

**Федеральное государственное образовательное
бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ
ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)**

**Департамент анализа данных и машинного обучения
Факультета информационных технологий и анализа больших данных**

Макрушин С.В., Горохова Р.И.

Практикум по программированию

Рабочая программа дисциплины
для студентов, обучающихся по направлению подготовки
01.03.02 - Прикладная математика и информатика,
ОП "Прикладное машинное обучение",
Профиль: "Прикладное машинное обучение"

Москва 2022

**Федеральное государственное образовательное
бюджетное учреждение высшего образования
«ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»
(Финансовый университет)**

**Департамент анализа данных и машинного обучения
Факультета информационных технологий и анализа больших данных**

УТВЕРЖДАЮ

Проректор по учебной и
методической работе

_____ Е.А. Каменева

24.05.2022 г.

Макрушин С.В., Горохова Р.И.

Практикум по программированию

Рабочая программа дисциплины
для студентов, обучающихся по направлению подготовки
01.03.02 - Прикладная математика и информатика,
ОП "Прикладное машинное обучение",
Профиль: "Прикладное машинное обучение"

*Рекомендовано Ученым советом
Факультета информационных технологий и анализа больших данных
(протокол №21 от 17.05.2022 г.)*

*Одобрено Советом учебно-научного
Департамента анализа данных и машинного обучения
(протокол № 9 от 28.04.2022 г.)*

Москва 2022

Рецензент: **В.Г. Феклин** – к.ф.-м.н., доцент Департамента анализа данных и машинного обучения

Макрушин С.В., Горохова Р.И. «Практикум по программированию». Рабочая программа дисциплины для студентов, обучающихся по направлению подготовки 01.03.02 - Прикладная математика и информатика, ОП "Прикладное машинное обучение", Профиль: "Прикладное машинное обучение" – М.: Финансовый университет, Департамент анализа данных и машинного обучения, 2022. – 46 с.

Дисциплина «Практикум по программированию» относится к Обще-профессиональному циклу дисциплин по направлению подготовки 01.03.02 - Прикладная математика и информатика, ОП "Прикладное машинное обучение", Профиль: "Прикладное машинное обучение".

В рабочей программе дисциплины определены ее цель, место в структуре ОП, требования к результатам освоения дисциплины, содержание программы, тематика аудиторных занятий, формы самостоятельной работы, оценочные средства для текущего контроля и промежуточной аттестации, учебно-методическое и информационное обеспечение.

Учебное издание

Сергей Вячеславович Макрушин, Римма Ивановна Горохова

Практикум по программированию

Рабочая программа дисциплины

Компьютерный набор, верстка С.В. Макрушин,
Р.И. Горохова

Формат 60х90/16. Гарнитура *Times New Roman*

Усл. п.л.0,9. Изд. № - 2022. Тираж - экз.

Заказ № _____

Отпечатано в Финансовом университете

© **Макрушин Сергей Вячеславович,
Римма Ивановна Горохова 2022**

© **Финансовый университет, 2022**

Содержание

1. Наименование дисциплины	4
2. Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	4
3. Место дисциплины в структуре образовательной программы	6
4. Объем дисциплины (модуля) в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	7
5. Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий.....	7
5.1. Содержание дисциплины	7
5.2. Учебно-тематический план.....	10
5.3. Содержание семинаров, практических занятий.....	12
6. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине.....	13
6.1. Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	13
6.2. Перечень вопросов, заданий, тем для подготовки к текущему контролю	14
7. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине:.....	32
8. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины:	41
9. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины.....	43
10. Методические указания для обучающихся по освоению дисциплины	45
11. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем.....	45
12. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	46

1. Наименование дисциплины

«Практикум по программированию».

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

Дисциплина «Практикум по программированию» обеспечивает формирование следующих компетенций: УК-10, ПКН-4.

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
УК-10	Способность осуществлять поиск, критически анализировать, обобщать и систематизировать информацию, использовать системный подход для решения поставленных задач.	1. Четко описывает состав и структуру требуемых данных и информации, грамотно реализует процессы их сбора, обработки и интерпретации. 2. Обосновывает сущность происходящего, выявляет закономерности, понимает природу вариабельности. 3. Формулирует признак классификации, выделяет соответствующие ему группы однородных «объектов», идентифицирует общие свойства элементов этих групп, оценивает полноту результатов классификации,	Знать: состав и структуру требуемых данных и информации. Уметь: грамотно реализовать процессы сбора, обработки и интерпретации данных и информации. Знать: закономерности реализуемых процессов и природу их вариабельности. Уметь: определять закономерности изучаемых процессов и данных, выявлять природу их вариабельности. Знать: особенности признаков классификации и оценки результатов классификации. Уметь: разрабатывать программный код, выполняющий классификацию по обозначенному признаку, ориентироваться в существующем коде, оценивать

