениями.
- Программа не завершается до явного выхода (например, по команде quit или после мата/пата — если эти правила добавлены).

2. Взаимодействие с игроками

- Программа поочерёдно запрашивает ходы у игроков за белых и чёрных;
- Для каждого хода программа явно сообщает, что ожидает от пользователя (например: «Ход белых. Укажите позицию фигуры (например, e2)», «Укажите целевую позицию (например, e4)»);
- Ввод проверяется на корректность:
- формат записи (допустимы только обозначения вида буква+цифра, например e2);
- принадлежность клетки игровому полю (координаты в диапазоне a1–h8);
- наличие фигуры нужного цвета на исходной позиции;
- соответствие хода правилам шахмат для данной фигуры.

3. Учёт ходов

- Программа ведёт счётчик общего количества сделанных ходов (суммарно за обе стороны);
- После каждого корректного хода счётчик увеличивается на 1;
- Текущее количество ходов отображается в интерфейсе (например, в заголовке или внизу поля).

4. Ограничения

- Программа **не делает ходов автоматически** — она лишь принимает и проверяет вводы игроков;
- Поддержка сложных правил (рокировка, взятие на проходе, превращение пешки, проверка на мат/пат) **не требуется** в базовой части (их можно вынести в дополнительные задания).

Индивидуальные задания:

Справка о шахматной нотации:

- Общая информация о шахматной нотации записи партий:

https://ru.wikipedia.org/wiki/Шахматная_нотация

- Партии в полной нотации: бесплатная база (для открытия партий нужно зарегистрироваться на ресурсе) записей партий в шахматной нотации (полной): <http://www.chessebook.com/openings.php?lan=ru&pa=pa> (для получения файлов с записью партий копируйте текст понравившихся партий в текстовый файл, скопированный текст не подвергать дополнительному редактированию и сохранить в файл).

- Партии в сокращенной нотации берем из обсуждений на kasparovchess.crestbook.com, например, из этой ветки: <http://kasparovchess.crestbook.com/threads/8210/> (для получения файлов с записью партий копируйте текст понравившихся партий, расположенных справа от блока с доской, в текстовый файл, скопированный текст не подвергать дополнительному редактированию (он во многих нюансах будет отличаться от партий с [chessebook.com](http://www.chessebook.com), так и должно быть) и сохранять файл).

1. Реализовать чтение записи шахматной партии из выбранного пользователем файла в полной нотации. После чтения должна быть возможность двигаться вперед и назад по записи партии (с соответствующим изменением на поле). Должна быть возможность в выбранной позиции перейти из режима просмотра партии в обычный режим игры.

Протестировать не менее чем на 20 реальных партиях с сайта.

Сложность 2

2. Реализовать чтение записи шахматной партии из выбранного пользователем файла в сокращенной нотации. После чтения должна быть возможность двигаться вперед и назад по записи партии (с соответствующим изменением на поле). Должна быть возможность в выбранной позиции перейти из режима просмотра партии в обычный режим игры.

Протестировать не менее чем на 20 реальных партиях с сайта.

Сложность 3 (если пункты 1 и 2 совместно, то суммарная сложность 4)

3. Реализовать возможность записи разыгрываемой шахматной партии в текстовый файл в полной (сокращенной, если студент выполнял

задание 2) нотации. Записать партию можно на любом ходу, с историей всей партии с самого начала. Записанная партия должна корректно воспроизводиться в режиме чтения записи партии. *Сложность 2*

4. Реализовать возможность «отката» ходов. С помощью специальной команды можно возвращаться на ход (или заданное количество ходов) назад вплоть до начала партии. *Сложность 1*

5. Реализовать функцию подсказки выбора новой позиции фигуры: после выбора фигуры для хода функция визуально на поле показывает поля доступные для хода или фигуры соперника, доступные для взятия, выбранной фигурой. *Сложность 1*

6. Реализовать функцию подсказки угрожаемых фигур: она возвращает информацию о том, какие фигуры ходящего игрока сейчас находятся под боем (т.е. могут быть взяты соперником на следующий ход) и визуально выделяет их на поле. Функция отдельно указывает на наличие шаха королю. *Сложность 1*

