

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

Владикавказский филиал Финуниверситета

УТВЕРЖДАЮ

Заместитель директора
по учебно-методической работе
Владикавказского филиала
Финуниверситета

З. Айларова З.К. Айларова
«31» августа 2026 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине

«ОП.13 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»

по специальности 09.02.13 Интеграция решений с применением технологий
искусственного интеллекта

Владикавказ 2026

Фонд оценочных средств по дисциплине разработан на основе Федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.13 Интеграция решений с применением технологий искусственного интеллекта.

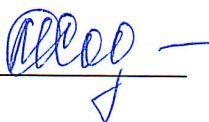
Разработчик:

Зембатова М.А., преподаватель

Фонд оценочных средств по дисциплине рассмотрен и рекомендован к утверждению на заседании предметной (цикловой) комиссии математики и информатики

Протокол от «28» 08 2026 г. № 1

Председатель предметной (цикловой)
комиссии математики и информатики



М.К. Ходова

1. Паспорт фонда оценочных средств
по дисциплине «ОП.13 Основы алгоритмизации и программирования»
специальность 09.02.13 Интеграция решений с применением технологий
искусственного интеллекта

Результаты обучения (знания, умения)	Общие и профессиональные компетенции	Наименование темы	Наименование оценочного средства	
			Текущий контроль успеваемости	Промежуточная аттестация
Знания: – определение алгоритма; – представление алгоритма; – виды алгоритмических структур (линейные, разветвляющиеся, циклические); – правила построения блок-схем. Умения: – составлять блок-схемы линейных алгоритмов; – составлять блок-схемы разветвляющихся алгоритмов; – составлять блок-схемы циклических алгоритмов с условием и со счетчиком.	ОК 01 ОК 02 ОК 09 ПК 1.1	Тема 1.1. Понятие алгоритмизации и алгоритма. Виды алгоритмических структур.	Вопросы для устного (письменного) опроса по теме «Понятие алгоритмизации и алгоритма. Виды алгоритмических структур» Комплект практических заданий по темам: – «Составление блок-схем линейных алгоритмов» – «Составление блок-схем разветвляющихся алгоритмов» – «Составление блок-схем циклических алгоритмов с условием» – «Составление блок-схем циклических алгоритмов со счетчиком» Самостоятельная работа студентов. Реферат, доклад Тест по разделу «Основы алгоритмизации»	Вопросы для устного (письменного) зачета по дисциплине
Знания: – понятие переменной и константы; – команда print() и её аргументы; – комментарии в	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.1. Переменные. Ввод, вывод данных и базовые операции.	Вопросы для устного (письменного) опроса по теме «Переменные. Ввод, вывод данных и базовые операции»	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине

Python; – операции с числами. Умения: – использовать среду программирования; – писать первую программу; – применять команды ввода/вывода данных.			Комплект практических заданий по темам: – «Знакомство со средой программирования. Написание первой программы» Самостоятельная работа студентов. Реферат, доклад	
Знания: – основные числовые типы данных; – тип данных int; – тип данных float; – преобразование типов. Умения: – выполнять операции с числовыми данными; – осуществлять преобразование числовых типов.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.2. Числовые типы данных.	Вопросы для устного (письменного) опроса по теме «Числовые типы данных» Комплект практических заданий по темам: – «Операции с числовыми данными» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – строковый тип данных str; – конкатенация и умножение строк; – преобразование чисел в строку; – срезы строк и шаг. Умения: – выполнять операции со строками; – использовать срезы строк.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.3. Строковый тип данных.	Вопросы для устного (письменного) опроса по теме «Строковый тип данных» Комплект практических заданий по темам: – «Операции со строками» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – список (list); – создание списков; – индексация и	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.4. Списки (List).	Вопросы для устного (письменного) опроса по теме «Списки (List)»	Вопросы и задания для устного (письменного) дифференцирова

срезы; – добавление и удаление элементов; – основные методы списков. Умения: – создавать и модифицировать списки; – применять методы списков.			Комплект практических заданий по темам: – «Операции над списками: создание и модификация» Самостоятельная работа студентов. Реферат, доклад	ного зачета по дисциплине
Знания: – кортежи (tuple); – отличия от списков; – неизменяемость кортежей; – распаковка кортежей. Умения: – выполнять операции над кортежами; – использовать распаковку кортежей.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.5. Кортежи (Tuple).	Вопросы для устного (письменного) опроса по теме «Кортежи (Tuple)» Комплект практических заданий по темам: – «Операции над кортежами» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – множества (set); – свойства множеств; – функция set(); – замороженное множество (frozenset); – основные операции над математическим и множествами. Умения: – выполнять операции над математическим и множествами; – использовать frozenset.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.6. Множества (Set).	Вопросы для устного (письменного) опроса по теме «Множества (Set)» Комплект практических заданий по темам: – «Операции над математическими множествами» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – словарь (dict); – создание словаря; – операции над	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 2.7. Словари (Dict).	Вопросы для устного (письменного) опроса по теме «Словари (Dict)»	Вопросы и задания для устного (письменного) дифференцирова

словарями; – добавление, удаление и изменение элементов. Умения: – выполнять операции над словарями; – управлять элементами словарей.			Комплект практических заданий по темам: – «Операции над словарями» Самостоятельная работа студентов. Реферат, доклад Тест по разделу «Основы языка программирования Python»	ного зачета по дисциплине
Знания: – арифметические операторы; – остаток от деления; – операторы сравнения; – инкремент и декремент; – приоритет операций. Умения: – реализовывать линейный алгоритм с арифметическим и вычислениями.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 3.1. Арифметические операторы.	Вопросы для устного (письменного) опроса по теме «Арифметические операторы» Комплект практических заданий по темам: – «Линейный алгоритм с арифметическими вычислениями» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – логический тип данных; – логическое умножение (and); – логическое сложение (or); – логическое отрицание (not); – оператор принадлежности (in, not in). Умения: – работать с логическими операторами и условиями.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 3.2. Логические операторы.	Вопросы для устного (письменного) опроса по теме «Логические операторы» Комплект практических заданий по темам: – «Работа с логическими операторами и условиями» Самостоятельная работа студентов. Реферат, доклад»	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – условные операторы if-	ОК 01 ОК 02 ОК 09	Тема 3.3. Условный оператор.	Вопросы для устного (письменного) опроса по теме	Вопросы и задания для устного

elif-else; – особенности работы булевых значений; – правильное оформление отступов; – проверка равенства различных типов данных; – цепочки сравнения. Умения: – реализовывать разветвляющиеся алгоритмы; – решать задачи с цепочками сравнения.	ПК 1.1 ПК 1.3		«Условный оператор» Комплект практических заданий по темам: – «Разветвляющиеся алгоритмы» – «Решение задач с цепочками сравнения» Самостоятельная работа студентов. Реферат, доклад Тест по разделу «Операторы Python»	(письменного) дифференцированного зачета по дисциплине
Знания: – основы цикла for; – переменная цикла for; – синтаксис цикла for; – итерируемые типы данных; – управляющие конструкции. Умения: – программировать циклические алгоритмы с параметром; – работать с итерируемыми типами данных; – использовать range() и последовательности.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 4.1. Цикл for.	Вопросы для устного (письменного) опроса по теме «Цикл for» Комплект практических заданий по темам: – «Программирование циклических алгоритмов: цикл с параметром» – «Итерируемые типы данных» – «Итерируемый тип данных range. Работа с последовательностями» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – цикл while; – синтаксис цикла while; – бесконечный цикл; – управление выполнением цикла.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 4.2. Цикл While.	Вопросы для устного (письменного) опроса по теме «Цикл While» Комплект практических заданий по темам: – «Программирование	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине

Умения: – программировать циклические алгоритмы с предусловием; – программировать вложенные циклы.			циклических алгоритмов: цикл с предусловием» – «Программирование вложенных циклов» Самостоятельная работа студентов. Реферат, доклад	
Знания: – операторы break и continue; – вложенные циклы; – оптимизация циклических процессов. Умения: – решать сложные задачи с использованием вложенных циклов; – управлять потоком выполнения циклов.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 4.3. Управление циклами.	Вопросы для устного (письменного) опроса по теме «Управление циклами» Комплект практических заданий по темам: – «Решение сложных задач с использованием вложенных циклов» Самостоятельная работа студентов. Реферат, доклад Тест по разделу «Циклы» Контрольная работа №4 по разделу «Циклы»	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – понятие функции; – популярные встроенные функции (type, len, sum, range, sorted, reversed, max, min, abs, round). Умения: – использовать встроенные функции для обработки данных.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 5.1. Встроенные функции Python.	Вопросы для устного (письменного) опроса по теме «Встроенные функции Python» Комплект практических заданий по темам: – «Использование встроенных функций для обработки данных» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – методы строк, списков, кортежей, множеств,	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 5.2. Методы типов данных.	Вопросы для устного (письменного) опроса по теме «Методы типов данных»	Вопросы и задания для устного (письменного) дифференцирова

словарей; – разница между функциями и методами. Умения: – работать со строковыми методами; – работать с методами списков и словарей; – выполнять комплексную обработку данных с использованием методов.			Комплект практических заданий по темам: – «Работа со строковыми методами» – «Работа с методами списков и словарей» – «Комплексная обработка данных с использованием методов» Самостоятельная работа студентов. Реферат, доклад	нного зачета по дисциплине
Знания: – функции без параметров; – локальные и глобальные переменные; – ключевое слово global; – функции с параметрами; – параметры по умолчанию. Умения: – создавать и вызывать функции без параметров; – создавать функции с различными типами параметров.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 5.3. Пользовательские функции.	Вопросы для устного (письменного) опроса по теме «Пользовательские функции» Комплект практических заданий по темам: – «Создание и вызов функции без параметров» – «Создание функции с различными типами параметров» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – функции с возвратом значения return; – lambda-функции; – синтаксис и применение lambda; – рекурсивные функции. Умения: – использовать оператор return	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 5.4. Возврат значения и анонимные функции.	Вопросы для устного (письменного) опроса по теме «Возврат значения и анонимные функции» Комплект практических заданий по темам: – «Оператор return для возврата значений из функций» – «Создание и	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине

для возврата значений; – создавать и применять лямбда-функции; – реализовывать рекурсию.			применение лямбда-функций и рекурсии» Самостоятельная работа студентов. Реферат, доклад Тест по разделу «Функции и модули»	
Знания: – процедурное программирование и ООП; – принципы ООП; – классы в Python; – объекты класса; – основы синтаксиса создания класса. Умения: – создавать первые классы и объекты.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 6.1. Введение в ООП. Классы и объекты.	Вопросы для устного (письменного) опроса по теме «Введение в ООП. Классы и объекты» Комплект практических заданий по темам: – «Создание первых классов и объектов» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – конструктор класса init ; – инициализация объектов; – атрибуты класса и экземпляра; – динамические атрибуты; – методы класса; – параметр self . Умения: – работать с атрибутами класса и экземпляра; – использовать self в методах.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 6.2. Атрибуты и методы класса.	Вопросы для устного (письменного) опроса по теме «Атрибуты и методы класса» Комплект практических заданий по темам: – «Атрибуты класса и экземпляра. Работа с self » Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – инкапсуляция; – приватные и защищенные атрибуты; – свойства (properties);	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 6.3. Инкапсуляция и свойства.	Вопросы для устного (письменного) опроса по теме «Инкапсуляция и свойства» Комплект	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине

– геттеры и сеттеры. Умения: – реализовывать инкапсуляцию; – использовать свойства (@property).			практических заданий по темам: – «Реализация инкапсуляции и свойств» Самостоятельная работа студентов. Реферат, доклад	
Знания: – наследование классов; – переопределение методов; – полиморфизм; – множественное наследование. Умения: – реализовывать наследование в иерархии классов; – применять полиморфизм и переопределять методы.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 6.4. Наследование и полиморфизм.	Вопросы для устного (письменного) опроса по теме «Наследование и полиморфизм» Комплект практических заданий по темам: – «Реализация наследования в иерархии классов» – «Полиморфизм и переопределение методов» Самостоятельная работа студентов. Реферат, доклад	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
Знания: – магические методы (str, add и др.); – абстрактные базовые классы (модуль abc); – декоратор @abstractmethod; – интерфейсы. Умения: – перегружать операторы с помощью магических методов; – создавать абстрактные классы и реализовывать интерфейсы.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 6.5. Магические методы и абстрактные классы.	Вопросы для устного (письменного) опроса по теме «Магические методы и абстрактные классы» Комплект практических заданий по темам: – «Перегрузка операторов и магические методы» – «Создание абстрактных классов и реализация интерфейсов» Самостоятельная работа студентов. Реферат, доклад Тест по разделу «Объектно-ориентированное программирование»	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине

Знания: – чтение и запись текстовых файлов; – режимы открытия файлов; – контекстный менеджер with open(); – форматы CSV и JSON; – обработка исключений (try-except). Умения: – читать и записывать текстовые файлы; – обрабатывать данные в форматах CSV и JSON; – защищать программу от ошибок ввода и отсутствия файлов; – решать комплексные задачи по сохранению и загрузке данных приложения.	ОК 01 ОК 02 ОК 09 ПК 1.1 ПК 1.3	Тема 7.1. Файлы и исключения.	Вопросы для устного (письменного) опроса по теме «Файлы и исключения» Комплект практических заданий по темам: – «Чтение и запись текстовых файлов» – «Обработка данных в форматах CSV и JSON» – «Защита программы от ошибок ввода и отсутствия файлов» – «Комплексная задача: сохранение и загрузка данных приложения» Самостоятельная работа студентов. Реферат, доклад на тему: « Тест по разделу «Работа с файлами и обработка данных»»	Вопросы и задания для устного (письменного) дифференцированного зачета по дисциплине
---	--	--	---	---

2. Комплект оценочных средств

1. Задания для текущего контроля успеваемости

Вопросы для устного (письменного) опроса (ОК 01, ОК 02, ОК 09, ПК 1.1, ПК 1.3)

Тема 1.1. Понятие алгоритмизации и алгоритма. Виды алгоритмических структур.

1. Дайте определение понятия «алгоритм». Перечислите основные свойства алгоритма.
2. Какие существуют способы представления алгоритмов? В чем их преимущества и недостатки?
3. Что такое блок-схема? Назовите основные геометрические фигуры, используемые при построении блок-схем, и их назначение.
4. Что такое линейная алгоритмическая структура? Приведите пример задачи, решаемой с помощью линейного алгоритма.
5. Что такое разветвляющаяся алгоритмическая структура? Какие виды условий могут использоваться в развилках?
6. Что такое циклическая алгоритмическая структура? Чем отличается цикл с предусловием от цикла с постусловием?
7. Что такое цикл со счетчиком (параметром)? Как он изображается на блок-схеме?
8. Что такое тело цикла? Что такое условие выхода из цикла?
9. Может ли алгоритм не иметь ни одного входного или выходного параметра? Обоснуйте ответ.
10. Что понимается под детерминированностью и массовостью алгоритма?

Тема 2.1. Переменные. Ввод, вывод данных и базовые операции.

1. Что такое переменная в программировании? Чем переменная отличается от константы?
2. Каков синтаксис команды присваивания в Python? Что находится слева, а что справа от знака '='?
3. Для чего используется функция `print()`? Перечислите ее основные аргументы (`'sep'`, `'end'`).
4. Какие функции используются для ввода данных с клавиатуры в Python? Какой тип данных возвращает функция `input()`?

5. Что такое комментарий в коде? Какие виды комментариев существуют в Python?
6. Что произойдет, если попытаться сложить строку и число без явного преобразования типов?
7. Как создать переменную, не присваивая ей `None` значения (если это возможно), или как обозначить отсутствие значения?
8. Что такое динамическая типизация в Python?
9. Как вывести несколько значений в одной строке через запятую?
10. Можно ли использовать зарезервированные слова Python (например, `if`, `for`) в качестве имен переменных?

Тема 2.2. Числовые типы данных.

1. Какие основные числовые типы данных существуют в Python?
2. В чем отличие типа `int` от типа `float` с точки зрения хранения в памяти и точности вычислений?
3. Что такое преобразование типов (type casting)? Какие функции используются для приведения к `int` и `float`?
4. Что произойдет при выполнении операции `int(3.9)` и `int(-3.9)`?
5. Почему результат выражения `0.1 + 0.2` в Python не всегда равен точно `0.3`?
6. Какие арифметические операции можно выполнять над целыми числами? Над вещественными?
7. Что такое оператор целочисленного деления `//` и оператор остатка от деления `%`?
8. Как возвести число в степень в Python?
9. Можно ли смешивать в одном выражении типы `int` и `float`? Каков будет тип результата?
10. Существуют ли ограничения на размер целых чисел в Python 3?

Тема 2.3. Строковый тип данных.

1. Что такое строковый тип данных `str`? Являются ли строки изменяемыми объектами в Python?
2. Что такое конкатенация строк? Какой оператор используется для этого?
3. Что означает операция умножения строки на число?
4. Что такое индексация строк? С какого индекса начинается нумерация символов?
5. Что такое срез (slice) строки? Объясните синтаксис `[start:stop:step]`.
6. Как получить последний символ строки, не зная её длины?

7. Что такое f-строки (форматированные строки)? В чем их преимущество перед обычной конкатенацией?
8. Как преобразовать число в строку и наоборот?
9. Что такое экранирование символов в строках? Приведите примеры специальных символов (``\n``, ``\t``).
10. Можно ли изменить отдельный символ в строке по индексу после её создания?

Тема 2.4. Списки (List).

1. Что такое список в Python? Является ли он изменяемым типом данных?
2. Как создать пустой список? Как создать список с начальными элементами?
3. Чем список отличается от кортежа и множества?
4. Какие методы позволяют добавлять элементы в список (``append``, ``insert``, ``extend``)? В чем их разница?
5. Как удалить элемент из списка по значению и по индексу?
6. Что такое отрицательная индексация в списках?
7. Как найти количество элементов в списке? Как найти индекс первого вхождения элемента?
8. Что происходит при присваивании одного списка другому (``a = b``)? Создается ли копия?
9. Как отсортировать список? В чем разница между методом ``sort()`` и функцией ``sorted()``?
10. Можно ли хранить в одном списке элементы разных типов данных?

Тема 2.5. Кортежи (Tuple).

1. Что такое кортеж? Зачем нужны неизменяемые последовательности?
2. Как создать кортеж с одним элементом? Почему важна запятая?
3. Можно ли изменить элемент кортежа после создания? А можно ли изменить объект, находящийся внутри кортежа (например, список)?
4. Что такое распаковка кортежа (unpacking)? Приведите пример.
5. Какие операции поддерживают кортежи? Отличаются ли они от операций со списками?
6. В каких случаях целесообразно использовать кортеж вместо списка?
7. Как проверить наличие элемента в кортеже?
8. Можно ли использовать кортеж в качестве ключа словаря? Почему?
9. Что такое именованный кортеж (namedtuple)?
10. Как объединить два кортежа?

Тема 2.6. Множества (Set).

1. Что такое множество в Python? Какие главные свойства отличают его от списка?
2. Как создать пустое множество? Почему нельзя использовать `{}`?
3. Что такое замороженное множество (`frozenset`)? Чем оно отличается от обычного `set`?
4. Какие математические операции над множествами поддерживает Python (объединение, пересечение, разность)?
5. Как добавить элемент во множество? Как удалить элемент?
6. Что произойдет, если попытаться добавить в множество дублирующийся элемент?
7. Можно ли хранить во множестве списки или словари? Почему?
8. Для каких практических задач удобно использовать множества (например, поиск уникальных значений)?
9. Как проверить, является ли одно множество подмножеством другого?
10. Как преобразовать список в множество и обратно? Сохранится ли порядок элементов?

Тема 2.7. Словари (Dict).

1. Что такое словарь? Из каких элементов он состоит?
2. Какие требования предъявляются к ключам словаря? Могут ли ключи быть изменяемыми объектами?
3. Как получить значение по ключу? Что произойдет, если ключа нет в словаре? Как избежать ошибки?
4. Как добавить новую пару «ключ-значение» в словарь? Как обновить существующее значение?
5. Как удалить элемент из словаря?
6. Какие методы позволяют перебирать ключи, значения и пары словаря?
7. Что такое вложенные словари? Приведите пример структуры.
8. Как объединить два словаря (в современных версиях Python)?
9. Для чего используется метод `.get()`?
10. Можно ли использовать список в качестве значения словаря?

Тема 3.1. Арифметические операторы.

1. Перечислите основные арифметические операторы в Python.
2. В чем разница между операторами `/` и `//`?
3. Что такое приоритет операций? Как изменить порядок выполнения операций?