		показывает прикладное назначение классификационных групп. 4. Грамотно, логично, аргументировано формирует собственные суждения и оценки. Отличает факты от мнений, интерпретаций, оценок и т.д. в рассуждениях других участников деятельности. 5. Аргументированно и логично представляет свою точку зрения посредством и на основе системного описания.	полноту результатов классификации. Знать: основы логических рассуждений, аргументации, оценки своих суждений и рассуждений других участников. Уметь: проводить сравнительный анализ мнений, интерпретаций, оценок в рассуждениях других участников деятельности. Знать: основы аргументированного и логического рассуждения. Уметь: проводить системное описание своей точки зрения, представлять ее аргументированно и логично
ПКН-4	Способен проектировать и реализовывать прикладные программные системы в соответствии с анализом задачи и требований к ним.	1. Демонстрирует базовые знания о существующих математических методах и системах программирования. 2. Использует и адаптирует существующие математические методы и системы программирования для решения прикладных задач.	Знать: существующие математические методы; существующие системы программирования. Уметь: выполнять выбор математических методов и систем программирования необходимый для решения конкретной задачи. Знать: существующие математические методы и системы программирования. Уметь: уметь адаптировать существующие математические методы и системы программирования для решения прикладных задач

		3. Владеет навыками проектирования и разработки компонентов программного обеспечения на основе современных парадигм, технологий и языков программирования. 4. Применяет методы машинного обучения для решения прикладных задач анализа данных	Знать: современные языки программирования; современные технологии программирования Уметь: проектировать и разрабатывать компоненты программного обеспечения на основе современных парадигм, технологий и языков программирования. Знать: методы машинного обучения для решения прикладных задач анализа данных. Уметь: применять методы машинного обучения для решения прикладных задач анализа данных.
--	--	---	---

3. Место дисциплины в структуре образовательной программы

Дисциплина «Практикум по программированию» относится к Общепрофессиональному циклу дисциплин по направлению подготовки 01.03.02 - Прикладная математика и информатика, ОП "Прикладное машинное обучение", Профиль: "Прикладное машинное обучение".

Дисциплина «Практикум по программированию» базируется на знаниях, приобретаемых в рамках дисциплин «Алгоритмы и структуры данных в языке Python» которую студенты изучают в течение первого года обучения.

Дисциплина «Практикум по программированию» служит для закрепления и углубления знаний и навыков в области программирования, получаемых при освоении других дисциплин, изучаемых в рамках направления подготовки 01.03.02-Прикладная математика и информатика.

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

Очная форма обучения

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

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 упаковка 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`, `xlsx`.

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 обработки 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 теку- щего контроля успеваемости	
		Всего	Контактная работа - Аудиторная работа				Самосто- ятельная работа
			Об- щая, в т.ч.:	Лек- ции	Семи- нары, практиче- ские за- нятия		
1.	Основы языка Python	24	4	-	4	20	Опрос, про- верка выпол- ненных зада- ний

2.	Функции и модули	24	6	-	6	18	Опрос, проверка выполненных заданий
3.	Обработка исключений. Работа с файлами средствами языка Python	24	4	-	4	20	Опрос, проверка выполненных заданий
4.	Объектно-ориентированное программирование на Python	24	6	-	6	18	Опрос, проверка выполненных заданий
5.	Функциональное программирование на Python	24	4	-	4	20	Опрос, проверка выполненных заданий
6.	Алгоритмы и структуры данных	24	6	-	6	18	Опрос, проверка выполненных заданий
7.	Паттерны проектирования	24	6	-	6	18	Опрос, проверка выполненных заданий
8.	Программирование графических интерфейсов	24	4	-	4	20	Опрос, проверка выполненных заданий
9.	Системное программирование на Python	24	6	-	6	18	Опрос, проверка выполненных заданий
10.	Сетевое программирование на Python	24	6	-	6	18	Опрос, проверка выполненных заданий
11.	Тестирование программ на Python	24	6	-	6	18	Опрос, проверка выполненных заданий
12.	Документирование и развертывание программ на Python	24	6	-	6	18	Опрос, проверка выполненных заданий
		288	64	-	64	224	Согласно учебному плану: проектные

	В целом по дисциплине						работы
	Итого в %	100	22	-	100	78	

5.3. Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарских, практических занятиях, рекомендуемые источники из разделов 8,9 (указывается раздел и порядковый номер источника)	Формы проведения занятий
1. Основы языка Python	Решение задач с использованием инструкции ветвления. Решение задач с использованием циклов. Создание и обработка списков. Многомерные списки. Обработка текстовой информации. Использование словарей и множеств. Совместное использование различных типов данных. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
2. Функции и модули	Создание и использование функций. Анонимные функции. Функции высшего порядка. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
3. Обработка исключений. Работа с файлами средствами языка Python	Работа с текстовыми файлами. Работа с двоичными файлами. Сохранение объектов в файл. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
4. Объектно-ориентированное программирование на Python	Создание классов и объектов. Наследование и полиморфизм. Специальные методы классов. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
5. Функциональное программирование на Python	Функции map, filter, reduce, any, all. Декораторы. Итераторы. Функции генераторы. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
6. Алгоритмы и структуры данных	Создание связанных списков. Использование стеков и очередей при решении задач. Алгоритмы сортировки и поиска. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере

