

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

Тульский филиал Финуниверситета

Кафедра «Математика и информатика»

СОГЛАСОВАНО
АО «ИНВЭК»

Заместитель генерального директора
по финансам


О.В. Кузина
«24» апреля 2024 г.

УТВЕРЖДАЮ

Директор Тульского филиала
Финуниверситета


Г.В. Кузнецов
«24» апреля 2024 г.

Жуков Р.А.

**ТЕХНОЛОГИИ ПРИКЛАДНОГО ПРОГРАММИРОВАНИЯ И
ОБРАБОТКИ ДАННЫХ**

Рабочая программа дисциплины

для студентов, обучающихся по направлению
38.03.0 «Экономика»,
Образовательная программа «Бизнес-анализ, налоги и аудит»,
профиль «Учет, анализ и аудит»

*Рекомендовано Ученым советом Тульского филиала Финуниверситета
(протокол от 23 апреля 2024 г. № 14)*

*Одобрено кафедрой «Математика и информатика»
(протокол от 5 марта 2024 г. № 8)*

Тула 2024

Содержание

№ п/п	Наименование разделов РПД	стр.
1.	Наименование дисциплины	3
2.	Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы	3
3.	Место дисциплины в структуре образовательной программы	4
4.	Объем дисциплины в зачетных единицах и в академических часах с выделением объема аудиторной (лекции, семинары) и самостоятельной работы обучающихся	4
5.	Содержание дисциплины, структурированное по темам (разделам) дисциплины с указанием их объемов (в академических часах) и видов учебных занятий	4
5.1.	Содержание разделов дисциплины	4
5.2.	Учебно-тематический план	6
5.3.	Содержание семинаров, практических занятий	7
6.	Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине	8
6.1.	Перечень вопросов, отводимых на самостоятельное освоение дисциплины, формы внеаудиторной самостоятельной работы	8
6.2.	Перечень вопросов, заданий, тем для подготовки к текущему контролю	8
7.	Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине	18
7.1.	Типовые контрольные задания или иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний.....	18
7.2.	Иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний	20
8.	Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины	20
9.	Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимых для освоения дисциплины	20
10.	Методические указания для обучающихся по освоению дисциплины	20
11.	Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень необходимого программного обеспечения и информационных справочных систем.....	23
11.1	Комплект лицензионного программного обеспечения	23
11.2	Современные профессиональные базы данных и информационные справочные системы	23
11.3	Сертифицированные программные и аппаратные средства защиты информации	23
12	Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине	23

1. Наименование дисциплины

Дисциплина «Технологии прикладного программирования и обработки данных»

2. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

Дисциплина «Технологии прикладного программирования и обработки данных» обеспечивает формирование следующих компетенций:

Код компетенции	Наименование компетенции	Индикаторы достижения компетенции	Результаты обучения (умения и знания), соотнесенные с индикаторами достижения компетенции
ПКП-5	Способность к использованию специальных программных продуктов, применяемых для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте	1.Использует специальные программные продукты для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте.	Знать: принципы использования специальных программных продуктов на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте Уметь: применять специальные программные продукты на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте
		2. Демонстрирует владение специальными программными продуктами, применяемыми для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте.	Знать: методологию использования специальных программных продуктов на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте Уметь: выбрать и

			применить специальные программные продукты на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте
--	--	--	--

3. Место дисциплины в структуре образовательной программы

Дисциплина «Технологии прикладного программирования и обработки данных» относится дисциплинам цикла профиля (элективного), части, формируемой участниками образовательных отношений.

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

Таблица 1 – Очно-заочная форма обучения

Вид учебной работы по дисциплине	Всего (в з.е. и часах)	Семестр 8 (в часах)
Общая трудоемкость дисциплины	3 з.е./108 час.	108
Контактная работа - Аудиторные занятия	16	16
Лекции	8	8
Семинары, практические занятия	8	8
Самостоятельная работа	92	92
Вид текущего контроля	Контрольная работа	Контрольная работа
Вид промежуточной аттестации	зачет	зачет

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

5.1. Содержание разделов дисциплины

Тема 1. Введение в программирование на языке Python

Задачи анализа данных, понятие набора данных (dataset). Подготовительные операции для выполнения анализа данных: загрузка данных, трансформация данных, изучение данных, очистка данных, визуализация данных.

Технологический стек анализа данных, построенный на базе языка программирования Python. Язык программирования Python: основные характеристики, возможности языка для решения задач анализа данных и машинного обучения. Версии языка программирования Python,

дистрибутивы и библиотеки Python. Знакомство с дистрибутивом Anaconda и составом инструментов для задач анализа данных и машинного обучения, входящих в дистрибутив. Интерактивная оболочка IPython notebook: принципы работы и применение для решения задач анализа данных и машинного обучения.

Тема 2. Основные синтаксические конструкции Python

Знакомство с типами данных и операциями, переменными. Возможности работы со строками в Python. Основные операции над строками, функции и методы для работы со строками. Структура программы. Инструкции выражений, операторы сравнения, логические операторы. Инструкция ветвления if...else. Инструкция цикла while. Инструкция цикла for и Инструкции break, continue, pass, else.

Работа со списками в Python. Создание списка. Операции над списками. Перебор элементов списка. Многомерные списки. Методы списков. Кортежи.

Работа со словарями в Python. Создание словаря. Операции над словарями. Перебор элементов словаря. Методы для работы со словарями. Множества.