7. Автоматически определять мат (правила определения: [https://ru.wikipedia.org/wiki/Мат_\(шахматы\)](https://ru.wikipedia.org/wiki/Мат_(шахматы))). *Сложность 2*

8. Реализовать поддержку выполнения рокировки по всем шахматным правилам (в базовой версии поддержка рокировки не обязательна). Правила рокировки см.: <https://ru.wikipedia.org/wiki/Рокировка> . *Сложность 1*

9. Реализовать поддержку для пешки сложных правил: «взятие на проходе» и замены на других фигуру при достижении крайней горизонтали (в базовой версии их поддержка не обязательна, но возможность первого хода на одну или две горизонтали - обязательно). Подробнее о правилах см.: https://ru.wikipedia.org/wiki/Правила_шахмат . *Сложность 1*

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях кафедры.

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

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

Примерные вопросы для подготовки к зачету (семестр 1)

1. Функции модуля math.
2. Инструкция if...else.
3. Инструкция цикла while.
4. Инструкция цикла for.
5. Функция range.
6. Строки. Операции над строками (+, *, in, доступ по индексу [], получение среза). Функция len.
7. Методы строк: strip, find, count, replace.
8. Списки в Python. Создание: создание пустого списка, методы append, split, функция list. Генераторы списков.
9. Создание копии списка (срезы, функции list и deepcopy).
10. Основные операции над списками (+, *, in, доступ по индексу [], получение среза). Перебор элементов списка.
11. Добавление и удаление элементов списка (методы append, insert, pop, remove). Методы reverse, join. Функция map.
12. Сортировка списков. Параметры метода sort: key, reverse.

- 13.Кортежи. Создание. Создание пустого кортежа и кортежа из одного элемента. Операции (+, *, in, доступ по индексу [], получение среза).
- 14.Функции tuple, len.
- 15.Распаковка последовательности.
- 16.Словари. Создание словаря. Функции dict, zip.
- 17.Операции над словарями ([], in, del). Функция len.
- 18.Перебор элементов словаря. Методы get, clear, copy, keys, values.
- 19.Множества. Создание. Функции set, len.
- 20.Операции над множествами: in, |, &, -, ^, <=, >=, <, >, ==. Методы add, discard.
- 21.Создание и вызов функции.
- 22.Передача аргументов в функцию. Необязательные параметры функций. Функции в качестве аргументов.
- 23.Глобальные и локальные переменные.
- 24.Анонимные функции.
- 25.Создание и использование модулей. Инструкции import и from.
- 26.Пакеты. Использование пакетов.

Примерные вопросы для подготовки к зачету (семестр 2)

1. Инструкция try ... except ... else ... finally.
2. Инструкции raise и assert.
3. Инструкция with ... as.
4. Классы встроенных исключений.
5. Работа с текстовыми файлами. Открытие файла (функция open) и закрытие файла (метод close). Чтение текстового файла (методы read, readline, readlines). Перебор строк файла в цикле for. Запись в текстовый файл (метод write, функция print с параметром file).
6. Сохранение объектов в файл (функции dump и load модуля pickle).
7. Понятие класса и объекта. Определение класса и создание экземпляра класса.

8. Методы класса. Параметр self. Статические методы. Закрытые методы.
9. Атрибуты класса и экземпляра класса. Доступ к атрибуту. Закрытые атрибуты.
10. Свойства. Создание и использование свойства.
11. Наследование. Базовый и производный классы. Переопределение методов.
12. Специальные методы. Перегрузка операторов.
13. Функции map, filter, reduce, any, all.
14. Декораторы функций.
15. Итераторы.
16. Функции-генераторы.
17. Массивы. Использование массивов.
18. Стеки. Использование стеков. Реализация стека.
19. Очереди. Использование очереди. Реализация очереди.
20. Связные списки. Использование связанных списков. Реализация связанных списков.
21. Бинарные деревья. Использование бинарных деревьев. Реализация бинарных деревьев.
22. Бинарный поиск.
23. Обменные сортировки.
24. Сортировка Шелла.
25. Быстрая сортировка.

