Комплект поисково-творческих заданий по программированию на Phyton

Комплект поисково-творческих заданий, разработан для активного поиска информации, анализа данных и формулировки собственных выводов. Задания нашего комплекта не только развивают навыки поиска, а также обучают коммуникациям в онлайн-среде программистов. В комплект включены задания для учащихся и рекомендации для учителя.

С помощью нашего конструктора веб-квестов, вы сможете сделать эти задания настоящим интеллектуальным приключением для своих учеников.

В каждой теме п.1 – условие поисково-творческого задания, п.2 – материал для учителя: рекомендации и возможный вариант результата выполнения задания.

Т.1. **Язык программирования Python, алгоритм и виды алгоритмов**

1	Представьте, что вы проектируете робот-доставщик. Какой **список команд** исполнителя может быть предложен? В качестве примера можно рассматривать робота-курьера от компании Яндекс.
2	Объясните учащимся, насколько важно для программирования понятие «список команд исполнителя». Предложите им провести поисково-исследовательскую работу и самостоятельно составить список команд для их собственного исполнителя «робот-доставщик». Можно использовать Конструктор квестов для проведения анализа, предложив 2-3 сайта с описанием роботов-доставщиков. Далее попросите учащихся ответить на вопрос «Какой список команд Вы можете предложить для робота-доставщика?» Пример списка команд: 1. Переместиться к точке назначения. 2. Взять посылку из ячейки/склада. 3. Доставить посылку по адресу получателя. 4. Отправить уведомление получателю о приближении к месту доставки. 5. Подтвердить доставку получателем. 6. Вернуться на базу для зарядки/обслуживания. 7. Сообщить о любых повреждениях или проблемах на пути.

Т.2. **Переменная и оператор присваивания. Типы переменных и операции с ними.**

1	Изучите особенности **типизации данных** в языках программирования Python и Java. Опишите это различие своими словами, сделайте выводы.
2	Объясните учащимся, что языки программирования обладают различными свойствами (удобство, гибкость, производительность и т.д.), поэтому выбор языка программирования зависит от конкретных требований проекта. Предложите им провести поисково-исследовательскую работу и самостоятельно составить объяснение о том, в чем различие типизации данных в языках программирования Python и Java, как и на что это различие влияет. Можно использовать Конструктор квестов для проведения этого мини-исследования, предложив 2-3 сайта для поиска информации. *Пример результата*:

	Различие в языках программирования Python и Java представляют собой разные подходы к работе с типами данных – динамическую и статическую типизацию соответственно. **Python** – это интерпретируемый язык программирования с динамической типизацией, что означает, что типы переменных определяются во время выполнения программы. Преимущества: · Простота использования. Динамическая типизация в Python позволяет разработчику быстро создавать и изменять типы переменных без явного указания типов. · Гибкость. Python позволяет легко работать с различными типами данных. **Java** – это компилируемый язык программирования со статической типизацией, что означает, что типы переменных проверяются на этапе компиляции. Преимущества: · Безопасность и надежность. Статическая типизация в Java помогает выявлять ошибки типов на этапе компиляции, что обеспечивает более высокий уровень безопасности и надежности программного обеспечения. · Производительность. Статическая типизация может способствовать более эффективной оптимизации кода на этапе компиляции и повышению производительности программ. **Выводы**: динамическая типизация в Python обеспечивает гибкость и удобство использования, в то время как статическая типизация в Java обеспечивает более высокий уровень безопасности и производительности

Т.3. **Ввод и вывод информации (print, input).**

1	Организация функционала ввода/вывода информации влияет на взаимодействие между пользователем и приложением/ или игрой, от него во многом зависит общая оценка всей разработки. Проведите исследование примеров **организации ввода и вывода информации** в различных категориях программных продуктов: ▪ исследование ввода в **текстовых квестах** – найдите пример текстового квеста или игры с выбором действий. Проанализируйте, как реализован ввод информации от игрока. Обратите внимание на использование текстового интерфейса и варианты ответов; ▪ анализ ввода в графических **приключенческих играх** – изучите графические приключенческие игры и анализируйте, как в них реализован ввод информации. Рассмотрите использование мыши, клавиатуры или сенсорных экранов. Выделите примеры хорошей и плохой реализации; ▪ исследование диалогов в **ролевых играх** – найдите примеры ролевых игр с диалогами. Проанализируйте, как организован ввод информации при выборе диалогов и влияние этих выборов на развитие сюжета (сравните подходы разных игр); ▪ анализ ввода в головоломках и **логических играх** – исследуйте головоломки и логические игры, где требуется вводить ответы на вопросы или решения задач. Рассмотрите, как представлен ввод, и оцените его удобство и эффективность; ▪ оценка **обратной связи** в играх – проанализируйте, как игры предоставляют обратную связь по результатам ввода пользователя. Рассмотрите использование звуков, анимаций, текстовых подсказок и других методов. Определите, насколько эффективно обеспечивается информирование пользователя
2	Предложите учащимся выбрать для проведения исследования одну из предложенных категорий. В качестве объекта исследования можно использовать игры или