Тема 3. Базовые технологии для анализа данных

Знакомство с библиотеками numpy и pandas и решением базовых задач подготовительных операций для выполнения анализа данных с помощью этих библиотек.

Постановки задач машинного обучения. Объекты и признаки. Типы признаков: бинарные, номинальные, порядковые, количественные. Типы задач машинного обучения: классификация, регрессия, прогнозирование, кластеризация. Примеры задач, решаемых методами машинного обучения. Проблема недообучения / переобучения.

Тема 4. Технологии работы со структурированными данными

Обзор технологий хранения данных: файловых систем, реляционных СУБД, OLAP, Data Warehouses, не реляционных (“NoSQL”) баз данных. Сравнительный анализ и области применения различных технологий хранения информации. Работа с файлами. Работа с реляционными базами данных на примере SQLite.

Краткий обзор основных видов не реляционных баз данных: хранилищ «ключ-значение», хранилище семейств колонок, документно-ориентированная СУБД, баз данных на основе графов. Сравнительный анализ и области применения не реляционных баз данных.

Хранение и обмен структурированной информацией в виде документов или сообщений. Форматы представления переносимой структурированной информации. Сравнение различных принципов представления структурированной информации: закрытые и открытые форматы, бинарное и текстовое представление данных.

Универсальные форматы хранения структурированной информации (разметки документов): CSV, XML, HTML (XHTML), JSON. Язык разметки XML: основные принципы построения и специфика использования. Построение схемы документа с помощью XML DTD или XML Schema. HTML

(XHTML) – отличие от XML, специфика использования. Формат представления структурированной информации JSON: принципы построения, специфика использования.

Тема 5. Технологии обработки данных

Знакомство с различными классами информационно-аналитических систем. Технологии Data Mining. Технологии анализа больших объемов данных (Big Data): причины возникновения, основные особенности функционирования и специфика создания приложений.

Сравнительный анализ различных подходов к анализу экономически значимой информации: от построения систем отчетов до алгоритмов машинного обучения. Особенности построения информационно-аналитических систем с применением алгоритмов машинного обучения. Основные этапы создания информационно-аналитических систем с использованием алгоритмов машинного обучения.

5.2. Учебно-тематический план

Таблица 2 – Для очно-заочной формы обучения

№ п/п	Наименование тем (разделов) дисциплины	Всего	Трудоемкость в часах			Самостоя тельная работа	Формы текущего контроля успеваемости
			Контактная работа - Аудиторная работа				
			Общая, в т.ч.:	Лекции	Семинары, практические занятия		
1.	Введение в программирование на языке Python	10	2	1	1	8	Устный опрос, выполнение практических заданий.
2	Основные синтаксические конструкции Python	20	2	1	1	18	Устный опрос, выполнение практических заданий. Срезочная контрольная работа.
3	Базовые технологии для анализа данных	24	4	2	2	20	Устный опрос, выполнение практических заданий.
4	Технологии работы со структурированным и данными	26	4	2	2	22	Устный опрос, выполнение практических

							заданий.
5	Технологии обработки данных	28	4	2	2	24	Устный опрос, выполнение практических заданий. Срезовая контрольная работа.
	В целом по дисциплине	108	16	8	8	92	Согласно учебному плану: контрольная работа.
	Итого в %	100	15	7,5	7,5	85	-

5.3. Содержание семинаров, практических занятий

Таблица 3

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
Тема 1. Введение в программирование на языке Python	Изучение технологического стека анализа данных, построенного на базе языка программирования Python. Рекомендуемые источники: 8.1 - 8.4	Индивидуальное выполнение заданий, обсуждение результатов
Тема 2. Основные синтаксические конструкции Python	Изучение базовых конструкций языка программирования Python. Рекомендуемые источники: 8.1 - 8.4	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий
Тема 3. Базовые технологии для анализа данных	Знакомство с информационными технологиями анализа данных Python. <i>Рекомендуемые источники: 8.1</i>	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий
Тема 4. Технологии работы со структурированными данными	Изучение примеров работы с форматами CSV, XML, XHTML, HTML, JSON при помощи библиотек на языке Python, проектирование собственного приложения работающего с одним из данных форматов. <i>Рекомендуемые источники: 8.2</i>	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий
Тема 5. Технологии обработки данных	Изучение примеров построения аналитических инструментов на языке Python, проектирование инструментария анализа данных собственного приложения <i>Рекомендуемые источники: 8.2 - 8.3.</i>	Индивидуальное выполнение заданий, групповой разбор результатов выполнения

		заданий
--	--	---------

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

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

Таблица 4

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
Тема 1. Введение в программирование на языке Python	Знакомство с интерактивной оболочкой IPython notebook. Изучение принципов работы в оболочке.	работа с литературой; работа с электронными источниками; разработка алгоритмов и программ.
Тема 2. Основные синтаксические конструкции Python	Работа с множествами, генераторы списков и словарей.	работа с литературой; работа с электронными источниками; разработка алгоритмов и программ.
Тема 3. Базовые технологии для анализа данных	Знакомство с библиотеками numpy и pandas и решением базовых задач подготовительных операций для выполнения анализа данных с помощью этих библиотек.	работа с литературой; работа с электронными источниками; разработка алгоритмов и программ.
Тема 4. Технологии работы со структурированными данными	Изучение библиотек Python для работ с данными в форматах XML, XHTML, HTML, JSON	работа с литературой; работа с электронными источниками; разработка алгоритмов и программ.
Тема 5. Технологии обработки данных	Изучение библиотек Python для анализа данных	работа с литературой; работа с электронными источниками; разработка алгоритмов и программ.

