

Федеральное государственное образовательное бюджетное учреждение
высшего образования

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

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

Документ подписан усиленной неквалифицированной электронной подписью
Организация: Финансовый университет при Правительстве РФ
Утверждено: Проректор по учебной и методической работе Е.А. Каменева
Сертификат: +9pocKN42Pd/M+thois8r9fawSg2wdTY
Дата: 25.03.2025 г.

Р. И. Горохова

Практикум по программированию

Рабочая программа дисциплины

для студентов, обучающихся по направлению подготовки

09.03.04 - Программная инженерия

Образовательная программа:

Разработка и внедрение информационно - аналитических систем

Профиль

Разработка и внедрение информационно - аналитических систем

Рекомендовано

*Факультет информационных технологий и анализа больших данных
(протокол №55 от 20.05.2025)*

Одобрено

*Кафедра информационных технологий
(протокол №4 от 19.05.2025)*

Москва 2025

Содержание

Наименование разделов РПД		Стр.
1.	Наименование дисциплины	5
2.	Перечень планируемых результатов освоения образовательной программы (перечень компетенций) с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине	5
3.	Место дисциплины в структуре образовательной программы	7
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	11
6.	Перечень учебно-методического обеспечения для	13

	самостоятельной работы обучающихся по дисциплине	
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	15
7.	Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	25
8.	Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	31
9.	Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	33
10.	Методические указания для обучающихся по освоению дисциплины	34
11.	Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных	34

	справочных систем	
12.	Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	35

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

Код компетенции	Наименование компетенции	Индикатор достижения компетенции	Результаты обучения
ОПК-6	Способен разрабатывать алгоритмы и программы, пригодные для практического использования, применять основы информатики и программирования к проектированию, конструированию и тестированию программных продуктов	Разрабатывает алгоритмы решения простых информационных задач и выражает их на языке программирования	Знать: 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
		Анализирует алгоритмы в части производительности, оптимальности, вырабатывает рекомендации для	Знать: технологии прикладного программирования, включая среды высокоуровневого программирования

		оптимизации алгоритмов программ	Уметь: разрабатывать программы решения задач с использованием среды высокоуровневого программирования
		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 программ-ной системы исходя

			из ее назначения
--	--	--	------------------

3. Место дисциплины в структуре образовательной программы

Дисциплина «Практикум по программированию» относится к «Общепрофессиональному циклу»

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

Очная форма обучения

Вид учебной работы по дисциплине	Всего (в з/е и часах)	Семестр 1 (в часах)	Семестр 2 (в часах)
Общая трудоемкость дисциплины	4/144	72	72
Контактная работа - Аудиторные занятия	32	16	16
<i>Лекции</i>	0	0	0
<i>Семинары, практические занятия</i>	32	16	16
Самостоятельная работа	112	56	56
Вид текущего контроля	Проектная работа; Проектная работа;	Проектная работа	Проектная работа
Вид промежуточной аттестации	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 упаковка параметров функции. Аннотации и документирование функций. 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 : с оздание классов и объектов; наследование и полиморфизм; функция super(); проверка принадлежности к классу. Базовые типы в Python.

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

Тема 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.

5.2. Учебно-тематический план

Очная форма обучения

№ п/п	Наименование тем(разделов) дисциплины	Трудоемкость в часах				
		Всего	Контактная работа - Аудиторная работа		Семинары, практические занятия	Самостоятельная работа
			Общая, в т.ч.:	Лекции		
1	Основы языка Python	24	4	0	4	20
2	Функции и модули	24	6	0	6	18
3	Обработка исключений. Работа с файлами и средствами языка Python	24	6	0	6	18
4	Объектно-ориентированное программирование на Python	26	6	0	6	20
5	Функциональное программирование	22	4	0	4	18

	ие на Python					
6	Алгоритмы и структуры данных	24	6	0	6	18
	Итого	144	32	0	32	112

5.3. Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарах, практических занятиях	Формы проведения занятий
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.	Интерактивная форма, работа на компьютере. Защита проектов.
Функции и модули	Создание и использование функций. Анонимные функции. Функции высшего порядка.	Интерактивная форма, работа на компьютере. Защита проектов.
Обработка исключений. Работа с файлами средствами языка Python	Понятие исключения. Инструкция try ... except ... else ... finally. Классы встроенных исключений. Инструкции raise и	Интерактивная форма, работа на компьютере. Защита проектов.

	assert. Инструкция with ... as. Работа с файлами в Python. Операции с файлами: открытие/закрытие файла, чтение и записи и другие методы для работы с файлами. Инструкция with ... as и ее использование для файлов. Работа с текстовыми файлами. Работа с двоичными файлами. Сохранение объектов в файл. Работа с файлами различных форматов: csv, xlsx.	
Объектно-ориентированное программирование на Python	Создание классов и объектов. Наследование и полиморфизм. Функция super(); проверка принадлежности к классу. Методы классов, статические переменные и методы в Python. Управление доступом к атрибутам класса в Python. Специальные методы классов.	Интерактивная форма, работа на компьютере. Защита проектов.
Функциональное программирование на Python	Функции map, filter, reduce, any, all. Итераторы в Python, итерируемый тип данных. Модуль itertools. Функции-генераторы и выражения-генераторы в Python. Декораторы в Python: использование и	Интерактивная форма, работа на компьютере. Защита проектов.