	приложения, которые они уже используют; также можно предложить ссылки на ресурсы онлайн-игр (в этом случае педагог должен быть уверен в безопасности такого ресурса). После проведения исследования ученики должны написать свое мнение о том, какие подходы им кажутся наилучшими, а какие можно было би ещё улучшить (можно привести варианты улучшений). Такая задача позволит им развивать критическое мышление и аналитические навыки.

Т.4. **Первый чат-бот.**

1	В этом задании нужно исследовать **функционал чат-ботов** и оценить, как различные боты решают определенные задачи. Проанализируйте особенности ввода и вывода информации, обработку пользовательских запросов и функционал, предоставляемый разными чат-ботами. Предлагаем следующий план проведения исследования. ▪ **Выбор** ботов**.** Выберите несколько чат-ботов с разными функциональными возможностями. Рассмотрите, как минимум одного бота, предназначенного для развлечений, одного для помощи в повседневных задачах и одного, предназначенного для образовательных целей. ▪ **Анализ** функционала. Изучите функционал каждого выбранного бота. Проанализируйте, какие команды и запросы пользователей они поддерживают. Оцените, насколько широкий и разнообразный функционал предоставляется. ▪ **Сравнение** ввода и вывода. Сравните методы ввода и вывода информации для каждого бота. Рассмотрите, как боты реагируют на текстовый ввод, ввод изображений, голосовые команды и другие возможности. Оцените, насколько эффективен каждый метод. ▪ **Решение** пользовательских запросов. Исследуйте, как чат-боты решают конкретные задачи, предлагаемые пользователями. Например, запрос на получение новостей, решение математических задач, предоставление советов и т.д. Оцените точность и полноту ответов. ▪ **Отчет**. Напишите отчет о результатах исследования, включив в него описание выбранных ботов, анализ функционала, сравнение методов ввода и вывода, оценку решения пользовательских запросов и обзор отзывов.
2	Предложите учащимся выбрать для проведения исследования два (или более) чат-бота с похожим кругом задач. Ресурсы для исследования: можно использовать официальные сайты чат-ботов, рецензии в магазинах приложений, обзоры блогеров и форумы для поиска информации и отзывов о выбранных ботах. Основная задача проведения исследования – научить грамотно анализировать программные разработки, их соответствие пользовательским запросам, решения для обеспечения коммуникаций между продуктом и потребителем. Отчет не должен быть объемным, нужно учиться формулировать своим выводы коротко и в доходчивой форме. Это очень важный навык для «самостоятельного» мышления обучающихся.

Т.5. **Операции с целыми числами: целочисленное деление, остаток от деления. Кратность чисел.**

1	Исследуйте **применение функций** деления с остатком, четности/нечетности и кратности чисел в программировании и приведите примеры из повседневной жизни, где это понятие играет важную роль.

2	Основная идея этого задания состоит в том, чтобы показать учащимся практическую значимость изучаемого материала. В качестве примера, используемых в практической жизни изучаемых функций, учащиеся могут приводить простые житейские задания. Ниже несколько таких примеров. 1. Распределение по коробкам. *Задача*: у вас есть ***N*** предметов, и вы хотите распределить их по коробкам так, чтобы в каждой коробке было одинаковое количество предметов. Какое минимальное количество коробок вам понадобится? 2. Парковка автомобилей. *Задача*: вам нужно парковать автомобили на участке, и у вас есть ***M*** мест. Какое максимальное количество автомобилей вы сможете припарковать, если каждое место подразумевает четное количество автомобилей? 3. Подготовка к экзамену. *Задача*: у студентов есть ***N*** задач для подготовки к экзамену. Студенты хотят разделить задачи между собой поровну. Какое максимальное количество студентов сможет получить одинаковое количество задач? 4. Обмен валюты. *Задача*: у вас есть ***X*** рублей, и вы хотите обменять их на иностранную валюту. Банк предоставляет вам купюры разных номиналов. Какое минимальное количество купюр вы получите, если обмен происходит с остатком? 5. Раскладка на полках. *Задача*: у вас есть ***K*** книг и ***S*** полок. Вы хотите расставить книги по полкам так, чтобы на каждой полке было четное количество книг. Сколько полок вам нужно? 6. Упаковка подарков. *Задача*: у вас есть ***N*** подарков, и вы хотите упаковать их в коробки так, чтобы каждая коробка содержала нечетное количество подарков. Сколько коробок вам понадобится?