7. Паттерны проектирования	Создание примеров программных реализаций для распространенных шаблонов проектирования. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
8. Программирование графических интерфейсов	Создание графического приложения с использованием встроенных возможностей языка и библиотек <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
9. Системное программирование на Python	Автоматизация рутинных административных задач. Создание Web-парсера. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
10. Сетевое программирование на Python	Создание многопоточного многопользовательского многофункционального сервера на сокет-ах. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
11. Тестирование программ на Python	Написание модульных тестов по описанию. Разработка программы по тестам. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере
12. Документирование и развертывание программ на Python	Документирование программы по описанию и исходному коду. Экспорт документации в формате сайта. <i>Основная литература - [8.1 - 8.3]</i> <i>Дополнительная литература - [8.1-8.2]</i>	Интерактивная форма, работа на компьютере

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
1. Основы языка Python	Функции модуля math, random, сорю. Регулярные выражения. Кортежи. Множества.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
2. Функции и модули	Создание и использование модулей и пакетов.	Работа с учебной литературой и документацией. Решение задач с

		использованием Jupyter Notebook. Выполнение домашних заданий.
3. Обработка исключений. Работа с файлами средствами языка Python	Работа с файлами различных форматов: csv, docx, xlsx.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
4. Объектно-ориентированное программирование на Python	Специальные методы классов.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
5. Функциональное программирование на Python	Функции any, all, zip.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
6. Алгоритмы и структуры данных	Бинарные деревья. Использование бинарных деревьев в прикладных задачах. Двоичное дерево поиска.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
7. Паттерны проектирования	Внедрение зависимости, инверсия управления.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
8. Программирование графических интерфейсов	Создание игр. Библиотека PyGame.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
9. Системное программирование на Python	Создание административных скриптов.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
10. Сетевое программирование на Python	Создание веб-сервиса.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
11. Тестирование программ на Python	Экстремальное программирование.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.
12. Документирование и развертывание программ на Python	Непрерывная интеграция.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook. Выполнение домашних заданий.

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

Примеры заданий проектных работ

Задание 1

1. Реализовать программу, с которой можно играть в логическую игру «Быки и коровы» (описание правил игры: <http://побомозг.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-slo-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-палуб-

ных корабля и 1 4-х палубный. Между любыми двумя кораблями по горизонтали 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.

Задание 2

Базовая часть:

Написать калькулятор для строковых выражений вида '<число> <операция> <число>', где <число> - не отрицательное целое число меньше 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". (Для реализации рекомендуется использовать алгоритмы основанные на польской инверсной записи см. https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%80%D0%B0%D1%82%D0%BD%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BB%D1%8C%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C) *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 пи. 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*

Задание 3

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 выч. 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 длиной 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*

Задание 4

Базовая часть (выполняется всеми самостоятельно!):

Реализовать программу, которая позволяет играть в шахматы на компьютере. Взаимодействие с программой производится через консоль (базовый вариант). Игровое поле изображается в виде 8 текстовых строк, плюс строки с буквенным обозначением столбцов (см. пример на Рис. 1) и перерисовывается при каждом изменении состояния поля. При запросе данных от пользователя программа сообщает, что ожидает от пользователя (например,

позицию фигуры для следующего хода белыми; целевую позицию выбранной фигуры) и проверяет корректность ввода (допускаются только ходы соответствующие правилам шахмат; поддержка рокировки, сложных правил для пешек и проверки мата вынесена в отдельные пункты). Программа должна считать количество сделанных ходов.

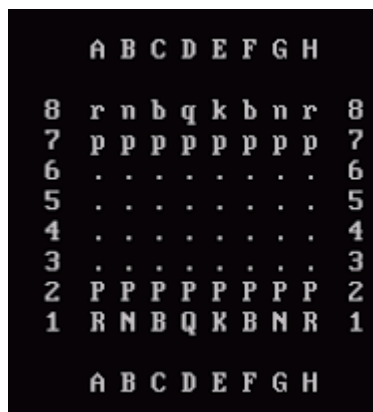


Рис. 1 Пример изображения шахматного поля в текстовом режиме

Сама программа НЕ ходит: т.е. не пытается выполнить ходы за одну из сторон, а предоставляет поочередно вводить ходы за белых и черных.