	создание собственных декораторов.	
Алгоритмы и структуры данных	Динамические массивы. Стеки, очереди, деки. Связные списки. Бинарные деревья. Использование бинарных деревьев в прикладных задачах. Двоичное дерево поиска. Асимптотическая оценка сложности алгоритма. Создание связных списков. Использование стеков и очередей при решении задач. Алгоритмы сортировки и поиска. Бинарный поиск. Простые методы сортировки: обменные сортировки, сортировка выбором, сортировка вставками. Сортировка Шелла. Быстрая сортировка. Сортировка слиянием.	Интерактивная форма, работа на компьютере. Защита проектов.

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
--	--	--

Основы языка Python	Функции модуля math, random, copy. Вывод текстовой информации. Регулярные выражения. Кортежи. Множества.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.
Функции и модули	Создание и использование функций. Анонимные функции. Функции высшего порядка. Создание и использование модулей и пакетов.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.
Обработка исключений. Работа с файлами средствами языка Python	Работа с файлами различных форматов: txt, pickle, csv, docx, xlsx. Работа с текстовыми файлами. Работа с двоичными файлами. Сохранение объектов в файл.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.
Объектно-ориентированное программирование на Python	Создание классов и объектов. Абстракция и инкапсуляция. Наследование и полиморфизм. Специальные методы классов. Декораторы встроенные и пользовательские.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.
Функциональное программирование на Python	Функции map, filter, reduce, sorted.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.

Python	Декораторы. Итераторы. Функции генераторы и выражения-генераторы. Функции any, all, zip.	документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.
Алгоритмы и структуры данных	Бинарные деревья. Использование бинарных деревьев в прикладных задачах. Создание связанных списков. Создание стеков и очередей для решения задач. Алгоритмы сортировки и поиска. Двоичное дерево поиска.	Работа с учебной литературой и документацией. Решение задач с использованием Jupyter Notebook или другой средой программирования. Выполнение домашних заданий. Подготовка проектов к защите.

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

Примерные задания проектных работ

Задание 1. 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.

Задание 2. 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. Модуль с базовыми операциями над таблицами, для каждой функции должно быть реализована генерация не менее одного вида исключительных ситуаций.

Задание 3. 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. Сама программа НЕ ходит: т.е. не

пытается выполнить ходы за одну из сторон, а предоставляет поочередно вводить ходы за белых и черных.

Задание 4. Реализовать программу, которая позволяет играть в шахматы на компьютере. «Шахматный симулятор: объектно-ориентированная версия».

На базе собственной реализации Задания 4 создать объектно-ориентированную реализацию программы для игры в шахматы.

Базовые требования к функциональности программы сохраняются прежними: 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.

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

Требования к реализации: Основные объекты и абстрактные сущности игры должны быть представлены в виде объектов, представителей соответствующих классов, часть классов должны быть организованы в виде иерархии. В частности: шахматные фигуры – объекты, представители классов, организованных в виде иерархии; доска – объект; ходы фигур – объекты. Вся основная информация должна храниться в атрибутах объектов или классов (например, информация о положении фигур, цвете фигур, символах, используемых для визуализации фигур и т.п.). Основная часть функционала должна программы должна быть организована в виде методов, закрепленных за соответствующими объектами или классами. Например, это касается методов определяющих допустимые ходы фигур. Организация иерархий классов, атрибутов

и методов должна позволять гибко расширять возможности программы с минимальными изменениям в уже созданном коде

Задание 5. Задача коммивояжера. Задача математического программирования по определению оптимального маршрута движения коммивояжера, цель которого состоит в том, чтобы посетить все объекты, записанные в задании, за кратчайший срок и с наименьшими затратами. В теории графов Задача коммивояжера - это поиск пути, связывающего два или более узла, с использованием критерия оптимальности.

Задача коммивояжёра (коммивояжёр — бродячий торговец) заключается в отыскании самого выгодного маршрута, проходящего через указанные города хотя бы по одному разу. В условиях задачи указываются критерий выгодности маршрута (кратчайший, самый дешёвый, совокупный критерий и т. п.) и соответствующие матрицы расстояний, стоимости и т. п. Как правило указывается, что маршрут должен проходить через каждый город только один раз, в таком случае выбор осуществляется среди гамильтоновых циклов.