Т.6. **Алгоритмы для решения математических задач без деления.**

1	Для **исследования применения** Алгоритмов для решения задач без деления в повседневной жизни выполните задание «Планирование бюджета с учетом различных финансовых целей». *Задание*: вы являетесь финансовым консультантом и вам поручено помочь клиенту составить бюджет на следующий год, учитывая его финансовые цели. Клиент хочет сэкономить на покупку нового автомобиля, планирует оплатить образование для своих детей и сохранить определенную сумму на пенсию. Вам нужно разработать план бюджета, который включает в себя ежемесячные траты на платежи по автокредиту, сбережения на образование детей, регулярные взносы в пенсионный фонд, а также базовые расходы на жилье, питание, транспорт и развлечения. Кроме того, вам нужно учесть возможные инвестиции, которые помогут клиенту достичь его финансовых целей быстрее. Вам предоставляются данные о ежемесячном доходе клиента, текущих затратах, процентных ставках по инвестициям и кредитам, а также прогнозируемая стоимость образования. Ваша задача - разработать детальный план бюджета на следующий год, учитывая финансовые цели клиента, и представить его в удобной для клиента форме, а также обосновать рекомендации по инвестициям
2	Основная идея этого задания состоит в том, чтобы показать учащимся практическую значимость изучаемого материала. Для решения поставленной задачи предложите учащимся выбрать один из банков, изучить его предложения. Очевидно, что задача носит условный характер, основная цель данного задания выработать навыки системного подхода.

Т.7. **Алгоритмы для решения математических задач с возведением в степень.**

1	Исследуйте **применение функций** возведения в степень и извлечение корня в программировании и приведите примеры из повседневной жизни, где это понятие играет важную роль.
2	Основная идея этого задания состоит в том, чтобы показать учащимся практическую значимость изучаемого материала. В качестве примера, используемых в практической жизни изучаемых функций, учащиеся могут приводить простые житейские задания.

Т.8. **Операторы условия if и else.**

1	Проведите **исследование эффективности** операторов ветвления в различных языках программирования.
2	Предложите ученикам на выбор два языка программирования (например, Python, Java, C++, JavaScript и т.п.). Сделав выбор, изучить синтаксис и особенности использования операторов ветвления в каждом из выбранных языков программирования; сравнить и проанализировать эффективность и недостатки каждого из операторов ветвления в контексте решения различных задач. В результате учащиеся могут сформулировать итоговое заключение о том, в каких случаях и в каком из языков программирования операторы ветвления демонстрируют оптимальную производительность и удобство использования. Этот проект поможет учащимся ознакомиться с синтаксисом операторов ветвления, развить навыки анализа и сравнения концепций программирования, а также сформулировать собственное мнение о преимуществах и недостатках каждого из операторов ветвления.

Т.9. **Сложные условия с помощью elif**

1	На онлайн-форумах разработчиков (сообщества программистов Phyton) найдите информацию **по теме каскадного ветвления**, изучите и составьте коротенькое эссе для начинающего разработчика с советами по таким вопросам: ▪ в каком порядке выполняются условные блоки кода в каскадном ветвлении; ▪ какой оператор используется для добавления альтернативных условий; ▪ в каких случаях рекомендуется использовать каскадное ветвление; ▪ какой тип ошибки может возникнуть при неправильном использовании каскадного ветвления; ▪ какие техники можно применить для улучшения читаемости кода с использованием каскадного ветвления; ▪ почему каскадное ветвление считается более гибким, чем простое условное ветвление.
2	Предложите ученикам на выбор несколько ссылок на форумы разработчиков (программистские онлайн-сообщества). Коротко объясните в чем состоит специфика жанра эссе (*это выражение личного мнения, обычно короткое, использует ясный стиль, излагает мысли четко и логично*). Эссе может содержать такие варианты ответов: ▪ условные блоки кода в каскадном ветвлении выполняются по порядку, начиная с первого условия и переходя к следующему только в случае невыполнения предыдущего; ▪ для добавления альтернативных условий используется оператор **elif** (**else if**);