Индивидуальные задания:

Справка о шахматной нотации:

1. Общая информация о шахматной нотации записи партий:
https://ru.wikipedia.org/wiki/Шахматная_нотация
2. 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).
3. 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, 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*

Задание 5

Простая анимация

Цель работы: Основная цель решения задач из данного списка - освоить приемы работы с графической библиотекой, а именно: построение графических примитивов, создание анимации, управление главным циклом работы программы.

Порядок выполнения:

Задания в этой лабораторной работе предусматривают индивидуальное самостоятельное выполнение. Преподаватель распределяет задания между студентами случайным образом. На семинаре преподаватель может по запросу студентов пояснить задание в части требований к выполнению или стоящей за заданием математической концепцией.

Все задания подразумевают творческий подход к их выполнению. Задание описывает только общую канву программы. В процессе выполнения студент может столкнуться с рядом дизайнерских решений - выбор скорости анимации, размера элементов, цветового решения и так далее. Все это ложится на студента как на разработчика программы. Преподаватель вправе оценить удачные решения и снизить оценку за неудачные.

По умолчанию предусматривается использование библиотеки tkinter. По желанию студента и по согласованию с преподавателем можно выбрать другой подобный инструмент решения задачи (pygame, p5 или аналогичные графические библиотеки).

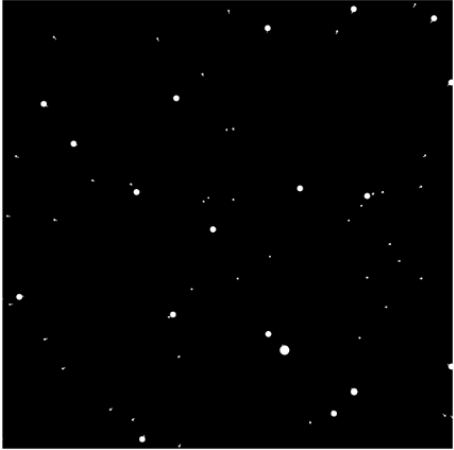
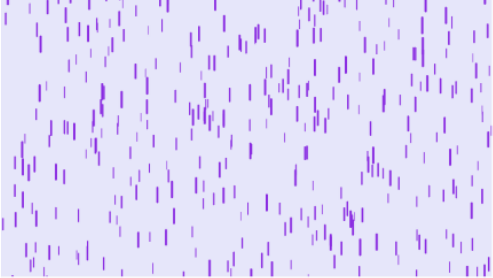
К каждой задаче предлагаются дополнительные задания (пронумерованы), которые расширяют или углубляют данное задание. Студент может самостоятельно предложить расширение функционала программы. Дополнительные задания оцениваются преподавателем с учетом их сложности.

Дополнительные задания предусматривают самостоятельное освоение студентом некоторых инструментов - обработки событий, работы с элементами пользовательского интерфейса, работу с мышью и клавиатурой. Следует помнить, что все эти навыки пригодятся для выполнения дальнейших заданий.

После самостоятельного выполнения задания преподаватель может

попросить студента провести публичную демонстрацию выполненного задания, включая устные комментарии написанного хода и алгоритмических решений, примененных студентом при решении задачи.

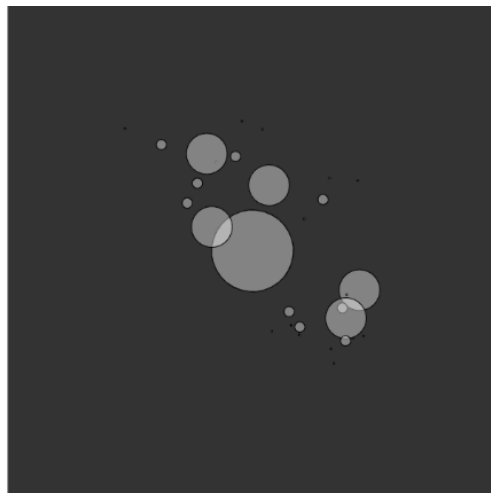
Задания для выполнения

Звездное поле В произвольном месте на экране появляются звезды, которые движутся по направлению от центра экрана. Должно складываться ощущение движения наблюдателя сквозь звездное поле. Сделайте звезды разного размера и цветов. 1. Реализуйте движение в 3D, где размер звезды зависит от расстояния до нее. 2. Скорость движения регулируется с клавиатуры. 3. Мышкой можно регулировать направление движения.	
Дождь На экране постоянно генерируются контрастные “капли”, равномерно движущиеся вниз. 1. Поэкспериментируйте с плотностью генерации капель. 2. Капли должны создавать ощущение трехмерности - дальние движутся медленнее и меньше в размерах.	

Солнечная система

В центре экрана находится неподвижное солнце. Вокруг него вращаются несколько планет по круговым орбитам с разной скоростью и на разном расстоянии.

1. Попробуйте добавить прозрачность.
2. У планет могут быть собственные спутники.
3. Добавьте пояс астероидов.