Примерные вопросы для подготовки к зачету (семестр 3)

1. Основные принципы объектно-ориентированного проектирования.
2. Шаблон проектирования: интерфейс
3. Шаблон проектирования: делегирование.
4. Шаблон проектирования: фабричный метод
5. Шаблон проектирования: абстрактная фабрика
6. Шаблон проектирования: строитель

7. Шаблон проектирования: адаптер
8. Шаблон проектирования: мост
9. Шаблон проектирования: декоратор
10. Шаблон проектирования: фасад
11. Шаблон проектирования: цепочка обязанностей
12. Шаблон проектирования: команда
13. Шаблон проектирования: посредник
14. Шаблон проектирования: наблюдатель
15. Шаблон проектирования: состояние
16. Шаблон проектирования: стратегия
17. Шаблон проектирования: посетитель
18. Шаблон проектирования: шаблонный метод.
19. Событийно-ориентированное программирование.
20. События и обработчики событий. Виды событий.
21. События мыши и клавиатуры.
22. tkinter, PyQt, PyGTK – особенности и отличия.
23. Главный цикл программы.
24. Основные элементы управления: кнопки, ползунки, поля ввода.
25. Работа с графикой.
26. Работа с путями в Windows и Linux.
27. Основные форматы хранения данных: CSV, XML, JSON.

Примерные вопросы для подготовки к зачету (семестр 4)

1. Получение и разбор HTML-страниц.
2. Библиотеки HTML-парсинга.
3. Сокеты.
4. Клиент-серверные приложения.
5. Обращение к внешним API.
6. Многопоточность.
7. Библиотеки многопоточности и многопроцессности.

8. Отправка и получение электронных писем.
9. Основные виды тестирования программного обеспечения.
10. Модульное и интеграционное тестирование.
11. Понятие регрессии и регрессионного тестирования.
12. Библиотеки автоматизированного тестирования.
13. Разработка через тестирование.

Примерные задания для подготовки к зачету (семестр 1)

1. Чему будет равно значение переменной A после выполнения оператора:
 $A = [i+10 \text{ for } i \text{ in range}(5,0,-1)]$
2. Имеется строка $S = "1234567890"$. Чему равно значение S1, если
 $S1 = S[1:2] + S[4: -1]$
4. Используя генератор словарей, для заданного натурального числа n создайте словарь D, в котором будет ровно n элементов. Элементами словаря являются $\{ 's1': 10, 's2': 20, 's3': 30, \dots \}$, т.е. значение равно номеру элемента (нумерация с 1), умноженному на 10, а ключ состоит из символа 's', к которому дописан номер.

Примерные задания для подготовки к зачету (семестр 2)

1. Имеется список L вида $[[1, 3, 65], [11, 5, 6], [15, 33, 11]]$. Отсортируйте список по возрастанию последнего значения элемента.
2. После выполнения приведенного кода будет напечатано ...

```
def data(d=[]):  
    d.append(1)  
    return d  
a=data()  
b=data()  
c=data([3])  
print(b,c)
```

 - a) [1] [3]
 - b) [2] [4]
 - c) [1,1] [3,1]
 - d) [1,2] [3,4]

е) Будет выведено сообщение об ошибке

3. После выполнения приведенного кода будет напечатано ...

```
class Class1:
    def __init__(self, n):
        self._x = n

    def gx(self):
        return self._x+2

x = property(gx)

c=Class1(2)
print(c.x)
```

- a) 2
- b) 4
- c) None
- d) Возникнет ошибка

Примерные задания для подготовки к зачету (семестр 3)

1. Приведите пример приватного IP-адреса

- a. 196.168.0.1
- b. 127.0.0.10
- c. 87.250.250.242
- d. 173.194.73.113

2. Приведите пример публичного IP-адреса

- a. 8.8.8.8
- b. 127.0.0.1
- c. 196.168.0.1
- d. 10.38.51.16

3. Соглашение о порядке и способе связи между компьютерами это:

- a. сетевой протокол
- b. сетевой интерфейс
- c. коммутация
- d. адресация

4. Как называется самый распространенный стандарт на архитектуру локальных сетей на основе кабельного соединения

- a. Ethernet
- b. Wi-Fi
- c. TCP
- d. GlobalNet X

Примерные задания для подготовки к зачету (семестр 4)

1. Приведите пример стандарта из серии IEEE 802
 - a. Wi-Fi
 - b. витая пара
 - c. POSIX
 - d. USB

2. Назовите команду, отображающую основную информацию о сетевых подключениях в ОС Linux
 - a. ipconfig
 - b. ifconfig
 - c. netstat
 - d. traceroute