	▪ каскадное ветвление рекомендуется использовать при необходимости проверки нескольких условий, когда простое условное ветвление может оказаться недостаточным; ▪ при неправильном использовании каскадного ветвления может возникнуть ошибка логики (например, неправильный порядок условий, недостижимый код и т. д.); ▪ для улучшения читаемости кода с использованием каскадного ветвления рекомендуется использовать комментарии, выравнивание кода и разделение условных блоков. ▪ каскадное ветвление считается более гибким, потому что оно позволяет проверять несколько условий и выполнять соответствующий код для каждого из них, в то время как простое условное ветвление позволяет проверять только одно условие. Этот проект поможет учащимся ознакомиться с синтаксисом операторов ветвления, развить навыки анализа и сравнения концепций программирования, а также сформулировать собственное мнение о преимуществах и недостатках каждого из операторов ветвления.

Т.10. **Вложенное условие. Решение квадратного уравнения**

1	Сравните **использование вложенных условий** в Python с их использованием в других языках программирования. Какие отличия существуют, и как они влияют на структуру кода? Результаты сравнительного анализа изложите, следуя следующему плану: ▪ укажите язык программирования, с которым вы сравниваете использование вложенных условий (например, C++); ▪ опишите синтаксис вложенных условий в Python: укажите, как они строятся с использованием отступов (пробелов) и какие ключевые слова используются; ▪ проведите сравнение синтаксиса вложенных условий в Python с синтаксисом в выбранном вами языке программирования и обратите внимание на наличие или отсутствие отступов, ключевых слов и других структурных элементов; ▪ оцените, как различия в синтаксисе влияют на читаемость кода; ▪ сравните, насколько легко управлять уровнем вложенности в различных языках
2	Предложите ученикам на выбор несколько ссылок на форумы разработчиков (программистские онлайн-сообщества). Объясните, что целью выполнения задания является выработка умения проводить сравнительный анализ и формировать собственные выводы на основе результатов анализа. Вывод может быть представлен таким вариантом сравнения. Использование вложенных условий в Python с языком C++ показывает такие отличия: ▪ в Python вложенные условия строятся с использованием отступов, а в языке C++ используются фигурные скобки для обозначения блоков кода; ▪ читаемость кода в Python представлена более выразительным языком, что может способствовать легкости чтения вложенных условий; ▪ структура кода в Python выглядит более плоской из-за отступов, в то время как в C++ блоки кода могут выглядеть более явно из-за фигурных скобок; ▪ фигурные скобки для обозначения блоков кода, возможно, позволяют увеличить уровень вложенности, но это усложняет структуру кода.

Т.11. **Составные условия, логические операторы and, or, not**

1	На онлайн-форумах разработчиков (сообщества программистов Phyton) найдите информацию по вопросам **применения логических операторов** и потом составьте свою собственную памятку для начинающих программистов по теме «Логические операторы в Phyton». Обязательно включите в эту памятку вопрос приоритета операторов и информацию о возможных ошибках их использования в коде программы.
2	Предложите ученикам на выбор несколько ссылок на форумы разработчиков и программистские онлайн-сообщества. Объясните, что целью выполнения задания является умения систематизировать полученную информацию, делать выводы и представлять их в доступной для понимания форме. Памятка может быть представлена в таком варианте. В Python порядок выполнения логических операторов определяется их приоритетом. Вот основные правила: ▪ **not** имеет наивысший приоритет; ▪ затем выполняются операции с **and**; ▪ самым последним выполняются операции с **or**. Используя приоритеты логических операторов, программист может более точно контролировать порядок выполнения условий в выражениях, что способствует правильному интерпретированию логических выражений. Часто встречаемые ошибки при использовании логических операторов в Python 1) неправильное использование скобок; *Ошибка заключается в том, что программисты могут забывать использовать скобки для явного определения порядка выполнения условий. Это может привести к неправильному результату из-за стандартного приоритета операторов.* 2) смешивание операторов с разными приоритетами. *Эта ошибка связана с тем, что программисты могут смешивать операторы с разными приоритетами без использования скобок, что может привести к недопониманию.* Исследование и понимание этих ошибок помогут избежать их в процессе написания логических выражений в Python.