4. Что делают операторы инкремента и декремента в других языках, и есть ли они в Python? Как реализовать аналогичное поведение?
5. Как работает оператор остатка от деления `%` для отрицательных чисел?
6. Что такое составные операторы присваивания (`+=`, `-=`, `\*=` и т.д.)?
7. Как выполнить деление с округлением до определенного количества знаков?
8. Что вернет выражение `10 // 3` и `10 % 3`?
9. Можно ли применять арифметические операторы к строкам? К каким именно?
10. Как возвести число в квадрат и извлечь квадратный корень?

Тема 3.2. Логические операторы.

1. Что такое булев тип данных (`bool`)? Какие значения он принимает?
2. Как работают логические операторы `and`, `or`, `not`?
3. Что такое «ленивые вычисления» (short-circuit evaluation) в логических операторах?
4. Какие объекты в Python считаются «ложными» (falsy) в логическом контексте?
5. Что такое оператор принадлежности `in` и `not in`?
6. Как проверить, что переменная равна одному из нескольких значений?
7. В чем разница между операторами `==` и `is`?
8. Как инвертировать логическое выражение?
9. Что вернет выражение `True and False or True`?
10. Можно ли применять логические операторы к числам и строкам? Какой будет результат?

Тема 3.3. Условный оператор.

1. Каков синтаксис конструкции `if-elif-else`?
2. Обязательно ли наличие блоков `elif` и `else`?
3. Что такое отступы в Python и почему они критически важны для условных операторов?
4. Что такое цепочки сравнения (chained comparisons)? Приведите пример.
5. Как записать условие «если x больше 5 И меньше 10»?
6. Что такое тернарный оператор (условное выражение) в Python?
7. Можно ли вкладывать одни условные операторы в другие? Есть ли ограничение на глубину вложенности?
8. Как проверить, является ли строка пустой, с помощью условия?

9. Что такое «truthy» и «falsy» значения в условиях?
10. Как обработать ситуацию, когда пользователь ввел некорректные данные, используя условный оператор?

Тема 4.1. Цикл for.

1. Каков синтаксис цикла ``for`` в Python?
2. Что такое итерируемый объект? Приведите примеры.
3. Для чего используется функция ``range()``? Объясните параметры ``start``, ``stop``, ``step``.
4. Как перебрать элементы списка с помощью цикла ``for``?
5. Как получить одновременно индекс и значение элемента при переборе (функция ``enumerate``)?
6. Можно ли изменять список, по которому идет итерация, прямо внутри цикла ``for``? К каким последствиям это приведет?
7. Что такое генератор списков (list comprehension)? В чем его преимущество перед обычным циклом?
8. Как перебрать несколько списков одновременно (функция ``zip``)?
9. Чем цикл ``for`` в Python отличается от цикла ``for`` в C++ или Pascal?
10. Что произойдет, если передать в ``range()`` отрицательный шаг?

Тема 4.2. Цикл While.

1. Каков синтаксис цикла ``while``?
2. В чем принципиальное отличие цикла ``while`` от цикла ``for``?
3. Что такое бесконечный цикл? В каких случаях он может быть полезен?
4. Как гарантировать завершение цикла ``while``?
5. Можно ли использовать цикл ``while`` для перебора элементов коллекции?
6. Что такое цикл с постусловием и как его эмулировать в Python?
7. Как прервать выполнение цикла ``while`` досрочно?
8. Можно ли использовать условие ``True`` в заголовке цикла ``while``?
9. Что произойдет, если условие цикла ``while`` изначально ложно?
10. Как реализовать меню пользователя с помощью цикла ``while``?

Тема 4.3. Управление циклами.

1. Для чего используется оператор ``break``?
2. Для чего используется оператор ``continue``? В чем его отличие от ``break``?
3. Что такое блок ``else`` у цикла? Когда он выполняется?

4. Как выйти сразу из нескольких вложенных циклов?
5. Что такое флаг (flag variable) и как он используется для управления циклами?
6. Можно ли использовать `'break'` и `'continue'` в цикле `'for'`?
7. Как пропустить первую итерацию цикла?
8. Что произойдет, если использовать `'continue'` в последней итерации цикла?
9. Как оптимизировать вложенные циклы для снижения временной сложности?
10. В каких случаях лучше заменить вложенные циклы на встроенные функции или библиотеки?

Тема 5.1. Встроенные функции Python.

1. Что такое функция в программировании?
2. Для чего используется функция `'len()'`? С какими типами данных она работает?
3. Что делает функция `'type()'`?
4. В чем разница между функциями `'sum()'`, `'max()'`, `'min()'`?
5. Для чего нужна функция `'sorted()'`? Возвращает ли она новый список или изменяет старый?
6. Что делает функция `'reversed()'`? Является ли её результат списком?
7. Для чего используется функция `'abs()'`?
8. Что делает функция `'round()'`? Как работает округление до четного (banker's rounding)?
9. Можно ли передавать в встроенные функции пользовательские объекты?
10. Где найти документацию по всем встроенным функциям Python?

Тема 5.2. Методы типов данных.

1. В чем фундаментальное отличие метода от функции?
2. Назовите основные методы строк для поиска и замены подстрок.
3. Какие методы позволяют проверять содержание строки (`'isdigit'`, `'isalpha'` и др.)?
4. Как разделить строку на список подстрок? Как объединить список в строку?
5. Назовите методы добавления и удаления элементов списка.
6. Как отсортировать список на месте?
7. Какие методы есть у словарей для безопасного получения значений?
8. Что делает метод `'update()'` у словаря?

9. Можно ли вызывать методы у литералов (например, `"hello".upper()`)?
10. Как узнать все доступные методы для объекта?

Тема 5.3. Пользовательские функции.

1. Как объявить функцию в Python? Какое ключевое слово используется?
2. Что такое параметры и аргументы функции? В чем разница?
3. Что такое возвращаемое значение? Какое значение возвращается по умолчанию?
4. Что такое локальные и глобальные переменные?
5. Для чего используется ключевое слово `global`? Почему его использование считается плохой практикой?
6. Что такое параметры по умолчанию? В чем опасность использования изменяемых объектов в качестве значений по умолчанию?
7. Что такое `*args` и `kwargs`?
8. Можно ли передать функцию в качестве аргумента другой функции?
9. Что такое docstring? Зачем он нужен?
10. Как вызвать функцию, определенную ниже по тексту программы?

Тема 5.4. Возврат значения и анонимные функции.

1. Сколько значений может вернуть функция? Как вернуть несколько значений?
2. Что такое lambda-функция? В чем её ограничения?
3. В каких случаях уместно использовать lambda-функции?
4. Что такое рекурсия? Какие две части обязательны в рекурсивной функции?
5. Что такое «база рекурсии» и «шаг рекурсии»?
6. Что произойдет, если забыть базу рекурсии?
7. Что такое стек вызовов и переполнение стека (RecursionError)?
8. Чем рекурсия отличается от итерации? Что эффективнее?
9. Можно ли присвоить lambda-функцию переменной? Стоит ли так делать?
10. Как использовать lambda вместе с функциями `map`, `filter`, `sorted`?

Тема 6.1. Введение в ООП. Классы и объекты.

1. Что такое объектно-ориентированное программирование?
2. Что такое класс? Что такое объект (экземпляр)?
3. В чем разница между процедурным и ООП подходом?
4. Как создать класс в Python?
5. Как создать экземпляр класса?

6. Что такое атрибут класса и атрибут экземпляра?
7. Можно ли создавать классы динамически?
8. Что такое пространство имен класса?
9. Как проверить, принадлежит ли объект к определенному классу?
10. Зачем нужно ООП при решении сложных задач?

Тема 6.2. Атрибуты и методы класса.

1. Что такое конструктор `__init__`? Когда он вызывается?
2. Что такое параметр `self`? Почему он обязателен в методах?
3. Как определить метод класса?
4. В чем разница между атрибутом класса и атрибутом экземпляра при обращении через `self`?
5. Можно ли добавлять атрибуты объекту после его создания?
6. Что такое статический метод (`@staticmethod`)?
7. Что такое метод класса (`@classmethod`)? Чем он отличается от обычного метода?
8. Как удалить атрибут объекта?
9. Можно ли переопределить метод родительского класса в дочернем?
10. Что такое `__dict__` объекта?

Тема 6.3. Инкапсуляция и свойства.

1. Что такое инкапсуляция? Зачем она нужна?
2. Как обозначаются защищенные (`_protected`) и приватные (`__private`) атрибуты в Python?
3. Является ли приватность в Python жестким ограничением доступа? Что такое name mangling?
4. Что такое геттеры и сеттеры?
5. Для чего используется декоратор `@property`?
6. Как создать свойство только для чтения?
7. Можно ли добавить валидацию данных в сеттер?
8. Зачем скрывать внутреннюю реализацию класса от пользователя?
9. Что такое публичный интерфейс класса?
10. Как обратиться к приватному атрибуту извне класса (технически)?

Тема 6.4. Наследование и полиморфизм.

1. Что такое наследование? Какие преимущества оно дает?
2. Кто такой родительский (базовый) класс и кто такой дочерний (производный)?

3. Как вызвать конструктор родительского класса из дочернего (`super()`)?
4. Что такое переопределение методов (overriding)?
5. Что такое полиморфизм? Приведите пример.
6. Что такое множественное наследование?
7. Что такое MRO (Method Resolution Order)? Как посмотреть порядок разрешения методов?
8. Что такое абстрактный метод?
9. Можно ли наследоваться от встроенных классов Python?
10. Что такое композиция и агрегация? Чем они отличаются от наследования?

Тема 6.5. Магические методы и абстрактные классы.

1. Что такое магические (dunder) методы? Как они называются?
2. Для чего используется метод `__str__`? Чем он отличается от `__repr__`?
3. Как перегрузить оператор сложения для своего класса?
4. Что такое абстрактный базовый класс (ABC)?
5. Для чего используется модуль `abc` и декоратор `@abstractmethod`?
6. Можно ли создать экземпляр абстрактного класса?
7. Зачем нужны интерфейсы в Python?
8. Какие еще магические методы вы знаете (`__len__`, `__getitem__`, `__call__`)?
9. Как сделать объект вызываемым (callable)?
10. Что произойдет, если не реализовать все абстрактные методы в дочернем классе?

Тема 7.1. Файлы и исключения.

1. Зачем нужна работа с файлами в программах?
2. Какие режимы открытия файлов существуют (`'r'`, `'w'`, `'a'`, `'x'`, `'b'`, `'t'`)?
3. Почему важно закрывать файл после работы с ним?
4. Что такое контекстный менеджер `with open(...)`? Какие гарантии он дает?
5. Как прочитать весь файл целиком? Как читать построчно?
6. Как записать данные в файл? В чем разница между `write()` и `writelines()`?
7. Что такое CSV и JSON? В чем их различия?
8. Как читать и записывать JSON-файлы в Python?
9. Что такое исключение (exception)? Зачем нужна обработка ошибок?