3. Назовите команду, отображающую основную информацию о сетевых подключениях в ОС Windows
 - a. ifconfig
 - b. ipconfig
 - c. netstat
 - d. tracert

4. Назовите пример сетевой топологии
 - a. звезда
 - b. снежинка
 - c. лабиринт
 - d. круговая

Примеры оценочных средств для проверки индикаторов достижения компетенций, формируемых дисциплиной

Код и наименование компетенции	Наименование индикатора достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции	Типовые контрольные задания
ОПК-6 Способен разрабатывать алгоритмы и программы, пригодные для практического	1. Разрабатывает алгоритмы решения простых информационных задач и выражает их	Знать: объектно-ориентированный язык программирования Python на уровне знания синтаксиса и семантики, основ стандартной	Вопросы: 1. Глобальные и локальные переменные. Создание и обработка списков. Работа с текстовыми файлами. Использование словарей и множеств. Совместное использование различных типов

о использовании, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов	на языке программирования.	библиотеки. Уметь: определять на уровне знания синтаксис и семантику, стандартные библиотеки языка Python, необходимые для решения прикладных задач.	данных. 2. Создание и использование функций. Создание классов и объектов Задача Напишите программу на Python, обрабатывающую текстовую строку так, чтобы в нем все заглавные буквы преобразовать в строчные. Напишите скрипт на Python обрабатывающий текстовый файл и получение на его основе словаря, ключами являются слова текста, а значениями количество встречаемости слов. Сохраните в файле CSV.
	2. Анализирует алгоритмы в части производительности, оптимальности, вырабатывает рекомендации для оптимизации алгоритмов программ.	Знать: технологии прикладного программирования, инструментальные средства программирования. Уметь: разрабатывать программы решения задач с использованием прикладного программирования, включая среды высокоуровневого программирования.	Вопросы: 1. Понятие класса и объекта. Атрибуты класса и экземпляра класса. Доступ к атрибуту. Закрытые атрибуты. 2. Создание примеров программных реализаций для распространенных шаблонов проектирования. Задача Используя программу Jupyter Notebook составьте код на Python, определяющий количество точек на плоскости, находящихся на заданном расстоянии от начала координат. Используйте библиотеку math. Выберите библиотеку Python для работы с массивами. Выполните программный код бинарного поиска данных.

	<p>3. Про водит ручное и автоматизированное тестирование программных продуктов по методам черного и белого ящика, составляет набор тестовых случаев.</p>	<p>Знать: особенности создания программного кода, основы проектирования различных видов интерфейса программной системы. Уметь: разрабатывать программный код, ориентироваться в существующем коде, применять знание общепринятых соглашений и политик в области оформления кода, разрабатывать текстовый, программный или графический интерфейс программной системы исходя из ее назначения.</p>	<p>Вопросы: 1. Основные принципы объектно-ориентированного проектирования. Шаблон проектирования 2.. Клиент-серверные приложения. Библиотеки автоматизированного тестирования 3. Сокеты Документирование программы по описанию и исходному коду. Задача Выберите библиотеку Python для создания и обработки деревьев данных. Реализуйте программный код вычисления различных арифметических операций, выбор которых осуществляется с помощью разработанного меню. 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.</p>
--	--	---	--

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

Основная литература

1. Гуриков, С. Р. Основы алгоритмизации и программирования на Python : учебное пособие / С.Р. Гуриков. — Москва: ИНФРА-М, 2022. — 343 с. — ISBN 978-5-16-017142-5. - URL: <https://znanium.com/catalog/product/1356003> – Режим доступа: Электронно-библиотечная система Znanium.com – Текст: электронный.

2. Жуков, Р. А. Язык программирования Python: практикум учебное пособие / Р.А. Жуков. — Москва: ИНФРА-М, 2026. — 216 с. - ISBN 978-5-16-107207-3 - URL: <https://znanium.ru/catalog/documentid=477700>. – Режим

доступа: Электронно-библиотечная система Znanium.com – Текст: электронный.