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

Примеры заданий контрольной работы

1.Процедурное программирование

1.1.Строки

1.1.1. Инвертировать последовательность слов, разделенных запятыми.

Пример: строка 'SIX, SEVEN, EIGHT, NINE, TEN' будет преобразована в: 'TEN, NINE, EIGHT, SEVEN, SIX'.

1.1.2. На основе строки, представляющей из себя предложение, построить вложенный список, содержащий символы всех слов в предложении.

Пример: строка 'Eeny, meeny, miney, moe; Catch a tiger by his toe.' будет преобразована в: [['E', 'e', 'n', 'y'], ['m', 'e', 'e', 'n', 'y'], ['m', 'i', 'n', 'e', 'y'], ['m', 'o', 'e'], ['C', 'a', 't', 'c', 'h'], ['a'], ['t', 'i', 'g', 'e', 'r'], ['b', 'y'], ['h', 'i', 's'], ['t', 'o', 'e']]

1.1.3. В строке содержащей последовательность слов, разделенных запятыми удалить все нечетные слова. Ответ представить в виде строки.

Пример: строка 'SIX,SEVEN,EIGHT,NINE,TEN' будет преобразована в: 'SIX,EIGHT,TEN'.

1.1.4. Из списка списков элементами которого являются текстовые символы собрать строку, в которой вложенные списки объединены в слова, а слова через запятую объединены в строку.

Пример список вида [['E', 'e', 'n', 'y'], ['m', 'e', 'e', 'n', 'y'], ['m', 'i', 'n', 'e', 'y'], ['m', 'o', 'e']] будет преобразован в строку 'Eeny,meeny,miney,moe'

1.2. Генераторы

1.2.1. Используя генератор списков (и не используя код вне него) преобразовать строку по следующей логике: для каждого символа исходной строки создать в итоговом списке строку, содержащую копии символа в количестве, равном номеру символа в исходной строке. Пример: 'abcd' -> ['a', 'bb', 'ccc', 'dddd']

1.2.2. Используя генератор словарей (и не используя код вне него) инвертировать словарь, т.е. сделать ключи словаря, его значениями и наоборот. Значения, которые в исходном словаре повторяются не добавлять в итоговый словарь.

Пример: {'a':1, 'b':3, 'c':4, 'd':3} -> {1:'a', 4:'c'}

1.2.3. Используя генератор словарей (и не используя код вне него) преобразовать словарь в котором ключами являются кортежи из целых чисел в словарь в котором ключом является среднее значение из чисел исходного ключа, значение оставить прежним.

Пример: {(2,4):'a', (1,1,1):'b', (2,3):'c'} -> {3.0:'a', 1.0:'b', 2.5:'c'}

1.2.4. Используя генератор списков (и не используя код вне него) преобразовать список кортежей в список кортежей по следующему правилу: если в кортеже четное количество элементов, то из него нужно удалить последний элемент. В остальных случаях кортежи оставить неизменными.

Пример: [(1,3,4), (2,1), (6,), (2,2,2,1)] -> [(1,3,4), (2,), (6,), (2,2,2,)]

1.2.5. Используя генератор списков (и не используя код вне него) преобразовать два списка (в первом содержатся целые числа, во второй строки, содержащие один символ) в словарь, в котором соответствующие друг другу пары значений из исходных списков преобразованы в целочисленный ключ и строку состоящую из повторенных символов (количество повторений равно значению ключа).

Пример [2, 4, 1, 3], ['a', 'b', 'c', 'd'] -> {2:'aa', 4:'bbbb', 1:'c', 3:'ddd'}

1.2.6. Используя генератор словарей (и не используя код вне него) преобразовать словарь в котором ключами и значениями являются целые числа в список, в котором содержатся суммы исходных пар ключей и значений, причем, в список включаются только суммы, являющиеся четными числами.

Пример: {2:4, 3:2, 12:6, 5:4, 1:3} -> [6, 18, 4]

Используя генератор списков (и не используя код вне него) преобразовать список содержащий положительные целые числа в список, элементами которого являются списки с длиной равной соответствующему числу в первом списке. Содержимым вложенных списков являются последовательно идущие целые числа начиная с 1. Пример: [3, 1, 4] -> [[1, 2, 3], [1], [1, 2, 3, 4]]

1.2.7. Используя генератор списков (и не используя код вне него) преобразовать строку по следующей логике: для каждого символа исходной строки создать в итоговом списке строку, содержащую копии символа в количестве, равном номеру символа рассчитанному с конца исходной строки.

Пример: 'abcd' -> ['aaaa', 'bbb', 'cc', 'd']

2.Объектно-ориентированное программирование

2.1.Иерархия.

2.1.1. Создать иерархию классов для фруктов, продающихся в магазине. Иерархия должна содержать не менее 5 классов, и не менее 3х уровней. Объекты должны содержать не менее 3х атрибутов и 2х методов (часть из которых должны быть перегружены). В конструкторах должны корректно использоваться конструкторы базовых классов.

Необходимо заполнить список представителями всех классов (всего не менее 10 объектов) и продемонстрировать работу полиморфизма.

2.1.2. Создать иерархию классов для фруктов, продающихся в магазине. Иерархия должна содержать не менее 3 классов. Объекты должны содержать не менее 4х атрибутов. Часть атрибутов должна быть защищена от изменения, а часть и от изменения, и от чтения. Необходимо заполнить список представителями всех классов (всего не менее 10 объектов) и продемонстрировать созданную защиту.