Т.12. **Цикл с условием. Алгоритм Евклида. Разбиение записи натурального числа на цифры**

1	Посетите форумы разработчиков и найдите обсуждения, касающиеся применения алгоритма Евклида в программировании. Исследуйте различные подходы и советы, предложенные сообществом. Рассмотрите, как этот алгоритм может быть применен **для решения конкретных задач** в разработке и составьте материалы (можно в форме презентации) для выступления по теме «Применение алгоритма Евклида в программировании».
2	Предложите ученикам на выбор несколько ссылок на форумы разработчиков и программистские онлайн-сообщества. Объясните, что целью выполнения задания является подготовка выступления, в котором будет демонстрироваться универсальность и эффективность этого алгоритма для решения разнообразных задач. В качестве примеров, которые могут быть включены в материалы выступления: ▪ в криптографии алгоритм Евклида может использоваться для генерации ключей и шифрования данных (например, в алгоритме RSA для генерации ключей используется нахождение наибольшего общего делителя);

	▪ в разработке алгоритмы, связанные с работой с целыми числами, могут использовать алгоритм Евклида для оптимизации различных задач (например, проверка чисел на взаимопростоту); ▪ некоторые алгоритмы сортировки, такие как "Быстрая сортировка", могут использовать алгоритм Евклида для определения порядка элементов; ▪ в компьютерных играх алгоритм Евклида может использоваться для определения коллизий и взаимодействия объектов в виртуальном пространстве; ▪ в области графического дизайна алгоритм Евклида может применяться для нахождения цветовых схем и определения соответствия цветов в изображении.

Т.13. **Символьные (строковые) переменные. Встроенные функции для обработки строк**

1	Посетите форумы разработчиков и найдите советы по эффективному форматированию строк в разработке на Python. Изучите различные способы форматирования, включая **f-строки**, метод **format** и **%-форматирование**. Составьте опорный конспект по этим материалам, включите в него свои выводы.
2	Предложите ученикам на выбор несколько ссылок на форумы разработчиков и программистские онлайн-сообщества. Объясните, что целью выполнения задания является расширение области знаний по вопросам форматирования строк в Phyton и умение делать свои вывода на основе изученного материала. Вариант опорного конспекта может выглядеть так. В разработке на Python существует несколько способов форматирования строк, каждый из которых имеет свои особенности. Рассмотрим три основных метода: f-строки, метод format и %-форматирование. **F-строки** представляют собой синтаксис, который позволяет вставлять значения переменных непосредственно в строку. Пример: name = "John" age = 25 formatted_string = f"Привет, меня зовут {name} и мне {age} лет." print(formatted_string) **Метод format** позволяет более гибко форматировать строки, вставляя значения переменных с использованием заполнителей. Пример: name = "Alice" age = 30 formatted_string = "Привет, меня зовут {} и мне {} лет.".format(name, age) print(formatted_string) **%-форматирование** предоставляет стиль форматирования, аналогичный тому, который используется в языке C. Пример: name = "Bob" age = 22 formatted_string = "Привет, меня зовут %s и мне %d лет." % (name, age) print(formatted_string) **%s** означает строку (**string**), заполнитель используется для вставки строковых значений в форматируемую строку. В данном случае, переменная **name** является строкой. **%d** означает целое число (**integer**), заполнитель используется для вставки целочисленных значений в форматируемую строку. В данном случае, переменная **age** является целым числом. Вывод. Каждый из этих методов подходит для различных сценариев. **F-строки** предоставляют чистый и читаемый синтаксис, метод **format** обеспечивает гибкость,

	а **%-форматирование** может быть удобным при работе с уже знакомым стилем форматирования из языка С. Важно выбирать подходящий метод в зависимости от конкретной задачи и предпочтений разработчика

Т.14. **Работа со срезами, получение элемента строки или подстроки**

1	Посетите форумы разработчиков и найдите материалы для подготовки сообщения на тему «Использование срезов **в многострочных строках**». В этом сообщении дайте определение понятиям: «многострочная строка» и «тройные кавычки» в Phyton. Найдите примеры использования срезов в многострочных строках.