3. Шелудько, В. М. Язык программирования высокого уровня Python. Функции, структуры данных, дополнительные модули: учебное пособие / В. М. Шелудько; Южный федеральный университет. - Ростов-на Дону; Таганрог : Издательство Южного федерального университета, 2017. - 107 с. – ЭБС ZNANIUM.com. - URL: <https://znanium.com/catalog/product/1021664> – Текст: электронный.

Дополнительная литература

4. Дроздов, С. Н. Структуры и алгоритмы обработки данных: учебное пособие / С. Н. Дроздов. – Таганрог: Южный федеральный университет, 2016. - 228 с. – ЭБС ZNANIUM.com. - URL: <https://znanium.com/catalog/product/991928> - Текст: электронный.

5. Северенс, Ч. Введение в программирование на Python / Ч. Северенс. – 2-е изд., испр. – Москва: Национальный Открытый Университет «ИНТУИТ», 2016. – 231 с. – ЭБС Университетская библиотека online. – URL: <https://biblioclub.ru/index.php?page=book&id=429184>). – Текст: электронный.

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

1. Электронно-библиотечная система BOOK.RU <http://www.book.ru>
2. Электронно-библиотечная система Znanium <http://www.znanium.com>
3. Электронно-библиотечная система «Университетская библиотека ОНЛАЙН» <http://biblioclub.ru/>
4. Электронно-библиотечная система издательства «ЮРАЙТ» <https://www.biblio-online.ru>
5. Электронная библиотека издательского дома «Гребенников» <https://grebennikon.ru>

6. Электронно-библиотечная система издательства «Лань»
<https://e.lanbook.com>

7. Python Data Analysis Library [Электронный ресурс]: сайт. – Режим доступа: <http://pandas.pydata.org/>.

8. Python Documentation [Электронный ресурс]: сайт. – Режим доступа: <http://python.org/doc/>.

9. Python Standard Library [Электронный ресурс]: сайт. – Режим доступа: <https://docs.python.org/2/library/>.

10. Scikit-learn Machine Learning in Python [Электронный ресурс]: сайт. – Режим доступа: <http://scikit-learn.org>.

10. Методические указания для обучающихся по освоению дисциплины

Методика освоения дисциплины предусматривает подготовку обучающихся к семинарам и практическим занятиям, выполнение студентами самостоятельной внеаудиторной работы, в том числе – проектных работ.

Рекомендации по подготовке к семинарам, практическим занятиям.

Студентам следует:

- проработать теоретический материал к занятию по рекомендованным литературным источникам и лекциям;
- использовать при подготовке к занятию нормативно-правовые документы, научные публикации, информационный материал, рекомендуемый преподавателем;
- перед занятиями задать вопросы по невыясненным в ходе самостоятельной подготовки темам или отдельным положениям темы;
- в ходе занятия давать четкие и исчерпывающие ответы на вопросы;
- на занятии демонстрировать понимание обсуждаемых тем и вопросов.

Студентам, по различным причинам пропустившим занятия,

необходимо перед очередным занятием отработать пропущенный материал, подготовив его самостоятельно.

Методические рекомендации по выполнению различных форм самостоятельной работы

Студентам при организации самостоятельной работы следует руководствоваться Приказом Финансового университета № 1040/о от 11.05.2021г. «Об утверждении методических рекомендаций по планированию и организации внеаудиторной самостоятельной работы студентов по образовательным программам бакалавриата и магистратуры в Финансовом университете».

Самостоятельная работа содержит в себе различные виды и формы работ. Самостоятельная работа студентов включает в себя выполнение различного рода заданий, которые ориентированы на более глубокое усвоение материала изучаемой дисциплины. По теме учебной дисциплины студентам предлагается перечень заданий для самостоятельной работы.

В ходе изучения дисциплины предусмотрены следующие формы самостоятельной работы:

- подготовка к опросу;
- выполнение заданий самостоятельной работы,
- решение задач;
- выполнение проектных работ;
- подготовка к зачетам.

К выполнению заданий для самостоятельной работы предъявляются следующие требования: задания должны выполняться самостоятельно и представляться в установленный срок, а также должны соответствовать установленным требованиям по оформлению.

Студентам следует:

- руководствоваться графиком самостоятельной работы, определенным РПД;
- выполнять все плановые задания, выдаваемые преподавателем для

самостоятельного выполнения, разбирать на занятиях и консультациях неясные вопросы;

- прорабатывать соответствующие теоретические и практические разделы дисциплины, фиксируя неясные фрагменты для их обсуждения на консультации.