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

3. Функции и функциональное программирование.

3.1.Функции

3.1.1. Реализовать функцию `summlate` для расчета накопленных сумм (произведений). Функция принимает одно или более числовое значение (количество параметров заранее не определено). На основе этих значений рассчитываются накопленные суммы, которые сохраняются в списке, список возвращается как результат функции.

Пример: параметры: 1, 3, 2, 2 -> [1, 4, 6, 8]. Необязательный булевский параметр `mul` должен позволять заменять суммирование умножением.

Пример: параметры: 1, 3, 2, 2 -> [1, 3, 6, 12].

3.1.2. Реализовать функцию `gerl`, которая принимает на вход строку и набор заранее неизвестных параметров. Результатом функции является строка, в которой слова совпадающие с именами параметров заменены на значения параметров.

Пример: строка: 'replace my val abc', параметры `my='s1'`, `abc='fff'` -> Результат: 'replace s1 val fff'

3.1.3. Реализовать функцию `psort`, которая принимает на вход набор заранее неизвестных поименованных параметров. Функция возвращает список значений параметров отсортированный по именам параметров.

Пример: `psort(c=21, a=22, ac=17, b=16)` -> [22, 17, 16, 21]

3.1.4. Реализовать функцию `psort`, которая принимает на вход набор заранее неизвестных поименованных параметров. Функция возвращает список имен параметров, отсортированный по значениям параметров.

Пример: `psort(c=21, a=22, ac=17, b=16)` -> [b, ac, c, a]

3.1.5. Реализовать функцию `nam_par`, которая принимает на вход заранее неизвестное количество параметров и необязательный параметр `name` в который можно передать строку. Функция возвращает словарь в котором переданные параметры являются значениями, ключами для них являются соответствующие (сопоставленные по порядку следования) символы из строки `name`. Если строка `name` не задана, то значения присваиваются по порядку английского алфавита.

Пример 1: `nam_par(7, 3, 1, 8, 10, 13, name='xyzafg')` -> {'x':7, 'y':3, 'z':1, 'a':8, 'f':10, 'g':13}

Пример 2: `nam_par(21, 'val', -3.5)` -> {'a':21, 'b':'val', 'c':-3.5}

3.1.6. Реализовать функцию `par_val`, которая принимает на вход заранее неизвестное количество именованных параметров (значения параметров - строки) и возвращает список имен параметров, которым соответствуют строки, содержащие более двух слов. Пример: `par_val(pp='abba war', fan='oneword', zr='a x')` -> [pp, zr]

4.1. Рекурсии

4.1.1. Рекурсивно реализовать функцию `fib(n)` вычисляющую значение n -го числа Фибоначчи. Числа Фибоначчи — элементы числовой последовательности в которой каждое последующее число равно сумме двух предыдущих чисел (Числа Фибоначчи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...)

4.1.2. Создать рекурсивную реализацию функции поиска элемента в отсортированном списке методом деления пополам (бинарного поиска) `sort_search(sorted_list, value, from, to)`. Аргументы функции: `sorted_list` – отсортированный список, в котором проводится поиск; `value` – искомое значение; `from` – индекс элемента в списке, начиная с которого осуществляется поиск; `to` – индекс элемента в списке, до которого (не включительно) осуществляется поиск. Функция возвращает индекс первого вхождения `value` или `None`, в случае отсутствия элемента.

4.1.3. Создать рекурсивную реализацию функции поиска максимального числа в списке, содержащем целые числа. В качестве основного

шага рекурсии использовать переход от поиска в данном списке к двум операциям поиска в списках вдвое меньшей длины.

Функция должна иметь вид `max_search(int_list, from, to)`. Аргументы функции: `int_list` – список, содержащий целые числа; `from` – индекс элемента в списке, начиная с которого осуществляется поиск; `to` – индекс элемента в списке, до которого (не включительно) осуществляется поиск. Функция возвращает значение максимального элемента.

5.1. Декораторы

5.1.1. Реализовать декоратор, который выводит на печать возвращаемые значения функции.

5.1.2. Реализовать декоратор с именем `not_none`, который генерирует исключительную ситуацию если декорируемая функция вернула значения `None`.

5.1.3. Реализовать декоратор с именем `print_type`, выводящий на печать тип значения, возвращаемого декорируемой функцией.

map/filter/reduce

5.1.4. При помощи механизма `map/filter/reduce` возвести в квадрат числа от 1 до 100, и рассчитать их сумму, не включая в сумму числа, кратные 10.

5.1.5. Дан список целых чисел. При помощи механизма `map/filter/reduce` рассчитать остаток от деления на 17 для каждого из чисел списка и получить произведение тех остатков, величина которых больше 7.

5.1.6. Дано предложение без знаков препинания. Превратить предложение в список слов. При помощи механизма `map/filter/reduce` отбросить у каждого слова последнюю букву и склеить в одну строку те обрезанные слова, длина которых больше 5.

4. Структуры данных

4.1. Стеки, очереди

4.1.1. При помощи стека (можно использовать любую реализацию стека) проверить что в строке содержащей открывающие и закрывающие скобки трех типов: `()`, `[]`, `{}`.

Соблюдается корректный баланс и вложенность скобок.