2	Предложите ученикам на выбор несколько ссылок на форумы разработчиков и программистские онлайн-сообщества. Объясните, что целью выполнения задания является расширение области знаний по вопросам работы со срезами в Phyton.
	Пример сообщения может выглядеть так.
	Тройные кавычки в Python (три одинарные ''' или три двойные """) используются для создания многострочных строк. Это удобно, когда вам нужно включить несколько строк текста в переменную. Преимущества тройных кавычек: «многострочность» – можно вставлять строки текста на разных строках без необходимости использования символов новой строки (\n), «сохранение форматирования» – текст внутри тройных кавычек сохраняет форматирование, включая отступы. Таким образом, тройные кавычки упрощают работу с многострочным текстом в Python, делая код более читаемым. Рассмотрим задачу с использованием срезов в многострочных строках.
	# Задана многострочная строка
	multiline_string = """Это
	многострочная
	строка
	с несколькими
	строками текста."""
	# Находим индекс начала интересующего нас фрагмента из строки
	start_index = multiline_string.find("многострочная")
	# Находим индекс конца интересующего нас фрагмента из строки
	end_index = multiline_string.find("строками") + len("строками")
	# Вырезаем нужный фрагмент
	text = multiline_string[start_index:end_index]
	# Выводим результат
	print(text)

Т.15. **Округление чисел: функция round и модуль math**

1	Поищите информацию о том, как можно использовать функцию **round** и модуль **math** для округления числа до ближайшего кратного. Приведите пример такого кода, для вывода результата используйте **f-строку**. Рассмотрите сценарии из практической жизни, где такое округление может быть важным для работы с кратными данными.

2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является не только стандартное решение типичной для округления задачи, а также о широком спектре применения этого решения в практической жизни.
	Пример решения задачи может выглядеть следующим образом.
	import math
	# Вводим число, которое нужно округлить до ближайшего кратного
	number = float(input("Введите число для округления: "))
	# Вводим кратное число, до которого нужно округлить

	multiple = float(input("Введите кратное число: "))
	# Вычисляем ближайшее кратное число с использованием функции round и модуля math
	rounded_number = round(number / multiple) * multiple
	# Выводим результат
	print(f"Ближайшее кратное {multiple} для числа {number} равно {rounded_number}")
	Сценарии использования такого округления в практической жизни:
	▪ количество товара в упаковке - супермаркеты могут округлять количество товара в упаковке, чтобы упростить управление запасами и инвентаризацию;
	▪ количество времени/ресурсов в проекте – в проекте, где требуется определенное количество времени или ресурсов для выполнения задачи, округление до ближайшего кратного может облегчить планирование и распределение ресурсов;
	▪ процентные доли в распределении бюджета – при распределении бюджета на различные области проекта, округление до ближайшего кратного процента может упростить выделение средств;
	▪ размеры деталей в производстве – в производственной отрасли округление размеров деталей до ближайшего кратного может быть важным для стандартизации и упрощения процессов производства.

Т.16. **Циклы с ограниченным количеством повторений. Подсчет частоты появления символа. Посимвольная обработка строк.**

1	Поищите информацию о том, как работает в Phyton функция **all()**, найдите примеры её применения в программном коде для проверки условий, созданных генераторными выражениями. Попробуйте составить программу, которая имитирует поиск слова на странице сайта. Ваша программа должна запрашивать текст (имитация содержания страницы веб-сайта) и паттерн (имитация ввода слова для поиска), далее программа «пробегает» весь текст для поиска паттерна и подсчитывает количество паттернов в тексте. В ходе выполнения задания попробуйте получить помощь от комьюнити программистов.

2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является не только написание программы, а также умение выстраивать конструктивный диалог на форумах разработчиков. В качестве примера результатов.
	1) функция **all** проверяет, все ли условия выполнены для элементов в переданном ей итерируемом объекте. Если для каждого элемента условие истинное, функция возвращает **True**, в противном случае — **False**.
	2) Код программы:
	web_text = input("введите исходный текст: ",)
	pattern = input("введите паттерн: ",)
	pattern_len = len(pattern)
	text_len = len(web_text)
	count = 0
	for i in range(text_len - pattern_len + 1):
	if all(web_text[i + j] == pattern[j] for j in range(pattern_len)):
	count += 1
	print("Количество паттернов = ", count)
	
	Работа кода:
	введите исходный текст: это текст, это тоже текст и это тоже текст

	введите паттерн: текст Количество паттернов = 3

Т.17. **Проверка делимости и другие алгоритмы для решения математических задач с циклами. Проверка числа на простоту.**

1	Поищите информацию о том, что такое **совершенное число** и как проверить, является ли заданное число совершенным. Найдите примеры кода, демонстрирующие алгоритм проверки числа на совершенность.

