Федеральное государственное образовательное бюджетное учреждение высшего образования

«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»

(Финансовый университет)

Новороссийский филиал

Кафедра «Информатика, математика и общегуманитарные науки»

УТВЕРЖДАЮ
Директор филиала
Е.Н. Сейфиева
«__25_» __марта_____ 2021 г.

Программирование и анализ данных с помощью Python

Рабочая программа дисциплины
для студентов, обучающихся по направлению подготовки
27.03.05 «Инноватика» очная форма обучения

Образовательная программа “Управление цифровыми инновациями”

Рекомендовано Ученым советом Новороссийского филиала Финуниверситета
протокол № 34 от 25 марта 2021 г.
.
Одобрено кафедрой «Информатика, математика и общегуманитарные науки»
№ 8 от 25 марта 2021 г.

Новороссийск 2021

УДК 004(073)
ББК 32.972.134

К64

Рецензент: В.И. Завгородний, д.э.н., профессор департамента анализа данных, принятия решений и финансовых технологий

Кондрашов Ю.Н. «Программирование и анализ данных с помощью Python». Рабочая программа дисциплины для студентов, обучающихся по направлению подготовки 27.03.05 «Инноватика», профиль «Управление цифровыми инновациями» очная форма обучения (программа подготовки бакалавра) – М.: Финансовый университет при Правительстве Российской Федерации, департамент «Анализа данных, принятия решений и финансовых технологий», 2019. - 32 с.

Дисциплина «Программирование и анализ данных с помощью Python» является дисциплиной «Естественнонаучного, математического и информационного модуля».

В рабочей программе дисциплины представлены цели и задачи дисциплины, требования к результатам освоения дисциплины, содержание дисциплины, тематика практических занятий и технология их проведения, формы самостоятельной работы студентов, система оценивания, учебно-методическое и информационное обеспечение дисциплины.

УДК 004(073)
ББК 32.972.134

Учебное издание
Кондрашов Юрий Николаевич
Программирование и анализ данных с помощью Python
Рабочая программа дисциплины

Компьютерный набор, верстка Ю.Н. Кондрашов

Формат 60x90/16. Гарнитура Times New Roman
Усл. п.л._____ . Изд. № ____ .Тираж - ____ экз.
Заказ №
Отпечатано в Финуниверситете

Оглавление

1. **Наименование дисциплины**

   «Программирование и анализ данных с помощью Python»

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

Код компетенции	**Наименование компетенции**	**Индикаторы достижения компетенции**	**Результаты обучения (владения, умения и знания), соотнесенные с компетенциями/индикаторами достижения компетенции**
ОПК-3	Способен использовать фундаментальные знания для решения базовых задач управления в технических системах с целью совершенствования в профессиональной деятельности	1. Демонстрирует навыки планирования целей и установления приоритетов при выборе способов принятия решений с учетом условий, средств, возможностей и временной перспективы достижения. 2.Владеет навыками применения знаний для создания приложений сервис-ориентированной архитектуры в практической и научной деятельности, методами и формами проведения научных исследований.	**Знать:** возможности языка программирования Python для решения стандартных задач профессиональной деятельности **Уметь:** применять язык программирования Python для решения стандартных задач профессиональной деятельности
УК-1	Способен осуществлять поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	1.Четко описывает состав и структуру требуемых данных и информации, грамотно реализует процессы их сбора, обработки и интерпретации 2.Обосновывает сущность происходящего,	**Знать:** возможности языка программирования Python при использовании информационно-коммуникационных технологий и баз данных **Уметь:** использовать язык программирования Python в составе информационно-коммуникационных технологий и баз данных

		выявляет закономерности, понимает природу вариабельности Формулирует признак классификации, выделяет соответствующие ему группы однородных «объектов», идентифицирует общие свойства элементов этих групп, оценивает полноту результатов классификации, показывает прикладное назначение классификационных групп. 4.Грамотно, логично, аргументировано формирует собственные суждения и оценки. Отличает факты от мнений, интерпретаций, оценок и т.д. в рассуждениях других участников деятельности. 5.Аргументированно и логично представляет свою точку зрения посредством и на основе системного описания.	