10. Как работает конструкция `try-except-else-finally`?
11. Можно ли перехватывать конкретные типы исключений? Зачем это нужно?
12. Что произойдет, если файл не существует при открытии в режиме `r`?
13. Как безопасно обработать ошибку ввода некорректных данных пользователем?
14. Что такое кодировка файла? Почему важно указывать `encoding='utf-8'`?
15. Как проверить существование файла перед открытием?

Критерии оценки устного (письменного) опроса

Оценка «отлично» выставляется, если обучающийся:

- полно раскрыл содержание материала в объеме, предусмотренном программой;
- изложил материал грамотным языком в определенной логической последовательности, точно используя терминологию дисциплины и символику;
- правильно выполнил рисунки, схемы, сопутствующие ответу;
- показал умение иллюстрировать теоретические положения конкретными примерами, применяя их в новой ситуации;
- продемонстрировал усвоение ранее изученных сопутствующих вопросов, сформированность и устойчивость используемых при ответе умений и навыков;
- выполнял работу самостоятельно без помощи преподавателя.

Возможны одна - две неточности при освещении второстепенных вопросов или выкладках, которые учащийся легко исправил по замечанию преподавателя.

Оценка «хорошо» выставляется, если обучающийся:

- полно раскрыл содержание материала в объеме, предусмотренном программой;
- изложил материал грамотным языком в определенной логической последовательности, точно используя терминологию дисциплины и символику;
- правильно выполнил рисунки, схемы, сопутствующие ответу;
- показал умение иллюстрировать теоретические положения конкретными примерами, применяя их в новой ситуации;

- продемонстрировал усвоение ранее изученных сопутствующих вопросов, сформированность и устойчивость используемых при ответе умений и навыков;

- выполнял работу самостоятельно без помощи преподавателя.

Ответ при этом имеет один из недостатков:

- в изложении допущены небольшие пробелы, не исказившие содержание ответа;

- допущены один - два недочета при освещении основного содержания ответа, исправленные по замечанию преподавателя;

- допущена ошибка или более двух недочетов при освещении второстепенных вопросов или выкладках, легко исправляемые по замечанию преподавателя.

Оценка «удовлетворительно» выставляется, если:

- неполно или непоследовательно раскрыто содержание материала, но показано общее понимание вопроса;

- имелись затруднения или допущены ошибки в определении понятий, использовании терминологии, схемах, выкладках, исправленные после нескольких наводящих вопросов преподавателя.

Оценка «неудовлетворительно» выставляется, если:

- не раскрыто основное содержание учебного материала;

- обнаружено незнание или непонимание учащимся большей или наиболее важной части учебного материала;

- допущены ошибки в определении понятий, при использовании терминологии, в рисунках или схемах, в выкладках, которые не исправлены после нескольких наводящих вопросов преподавателя.

**Комплекты практических заданий по темам
(ОК 01, ОК 02, ОК 09, ПК 1.1, ПК 1.3)**

Тема 1.1. Понятие алгоритмизации и алгоритма. Виды алгоритмических структур.

Практическое занятие «Составление блок-схем линейных алгоритмов»

1. Составить блок-схему вычисления площади треугольника по трем сторонам (формула Герона). Входные данные: стороны a , b , c . Выходные данные: площадь S .

2. Составить блок-схему обмена значениями двух переменных X и Y с использованием третьей вспомогательной переменной Z .
3. Составить блок-схему перевода температуры из градусов Цельсия (C) в градусы Фаренгейта (F) по формуле $F = C \times 9/5 + 32$.
4. Составить блок-схему вычисления периметра прямоугольника по заданным длинам сторон a и b .

Практическое занятие «Составление блок-схем разветвляющихся алгоритмов»

1. Составить блок-схему определения наибольшего из трех чисел A , B , C .
2. Составить блок-схему решения квадратного уравнения $ax^2 + bx + c = 0$ с учетом всех возможных случаев ($D > 0$, $D = 0$, $D < 0$).
3. Составить блок-схему проверки принадлежности точки с координатами (x, y) заштрихованной области (например, круг радиусом R с центром в начале координат).
4. Составить блок-схему определения тарифа на электроэнергию: до 100 кВт·ч — одна цена, свыше 100 кВт·ч — повышенная цена.

Практическое занятие «Составление блок-схем циклических алгоритмов с условием»

1. Составить блок-схему вычисления суммы ряда $S = 1 + 1/2 + 1/3 + \dots + 1/n$, где n вводится пользователем. Цикл выполняется пока счетчик $\leq n$.
2. Составить блок-схему нахождения первого члена геометрической прогрессии, превышающего заданное число M ($b_1 = 1, q = 2$).
3. Составить блок-схему алгоритма Евклида для нахождения НОД двух натуральных чисел A и B (цикл пока $A \neq B$).
4. Составить блок-схему подсчета количества цифр в целом положительном числе N (деление на 10 в цикле пока $N > 0$).

Практическое занятие «Составление блок-схем циклических алгоритмов со счетчиком»

1. Составить блок-схему вычисления факториала числа N ($N! = 1 \times 2 \times \dots \times N$).
2. Составить блок-схему вывода таблицы значений функции $y = x^2 - 3x + 5$ на отрезке $[a, b]$ с шагом h .
3. Составить блок-схему вычисления среднего арифметического N введенных чисел.

4. Составить блок-схему поиска минимального элемента в последовательности из N чисел.

Тема 2.1. Переменные. Ввод, вывод данных и базовые операции. Практическое занятие «Знакомство со средой программирования. Написание первой программы»

1. Написать программу, которая запрашивает у пользователя имя и возраст, а затем выводит приветствие: «Привет, [Имя]! Тебе [Возраст] лет. Через 10 лет тебе будет [Возраст+10] лет».

2. Написать программу, вычисляющую площадь круга по введенному радиусу. Использовать константу `math.pi`. Результат вывести с двумя знаками после запятой.

3. Написать программу конвертера валют: ввод суммы в рублях и курса доллара, вывод эквивалента в долларах. Обработать случай ввода некорректных данных (без try-ехсепт, только проверка типов).

4. Создать скрипт, который принимает три числа и выводит их сумму, произведение и среднее арифметическое в формате f-string.

Тема 2.2. Числовые типы данных. Практическое занятие «Операции с числовыми данными»

1. Дано целое число N. Найти сумму его цифр. (Использовать операции // и %).

2. Написать программу, определяющую, является ли введенное год високосным (кратен 4, но не кратен 100, либо кратен 400).

3. Реализовать калькулятор индекса массы тела (ИМТ): $I = m/h^2$, $I = m/h^2$. Вывести категорию веса (недостаточный, нормальный, избыточный) на основе диапазонов ИМТ.

4. Написать программу перевода секунд в формат «Часы:Минуты:Секунды» (например, 3661 сек -> 1:01:01).

Тема 2.3. Строковый тип данных. Практическое занятие «Операции со строками»

1. Написать программу, проверяющую, является ли введенная строка палиндромом (читается одинаково слева направо и справа налево, без учета регистра).

2. Дана строка текста. Подсчитать количество слов в ней (разделитель — пробел). Удалить лишние пробелы между словами.

3. Написать функцию шифрования строки методом Цезаря (сдвиг каждой буквы на N позиций по алфавиту).

4. Извлечь из строки формата «Фамилия Имя Отчество» инициалы (например, «Иванов Иван Иванович» -> «Иванов И.И.»).

Тема 2.4. Списки (List). Практическое занятие «Операции над списками: создание и модификация»

1. Дан список целых чисел. Найти максимальный и минимальный элементы, а также их индексы (без использования встроенных max/min/index).
2. Написать программу, удаляющую все дубликаты из списка, сохраняя порядок первых вхождений элементов.
3. Реализовать сортировку списка методом «пузырька» (Bubble Sort) вручную.
4. Создать список списков (матрицу 3x3). Написать программу транспонирования этой матрицы.

Тема 2.5. Кортежи (Tuple). Практическое занятие «Операции над кортежами»

1. Дан кортеж координат точек плоскости. Найти точку, наиболее удаленную от начала координат.
2. Написать функцию, принимающую список и возвращающую кортеж из двух элементов: (сумма четных элементов, сумма нечетных элементов).
3. Распаковать кортеж (name, age, city) в отдельные переменные и вывести информацию о человеке. Обработать ошибку, если длина кортежа не совпадает с количеством переменных.
4. Преобразовать список словарей [{'id':1, 'val':10}, ...] в словарь {1: 10, ...}, используя генератор словарей и распаковку.

Тема 2.6. Множества (Set). Практическое занятие «Операции над математическими множествами»

1. Даны два списка студентов, сдавших экзамены по математике и физике. Найти студентов, сдавших оба экзамена (пересечение), хотя бы один (объединение) и только математику (разность).
2. Написать программу проверки уникальности элементов в списке путем преобразования его во множество. Если длины не равны — вывести дубликаты.
3. Реализовать симметрическую разность двух множеств (элементы, входящие только в одно из множеств) без использования оператора ^.
4. Проверить, является ли одно множество подмножеством другого, используя методы .issubset() и оператор <=.

Тема 2.7. Словари (Dict). Практическое занятие «Операции над словарями»

1. Дан текст. Подсчитать частоту встречаемости каждого слова и вывести топ-5 самых частых слов.
2. Написать программу «Телефонная книга»: добавление, удаление, поиск контакта по имени. Хранение данных в словаре {имя: телефон}.
3. Объединить два словаря цен на товары. Если товар есть в обоих, сложить цены; если нет — добавить как есть.
4. Инвертировать словарь {ключ: значение} в {значение: ключ}. Учесть случай, когда значения повторяются (создать список ключей для одного значения).

Тема 3.1. Арифметические операторы. Практическое занятие «Линейный алгоритм с арифметическими вычислениями»

1. Написать программу вычисления корней квадратного уравнения с использованием комплексных чисел (модуль `cmath`) при отрицательном дискриминанте.
2. Реализовать конвертер систем счисления: перевод десятичного числа в двоичное, восьмеричное и шестнадцатеричное (используя арифметические операции деления с остатком).
3. Написать программу расчета сложных процентов: $S = P(1 + r/n)^{nt}$. Ввод, ставки, частоты начисления и срока.
4. Вычислить расстояние между двумя точками на плоскости (x_1, y_1) и (x_2, y_2) по формуле.