2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является не только изучение нового понятия «совершенные числа», а также применение уже полученных знаний для составления программы для решения задач на определение совершенности заданного пользователем числа.
	Пример определения нового понятия.
	Совершенные числа — это такие натуральные числа, сумма делителей которых (кроме самого числа) равна самому числу.
	Пример совершенного числа - 28, так как 1 + 2 + 4 + 7 + 14 = 28.
	Код программы для определения совершенности числа.
	# Получаем число от пользователя
	user_number = int(input("Введите число: "))
	# Инициализируем сумму делителей
	sum = 0
	# Перебираем числа от 1 до половины заданного числа (так как делители не могут быть больше половины числа)
	for i in range(1, user_number // 2 + 1):
	# Если i является делителем, добавляем его к сумме
	if user_number % i == 0:
	sum = sum + i
	# Проверяем, является ли сумма делителей равной заданному числу
	if sum == user_number:
	print("это совершенное число.")
	else:
	print(" это не совершенное число.")

Т.18. **Циклы с неограниченным количеством повторений.**

1	Поищите информацию о том, что принято **считать «счастливым» билетом**. Найдите различные варианты реализации программ для определения является билет с данным номером «счастливым». Попробуйте самостоятельно составить программу, которая запрашивает у пользователя номер билета и определяет, является ли он счастливым. Пользователь должен иметь возможность остановить работу программы, введя кодовое слово или число (например, слово «стоп»). При желании, можно обратиться к помощи сообщества форума программистов.
2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является не только изучение нового понятия «счастливый билет», а также поиск различных вариантов решения задачи и формирование собственного решения. Пример определения нового понятия. Счастливый билет — номер билета является "счастливым", если первая половина совпадает с зеркально отраженной второй половиной.

	Пример счастливого билета - 123321
	Код программы для определения счастливого билета.
	while True:
	  user_input = input("Введите номер билета или введите 'Стоп' для выхода: ")
	  if user_input.lower() == "стоп":
	  print("Программа завершена.")
	  break
	  # Проверка на "счастливый" билет
	  if len(user_input) == 6 and user_input[:3] == user_input[3:][::-1]:
	  print("Билет - счастливый!")
	  else:
	  print("Билет - несчастливый.")

Т.19. **Циклы с неограниченным количеством повторений для решения математических задач.**

1	Поищите информацию о том, что из себя представляет понятие **«гармонический ряд»**. Создайте собственное задание «Математическая гармония». Цель задания – создать программу, которая позволяет пользователю ввести значение, к которому нужно приблизить сумму гармонического ряда. Программа использует цикл, чтобы приближаться к этому значению, увеличивая число шагов. Как только сумма гармонического ряда превысит или станет равной целевому значению, программа завершит выполнение и выведет результат.

2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является знакомство с классическими задачами программирования. Учащиеся также практикуются анализировать и редактировать «под себя» уже существующее программное решение.
	Пример описания нового понятия.
	Гармонический ряд – представляет собой бесконечную сумму обратных значений натуральных чисел. Формально гармонический ряд записывается следующим образом:
	$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \ldots$
	Таким образом, каждый член гармонического ряда представляет собой обратное значение натурального числа. Гармонический ряд известен тем, что его сумма (гармоническая сумма) неограниченно возрастает по мере добавления новых членов. То есть, чем больше членов ряда учитывается, тем больше становится его сумма.
	Пример условия задания «Математическая гармония».
	Создайте программу, используя цикл, которая определит, как быстро сумма обратных чисел (гармонический ряд) приближается к определенному значению.
	Пример кода решения.
	# Ввод значения, к которому приближается сумма гармонического ряда
	target_value = float(input("Введите значение, к которому приближается сумма гармонического ряда: "))
	# Инициализация переменных
	harmonic_sum = 0
	current_term = 1
	# Цикл для пошагового приближения к целевому значению
	while harmonic_sum < target_value:
	  harmonic_sum += 1 / current_term
	  current_term += 1