3. Место дисциплины в структуре образовательной программы

Дисциплина «Программирование и анализ данных с помощью Python» является дисциплиной Естественнонаучного, математического и информационного модуля направления подготовки 27.03.05 «Инноватика», профиль «Управление цифровыми инновациями».

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

Общая трудоёмкость дисциплины составляет 5 зачётных единиц.

Вид промежуточной аттестации – экзамен.

Вид текущего контроля – контрольная работа.

Вид учебной работы по дисциплине	**Всего (в з/ед. и часах)**	**Семестр 3 (в часах)**	**Семестр 4 (в часах)**
Общая трудоемкость дисциплины	6/216	108	108
Контактная работа – Аудиторные занятия	100	50	50
Лекции	32	16	16
Семинары, практические занятия	68	34	34
Самостоятельная работа	116	58	58
Вид текущего контроля	Проектная работа	-	Проектная работа
Вид промежуточной аттестации	Экзамен, зачет	Зачет	Экзамен

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

5.1 Содержание дисциплины

Тема 1. Введение в программирование на 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, документо-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

№ п/п	Наименование тем (разделов) дисциплины	Всего	Трудоемкость в часах				Самостоятельная работа	Формы текущего контроля успеваемости
			Аудиторная работа					
			Общая, в т.ч.	Лекции	Семинары, практические занятия	Занятия в интерактивных формах		
1.	Введение в программирование на языке Python	29	20	4	12	12	12	Устный опрос, проверка практических заданий
2	Основные синтаксические конструкции Python	39	19	7	12	12	22	Устный опрос, проверка практических заданий
3	Базовые технологии для анализа данных	43	19	5	12	12	26	Устный опрос, проверка практических заданий
4	Технологии работы со структурированными данными	53	27	9	16	16	28	Устный опрос, проверка практических заданий
5	Технологии обработки данных	52	15	7	16	16	28	Устный опрос, проверка практических заданий
В целом по дисциплине		216	100	32	68	68	116	Контрольная работа

Итого в %					68%		

5.3 Содержание семинаров, практических занятий

Наименование тем (разделов) дисциплины	Перечень вопросов для обсуждения на семинарских, практических занятиях, рекомендуемые источники из разделов 8,9 (указывается раздел и порядковый номер источника)	Формы проведения занятий
Тема 1. Введение в программирование на языке Python	Изучение технологического стека анализа данных, построенного на базе языка программирования Python. *Рекомендуемые источники:8.1; 8.2; 8.3; 8.4*	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (30% времени на интерактивные технологии)
Тема 2. Основные синтактические конструкции Python	Изучение базовых конструкций языка программирования Python. *Рекомендуемые источники:8.1; 8.2; 8.3; 8.4*	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (33% времени на интерактивные технологии)
Тема 3. Базовые технологии для анализа данных	Знакомство с информационными технологиями анализа данных Python. *Рекомендуемые источники: 8.5; 8.8; 8.9-8.12;*	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (33% времени на интерактивные технологии)
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. *Рекомендуемые источники: 8.2-8.5; 8.8-8.12;*	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (25% времени на интерактивные технологии)
Тема 5. Технологии обработки данных	Изучение примеров построения аналитических инструментов на языке Python, проектирование инструментария анализа данных собственного приложения *Рекомендуемые источники: 8.2 - 8.5; 8.8 - 8.12*	Индивидуальное выполнение заданий, групповой разбор результатов выполнения заданий (25% времени на интерактивные технологии)

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

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

Наименование тем (разделов) дисциплины	Перечень вопросов, отводимых на самостоятельное освоение	Формы внеаудиторной самостоятельной работы
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 синтактические 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 базо вых 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

Примерный вариант 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 ключем 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 неизменнымии.

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 втором 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]

1.2.7. Используя генератор списков (и не используя код вне него)

преобразовать список, содержащий положительные целые числа в список, элементами которого являются списки с длинной 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]]

1.2.8. 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 cummulate 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 repl , 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 значе ниям 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 па раметр name в который можно передать строку. Функция возвра щает словарь в котором переданные параметры являются значени ями, ключами для них являются соответсвующие (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]

1.1. Рекурсии

1.1.1. Рекурсивно реализовать функцию fib(n) вычисляющую значение n-го числа Фибоначи. 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, …).