Методические рекомендации для обучающихся по выполнению проектной работы

Проектная работа является обязательной формой внеаудиторной самостоятельной работы студентов по дисциплине.

Целью проектной работы является развитие у студентов способности прогнозировать, проектировать, моделировать, формирование учебно-исследовательских навыков, закрепление умений самостоятельно работать с различными источниками информации; проверка сформированности компетенций.

Проектная работа может выполняться как индивидуально, так и в составе группы. Количество групп и их численный состав определяет преподаватель, ведущий семинарские занятия.

Заказчиками выполнения проекта могут являться представители работодателей. В этом случае проектная работа выполняется исходя из потребностей заказчика.

Выполнение проекта предполагает:

- диагностику ситуации (проблематизация, целеполагание, конкретизация цели, форматирование проекта);
- проектирование (уточнение цели, функций, задач и плана работы; теоретическое моделирование методов и средств решения задач; детальная проработка этапов решения конкретных задач; пошаговое выполнение запланированных проектных действий; систематизация и обобщение полученных результатов, конструирование предполагаемого результата, пошаговое выполнение проектных действий);

- рефлексия (выяснение соответствия полученного результата замыслу; определение качества полученного продукта; перспективы его развития и использования);

- фиксация результатов в виде исполненного проекта.

Проектная работа состоит из нескольких частей. Состав проектной работы и очередность размещения отдельных частей:

- титульный лист;
- основная часть;
- список использованных источников;
- приложения (при наличии).

Титульный лист является первой страницей проектной работы и заполняется по определенным правилам.

Основная часть выполняется согласно заданию преподавателя или исходя из потребностей заказчика.

В список использованных источников включаются названия законодательных актов, нормативных документов, книг, статей, учебных пособий и т. п., которые, так или иначе, использовались студентом при выполнении работы.

Иллюстративный материал (схемы, диаграммы, рисунки, таблицы и др.) встраивается в текст работы или выносится в Приложения. В Приложения выносятся вспомогательные материалы, которые не содержат основную информацию, либо материалы, которые сложно разместить по тексту работы (большие схемы, таблицы, графические материалы, расчетные справочные данные, образцы первичных документов и т.п.). Непременным условием включения данных материалов в приложение является ссылка на них в тексте работы.

Требования к оформлению проектной работы.

Проектная работа выполняется на компьютере на одной стороне белой бумаги формата А4 (210x297 мм). Размер шрифта -13 или 14, междустрочный интервал – одинарный или полуторный.

Размеры полей: левое - 30 мм, правое - 15 мм, верхнее - 20 мм, нижнее - 20 мм. Отступ первой строки абзаца - 1,25. Нумерация страниц – внизу в центре.

При написании допускаются только общепринятые сокращения (например, тыс. руб.).

Общий объем проектной работы составляет не более 10 страниц, не включая таблицы, графики и т.п. (при наличии), а также приложения (при наличии).

В тексте обязательны ссылки на литературные источники, лучше всего постраничные.

Законченная проектная работа, содержащая все требуемые элементы оформления, вставленная в папку (или файл) и скрепленная по левому краю, сдается на кафедру или непосредственно руководителю проектной работы – преподавателю; ведущему семинарские (практические) занятия по дисциплине. Он осуществляет проверку проектной работы, а также оказывает помощь при подготовке к ее защите.

Проектная работа защищается в назначенные сроки. Защита проектной работы проводится до начала сессии (в крайнем случае, до начала экзамена по соответствующему предмету). При защите студент кратко излагает основные положения работы, последовательность ее выполнения, свои предложения.

При защите проектной работы студент должен свободно ориентироваться в изложенном материале работы; ответить на все замечания преподавателя; уметь отвечать на вопросы преподавателя по проектной работе.

Оценка проектных работ студентов проводится в процессе текущего контроля успеваемости.