4.1.2. Реализовать функцию `st_reverse(a_string)`, которая при помощи стека инвертирует строку (меняет порядок букв на обратный). Пример: `st_reverse('abcd') -> 'dcba'`

4.1.3. Реализовать функцию `list3_to2(list1, list2, list3)`, которая при помощи очереди превращает 3 списка одинаковой длины в 2 списка, длина которых не различается больше чем на 1, в полученных очередях должны чередоваться элементы из разных исходных списков и не должен нарушаться их порядок (т.е. если 'А' шло раньше 'В' в исходном списке, то 'В' не может идти раньше 'А' в итоговом списке).

5. Сортировки.

5.1. Простые сортировки

5.1.1. Реализовать алгоритм обменной сортировки.

5.1.2. Реализовать алгоритм сортировки выбором.

5.1.3. Реализовать алгоритм сортировки вставками.

- 5.1.4. Эффективные сортировки
- 5.1.5. Реализовать алгоритм сортировки Шелла.
- 5.1.6. Реализовать алгоритм быстрой сортировки.
- 5.1.7. Реализовать алгоритм сортировки слиянием.

Примерный перечень вопросов к контрольной работе

1. Парадигмы программирования их суть и сильные стороны. Типичные представители различных парадигм, применение различных парадигм в Python.
2. Специфика типизации в языках программирования (различные аспекты типизации). Реализация типизации в Python.
3. Именованная переменная и другие объекты в Python (правила и соглашения). Числовые типы: литералы, объявление и операции.
4. Присвоение по ссылке и по значению. Специфика создания объектов и присвоения в Python, особенности работы с объектами целочисленного типа.
5. Разница между копированием и присвоением. Проблема утечки динамической памяти, сборка мусора. Копирование, присвоение и стратегия управления динамической памятью в Python.
6. Булевский тип, сравнения и условные операторы в Python.
7. Циклы в Python, работа и устройство цикла for, типичное применение range и enumerate в цикле for.
8. Строки в Python. Принципы работы и основные операторы и функции.
9. Списки в Python. Различные способы создания и копирования списков в Python. Обход списка и поиск элементов в списке.
10. Списки в Python. Обращение к элементам списка и создание срезов. Стандартные агрегирующие функции, работающие со списками.
11. Списки в Python. Ключевые операции, проводящие к изменению списка и порождающие измененные списки.
12. Словари в Python. Основные способы создания, получения и изменения значений. Обход словарей.
13. Преобразование между словарями и списками в Python. Операции с представлениями словарей.
14. Операции со словарями, учитывающие возможное отсутствие ключа. Операции многоэлементного изменения словарей. Операции поэлементного извлечения из словаря и их использование.
15. Множества в Python. Основные способы создания, получения и изменения значений. Обход множеств.
16. Выполнение основных операций с парой множеств в Python.
17. Кортежи в Python. Отличия кортежей от списков. Распаковка и частичная распаковка кортежей.
18. Выражения генераторы и генераторы списков в Python. Использование условий в генераторах.
19. Генераторы множеств и словарей в Python. Использование

условий в генераторах.

20. Функции стандартной библиотеки для работы с контейнерами.
21. Объявление и вызов функции в Python. Параметры функции со значением по умолчанию и комментирование функции. Получение информации о функции. Способы передачи параметров при вызове функции.
22. Передача переменного количества параметров (именованных и не именованных) в функции Python. Вызов функции с позиционными параметрами, находящимися в списке, и именованными параметрами, находящимися в словаре.
23. Анонимные функции в Python их возможности и ограничения. Типичные сценарии использования анонимных функций.
24. Синтаксис и семантика обработки исключительных ситуаций в Python.
25. Создание пользовательских исключений и инструкция `assert`.
26. Базовые операции для работы с файлами в Python.
27. Использование инструкции `with ... as` на примере работы с файлами.
28. Использование модулей `pickle` и `shelve` для сохранения объектов в файл и их восстановления.
29. Модули в Python и их отличие от скриптов Python. Варианты синтаксиса импорта модуля и объектов модуля. Применение импортированных объектов. Порядок поиска модулей и специфика их загрузки. Загрузка модулей из глобального репозитория.
30. Импорт кода из пакетов. Организация пакетов в Python.
31. Концепция класса и объекта. Принципы и механизмы ООП.
32. Объявление класса, конструктор, создание объектов и одиночное наследование в Python.
33. Управление доступом к атрибутам класса в Python.
34. Полиморфизм и утиная типизация и проверка принадлежности объекта к классу в языке Python.
35. Методы классов и статические переменные и методы в Python.
36. Интроспекция и динамические операции с объектами в Python.
37. Специальные методы для использования пользовательских классов со стандартными операторами и функциями.
38. Основные возможности, поддерживаемые функциональными языками программирования. Поддержка функционального программирования в Python.
39. Концепция «функции – граждане первого класса» в языке программирования, поддержка этой концепции в Python. Специфика лямбда-функций в Python.
40. Глобальные и локальные переменные в функциях на примере Python. Побочные эффекты вызова функций и их последствия.
41. Вложенные функции и замыкания, специфика реализации в Python.
42. Функции высшего порядка и декораторы в Python.