1.1.2. 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.

1.1.3. 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. 1.2.Декораторы

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

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

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

1.3. map/filter/reduce

1.3.1. 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.

1.3.2. 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.

1.3.3. 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.

2. Структуры данных

2.1. Стеки, очереди

2.1.1. При помощи стека (можно исопльзовать любую реализа цию стека) проверить что в строке содержащей открывающие и закрывающие скобки трех типов: (), [], {}.

Соблюдается корректный баланс и вложенность скобок.

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

2.1.3. 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 ‘A’ шло раньше ‘B’ в исходном списке, то ‘B’ не может идти раньше ‘A’ в итоговом списке.

3. Сортировки.

3.1. Простые сортировки

3.1.1. Реализовать алгоритм обменной сортировки.

3.1.2. Реализовать алгоритм сортировки выбором.

3.1.3. Реализовать алгоритм сортировки вставками.

3.2. Эффективные сортировки

3.2.1. Реализовать алгоритм сортировки Шелла.

3.2.2. Реализовать алгоритм быстрой сортировки.

3.2.3. 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. Именование переменных и других объектов 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 вариация в Python.

44. Концепция map/filter/reduce. Работа filter() в Python. Использования 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 работа 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.

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 с указанием индикаторов их достижения и планируемых результатов обучения по дисциплине».**

Типовые контрольные задания или иные материалы,

необходимые для оценки уровня сформированности компетенций

Код компетенции	Наименование компетенции	Примеры заданий для оценки сформированности компетенции
ОПК-7	Способен использовать информационнокоммуникационные компьютерные технологии, базы данных, пакеты прикладных программ для решения инженерно-технических и техникоэкономических задач планирования и управления работами по инновационным проектам	**Задание 1.** Написать программу для вычисления факториала циклом. **Задание 2.** Написать программу сортировки выбором
ОПК-10	Способен разрабатывать и применять алгоритмы и программные приложения для решения практических задач цифровизации в области профессиональной деятельности	**Задание 1.** Разобрать на отдельные составляющие текущую дату и вывести значения на экран в следующем порядке (вместо многоточий): "Сегодня: День = …, Месяц = …, Год = **Задание 2.** Создать пример иерархии классов. Иерархия должна содержать не менее 3 классов, и не менее 2 уровней. **Задание 3.** Написать функцию is_year_leap, принимающую 1 аргумент — год, и возвращающую True, если год високосный, и False иначе. **Задание 4.** Имеется список названий месяцев: [‘января’, ‘февраля’, ‘марта’, ‘апреля’, ’мая’, ‘июня’, ‘июля’, ‘августа’, ‘сентября’, ‘октября’, ‘ноября’, ‘декабря’]. Создайте по этому списку словарь, в котором название месяца будет ключом, а номер месяца (от 1 до 12) – значением. Используя полученный словарь преобразуйте строку с датой вида «1 января 2016» в строку «1.01.2016»

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 возвращать1

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. A получается наследованием B

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 Фибоначи. 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'.

Примерные вопросы для подготовки к экзамену

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.

23. Статические методы класса.

24. Свойства класса.

25. Исключения. Обработка исключений.

26. Пользовательские исключения.

27. Событие. Обработчик события. Цикл обработки событий.

28. Элемент Кнопка. Создание и настройка.

29. Элемент Кнопка. Создание обработчика события.

30. Элементы Надпись и Текстовое поле. Создание и настройка. Метод get().

31. Элемент Флажок. Создание, настройка, получение статуса флажка.

32. Элемент переключатель. Создание, настройка, доступ к значению.

33. Классы date, time и datetime.

34. Возможности библиотеки SymPy.

35. Возможности библиотеки NumPy.