Тема 3.2. Логические операторы. Практическое занятие «Работа с логическими операторами и условиями»

1. Написать программу проверки корректности пароля: длина ≥ 8 , есть заглавная буква, есть цифра, нет пробелов. Использовать логические операторы `and`, `or`, `not`.
2. Реализовать логику работы светофора: ввод цвета $\rightarrow$ вывод действия. Обработать некорректный ввод через `else`.
3. Написать программу определения сезона по номеру месяца (1-12). Использовать цепочки сравнений или оператор `in`.
4. Проверить, лежит ли точка (x, y) внутри прямоугольника с диагональю $(x_1, y_1)-(x_2, y_2)$, используя составное условие.

Тема 3.3. Условный оператор. Практическое занятие «Разветвляющиеся алгоритмы»

1. Написать программу классификации треугольников по сторонам (равносторонний, равнобедренный, разносторонний) с проверкой существования треугольника.
2. Реализовать меню банковского терминала: выбор операции (баланс, снятие, пополнение) через if-elif-else.
3. Написать программу определения знака зодиака по дате рождения (день и месяц).
4. Решить систему линейных неравенств графически (вывести текстовое описание области решений).

Практическое занятие «Решение задач с цепочками сравнения»

1. Написать программу сортировки трех чисел по возрастанию, используя только цепочки сравнений (без циклов и списков).
2. Определить четверть координатной плоскости, в которой находится точка (x, y), используя chained comparisons ($x > 0$ and $y > 0$ и т.д.).
3. Проверить, попадает ли оценка студента в диапазон «хорошо» ($4 \leq \text{grade} < 5$) или «отлично» ($\text{grade} \geq 5$), используя цепочки.
4. Написать валидатор даты: проверить корректность дня, месяца и года (учесть високосные годы) одной сложной логической конструкцией.

4. Циклы

Тема 4.1. Цикл for. Практическое занятие «Программирование циклических алгоритмов: цикл с параметром»

1. Вывести таблицу умножения на N (от 1 до 10) в столбик.
2. Найти сумму всех четных чисел от 1 до 100.
3. Написать программу, выводящую первые N чисел Фибоначчи.
4. Построить пирамиду из символов * высотой N строк.

Практическое занятие «Итерируемые типы данных»

1. Дан список слов. Вывести только те слова, длина которых больше 5 символов.
2. Пройтись по словарю оценок студентов и вывести имена тех, у кого оценка ниже 3.
3. Перебрать строку и заменить все гласные буквы на символ *.
4. Создать генератор списка квадратов чисел от 1 до N, которые делятся на 3.

Практическое занятие «Итерируемый тип данных range. Работа с последовательностями»

1. Вывести числа от N до 1 в обратном порядке с шагом -2.
2. Сгенерировать список из 20 случайных чисел в диапазоне [-50, 50] и найти среди них положительные.
3. Написать программу, выводящую индекс и значение каждого элемента списка с помощью enumerate.
4. Создать последовательность дат (день, месяц) на текущий месяц с помощью range и модуля datetime.

Тема 4.2. Цикл While. Практическое занятие «Программирование циклических алгоритмов: цикл с предусловием»

1. Написать игру «Угадай число»: компьютер загадывает число, пользователь угадывает. Цикл работает пока ответ неверен.
2. Реализовать алгоритм поиска НОД методом вычитания (while a != b).
3. Написать программу возведения числа A в степень N через умножение в цикле while.
4. Симуляция накопления денег: пока сумма < цели, добавлять ежемесячный взнос и процент. Вывести количество месяцев.

Практическое занятие «Программирование вложенных циклов»

1. Вывести таблицу Пифагора (умножение 10x10) с выравниванием столбцов.
2. Найти все простые числа в диапазоне от 2 до N (вложенный цикл для проверки делимости).
3. Построить шахматную доску 8x8 из символов # и ..
4. Найти все тройки Пифагора (a, b, c), где $a^2 + b^2 = c^2$ и $c < 100$.

Тема 4.3. Управление циклами. Практическое занятие «Решение сложных задач с использованием вложенных циклов»

1. Написать программу поиска элемента в двумерном массиве (матрице). При нахождении — прервать оба цикла (break + флаг или исключение).
2. Реализовать пропуск четных итераций во внешнем цикле и нечетных во внутреннем (continue).
3. Написать бесконечный цикл обработки команд пользователя с возможностью выхода по команде «exit» (break).
4. Оптимизировать поиск делителей числа N: перебирать только до $\sqrt{N}$ и использовать break при нахождении первого составного делителя.

Тема 5.1. Встроенные функции Python. Практическое занятие «Использование встроенных функций для обработки данных»

1. Дан список температур. Используя `map` и `lambda`, перевести их из Цельсия в Фаренгейт.
2. Отфильтровать список строк, оставив только те, что начинаются с заглавной буквы (`filter + str.startswith`).
3. Найти самое длинное слово в предложении, используя `max` с параметром `key=len`.
4. Отсортировать список словарей сотрудников по зарплате (по убыванию), используя `sorted` и `lambda`.

Тема 5.2. Методы типов данных. Практическое занятие «Работа со строковыми методами»

1. Написать программу нормализации ФИО: привести к виду «Иванов Иван Иванович» (удаление лишних пробелов, капитализация).
2. Разбить многострочный текст на предложения (разделитель `!?`) и посчитать количество слов в каждом.
3. Заменить в тексте все email-адреса на `[EMAIL]`, используя методы поиска и замены.
4. Проверить, состоит ли пароль только из букв и цифр (`isalnum`), и содержит ли он спецсимволы.

Практическое занятие «Работа с методами списков и словарей»

1. Реализовать стек (LIFO) на основе списка: методы `push` (`append`), `pop`, `peek`.
2. Написать программу слияния двух отсортированных списков в один отсортированный (алгоритм `merge`).
3. Обновить словарь настроек программы значениями из файла (метод `.update()`), сохраняя старые ключи, если новых нет.
4. Получить список всех уникальных значений из словаря, где значения — списки.

Практическое занятие «Комплексная обработка данных с использованием методов»

1. Парсинг CSV-строки вручную: разбить по запятым, учесть кавычки, преобразовать типы данных.
2. Анализ логов: извлечь IP-адреса, посчитать частоту обращений, вывести топ-10 нарушителей.

3. Трансформация данных: список объектов JSON -> список словарей -> агрегация статистики.

4. Валидация формы регистрации: проверка email, телефона, пароля с использованием цепочки методов строк.

Тема 5.3. Пользовательские функции. Практическое занятие «Создание и вызов функции без параметров»

1. Написать функцию `print_menu()`, выводящую меню программы.

2. Создать функцию `generate_password()`, возвращающую случайный пароль из 8 символов.

3. Написать функцию `get_current_time()`, возвращающую текущее время в формате HH:MM:SS.

4. Реализовать функцию очистки консоли `clear_screen()` (кроссплатформенно).

Практическое занятие «Создание функции с различными типами параметров»

1. Написать функцию `calculate(a, b, operation='sum')`, поддерживающую '+', '-', '*', '/'. Параметр `operation` по умолчанию.

2. Создать функцию `concatenate(*args, sep=' ')`, объединяющую любое количество строк.

3. Написать функцию `create_user(**kwargs)`, создающую словарь пользователя из именованных аргументов.

4. Реализовать функцию логирования `log(message, level='INFO', *tags)`, принимающую сообщение, уровень и теги.

Тема 5.4. Возврат значения и анонимные функции. Практическое занятие «Оператор return для возврата значений из функций»

1. Написать функцию `solve_quadratic(a, b, c)`, возвращающую корни уравнения или сообщение об отсутствии решений.

2. Создать функцию `divmod_custom(a, b)`, возвращающую кортеж (частное, остаток) без использования встроенной `divmod`.

3. Написать функцию `find_element(lst, target)`, возвращающую индекс элемента или -1, если не найден.

4. Реализовать функцию `safe_divide(a, b)`, возвращающую результат деления или `None` при делении на ноль.

Практическое занятие «Создание и применение лямбда-функций и рекурсии»

1. Написать lambda-функцию для сортировки списка кортежей (name, age) по возрасту.
2. Использовать map с lambda для возведения элементов списка в квадрат.
3. Реализовать рекурсивную функцию вычисления факториала factorial(n).
4. Написать рекурсивную функцию flatten(lst), превращающую вложенный список любой глубины в плоский.

Тема 6.1. Введение в ООП. Классы и объекты. Практическое занятие «Создание первых классов и объектов»

1. Создать класс Book с атрибутами title, author, year. Добавить метод display_info().
2. Создать класс Counter с атрибутом value и методами increment(), decrement(), reset().
3. Реализовать класс Point для работы с координатами на плоскости.
4. Создать класс Student и instantiate 3 объекта, поместить их в список и вывести информацию.

Тема 6.2. Атрибуты и методы класса. Практическое занятие «Атрибуты класса и экземпляра. Работа с self»

1. Модифицировать класс Student: добавить атрибут класса university_name и счетчик созданных студентов.
2. Создать класс BankAccount с приватным балансом и методами deposit, withdraw, get_balance.
3. Реализовать класс Vector с методами сложения, вычитания и скалярного произведения векторов.
4. Написать класс TemperatureConverter со статическими методами для конвертации C->F и F->C.

Тема 6.3. Инкапсуляция и свойства. Практическое занятие «Реализация инкапсуляции и свойств»

1. Создать класс Person с property age: запретить установку возраста < 0 и > 150.
2. Реализовать класс EmailValidator с property email: автоматическая проверка формата при установке.
3. Написать класс ReadOnlyConfig, где атрибуты можно задать только при создании, но нельзя изменить потом.

4. Создать класс SafeList, наследующий list, но с проверкой типов добавляемых элементов через setter.

Тема 6.4. Наследование и полиморфизм. Практическое занятие «Реализация наследования в иерархии классов»

1. Создать базовый класс Shape и дочерние Circle, Rectangle, Triangle. Переопределить метод area().
2. Реализовать иерархию Employee -> Manager, Developer. Добавить специфичные атрибуты и методы.
3. Создать класс Vehicle и наследников Car, Bike. Реализовать метод move() с разной логикой.
4. Написать класс Notification и наследников EmailNotification, SMSNotification.

Практическое занятие «Полиморфизм и переопределение методов»

1. Написать функцию print_area(shape), принимающую любой объект Shape и выводящую его площадь (полиморфизм).
2. Реализовать метод __str__ и __repr__ для классов иерархии Shape.
3. Создать список разношерстных объектов (Car, Bike, Person) и вызвать у них общий метод describe().
4. Переопределить метод __eq__ для класса Student (сравнение по ID, а не по ссылке).