Задание 6

Сложная анимация

Цель работы

Цель выполнения задач из данного списка - применить полученные навыки работы с графикой и анимацией для выполнения более сложных проектов. Каждая задача предполагает продумывание структуры программы, то есть этап проектирования.

Порядок выполнения

Задания в этой проектной работы предусматривают индивидуальное самостоятельное выполнение. Преподаватель распределяет задания между студентами случайным образом. На семинаре преподаватель может по запросу студентов пояснить задание в части требований к выполнению или стоящей за заданием математической концепцией.

Все задания подразумевают творческий подход к их выполнению. Задание описывает только общую канву программы. В процессе выполнения студент может столкнуться с рядом дизайнерских решений - выбор скорости анимации, размера элементов, цветового решения и так далее. Все это ложится на студента как на разработчика программы. Преподаватель вправе оценить удачные решения и снизить оценку за неудачные.

Многие задания из данного списка предполагают освоение какой-то математической концепции, необходимой для понимания задания или путей его выполнения. Этап исследования - важный и необходимый при написании прикладного программного обеспечения.

Строго рекомендуется при выполнении заданий из данного сборника использовать объектно-ориентированный подход. Вы должны следить за оформлением кода - соблюдением стилевых соглашений, необходимости комментировать код, его читаемости. Дополнительным плюсом будет использование системы контроля версий.

К каждой задаче предлагаются дополнительные задания (пронумерованы), которые расширяют или углубляют данное задание. Студент может

самостоятельно предложить расширение функционала программы. Дополнительные задания оцениваются преподавателем с учетом их сложности.

Дополнительные задания предусматривают самостоятельное освоение студентом некоторых инструментов - обработки событий, работы с элементами пользовательского интерфейса, работу с мышью и клавиатурой. Следует помнить, что все эти навыки пригодятся для выполнения дальнейших заданий.

После самостоятельного выполнения задания преподаватель может попросить студента провести публичную демонстрацию выполненного задания, включая устные комментарии написанного хода и алгоритмических решений, примененных студентом при решении задачи.

1. *Фейерверки (152)*. На экране случайно генерируются фейерверки. Сам фейерверк и его частицы имеют ограниченное время жизни. Добавьте генерацию фейерверков разных цветов и прозрачность. Поэкспериментируйте с гравитацией частиц.

2. *Притяжение (113)*. Реализуйте систему частиц, каждая из которых притягивается к определенному месту на экране. Одновременно, каждая частица отталкивается от текущих координат мышки. Должно получиться, что курсор "разбрасывает" частицы от себя. Нарисуйте местами притяжения частиц какой-нибудь рисунок или надпись. Реализуйте импорт рисунка из внешнего файла.

3. *Анимация круга (78)* С помощью полярных координат реализующее превращение круга в треугольник как на приведенной анимации

4. *Папоротник Барнсли (48)* При помощи интерактивного алгоритма постройте из точек фрактальную фигуру. Манипулируя параметрами уравнения, получите другие виды фрактальных растений.

Задание 7

Простые игры

Цель работы

Цель выполнения работ из данного списка - применить на практике навыки работы с двумерной анимацией в разработке простых аркадных игр и получить навыки командной работы, проектирования и управления длительной разработкой.

Порядок выполнения

Задания в этой лабораторной работе предусматривают самостоятельное выполнение в малых группах (2 человека). Преподаватель распределяет задания между студентами случайным образом. На семинаре преподаватель может по запросу студентов пояснить задание в части требований к выполнению или стоящей за заданием математической концепцией.

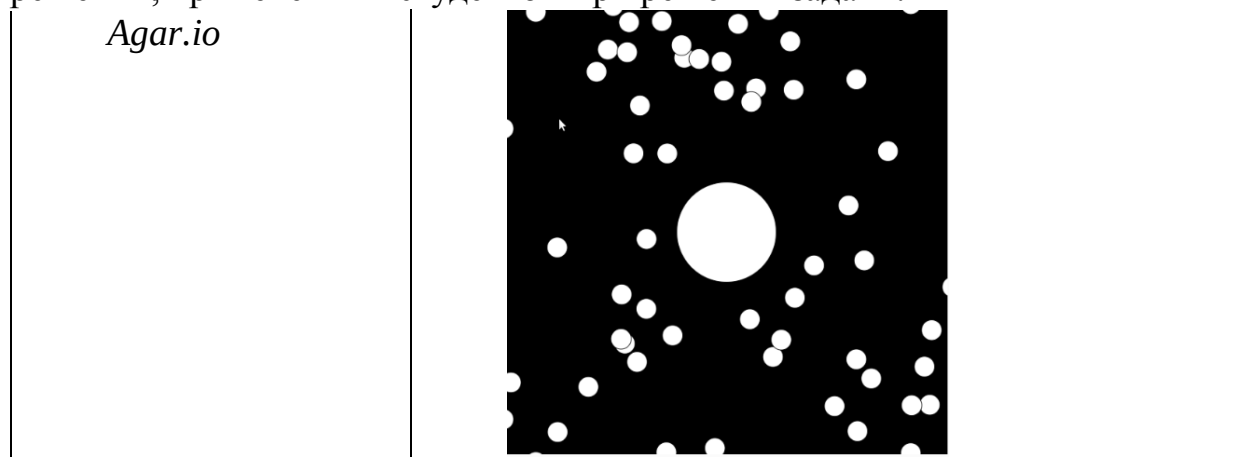
Все задания подразумевают творческий подход к их выполнению. Задание описывает только общую канву программы. В процессе выполнения студент может столкнуться с рядом дизайнерских решений - выбор скорости анимации, размера элементов, цветового решения и так далее. Все это ложится на студента как на разработчика программы. Преподаватель вправе оценить удачные решения и снизить оценку за неудачные.