Критерии оценки проектной работы

Оценка «отлично» (5 баллов) выставляется студенту, если проектная работа отличается творческим (креативным) подходом, собственным

оригинальным отношением автора к идее проекта; содержит полную диагностику ситуации, а также теоретическое моделирование методов и детальную проработку этапов решения конкретных задач; в работе сделаны необходимые выводы, намечены перспективы использования проекта, спланированы действия по его продвижению; работа отличается грамотным оформлением в точном соответствии с установленными правилами, с соблюдением логической последовательности изложения материала; студент в работе выдвигает новые идеи и трактовки, демонстрирует способность анализировать материал; на дополнительные вопросы при защите проектной работы даны полные ответы.

Оценка «хорошо» (3-4 балла) выставляется студенту, если проектная работа содержит достаточно полную диагностику ситуации, а также теоретическое моделирование методов и этапов решения конкретных задач; в работе сделаны выводы, намечены перспективы использования проекта; работа оформлена правильно с учетом 1-2 мелких погрешностей или 2-3 недочетов, исправленных самостоятельно или по требованию преподавателя; в работе соблюдена логическая последовательность изложения материала; студент в работе демонстрирует творческие способности и хорошую способность анализировать материал. На дополнительные вопросы при защите проектной работы даны не совсем полные ответы.

Оценка «удовлетворительно» (1-2 балла) выставляется студенту, если проектная работа содержит отдельные элементы моделирования методов и этапов решения конкретных задач; в работе сделаны выводы, намечены перспективы использования проекта; работа выполнена и оформлена правильно, но в ней допущены 1-2 погрешности или одна грубая ошибка; в работе соблюдена логическая последовательность изложения материала; студент в работе демонстрирует удовлетворительную способность анализировать материал; допущены ошибки при ответе на дополнительные вопросы при защите проектной работы.

Оценки «неудовлетворительно» (0 баллов) заслуживает студент, если в работе отсутствуют элементы моделирования; студент в работе не проявил способность анализировать, прогнозировать и проектировать; в работе отсутствует логическая последовательность изложения материала, допущены грубые ошибки, которые обучающийся не может исправить даже по требованию преподавателя или работа не выполнена полностью.

При оценивании проектной работы на «неудовлетворительно» работа должна быть переделана (исправлена) в соответствии с полученными замечаниями, сдана на проверку заново и защищена не позднее срока окончания ее приёма и защиты.

Оценка результатов текущего контроля успеваемости и промежуточной аттестации обучающихся осуществляется в соответствии с Балльно-рейтинговой системой Финансового университета (Приказ Финансового университета № 2187/о от 01.10.2024 г. «Об утверждении Положения о проведении текущего контроля успеваемости и промежуточной аттестации студентов, обучающихся по образовательным программам высшего образования в Финансовом университете»).

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

11.1 Комплект лицензионного программного обеспечения

- 1) Антивирусная защита Kaspersky Security для виртуальных и облачных сред;
- 2) Windows, Microsoft Office или Astra Linux, Libre Office.
- 3) Дистрибутив Python Anaconda (свободно распространяемое ПО).
- 4) Браузер.

11.2 Современные профессиональные базы данных, и информационные справочные системы

1. Информационно-правовая система «Гарант»:
<https://www.garant.ru>
2. Большая Российская энциклопедия: <https://bigenc.ru/>
3. Система комплексного раскрытия информации «СКРИН» -
<http://www.skrin.ru/>.

11.3 Сертифицированные программные и аппаратные средства защиты информации

Не используются

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

Учебная аудитория для проведения учебных занятий, предусмотренных программой бакалавриата, оснащенная оборудованием и техническими средствами обучения:

Аудитория № 47

Специализированная мебель:

Стол компьютерный – 20 шт.

Стол (двухместный) – 7 шт.

Стул – 34 шт.

Шкаф – 1 шт.

Технические средства обучения:

Компьютер в сборе – 20 шт.

Мультимедиа-проектор – 1 шт.

Экран настенный – 1 шт.

Подключение к сети «Интернет» и обеспечение доступа в электронную информационно-образовательную среду Финансового университета

Помещение для самостоятельной работы обучающихся

Кабинет № 55. Читальный зал:

Специализированная мебель:

Стол – 20 шт.

Стул – 40 шт.

Шкаф для книг – 4 шт.

Стеллаж книжный – 13 шт.

Стеллаж выставочный – 4 шт.

Технические средства обучения:

Компьютер в сборе – 6 шт.

Телевизор – 1 шт.

Помещение для самостоятельной работы обучающихся оснащено компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Финансового университета