Тема 6.5. Магические методы и абстрактные классы. Практическое занятие «Перегрузка операторов и магические методы»

1. Реализовать класс Matrix с перегрузкой операторов +, -, * (умножение на число и матричное умножение).
2. Создать класс MyList с поддержкой индексации [], срезов и итерации (__getitem__, __iter__).
3. Написать класс CallableClass, экземпляры которого можно вызывать как функции (__call__).
4. Реализовать контекстный менеджер Timer для замера времени выполнения блока кода (__enter__, __exit__).

Практическое занятие «Создание абстрактных классов и реализация интерфейсов»

1. Создать абстрактный базовый класс Serializer с методами serialize, deserialize. Реализовать JsonSerializer, XMLSerializer.

2. Написать абстрактный класс `DatabaseConnection` с методами `connect`, `query`, `close`.
3. Реализовать интерфейс `Comparable` с методом `compare_to` для сортировки пользовательских объектов.
4. Создать абстрактный класс `Animal` с абстрактным методом `make_sound`. Попытаться создать экземпляр `Animal` (ошибка) и реализовать `Dog`, `Cat`.

Тема 7.1. Файлы и исключения. Практическое занятие «Чтение и запись текстовых файлов»

1. Написать программу копирования содержимого одного текстового файла в другой с заменой табуляций на 4 пробела.
2. Реализовать функцию `count_words_in_file(filename)`, подсчитывающую количество слов в файле.
3. Написать программу нумерации строк файла: чтение `source.txt` -> запись в `numbered.txt` с номерами строк.
4. Создать логгер, записывающий действия пользователя в файл `log.txt` с временными метками.

Практическое занятие «Обработка данных в форматах CSV и JSON»

1. Прочитать CSV-файл с данными о студентах, найти средний балл и сохранить результат в JSON.
2. Написать программу конвертации JSON-конфига в CSV-таблицу.
3. Реализовать сохранение и загрузку состояния игры (`score`, `level`, `inventory`) в JSON-файл.
4. Парсинг API-ответа (JSON): извлечение нужных полей и запись в структурированный CSV.

Практическое занятие «Защита программы от ошибок ввода и отсутствия файлов»

1. Написать `robust`-функцию чтения файла: обработка `FileNotFoundException`, `PermissionError`, `UnicodeDecodeError`.
2. Реализовать безопасный ввод целого числа: цикл `while True` с `try-except ValueError`.
3. Создать декоратор `@safe_execute`, перехватывающий исключения в любой функции и логирующий их.
4. Написать программу обработки списка URL: пропуск недоступных ресурсов без падения программы.

Практическое занятие «Комплексная задача: сохранение и загрузка данных приложения»

1. Разработать мини-приложение «Блокнот»: CRUD операции с заметками, сохранение в JSON, загрузка при старте.
2. Реализовать систему настроек приложения: чтение config.json, валидация, запись изменений, backup старого конфига.
3. Написать программу анализа продаж: чтение CSV -> агрегация -> отчет в TXT + график (matplotlib) -> сохранение результатов.
4. Создать базу данных контактов в JSON с функциями поиска, фильтрации, экспорта в CSV и импорта из CSV.

Критерии оценки выполнения практических заданий

Оценка «отлично» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы (корректно работающий алгоритм/программный код);
- правильно выполнено 90-100% работы.

Оценка «хорошо» ставится, если:

- правильно выполнена большая часть работы (80-89%);
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи (например, неоптимальный алгоритм или избыточный код).

Оценка «удовлетворительно» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок.

Оценка «неудовлетворительно» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками или значительная часть работы выполнена не самостоятельно;
- работа показала полное отсутствие у обучающегося обязательных знаний и навыков по проверяемой теме.

Тестовые задания

(ОК 01, ОК 02, ОК 09, ПК 1.1, ПК 1.3)

Раздел 1. Основы алгоритмизации

1. Что такое алгоритм?
 - а) Программа, написанная на языке Python

б) Конечная последовательность точно определенных действий, приводящая к решению задачи за конечное число шагов

в) Графическая схема выполнения программы

г) Описание задачи на естественном языке

2. Какое свойство алгоритма означает, что каждый шаг должен быть понятен исполнителю и не допускать двоякого толкования?

а) Массовость

б) Результативность

в) Детерминированность

г) Конечность

3. Какой геометрической фигурой в блок-схеме обозначается условие (разветвление)?

а) Прямоугольник

б) Ромб

в) Параллелограмм

г) Овал

4. В какой алгоритмической структуре команды выполняются строго последовательно друг за другом?

а) Циклическая

б) Разветвляющаяся

в) Линейная

г) Рекурсивная

5. Чем цикл с предусловием отличается от цикла с постусловием?

а) В предусловии тело цикла выполняется хотя бы один раз

б) В предусловии условие проверяется до выполнения тела цикла

в) Цикл с постусловием не может быть бесконечным

г) Разницы нет, это синонимы

6. Что обозначает блок «Параллелограмм» в блок-схеме?

а) Начало/конец алгоритма

б) Вычислительный процесс

в) Ввод или вывод данных

г) Вызов подпрограммы

7. Что такое тело цикла?

а) Условие продолжения выполнения цикла

б) Последовательность действий, повторяющихся в цикле

в) Переменная, изменяющаяся при каждой итерации

г) Команда выхода из цикла

8. Какое свойство алгоритма означает возможность его применения к целому классу однотипных задач?

- а) Дискретность
- б) Массовость
- в) Детерминированность
- г) Понятность

9. Цикл со счетчиком (параметром) характеризуется тем, что:

- а) Число повторений заранее неизвестно
- б) Условие проверяется после тела цикла
- в) Число повторений определяется начальным, конечным значением и шагом
- г) Он всегда выполняется бесконечно

10. Может ли алгоритм не иметь входных данных?

- а) Нет, любой алгоритм требует ввода
- б) Да, если все исходные данные заданы внутри алгоритма или являются константами
- в) Только линейные алгоритмы
- г) Только циклические алгоритмы

Раздел 2. Основы языка программирования Python

1. Какой тип данных возвращает функция `input()`?

- а) `int`
- б) `float`
- в) `str`
- г) `bool`

2. Как преобразовать строку "42" в целое число?

- а) `float("42")`
- б) `str(42)`
- в) `int("42")`
- г) `list("42")`

3. Какой тип данных в Python является неизменяемым (`immutable`)?

- а) `list`
- б) `dict`
- в) `tuple`
- г) `set`

4. Что вернет выражение `len([1, [2, 3], 4])`?

- а) 4
- б) 3
- в) 5
- г) Ошибку

5. Как получить последний элемент списка `lst`?

- а) `lst[1]`
- б) `lst[-1]`
- в) `lst.last()`
- г) `lst.end()`

6. Какой метод добавляет элемент в конец списка?

- а) `.insert()`
- б) `.add()`
- в) `.append()`
- г) `.push()`

7. Чем множество (`set`) принципиально отличается от списка?

а) Множество хранит только уникальные элементы и не поддерживает индексацию

- б) Множество может содержать дубликаты
- в) Множество упорядочено по возрастанию
- г) Множество изменяемо, а список нет

8. Как создать пустой словарь?

- а) `[]`
- б) `()`
- в) `{}`
- г) `set()`

9. Что такое срез строки `[start:stop:step]`?

а) Удаление части строки

б) Извлечение подстроки с указанием начального, конечного индекса и шага

- в) Разбиение строки на список слов
- г) Замена символов в строке

10. Можно ли изменить отдельный символ в строке по индексу после её создания (например, `s[0] = "A"`)?

- а) Да, строки изменяемы
- б) Нет, возникнет ошибка `TypeError`, так как строки неизменяемы
- в) Да, но только в Python 2
- г) Да, если использовать метод `.replace()`

Раздел 3. Операторы Python

1. Чему равно выражение `10 // 3`?

- а) 3.33
- б) 3
- в) 1
- г) 4

2. Какой оператор используется для возведения числа в степень?

- а) ^
- б) *
- в) **
- г) //

3. Результат логического выражения True and False равен:

- а) True
- б) False
- в) None
- г) 1

4. Что делает оператор in?

- а) Присваивает значение
- б) Проверяет принадлежность элемента к коллекции
- в) Сравнивает два объекта на идентичность
- г) Выполняет логическое отрицание

5. В каком порядке выполнятся операции в выражении $2 + 3 * 4$?

- а) Сложение, затем умножение
- б) Умножение, затем сложение
- в) Слева направо
- г) Справа налево

6. Что вернет цепочка сравнения $1 < 5 < 10$?

- а) True
- б) False
- в) 5
- г) Ошибку синтаксиса

7. В чем разница между операторами == и is?

а) == сравнивает значения, is сравнивает идентичность объектов в памяти

- б) == сравнивает типы, is сравнивает значения
- в) Разницы нет
- г) is работает только с числами

8. Какие объекты в Python считаются «ложными» (falsy) в логическом контексте?

- а) Только False
- б) 0, "", [], None, False
- в) Любое число меньше 10
- г) Только пустые строки

9. Чему равно выражение $10 \% 3$?

- а) 3

- б) 1
- в) 0
- г) 10

10. Что вернет выражение `not (5 > 3)`?

- а) True
- б) False
- в) 5
- г) 3

Раздел 4. Циклы

1. Сколько раз выполнится тело цикла `for i in range(5):`?

- а) 4 раза
- б) 5 раз
- в) 6 раз
- г) Бесконечно

2. Что генерирует `range(1, 10, 2)`?

- а) [1, 2, 3, 4, 5, 6, 7, 8, 9]
- б) [1, 3, 5, 7, 9]
- в) [2, 4, 6, 8]
- г) [1, 10, 2]

3. Какой оператор полностью прерывает выполнение цикла?

- а) `continue`
- б) `pass`
- в) `break`
- г) `stop`

4. Какой оператор пропускает остаток текущей итерации и переходит к следующей?

- а) `break`
- б) `continue`
- в) `return`
- г) `exit`

5. Когда выполняется блок `else` у цикла `for` или `while`?

- а) Никогда
- б) Только если цикл был прерван через `break`
- в) Если цикл завершился естественно (без `break`)
- г) После каждой итерации

6. Для чего используется функция `enumerate()`?