- 43. Концепция map/filter/reduce. Работа map() в различных вариациях в Python.
- 44. Концепция map/filter/reduce. Работа filter() в Python. Использование модуля operator при работе с filter.
- 45. Концепция map/filter/reduce. Работа reduce() в Python. Использование reduce() с начальным значением, имеющим тип, отличный от возвращаемого редуцирующей функцией.
- 46. Итераторы в Python: встроенные итераторы, создание собственных итераторов, типичные способы обхода итераторов и принцип их работы.
- 47. Встроенные функции для работы с итераторами и возможности модуля itertools.
- 48. Функции генераторы и выражения генераторы: создание и применение в Python.
- 49. Специфика массивов, как структур данных. Динамические массивы – специфика работы, сложность операций. Специфика работы с array в Python.
- 50. Абстрактная структура данных стек: базовые и расширенные операции, их сложность. Реализация стека в Python.
- 51. Абстрактная структура данных очередь: базовые и расширенные операции, их сложность. Реализация очереди в Python.
- 52. Специфика реализации и скорости основных операций в очереди на базе массива и связанного списка.
- 53. Связанные списки: однонаправленные и двунаправленные. Сравнение скорости выполнения основных операций в связанных списках и в динамическом массиве.
- 54. Алгоритм сортировки выбором, сложность сортировки и возможности по ее улучшению.
- 55. Алгоритм сортировки вставками, его сложность. Алгоритм быстрого поиска в отсортированном массиве. Сложность поиска в отсортированном и не отсортированном массиве.
- 56. Алгоритм сортировки Шелла, сложность сортировки и возможности по ее улучшению.
- 57. Алгоритм быстрой сортировки, сложность сортировки и возможности по ее улучшению.
- 58. Алгоритм сортировки слиянием, сложность сортировки.

Примеры тестовых заданий

- 1. Какие ошибки здесь допущены?

```
def factorial(n):  
    if n == 0:  
        return 1  
    else:  
        return n * factorial(n - 1)
```

```
print factorial(5)
```

- Функция не может вызывать сама себя
- В коде нет никаких ошибок
- *Необходимо указать тип возвращаемого значения
- Функция всегда будет возвращать 1

2. Имеется определение класса:

```
class Ex:
```

```
def __init__(self, x, y):
```

```
    xy = x, y
```

```
    self.position = xy
```

```
    self._length = self.__len(x, y)
```

```
def __len(self, x, y):
```

```
    return abs(x) + abs(y)
```

```
def getlen(self):
```

```
    return self._length
```

```
p = Ex(1, 2)
```

Какой из вариантов его применения не допустим?

- `print p.getlen()`
- `print p.position`
- *`print p.__len(1,2)`

3. Дан массив:

```
>>> c = array([[1,2], [2,3], [4,5]])
```

Чему равен срез `c[:,1]`?

- `array([1, 2, 4])`
- *`array([2, 3, 5])`
- `array([1, 2])`
- `array([2, 3])`

4. Как называется отношение, которое имеют следующие два класса:

```
class A(object):
```

```
def __init__(self, x):
```

```
    self._mydata = x
```

```
def m1(self):
```

```
    raise NotImplementedError
```

```
class B(A):
```

```
def __init__(self, x):
```

```
    super(B, self).__init__(x)
```

```
def m1(self):
```

```
    return self._mydata
```

- агрегация. Экземпляры A содержат экземпляры класса B
- *наследование. B получается наследованием A
- ассоциация. Экземпляры A содержат ссылки на экземпляры

класса B

- наследование. А получается наследованием В

Примеры практико-ориентированных (ситуационных) заданий

1. При помощи стека (можно использовать любую реализацию стека) проверить что в строке содержащей открывающие и закрывающие скобки трех типов: (), [], {}. Соблюдается корректный баланс и вложенность скобок
2. Рекурсивно реализовать функцию fib(n) вычисляющую значение n-го числа Фибоначчи. Числа Фибоначчи — элементы числовой последовательности в которой каждое последующее число равно сумме двух предыдущих чисел (Числа Фибоначчи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...).
3. Инвертировать последовательность слов, разделенных запятыми. Пример: строка 'SIX, SEVEN, EIGHT, NINE, TEN' будет преобразована в: 'TEN, NINE, EIGHT, SEVEN, SIX'.

Критерии балльной оценки различных форм текущего контроля успеваемости

Организация текущего контроля успеваемости обучающихся по итогам семестра (модуля), по программам бакалавриата и магистратуры в соответствии с учебными планами по направлениям подготовки высшего образования утверждена приказом ректора Финансового университета по основной деятельности от 23.03.2017 №0557/о «Об утверждении Положения о проведении текущего контроля успеваемости и промежуточной аттестации обучающихся по программам бакалавриата и магистратуры в Финансовом университете».

Таблица 5 - Виды работ обучающегося, формирующие текущий контроль по дисциплине

№ п/п	Вид учебной деятельности	Баллы	Максимум за семестр (модуль)
1	Выполнение и защита контрольной работы	10	10
2	Аудиторная контрольная/срезовая работа	5	10
3	Тестирование/ опросы	0,2	10
4	Посещение занятий, ведение конспекта лекции/семинара и работа с ним	0,1	4
5	Активное участие в интерактивном процессе выполнения практических заданий на семинарском занятии	0,1	6
6	Всего за семестр (модуль)		40

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

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

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

Таблица 6

Компетенция	Типовые задания
ПКП-5 Способность к использованию специальных программных продуктов, применяемых для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте	Индикатор 1.Использует специальные программные продукты для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте. Знать: принципы использования специальных программных продуктов на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте. Уметь: применять специальные программные продукты на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте. Задание 1. Создайте рекурсивную реализацию функции поиска максимального числа в списке, содержащем целые числа. Задание 2. Напишите программу сортировки выбором. Индикатор 2.Демонстрирует владение специальными программными продуктами, применяемых для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте. Знать: методологию использования специальных программных продуктов на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте. Уметь: выбрать и применить специальные программные продукты на базе современных технологий прикладного программирования и обработки данных для выполнения бухгалтерско-аналитических и контрольных функций в экономическом субъекте. Задание 1.