Необходимо:

1. Реализовать графическую модель.
2. Реализовать метод полного перебора.
3. Представить один из методов решения.

Задание 6. «Реализация собственного пакета модулей для манипулирования плоскими фигурами». Базовые требования к функциональности программы:

Реализовать `api`, которое позволяет: генерировать, преобразовывать и визуализировать последовательность плоских полигонов, представленных в виде кортежа кортежей (например: $((0,0), (0,1), (1,1), (1,0))$ – представление для квадрата). Последовательности представлений полигонов представляют из себя итераторы (далее: последовательности полигонов). Решать задачи с использованием функционального стиля программирования, в том числе активно использовать функции из модуля `itertools` и `functools`.

- Реализовать функцию визуализации последовательности полигонов, представленной в виде итератора;
- Реализовать функции, генерирующие бесконечную последовательность не пересекающихся полигонов с различающимися координатами (например, «ленту»);
- Реализовать операции: параллельный перенос (`tr_translate`) ; поворот (`tr_rotate`) ; симметрия (`tr_symmetry`) ; гомотетия (`tr_homothety`), которые можно применить к последовательности полигонов с помощью функции `map`;
- С помощью данных функций создать и визуализировать: 3 параллельных «ленты» из последовательностей полигонов, расположенных под острым углом к оси *x*; две пересекающихся «ленты» из последовательностей полигонов, пересекающихся не в начале координат; две параллельных ленты треугольников, ориентированных симметрично друг к другу; последовательность четырехугольников в разном масштабе, ограниченных двумя прямыми, пересекающимися в начале координат.
- Реализовать операции: фильтрации фигур, являющихся выпуклыми многоугольниками (`flt_convex_polygon`); фильтрации фигур, имеющих хотя бы один угол, совпадающий с заданной точкой (`flt_angle_point`); фильтрации фигур, имеющих площадь менее заданной (`flt_square`); фильтрации фигур, имеющих кратчайшую сторону менее заданного значения (`flt_short_side`); фильтрации выпуклых многоугольников, включающих заданную точку (внутри многоугольника) (`flt_point_inside`); фильтрации выпуклых многоугольников, включающих любой из углов заданного многоугольника (`flt_polygon_angles_inside`); которые можно применить к последовательности полигонов с помощью функции `filter`.

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

Дополнительная информация:

тема "Реализация алгоритмической игры с использованием основных алгоритмических структур"

Примерные задания проектных работ

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. Начинающий 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 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.

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

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

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

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

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

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

Знать: объектно-ориентированный язык программирования Python на уровне знания синтаксиса и семантики, основ стандартной библиотеки

Уметь: определять на уровне знания синтаксис и семантику, стандартные библиотеки языка Python, необходимые для решения прикладных задач

Типовые контрольные задания

1. Напишите программу на Python, обрабатывающую текстовую строку так, чтобы в нем все заглавные буквы преобразовать в строчные.

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

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

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

Знать: технологии прикладного программирования, включая среды высоко-уровневого программирования

Уметь: разрабатывать программы решения задач с использованием среды высокоуровневого программирования

Типовые контрольные задания

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

2. 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.

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

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

Знать: 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 программ-ной системы исходя из ее назначения

Типовые контрольные задания

1. Реализуйте программный код вычисления различных арифметических операций, выбор которых осуществляется с помощью разработанного меню.
2. Реализуйте программу на Python, в 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.

Примеры практико-ориентированных заданий

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.

4. Имеется список L вида [[1, 3, 65], [11, 5, 6], [15, 33, 11]]. Отсортируйте список по возрастанию последнего значения элемента.

5. 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

6. 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 вопросы для подготовки к Зачету, Зачету

Примерные вопросы для подготовки к зачету (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.

Примерные вопросы для подготовки к зачету (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.

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

Нормативно-правовые акты:

- не предусмотрены

Основная литература:

1. Гниденко , Ирина Геннадиевна Технологии и методы программирования : учебник для вузов / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. 2-е изд. , пер. и доп Электрон. дан. Москва : Юрайт , 2025 248 с (Высшее образование) URL: <https://urait.ru/bcode/560978> (дата обращения: 08.04.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз. пользователей <https://urait.ru/bcode/560978> ISBN 978-5-534-18130-2 : 1289.00
2. Практикум по программированию на языке Python : учебное пособие для студентов, обучающихся по направлениям "Прикладная математика и информатика", "Прикладная информатика", "Информационная безопасность", "Бизнес-информатика" (программа подготовки бакалавра) / Р.И. Горохова, Е.П. Догадина, В.И. Долгов, В.И. Макрушин ; Финуниверситет, Департамент анализа данных и машинного обучения факультета информационных технологий и анализа больших данных Москва :