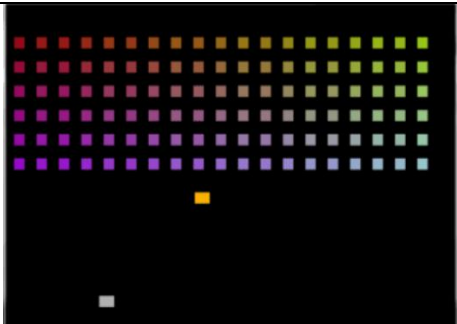
Задания из данного списка построены на классических играх. К каждому заданию прилагается анимация, иллюстрирующая механику игры. При разработке игр рекомендуется предварительно ознакомиться с существующими реализациями, правилами игры и их вариациями. Следует в первую очередь реализовать именно классический вариант данной игры. При желании в механику и правила игры можно внести изменения.

Разработка программ из данного списка может предполагать работу с графическими элементами - фоновыми изображениями, спрайтами, моделями. По возможности следует создать эти элементы самостоятельно (это может стать отдельной ролью студента в проекте), либо найти свободно распространяемые варианты.

Строго рекомендуется при выполнении заданий из данного сборника использовать объектно-ориентированный подход. Вы должны следить за оформлением кода - соблюдением стилевых соглашений, необходимости комментировать код, его читаемости. Строго рекомендуется использование системы контроля версий для организации коллективной работы.

После самостоятельного выполнения задания преподаватель может попросить студента провести публичную демонстрацию выполненного задания, включая устные комментарии написанного кода и алгоритмических решений, примененных студентом при решении задачи.



<i>Блек Джекс</i>	
<i>Space invaders</i>	

Критерии балльной оценки различных форм текущего контроля успеваемости

Критерии балльной оценки различных форм текущего контроля успеваемости содержатся в соответствующих методических рекомендациях Департамента анализа данных и машинного обучения

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

Наименование компетенции	Наименование индикаторов до-	Результаты обучения (умения и знания), соотнесенные	Типовые контрольные задания
--------------------------	------------------------------	---	-----------------------------

	стижения компетенции	с индикаторами достижения компетенции	
Способность осуществлять поиск, критически анализировать, обобщать и систематизировать информацию, использовать системный подход для решения поставленных задач (УК-10)	1. Четко описывает состав и структуру требуемых данных и информации, грамотно реализует процессы их сбора, обработки и интерпретации.	Знать: состав и структуру требуемых данных и информации. Уметь: грамотно реализовать процессы сбора, обработки и интерпретации данных и информации.	Выполните обработку текстового файла, 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. Опишите структуру CSV файла фиксирующего факты продаж, необходимого для подсчета суммарных продаж по различным географическим подразделениям фирмы.
	2. Обосновывает сущность происходящего, выявляет закономерности, понимает природу вариативности.	Знать: закономерности реализуемых процессов и природу их вариативности. Уметь: определять закономерности изучаемых процессов и данных, выявлять природу их вариативности.	Опишите закономерности/вариативность в синтаксических конструкциях Python для работы со списками и словарями. Определите количество элементов списка, значения которых совпадают с ключами заданного словаря и количество элементов списка, значения которых равны значениям в словаре.
	3. Формулирует признак классификации, выделяет соответствующие ему группы однородных «объектов», идентифицирует общие свойства элементов этих групп, оценивает полноту результатов классификации, показывает прикладное	Знать: особенности признаков классификации и оценки результатов классификации. Уметь: разрабатывать программный код, выполняющий классификацию по обозначенному при-	Опишите подходы используемые в Python для группировки функций, ориентированных на обработку однотипных объектов и решение близких задач. Выполните классификацию с использованием бинарного дерева. Оцените количество операций и глубину дерева