- а) Для сортировки списка
- б) Для получения пары (индекс, значение) при переборе коллекции
- в) Для фильтрации элементов

- г) Для объединения двух списков
- 7. Можно ли безопасно изменять список, по которому идет итерация в цикле for?
 - а) Да, Python автоматически создает копию
 - б) Нет, это приводит к непредсказуемому поведению или пропуску элементов
 - в) Да, но только для кортежей
 - г) Да, если использовать while
- 8. Что такое генератор списков (list comprehension)?
 - а) Функция для создания случайных списков
 - б) Синтаксическая конструкция [выражение for элемент in коллекция if условие]
 - в) Метод .generate()
 - г) Цикл while в одну строку
- 9. Как перебрать элементы двух списков параллельно?
 - а) Использовать вложенные циклы
 - б) Использовать функцию zip()
 - в) Использовать enumerate()
 - г) Сложить списки оператором +
- 10. Чем цикл while отличается от цикла for?
 - а) while работает только с числами
 - б) for предназначен для перебора итерируемых объектов, while выполняется пока истинно условие
 - в) while быстрее работает
 - г) Разницы нет

Раздел 5. Функции и модули

- 1. Какое ключевое слово используется для объявления функции в Python?
 - а) func
 - б) function
 - в) def
 - г) lambda
- 2. Что возвращает функция, если в ней отсутствует оператор return?
 - а) 0
 - б) ""
 - в) None
 - г) Ошибку
- 3. Что такое *args в параметрах функции?

- а) Словарь именованных аргументов
- б) Кортеж позиционных аргументов переменной длины
- в) Указатель на память
- г) Обязательный параметр

4. Что такое `**kwargs`?

- а) Кортеж аргументов
- б) Словарь именованных аргументов переменной длины
- в) Список ключей
- г) Декоратор функции

5. Какое правило определяет порядок поиска переменных в Python?

- а) FIFO
- б) LIFO
- в) LEGB (Local, Enclosing, Global, Built-in)
- г) MRU

6. Что такое `lambda`-функция?

- а) Именованная функция с документацией
- б) Анонимная однострочная функция
- в) Рекурсивная функция
- г) Функция без параметров

7. Чем `sorted()` отличается от метода `.sort()`?

- а) Ничем
- б) `sorted()` возвращает новый отсортированный список, `.sort()` изменяет список на месте и возвращает `None`
- в) `.sort()` работает быстрее с числами
- г) `sorted()` сортирует только строки

8. Что такое рекурсия?

- а) Цикл `for` внутри `while`
- б) Вызов функцией самой себя
- в) Импорт модуля внутри функции
- г) Многопоточное выполнение

9. Обязателен ли базовый случай (условие выхода) в рекурсивной функции?

- а) Нет, интерпретатор остановит её автоматически
- б) Да, иначе возникнет `RecursionError` из-за переполнения стека
- в) Только для функций с возвратом значения
- г) Только если функция принимает аргументы

10. Для чего используется `docstring`?

- а) Для ускорения выполнения кода
- б) Для документирования функции/модуля (строка документации)

- в) Для объявления переменных
- г) Для обработки исключений

Раздел 6. Объектно-ориентированное программирование (ООП)

1. Какое ключевое слово используется для создания класса?
 - а) object
 - б) struct
 - в) class
 - г) type
2. Для чего нужен параметр self в методах класса?
 - а) Для доступа к глобальным переменным
 - б) Для ссылки на текущий экземпляр объекта
 - в) Для передачи аргументов между методами
 - г) Это зарезервированное слово, его нельзя изменить
3. Когда вызывается метод `__init__`?
 - а) При удалении объекта
 - б) При создании (инициализации) объекта
 - в) При вызове любого метода класса
 - г) При импорте модуля
4. Чем атрибут класса отличается от атрибута экземпляра?
 - а) Атрибут класса общий для всех экземпляров, атрибут экземпляра уникален для каждого объекта
 - б) Атрибут класса можно менять, атрибут экземпляра нет
 - в) Атрибут класса хранится в оперативной памяти, экземпляр на диске
 - г) Разницы нет
5. Как в Python обозначается приватный атрибут?
 - а) Один подчеркивание в начале (`_attr`)
 - б) Два подчеркивания в начале (`__attr`)
 - в) Ключевое слово `private`
 - г) Модификатор `final`
6. Для чего используется декоратор `@property`?
 - а) Для создания статических методов
 - б) Для реализации геттеров/сеттеров и управления доступом к атрибутам
 - в) Для наследования
 - г) Для перегрузки операторов
7. Что такое полиморфизм в ООП?
 - а) Создание нового класса на основе старого

б) Способность разных объектов реагировать на один и тот же метод по-разному

в) Соккрытие внутренней реализации

г) Множественное наследование

8. Как вызвать метод или конструктор родительского класса из дочернего?

а) parent()

б) super()

в) base()

г) this()

9. Что такое MRO (Method Resolution Order)?

а) Порядок инициализации атрибутов

б) Порядок разрешения методов при наследовании (особенно множественном)

в) Список всех методов класса

г) Алгоритм сортировки методов

10. Можно ли создать экземпляр абстрактного базового класса (ABC)?

а) Да, всегда

б) Нет, возникнет ошибка TypeError

в) Да, но только с параметрами

г) Да, если переопределить __new__

Раздел 7. Работа с файлами и обработка данных

1. Какой режим открытия файла используется только для чтения?

а) 'w'

б) 'r'

в) 'a'

г) 'x'

2. Что делает режим 'a' (append)?

а) Перезаписывает файл целиком

б) Создает файл, если его нет, и открывает для чтения

в) Открывает файл для дозаписи в конец, не удаляя старые данные

г) Открывает файл в бинарном режиме

3. Для чего используется конструкция with open(...) as f:?

а) Для ускорения чтения файла

б) Для автоматического закрытия файла после выхода из блока

в) Для шифрования содержимого

г) Для проверки прав доступа

4. Какой модуль стандартной библиотеки используется для работы с JSON?

- а) csv
- б) xml
- в) json
- г) pickle

5. Какое исключение возникает при попытке открыть несуществующий файл в режиме 'r'?

- а) ValueError
- б) TypeError
- в) FileNotFoundError
- г) IndexError

6. Когда выполняется блок else в конструкции try-except-else-finally?

- а) Никогда
- б) Если в блоке try не возникло исключений
- в) Если возникло исключение
- г) Всегда после finally

7. Выполняется ли блок finally всегда?

- а) Нет, только при ошибках
- б) Да, независимо от наличия исключений или оператора return/break
- в) Только в циклах
- г) Только при успешном выполнении try

8. Чем формат CSV отличается от JSON?

- а) CSV хранит данные в виде таблицы (разделитель запятая), JSON – в виде структур/словарей
- б) CSV поддерживает вложенность, JSON нет
- в) JSON быстрее читается
- г) Разницы нет

9. Зачем при открытии текстовых файлов указывать encoding='utf-8'?

- а) Для сжатия файла
- б) Для корректной обработки кириллицы и специальных символов
- в) Для ускорения записи
- г) Для перевода файла в бинарный режим

10. Что делает метод .dump() модуля json?

- а) Читает JSON из файла
- б) Записывает Python-объект в файл в формате JSON
- в) Валидирует JSON-строку
- г) Конвертирует CSV в JSON

Критерии оценки выполнения тестовых заданий

Для **оценки результатов тестирования** предусмотрена следующая система оценивания учебных достижений студентов: за каждый правильный ответ ставится 1 балл, за неправильный ответ – 0 баллов.

Оценка «отлично» (например, 3 балла) выставляется при условии правильного ответа студента не менее чем на 86 % тестовых заданий;

Оценка «хорошо» (например, 2 балла) выставляется при условии правильного ответа студента на 70-85 % тестовых заданий;

Оценка «удовлетворительно» (например, 1 балл) выставляется при условии правильного ответа студента на 50-69 % тестовых заданий;

Оценки «неудовлетворительно» (0 баллов) заслуживает студент при условии правильного ответа студента менее чем на 50 % тестовых заданий.

Задания для самостоятельной работы студентов

Тематика рефератов, докладов (ОК 01, ОК 02, ОК 09, ПК 1.1, ПК 1.3)

Тема 1.1. Понятие алгоритмизации и алгоритма. Виды алгоритмических структур.

1. «История развития понятия алгоритма: от Аль-Хорезми до Тьюринга»

Тема 2.1. Переменные. Ввод, вывод данных и базовые операции.

2. «Эволюция языков программирования: место Python в современной IT-индустрии»

Тема 2.2. Числовые типы данных.

3. «Представление чисел в памяти компьютера: почему $0.1 + 0.2$ не всегда равно 0.3 ?»

Тема 2.3. Строковый тип данных.

4. «Кодировки текста (ASCII, UTF-8): как компьютер понимает буквы и символы»

Тема 2.4. Списки (List).

5. «Списки в Python: реализация динамических массивов и оценка производительности операций»

Тема 2.4. Кортежи (Tuple).

6. «Неизменяемость кортежей: зачем она нужна и как это влияет на безопасность кода»

Тема 2.6. Множества (Set).

7. «Математические множества в программировании: применение в базах данных и поиске уникальных значений»

Тема 2.7. Словари (Dict).

8. «Хеширование в словарях Python: как работает поиск по ключу за $O(1)$ »

Тема 3.1. Арифметические операторы.

9. «Приоритет операций в Python: таблица приоритетов и типичные ошибки новичков»

Тема 3.2. Логические операторы.

10. «Булева алгебра в программировании: законы де Моргана и упрощение сложных условий»

Тема 3.3. Условный оператор.

11. «Вложенные условия vs Логические операторы: что читабельнее и эффективнее?»

Тема 4.1. Цикл for.

12. Сравнительный анализ синтаксиса цикла **for** в языках Pascal, C++ и Python»

Тема 4.2. Цикл While.

13. Бесконечный цикл и способы его предотвращения

Тема 4.3. Управление циклами.

14. «Оптимизация вложенных циклов: сложность алгоритмов $O(n^2)$ и способы ее снижения»

Тема 5.1. Встроенные функции Python.

15. «Функциональное программирование: парадигма, преимущества и встроенные инструменты Python»

Тема 5.2. Методы типов данных.

16. «Изменяемые и неизменяемые типы данных: подводные камни передачи аргументов в функции»

Тема 5.3. Пользовательские функции.

17. «Области видимости переменных (LEGB): глобальные, локальные и нелокальные переменные»

Тема 5.4. Возврат значения и анонимные функции.

18. Рекурсия vs Итерация: когда стоит использовать рекурсивный подход, а когда нет»

Тема 6.1. Введение в ООП. Классы и объекты.