	Напишите функцию <code>is_year_leap</code>, принимающую 1 аргумент — год, и возвращающую <code>True</code>, если год високосный, и <code>False</code> иначе. Задание 2. В строке, содержащей последовательность слов, разделенных запятыми, удалите все нечетные слова. Ответ представьте в виде строки. Задание 3. Разберите на отдельные составляющие текущую дату и вывести значения на экран в следующем порядке (вместо многоточий): "Сегодня: День = ..., Месяц = ..., Год =
--	---

Вопросы для подготовки к зачету

1. Встроенные числовые типы языка Python.
2. Списки. Создание, основные операции.
3. Основные методы списка.
4. Кортежи. Создание, основные методы и операции.
5. Словари. Создание, основные операции. Методы для работы со словарями.
6. Множества. Создание, основные методы и операции.
7. Переменные. Правила именования переменных.
8. Динамическая типизация.
9. Операторы сравнения и логические операторы.
10. Инструкция `if...else`.
11. Инструкция цикла `while`.
12. Инструкция цикла `for`.
13. Создание и вызов функции.
14. Передача аргументов функцию.
15. Функции-генераторы.
16. Лямбда-функции.
17. Модули. Инструкции `import` и `from`.
18. Базовые принципы объектно-ориентированного программирования.
19. Класс, метод класса, атрибут класса. Определение класса и создание экземпляра класса.
20. Конструктор и деструктор.
21. Наследование.
22. Абстрактные методы класса.
24. Статические методы класса.
25. Свойства класса.
26. Исключения. Обработка исключений.
27. Пользовательские исключения.
28. Событие. Обработчик события. Цикл обработки событий.
29. Элемент Кнопка. Создание и настройка.
30. Элемент Кнопка. Создание обработчика события.
31. Элементы Надпись и Текстовое поле. Создание и настройка. Метод `get()`.
32. Элемент Флажок. Создание, настройка, получение статуса флажка.

33. Элемент переключатель. Создание, настройка, доступ к значению.
34. Классы date, time и datetime.
35. Возможности библиотеки SymPy.
36. Возможности библиотеки NumPy.

7.2. Иные материалы, необходимые для оценки индикаторов достижения компетенций, умений и знаний

Порядок проведения промежуточной аттестации и система оценивания результатов промежуточной аттестации по итогам семестра (модуля) по программам бакалавриата и магистратуры в соответствии с учебными планами по направлениям подготовки высшего образования утверждена приказом ректора Финансового университета по основной деятельности № 0557/о от 23.03.2017.

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

Основная литература

1. Гуриков С. Р. Основы алгоритмизации и программирования на Python [Электронный ресурс]: учебное пособие. М.: ИНФРА-М, 2023. 343 с. <https://znanium.ru/catalog/product/1927269> (дата обращения: 18.04.2024).

Дополнительная литература

2. Келлехер Д. Наука о данных: базовый курс [Электронный ресурс]: учебное пособие. М.: Альпина Паблишер, 2020. 224 с. <https://biblioclub.ru/index.php?page=book&id=598235> (дата обращения: 18.04.2024).
3. Колдаев В. Д. Структуры и алгоритмы обработки данных [Электронный ресурс]: учебное пособие. М.: РИОР; ИНФРА-М, 2021. 296 с. <https://znanium.ru/catalog/product/1230215> (дата обращения: 18.04.2024).
4. Макшанов А. В., Журавлев А. Е., Тындыкарь Л. Н. Большие данные. Big Data [Электронный ресурс]. 4-е изд., стер. СПб.: Лань, 2024. 188 с. <https://e.lanbook.com/book/362318> (дата обращения: 18.04.2024).

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

1. ЭБС Знаниум - Режим доступа: <http://znanium.com/index.php>, доступ по логину и паролю. - Загл. с экрана
2. Единое окно доступа к образовательным ресурсам: портал [Электронный ресурс]. - Режим доступа : <http://window.edu.ru>. - Загл. с экрана.

10. Методические указания для обучающихся по освоению дисциплины

При изложении лекции используется проблемный подход, что значительно расширяет предоставленный материал. На преподавательском

диске находятся тексты лекций, материалы практических занятий, разбитых по темам. Там же приведены постановки задач, образцы программ решения типовых задач и справочные материалы.

Для получения доступа к облачному хранилищу студенты должны получить соответствующую ссылку от преподавателя.

При переходе к новой теме проводится тестирование, направленное на оценивание теоретических знаний. Помимо тестирования, может проводиться выборочный устный опрос студентов.

Практические навыки оцениваются путем разработки прикладных программ. Студенты должны самостоятельно и вовремя решать поставленные преподавателем задачи. Преподаватель должен отмечать и поощрять наиболее исполнительных студентов.

Примеры практических заданий для самостоятельного выполнения студентами

1. Инвертировать последовательность слов, разделенных запятыми. Пример: строка 'SIX, SEVEN, EIGHT, NINE, TEN' будет преобразована в: 'TEN, NINE, EIGHT, SEVEN, SIX'.