	назначение классификационных групп.	знаку, ориентироваться в существующем коде, оценивать полноту результатов классификации.	
	4. Грамотно, логично, аргументировано формирует собственные суждения и оценки. Отличает факты от мнений, интерпретаций, оценок и т.д. в рассуждениях других участников деятельности.	Знать: основы логических рассуждений, аргументации, оценки своих суждений и рассуждений других участников. Уметь: проводить сравнительный анализ мнений, интерпретаций, оценок и т.д. в рассуждениях других участников деятельности.	Сформируйте собственные суждения о применимости Python для решения задач машинного обучения. Выполните задачу построения бинарного дерева двумя способами: на основе списков и на основе узлов и ссылок. Оцените эффективность работы каждого способа.
	5. Аргументированно и логично представляет свою точку зрения посредством и на основе системного описания.	Знать: основы аргументированного и логического рассуждения. Уметь: проводить системное описание своей точки зрения, представлять ее аргументированно и логично	Сформулируйте задачи по обработке массивов, которые вы сможете решить на основе языка Python Выскажите свою точку зрения о применимости Python для создания крупной корпоративной транзакционной системы на языке программирования Python.
Способен проектировать и реализовывать прикладные программные системы в соответствии с анализом задачи и требований к ним. (ПKN-4)	1. Демонстрирует базовые знания о существующих математических методах и системах программирования.	Знать: существующие математические методы; существующие системы программирования. Уметь: выполнять выбор	Выполните обработку текстового файла, 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. Опишите структуру CSV файла фиксирующего факты

		математических методов и систем программирования необходимый для решения конкретной задачи.	продаж, необходимого для подсчета суммарных продаж по различным географическим подразделениям фирмы.
	2.Использует и адаптирует существующие математические методы и системы программирования для решения прикладных задач.	Знать: существующие математические методы и системы программирования. Уметь: уметь адаптировать существующие математические методы и системы программирования для решения прикладных задач.	Опишите закономерности/вариабельность в синтаксических конструкциях Python для работы со списками и словарями. Определите количество элементов списка, значения которых совпадают с ключами заданного словаря и количество элементов списка, значения которых равны значениям в словаре.
	3.Владеет навыками проектирования и разработки компонентов программного обеспечения на основе современных парадигм, технологий и языков программирования.	Знать: современные языки программирования; современные технологии программирования Уметь: проектировать и разрабатывать компоненты программного обеспечения на основе современных парадигм, технологий и языков программирования.	Опишите подходы используемые в Python для группировки функций, ориентированных на обработку однотипных объектов и решение близких задач. Выполните классификацию с использованием бинарного дерева. Оцените количество операций и глубину дерева.
	4. Применяет методы машинного обучения для решения прикладных задач анализа данных.	Знать: методы машинного обучения для решения прикладных задач анализа данных. Уметь: применять методы машинного обучения для решения	Сформируйте собственные суждения о применимости Python для решения задач машинного обучения. Выполните задачу построения бинарного дерева двумя способами: на основе списков и на

		прикладных задач анализа данных.	основе узлов и ссылок. Оцените эффективность работы каждого способа.
--	--	----------------------------------	--

Примерные вопросы для подготовки к зачетам:

1 семестр

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.

2 семестр

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.

3 семестр

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.

4 семестр

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.

Примеры заданий для подготовки к зачету

1 семестр

1. Чему будет равно значение переменной A после выполнения оператора:
`A = [i+10 for i in range(5,0,-1)]`
2. Имеется строка `S="1234567890"`. Чему равно значение S1, если `S1 = S[1:2] + S[4: :-1]`
3. 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, ...}`, т.е. значение равно номеру элемента (нумерация с 1), умноженному на 10, а ключ состоит из символа 's', к которому дописан номер.

2 семестр

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]
- e) 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

3 семестр

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

4 семестр

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
с. netstat
d.tracert

4. Назовите пример сетевой топологии

а. звезда
б. снежинка
с. лабиринт
д. круговая

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. — (Высшее образование: Бакалавриат). — ЭБС ZNANIUM.com. - URL: <https://znanium.com/catalog/product/1356003> (дата обращения: 15.06.2022). — Текст : электронный.
2. 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> (дата обращения: 15.06.2022). — Текст: электронный.
3. Жуков, Р. А. Язык программирования Python: практикум : учебное пособие / Р. А. Жуков. — Москва : ИНФРА-М, 2022. — 216 с. — (Высшее образование: Бакалавриат). - ЭБС ZNANIUM.com. - URL: <https://znanium.com/catalog/product/1689648> (дата обращения: 15.06.2022). - Текст : электронный.

Дополнительная литература:

1. 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> (дата обращения: 15.06.2022). – Текст : электронный.
2. 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> (дата обращения 15.06.2022). - Текст : электронный.