19. «Четыре столпа ООП: краткий обзор принципов инкапсуляции, наследования, полиморфизма и абстракции»

Тема 6.2. Атрибуты и методы класса.

20. «Жизненный цикл объекта в Python: создание, использование и сборка мусора (Garbage Collection)»

Тема 6.3. Инкапсуляция и свойства.

21. «Инкапсуляция в Python: соглашение об именовании (`_protected`, `__private`) и декоратор `@property`»

Тема 6.4. Наследование и полиморфизм

22. «MRO (Method Resolution Order): порядок разрешения методов при множественном наследовании»

Критерии оценки доклада, реферата

1. Актуальность темы исследования.
2. Соответствие содержания теме.
3. Глубина проработки материала.
4. Правильность и полнота использования источников.
5. Соответствие оформления стандартам.

Оценка **«отлично»** ставится, если выполнены все требования к написанию и защите доклада (реферата): обозначена проблема и обоснована

ее актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложено собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объем, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы.

Оценка **«хорошо»** - основные требования к докладу (реферату) и его защите выполнены, но при этом допущены недочеты. В частности, имеются не точности в изложении материала; отсутствуют логическая последовательность в суждениях; не выдержан объем доклада (реферата); имеются упущения в оформлении; на дополнительные вопросы при защите даны не полные ответы.

Оценка **«удовлетворительно»** - имеются существенные отступления от требований. В частности, тема освещена лишь частично; допущены фактические ошибки в содержании доклада (реферата) или при ответе на дополнительные вопросы; во время защиты отсутствует вывод.

Оценка **«неудовлетворительно»** - тема доклада (реферата) не раскрыта, обнаруживается существенное непонимание проблемы.

2. Вопросы и задания для промежуточной аттестации

Вопросы для устного (письменного) зачета по дисциплине

(ОК 01, ОК 02, ОК 09, ПК 1.1, ПК 1.3)

1. Понятие алгоритма, его основные свойства (детерминированность, массовость, конечность) и способы представления.
2. Виды алгоритмических структур: линейная, разветвляющаяся, циклическая.
3. Основные элементы и правила построения блок-схем алгоритмов.
4. Переменные и константы в Python. Динамическая типизация данных.
5. Числовые типы данных (int, float): основные операции и преобразование типов.
6. Строковый тип данных (str): индексация, срезы, основные методы обработки.
7. Списки (list) и кортежи (tuple): принципиальные различия, создание и основные методы.
8. Множества (set) и словари (dict): особенности хранения, уникальности и доступа к данным.
9. Арифметические операторы, операторы сравнения и приоритет выполнения операций в Python.
10. Логические операторы (and, or, not) и операторы принадлежности (in, not in).
11. Синтаксис и правила работы условного оператора if-elif-else.
12. Цикл for: синтаксис, итерируемые объекты, функция range().
13. Цикл while: синтаксис, условия завершения и бесконечные циклы.
14. Операторы управления потоком в циклах: break, continue и блок else.
15. Вложенные циклы: назначение, правила реализации и области применения.
16. Понятие функции в программировании. Популярные встроенные функции Python и их назначение.
17. Создание пользовательских функций: синтаксис объявления, параметры и аргументы.
18. Локальные и глобальные переменные. Оператор return и возвращаемые значения.
19. Анонимные функции (lambda) и основы рекурсивных алгоритмов.
20. Основные принципы объектно-ориентированного программирования (ООП).

21. Понятие класса и объекта. Синтаксис создания класса и конструктор `__init__`.
22. Атрибуты класса и экземпляра. Назначение параметра `self` в методах.
23. Инкапсуляция: уровни доступа к атрибутам и использование декоратора `@property`.
24. Наследование и полиморфизм в Python. Вызов методов родительского класса (`super()`).
25. Магические методы (dunder-методы) и абстрактные базовые классы (ABC).
26. Работа с файлами в Python: режимы открытия и контекстный менеджер `with open()`.
27. Форматы хранения структурированных данных: особенности чтения и записи CSV и JSON.
28. Понятие исключений (ошибок) выполнения. Конструкция `try-except`.
29. Блоки `else` и `finally` в обработке исключений: порядок и условия их выполнения.
30. Принципы валидации входных данных и защиты программы от сбоев (Defensive Programming).

Задания для устного (письменного) зачета по дисциплине

(ОК 01, ОК 02, ОК 09, ПК 1.1, ПК 1.3)

Вариант 1

Задание 1. Условные операторы и базовые вычисления (Разделы 1-3)

Написать программу-калькулятор геометрических фигур. Пользователь выбирает фигуру (круг, прямоугольник, треугольник) и вводит необходимые параметры. Программа должна вычислять площадь выбранной фигуры. Предусмотреть проверку корректности введенных данных (например, стороны треугольника должны удовлетворять неравенству треугольника) с использованием условных операторов `if-elif-else` и логических операторов.

Задание 2. Циклы и функции (Разделы 4-5)

1. Написать функцию `get_primes(n)`, которая возвращает список всех простых чисел от 2 до N, используя цикл `for` и операторы `break/continue` для оптимизации.

2. Написать `lambda`-функцию для сортировки списка кортежей `[('apple', 2), ('banana', 1), ('cherry', 3)]` по второму элементу (числу).

Задание 3. Объектно-ориентированное программирование (Раздел 6)

Создать класс BankAccount (Банковский счет).

- Реализовать инкапсуляцию: атрибут баланса должен быть приватным.
- Добавить методы deposit(amount) (пополнение) и withdraw(amount) (снятие) с проверкой достаточности средств.
- Реализовать магический метод __str__ для красивого вывода информации о счете.
- Добавить свойство (property) balance только для чтения.

Задание 4. Файлы и обработка исключений (Раздел 7)

Написать скрипт, который читает JSON-файл students.json, содержащий список словарей с данными студентов (имя и оценки по предметам). Программа должна вычислить средний балл для каждого студента и записать итоговый отчет в текстовый файл report.txt. Обязательно использовать конструкцию try-except для обработки ошибок FileNotFoundError и json.JSONDecodeError.

Вариант 2

Задание 1. Циклы и арифметические операторы (Разделы 1-3)

Написать программу, которая переводит целое десятичное число, введенное пользователем, в двоичную систему счисления. Реализовать алгоритм с использованием цикла while и операторов целочисленного деления // и остатка от деления %. Вывести результат в виде строки.

Задание 2. Функции и рекурсия (Разделы 4-5)

1. Написать рекурсивную функцию factorial(n) для вычисления факториала числа. Обработать случай отрицательного числа.
2. Используя встроенные функции map и filter, написать код, который принимает список чисел, оставляет только положительные и возводит их в квадрат.

Задание 3. Наследование и абстрактные классы (Раздел 6)

1. Создать абстрактный базовый класс Shape (Фигура) с абстрактным методом area().
2. Создать классы-наследники Circle и Rectangle, которые реализуют метод area().
3. Продемонстрировать полиморфизм: создать список из разных фигур и в цикле вывести площадь каждой из них. Попытаться создать экземпляр абстрактного класса Shape и обработать возникшую ошибку.

Задание 4. Работа с CSV и защита от ошибок (Раздел 7)

Написать программу, которая читает CSV-файл `products.csv` (колонки: Название, Цена, Количество). Программа должна отфильтровать товары с ценой выше 1000 рублей и сохранить их в новый файл `expensive_products.json`. Использовать `try-except` для обработки ошибок преобразования типов (например, если в колонке «Цена» окажется текст) и отсутствия файла.

Вариант 3

Задание 1. Строки и логические операторы (Разделы 1-3)

Написать программу-валидатор пароля. Пользователь вводит строку. Программа проверяет, соответствует ли пароль требованиям:

- Длина не менее 8 символов;
- Содержит хотя бы одну заглавную букву;
- Содержит хотя бы одну цифру;
- Не содержит пробелов. Использовать методы строк (`isupper()`, `isdigit()` и т.д.) и логические операторы. Вывести подробный отчет о том, каким требованиям пароль не соответствует.

Задание 2. Словари и циклы (Разделы 4-5)

Написать функцию `analyze_text(text)`, которая принимает строку текста, приводит её к нижнему регистру, удаляет знаки препинания и возвращает словарь, где ключами являются слова, а значениями — частота их встречаемости. Найти и вывести 3 самых частых слова.

Задание 3. Инкапсуляция и свойства (Раздел 6)

Создать класс `Employee` (Сотрудник) с атрибутами `name`, `age` и `salary`.

- Использовать декоратор `@property` для атрибута `age`: запретить установку возраста меньше 18 и больше 100 лет.
- Использовать декоратор `@property` для атрибута `salary`: запретить установку отрицательной зарплаты.
- Создать класс-наследник `Manager`, который добавляет атрибут `bonus` и переопределяет метод `__str__` для вывода общей суммы дохода.

Задание 4. Контекстный менеджер и JSON (Раздел 7)

Написать программу, которая создает базу данных контактов.

1. При запуске программа пытается загрузить контакты из файла `contacts.json`. Если файла нет, создается пустой словарь.
2. Пользователь может добавить контакт (имя и телефон) или удалить его.
3. При завершении работы (или по команде) данные сохраняются обратно в JSON-файл. Использовать контекстный менеджер `with open(...)` и обработку исключений `try-except-else-finally`.

Критерии оценки ответов на дифференцированном зачете

Оценка «отлично» — теоретическое содержание дисциплины освоено полностью, сформированы необходимые практические навыки и умения, выполнены все учебные задания. Код работает без ошибок, соответствует стандартам PEP 8 (ПК 1.3), реализованы все требования задачи, включая обработку граничных условий и исключений.

Оценка «хорошо» — теоретическое содержание дисциплины освоено полностью, сформированы необходимые практические навыки и умения не в полном объеме, выполнены все учебные задания, при выполнении которых были обнаружены незначительные ошибки и недочеты (например, неоптимальный алгоритм, мелкие стилистические недочеты в коде), которые студент может исправить самостоятельно.

Оценка «удовлетворительно» — теоретическое содержание дисциплины освоено частично, но пробелы не носят существенного характера, сформированы в основном необходимые практические навыки и умения, выполнено большинство учебных заданий, при выполнении которых были обнаружены ошибки и недочеты (программа работает не на всех входных данных, требует помощи преподавателя для отладки).

Оценка «неудовлетворительно» — теоретическое содержание дисциплины не освоено, не сформированы необходимые практические навыки и умения, выполненные учебные задания содержат существенные ошибки и недочеты (код не написан, не запускается, не решает поставленную задачу, студент не понимает базовых принципов алгоритмизации и синтаксиса Python).