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

Основная литература:

1. Колдаев В.Д. Основы алгоритмизации и программирования [Электронный ресурс]: учебное пособие / В.Д. Колдаев; под ред. Л.Г. Гагариной. – Москва: ИД ФОРУМ: ИНФРА-М, 2012, 2015, 2017. – 416 с. – Режим доступа: http://znanium.com/go.php?id=902236

2. Колдаев В.Д. Структуры и алгоритмы обработки данных [Электронный ресурс]: учебное пособие / В.Д. Колдаев. – Москва: ИЦ РИОР: НИЦ ИНФРА-М, 2014. – 296 с. – Режим доступа: http://znanium.com/catalog.php?bookinfo=418290

Дополнительная литература

3. Прохоренок Н.А. Python. Самое необходимое / Н.А. Прохоренок. – Санкт-Петербург: БХВ-Петербург, 2011. – 416 с.

4. Прохоренок Н.А. Python и PyQt. Разработка приложений / Н.А. Про-хоренок. – Санкт-Петербург: БХВ-Петербург, 2012.

5. Маккинли У. Python и анализ данных / У. Маккинли; пер. с англ. А.А. Слинкин. – Москва: ДМК Пресс, 2015.

6. Прохоренок Н.А. Python 3. Самое необходимое / Н.А. Прохоренок, В.А. Дронов. – Санкт-Петербург: БХВ-Петербург, 2016. – 464 с.

7. Доусон М. Программируем на Python / М. Доусон; пер. с англ. – Санкт-Петербург: Питер, 2015. – 416 с.

8. Лутц М. Изучаем Python, / М. Лутц; пер. с англ. – 4-е издание. – Санкт-Петербург: Символ-Плюс, 2011. – 1280 с.

9. Лутц М. Программирование на Python, том I / М. Лутц; пер. с англ.– 4-е издание. — Санкт-Петербург: Символ-Плюс, 2011. – 992 с.

10. Лутц М. Программирование на Python, том II / М. Лутц; пер. с англ.– 4-е издание. – Санкт-Петербург: Символ-Плюс, 2011. – 992 с.

11. SciPy // http://docs.scipy.org/doc/scipy/reference/

12. NumPy User Guide // http://docs.scipy.org/doc/numpy/user/index.html

13. The Python Standard Library //

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

1. Pylru 1.0.9 [Электронный ресурс]: сайт. – Режим доступа: https://pypi.python.org/pypi/pylru

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

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

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

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

6. Официальный сайт продукта https://www.python.org/

7. Информационно-образовательный портал Финансового университета при Правительстве Российской Федерации http://portal.ufrf.ru/

8. Каталог курсов Интернет Университета Информационных Технологий http://www.intuit.ru/

9. Электронная библиотека Финансового университета (ЭБ) http://elib.fa.ru/ (http://library.fa.ru/files/elibfa.pdf)

10. Электронно-библиотечная система Znanium http://www.znanium.com

10. Методические указания 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.

Значительная часть семинарских занятий проводятся в интерактивном режиме с подробным обсуждением изучаемых тем. Активная работа компьютерных классах и самостоятельная работа являются обязательным условием формирования знаний, умений и навыков самостоятельного проектирования и сопровождения баз данных.

Большое значение в образовательном процессе придается выполнению индивидуальных заданий на основе вариантов и контрольной работы. Результаты выполнения индивидуальных заданий завершается представлением результатов преподавателю.

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

11. 1. Комплект лицензионного программного обеспечения:

1. Windows, Microsoft Office.
2. Антивирус ESET Endpoint Security

11.2. Современные профессиональные базы данных и информационные справочные системы

1. Информационно-правовая система «Гарант»
2. Информационно-правовая система «Консультант Плюс»
3. Электронная энциклопедия: http://ru.wikipedia.org/wiki/Wiki
4. Система комплексного раскрытия информации «СКРИН» - http://www.skrin.ru/

11.3. Сертифицированные программные и аппаратные средства защиты информации – не используются.

11.4. Браузер Google Chrom.

11.5. Дистрибутив языка Python 3.4 (или более поздней версии)

11.6. Anaconda 3

11.7. Для манипулирования с файлами файловый менеджер Far

11.8. Архиватор.

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

Лекционные занятия проводятся в мультимедийных аудиториях, а семинарские занятия – в компьютерных классах.