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

1. Pylru 1.0.9 [Электронный ресурс]: сайт. – Режим доступа: <https://pypi.python.org/pypi/pylru>.
2. Python Data Analysis Library [Электронный ресурс]: сайт. – Режим доступа: <http://pandas.pydata.org/>.
3. Python Documentation [Электронный ресурс]: сайт. – Режим доступа: <http://python.org/doc/>.
4. Python Standard Library [Электронный ресурс]: сайт. – Режим доступа: <https://docs.python.org/2/library/>.
5. Scikit-learn Machine Learning in Python [Электронный ресурс]: сайт. – Режим доступа: <http://scikit-learn.org>.
6. Официальный сайт продукта <https://www.python.org/>.
7. Портал Финансового университета <http://www.fa.ru/>.
8. Каталог курсов Интернет Университета Информационных Технологий <http://www.intuit.ru/>.
9. The Python Tutorial // <https://docs.python.org/3/tutorial/index.html>.
10. The Python Standard Library <https://docs.python.org/3/library/index.html>.
11. SciPy // <http://docs.scipy.org/doc/scipy/reference/>.
12. NumPy User Guide // <http://docs.scipy.org/doc/numpy/user/index.html>.
13. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/>.
14. Электронно-библиотечная система BOOK.RU <http://www.book.ru>.
15. Электронно-библиотечная система «Университетская библиотека ОН-ЛАЙН» <http://biblioclub.ru/>.

16. Электронно-библиотечная система Znanium <http://www.znanium.com>.
17. Электронно-библиотечная система издательства «ЮРАЙТ» <https://urait.ru/>.
18. Электронно-библиотечная система издательства Проспект <http://ebs.prospekt.org/books>.
19. Электронно-библиотечная система издательства «Лань» <https://e.lanbook.com/>.
20. Электронная библиотека Издательского дома «Гребенников» <https://grebennikon.ru/>.
21. Деловая онлайн-библиотека Alpina Digital <http://lib.alpinadigital.ru/>.
22. Научная электронная библиотека eLibrary.ru <http://elibrary.ru>.
23. Национальная электронная библиотека <http://нэб.рф/>.
24. Финансовая справочная система «Финансовый директор» <http://www.1fd.ru/>.
25. Ресурсы информационно-аналитического агентства по финансовым рынкам Cbonds.ru <https://cbonds.ru/>.
26. СПАРК <https://spark-interfax.ru/>.
27. Academic Reference <http://ar.cnki.net/ACADREF>.
28. Bank Focus <http://library.fa.ru/resource.asp?id=527>.
29. Пакет баз данных компании EBSCO Publishing, крупнейшего агрегатора научных ресурсов ведущих издательств мира <http://search.ebscohost.com>.
30. Электронные продукты издательства Elsevier <http://www.sciencedirect.com>.
31. Emerald: Management eJournal Portfolio <https://www.emerald.com/insight/>
32. Информационно-аналитическая база данных EMIS Global <https://www.emis.com/php/companies/overview/index>.
33. Реферативная база данных по математике MathSciNET <https://mathscinet.ams.org/mathscinet/>.
34. Oxford Scholarship Online <https://oxford.universitypressscholarship.com/>
35. Коллекция научных журналов Oxford University Press <https://academic.oup.com/journals/>.
36. ProQuest: База данных Business Ebook Subscription на платформе Ebook Central <https://search.proquest.com/>.
37. ProQuest Dissertations & Theses A&I <https://search.proquest.com/>.
38. База данных RUSLANA компании Bureau van Dijk <https://ruslana.bvdep.com/>.
39. Scopus <https://www.scopus.com>.

40. Электронная коллекция книг издательства Springer: Springer eBooks <http://link.springer.com/>.
41. Интерактивная финансовая информационная система компании Bloomberg.
42. Система Thomson Reuters Eikon.
43. Web of Science <http://apps.webofknowledge.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.

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. Операционная система. Пакет офисных программ.
2. Антивирус Kaspersky.
3. Дистрибутив Python Anaconda (свободно распространяемое ПО).
4. Браузер.
5. Файловый менеджер Far.

11.2. Современные профессиональные базы данных и информационные справочные системы

1. Информационно-правовая система «Гарант».
2. Информационно-правовая система «Консультант Плюс».
3. Электронная энциклопедия: <http://ru.wikipedia.org/wiki/Wiki>
4. 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

Материально-техническая база Финансового университета, необходимая для осуществления образовательного процесса по данной дисциплине, в соответствии с требованиями ФОС ВО включает в себя специальные помещения для проведения лекций, семинарских занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации; помещения укомплектованы специализированной мебелью и техническими средствами обучения, необходимыми для представления информации большой аудитории.

Помещения для самостоятельной работы студентов включают в себя библиотеку с читальным залом, укомплектованную в соответствии с существующими нормами необходимой учебной и учебно-методической литературой и иными материалами; медиатеку с выходом в Интернет, компьютерные классы с возможностью работы в Интернет; аудитории для консультационной деятельности.