Финуниверситет , 2023 1 файл (6,93 Мб) Свободный доступ из сети Интернет (чтение, печать, копирование) (дата обращения 09.10.2023)
+ Текст : электронный

3. Федоров , Дмитрий Юрьевич Программирование на python : учебное пособие для вузов / Д. Ю. Федоров. 6-е изд. , пер. и доп Электрон. дан. Москва : Юрайт , 2025 187 с (Высшее образование) URL: <https://urait.ru/bcode/556864> (дата обращения: 08.04.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз. пользователей <https://urait.ru/bcode/556864> ISBN 978-5-534-19666-5 : 829.00

Дополнительная литература:

1. Языки программирования. Python: решение сложных задач [Электронный ресурс] : учебное пособие для вузов / Борзунов С. В.,Кургалин С. Д. Санкт-Петербург : Лань , 2023 192 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/319394> ISBN 978-5-507-45923-0

2. Шелудько , Виктория Михайловна Основы программирования на языке высокого уровня Python : Учебное пособие Ростов-на-Дону : Издательство Южного федерального университета (ЮФУ) , 2017 146 с. ВО - Бакалавриат <https://znanium.com/catalog/document?id=339834> ISBN 978-5-9275-2649-9

3. Программирование. Основы Python для инженеров [Электронный ресурс] : учебное пособие для вузов / Никитина Т. П.,Королев Л. В. ; Королев Л. В. 2-е изд., стер. Санкт-Петербург : Лань , 2025 156 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/454463> ISBN 978-5-507-50668-2

4. Алгоритмы и структуры данных на Python : Учебное пособие / С.А. Чернышев Электрон. дан. Москва : КноРус , 2024 326 с. Режим доступа: book.ru Internet access <https://book.ru/book/949701> ISBN 978-5-406-11683-8

5. Чернышев , Станислав Андреевич Основы программирования на Python : учебник для вузов / С. А. Чернышев. 2-е изд. , пер. и доп Электрон. дан. Москва : Юрайт , 2025 349 с (Высшее образование) URL: <https://urait.ru/bcode/567821> (дата обращения: 08.04.2025). Режим доступа: Электронно-библиотечная система Юрайт, для авториз.

пользователей <https://urait.ru/bcode/567821> ISBN 978-5-534-17139-6 : 1729.00

6. Разработка приложений с графическим пользовательским интерфейсом на языке Python [Электронный ресурс] : учебное пособие для вузов / Букунов С. В., Букунова О. В. ; Букунова О. В. Санкт-Петербург : Лань , 2023 88 с. Книга из коллекции Лань - Информатика <https://e.lanbook.com/book/292856> ISBN 978-5-507-45191-3

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

1. Электронная библиотека Финансового университета (ЭБ) <http://elib.fa.ru/> (<http://library.fa.ru/files/elibfa.pdf>)
2. Электронно-библиотечная система BOOK.RU <http://www.book.ru>
3. Информационно-образовательный портал Финуниверситета: <https://org.fa.ru>
4. Личный кабинет обучающегося <https://org.fa.ru>
5. Электронно-библиотечная система издательства «ЮРАЙТ» <https://www.biblio-online.ru/>
6. Официальный сайт компании Блумберг. URL:<http://www.bloomberg.com>
7. Python Documentation <http://python.org/doc/>
8. Python Standard Library <https://docs.python.org/2/library/>
9. Дистрибутив anaconda: <http://anaconda.org/>
10. <http://pandas.pydata.org/>
11. Библиотечно-информационный комплекс Финуниверситета (электронная библиотека, ресурсы на русском языке): http://www.library.fa.ru/res_mainres.asp?cat=rus

10. Методические указания для обучающихся по освоению дисциплины

Студентам при подготовке следует использовать нормативные документы Финансового университета, Методические рекомендации по планированию и организации внеаудиторной самостоятельной работы студентов по образовательным программам бакалавриата и магистратуры в Финансовом университете, утвержденные приказом Финуниверситета от 11.05.2021 г. № 1040/о (см. сайт Финансового Университета: на главной странице раздел "Наш университет" → "Единая правовая база Финуниверситета" → "Организация учебного процесса" → "Нормативные документы по самостоятельной работе").

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

- Комплект лицензионного программного обеспечения:

1. Kaspersky
2. Пакет офисных программ

- Современные профессиональные базы данных и информационные справочные системы:

1. Электронная энциклопедия: <http://ru.wikipedia.org/wiki/Wiki>
2. Язык программирования Python 3. <https://pythonworld.ru/>
3. Размещение ЭУК: <https://www.campus.fa.ru/>
4. Создание тестов, система Moodle, осуществление прокторинга (асинхронный, синхронный): <https://edu.fa.ru/>
5. VK-звонки
6. VK Mail

- Сертифицированные программные и аппаратные средства защиты информации

1. не предусмотрены

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

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

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

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