2. На основе строки, представляющей из себя предложение, построить вложенный список, содержащий символы всех слов в предложении. Пример: строка 'Eeny, meeny, miney, moe; Catch a tiger by his toe.' будет преобразована в: [['E', 'e', 'n', 'y'], ['m', 'e', 'e', 'n', 'y'], ['m', 'i', 'n', 'e', 'y'], ['m', 'o', 'e'], ['C', 'a', 't', 'c', 'h'], ['a'], ['t', 'i', 'g', 'e', 'r'], ['b', 'y'], ['h', 'i', 's'], ['t', 'o', 'e']]

3. Используя генератор списков (и не используя код вне него) преобразовать строку по следующей логике: для каждого символа исходной строки создать в итоговом списке строку, содержащую копии символа в количестве, равном номеру символа в исходной строке. Пример: 'abcd' -> ['a', 'bb', 'ccc', 'dddd']

4. Реализовать функцию `summate` для расчета накопленных сумм (произведений). Функция принимает одно или более числовое значение (количество параметров заранее не определено). На основе этих значений рассчитываются накопленные суммы, которые сохраняются в списке, список возвращается как результат функции. Пример: параметры: 1, 3, 2, 2 -> [1, 4, 6, 8] . Необязательный булевский параметр `mul` должен позволять заменять суммирование умножением. Пример: параметры: 1, 3, 2, 2 -> [1, 3, 6, 12]

5. Рекурсивно реализовать функцию `fib(n)` вычисляющую значение *n*-го числа Фибоначчи. Числа Фибоначчи — элементы числовой последовательности в которой каждое последующее число равно сумме двух предыдущих чисел (Числа Фибоначчи: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...).

6. Реализовать декоратор, который выводит на печать возвращаемые значения функции.

7. При помощи стека (можно использовать любую реализацию стека) проверить что в строке содержащей открывающие и закрывающие скобки трех типов: (), [], {}. Соблюдается корректный баланс и вложенность скобок

8. Реализовать функцию `st_reverse(a_string)`, которая при помощи стека инвертирует строку (меняет порядок букв на обратный). Пример: `st_reverse('abcd') -> 'dcba'`

9. Дано предложение без знаков препинания. Превратить предложение в список слов. При помощи механизма `map/filter/reduce` отбросить у каждого слова последнюю букву и склеить в одну строку те обрезанные слова, длина которых больше 5.

10. Реализовать алгоритм обменной сортировки.

Оформление контрольной работы

Контрольная работа должна быть представлена студентом в установленный графиком срок в виде отчета. Отчет оформляется на бумаге формата А 4 в печатном виде или написанным от руки с установкой следующих полей: с правой стороны — 10 мм, с левой — 30 мм, вверху — 20 мм, внизу — 25 мм. Номера страниц проставляются арабскими цифрами в верхнем правом углу. Отчет должен содержать титульный лист, содержание, введение, основную часть (2 главы), заключение, список литературы, приложения.

Нумерация разделов в контрольной работе производится арабскими цифрами с точкой. Введение и заключение не нумеруются. Точку в конце заголовка раздела не ставят. Каждый раздел начинается с новой страницы. Расстояние между заголовком и текстом — 15 мм.

Рисунки располагаются после ссылки на них в тексте. Под рисунком указывается его номер и название. Таблицы должны иметь номер и название.

Все рисунки и таблицы должны иметь сквозную нумерацию.

Формулы помещаются на отдельной строке и нумеруются цифрами в скобках с правой стороны от формулы (нумерация сквозная). Каждая формула должна содержать расшифровку условных обозначений с указанием размерности. При ссылке в тексте на формулу указывается ее номер.

Список литературы оформляется в следующем порядке: первыми указываются законы и государственные документы, затем в алфавитном порядке приводятся учебники, справочники, словари и т.п. Все литературные источники нумеруются, и в тексте дается ссылка на номер. Последовательность записи такова: фамилия и инициалы авторов; полное название книги или статьи; место выпуска и наименование издательства; год выпуска; номер журнала; количество страниц.

Приложения помещаются после списка литературы на последующих страницах. Каждое приложение начинается с нового листа с указанием в правом верхнем углу слова "Приложение" с порядковым номером. Каждое приложение имеет заголовок.

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

лицензионное отечественное:

1. Антивирусная защита Kaspersky Endpoint Security.
2. Astra Linux.
3. LibreOffice.

лицензионное импортное:

ПО для ТСО лиц с ограниченными возможностями здоровья

свободно распространяемое:

Архиватор KDE (Ark)

11.2 Современные профессиональные базы данных и информационные справочные системы

Например,

1. Информационно-правовая система «Гарант»
2. Информационно-правовая система «Консультант Плюс»
3. Информационно-образовательный портал Финансового университета: URL: <http://www.fa.ru>.

11.3 Сертифицированные программные и аппаратные средства защиты информации

Не используются.

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

Образовательный процесс по учебной дисциплине осуществляется в учебных аудиториях для проведения учебных занятий. При освоении дисциплины используются технические средства мультимедийной техники аудиторий. Для проведения учебных занятий используются аудитории, оборудованные следующими техническими средствами: мультимедиа-проектором, экраном настенным или доской интерактивной с встроенной акустической системой для воспроизведения аудио файла, персональным (-ными) компьютером (-рами) с доступом к Internet-ресурсам и электронно-образовательной среде Финансового университета.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения в сети «Интернет» и обеспечением доступа в электронно-образовательную среду Финансового университета.