	# Вывод результатов
	print(f"Сумма гармонического ряда приблизилась к {target_value} за {current_term - 1} шагов.")

Т.20. **Рисование символами: строка символов, основные операции и функции.**

1	Многие культуры имеют уникальные **национальные узоры**, состоящие из специфических элементов и символов. Поищите информацию в сети интернет, изучите и опишите характерные элементы для выбранного национального узора, а затем создайте программу на языке программирования Python, которая будет формировать этот узор с использованием символов.
2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Это задание поможет ученикам не только изучить различные культуры, но и применить свои навыки программирования для создания интересных символьных решений для практической жизни. Предложите ученикам следующий план исследования и воссоздания узора: ▪ Выбор национального узора – выберите национальный узор, который вас заинтересовал. Можете рассмотреть узоры из различных стран, таких как японские, индийские, африканские и др. ▪ Исследование узора – опишите основные элементы, которые характеризуют выбранный национальный узор. Это могут быть фигуры, символы, линии и т.д. ▪ Создание программы – напишите программу на Python, которая формирует узор, используя описанные вами элементы. Программа должна быть способной воссоздавать узор с различными параметрами (например, размер узора, цвета). ▪ Визуализация – запустите программу и визуализируйте созданный узор. Можете использовать различные символы для представления элементов узора. ▪ Дополнительные возможности – добавьте параметры для пользовательского ввода, чтобы пользователь мог настраивать различные аспекты узора. Можно предложить учащимся поделиться своим исследованием и программой с классом, объяснив, почему выбран именно этот национальный узор. К примеру, учащийся может выбрать национальный узор Греции и составить программу, выводящую символами «греческий ключ» (меандр).

Т.21. **Рисование символами: вывод строк с помощью цикла с ограниченным количеством повторений.**

1	Поищите информацию о том, что представляет из себя понятие **«символьная гипноз-спираль»** в псевдографике. Рассмотрите различные методы создания гипнотических спиралей из символов. Опишите, как использовать циклы для создания визуального эффекта спирали. Напишите свое условие задания по созданию символьной гипноз-спирали и предложите вариант кода для её решения.
2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Ученик должен провести исследование по различным методам создания гипнотических спиралей из символов и описать их. Это может включать в себя рассмотрение различных подходов к распределению символов, использование разных символов для создания эффекта, изменение размеров и форм спиралей, а также эксперименты с разными угловыми скоростями. Ученик должен объяснить, как использовать циклы для создания визуального эффекта спирали. Это может включать в себя описание логики циклов, вычисление координат для размещения символов, изменение параметров для создания различных эффектов и т. д.

	Пример описания нового понятия.
	Термин "символьная гипноз-спираль" используется для описания визуального эффекта, когда символы или изображения формируют спиральную структуру, создавая при этом определенный визуальный паттерн, который может вызывать гипнотическое или завораживающее восприятие у зрителя. Этот эффект может быть достигнут различными методами, такими как вращение, масштабирование или другие преобразования символов вокруг центральной точки.
	В программировании задача "символьной гипноз-спирали" может включать в себя использование циклов для повторяющегося вывода символов в форме спирали, создавая при этом визуальный эффект, аналогичный гипнотической спирали.
	Пример условия задания.
	Напишите программу для создания «символьной гипноз-спирали», используя возможности библиотеки **math.** Программа должна выводить заданный символ на экран, создавая гипнотическую спираль. Угол поворота увеличивается на заданное количество градусов на каждой итерации цикла, что создает визуальный эффект вращения. Программа запрашивает у пользователя радиус, скорость вращения и символ для отображения на спирали, что делает ее более интерактивной
	Код решения может иметь такой вид.
	import math
	# Запрашиваем параметры спирали у пользователя
	radius = float(input("Введите радиус спирали: "))
	rotation_speed = float(input("Введите скорость вращения спирали: "))
	symbol = input("Введите символ для отображения на спирали: ")
	# Цикл для создания гипнотической спирали
	for angle in range(0, 720, 5):
	  # Рассчитываем координаты точки на спирали
	  x = int(radius * math.cos(math.radians(angle)))
	  y = int(radius * math.sin(math.radians(angle)))
	  # Выводим символ в заданных координатах
	  print(' ' * (x + 20) + symbol)
	  # Изменяем радиус для создания эффекта спирали
	  radius = radius + rotation_speed
	примечание:
	range(0, 720, 5) представляет собой генератор последовательности чисел от 0 до 720 с шагом 5. Это используется для итерации через углы от 0 до 720 градусов с интервалом в 5 градусов при создании символьной гипнотической спирали. Если вы хотите усложнить задание, то сделайте так, чтобы пользователь вводил эти параметры, использовав **input** для запроса значений у пользователя перед использованием функции **range**.

Т.22. **Рисование символами: вывод строк с помощью цикла с неограниченным количеством повторений. Создание эффекта анимации.**

1	Поищите в сети интернет информацию о том, что такое визуальные свойства символов, а на специализированных сайта изучите, как работает **end=""** и **flush=True** – параметры функции **print()**. На основе этих материалов напишите условие задания для реализации эффекта анимации в текстовом виде с помощью цикла с неограниченным количеством повторений и модуля **time**. Задачу с анимацией текста и символов, которые поочередно быстро отображаются рядом, частно называют **«бегущая строка».**

2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, какие свойства символов влияют на их визуальные свойства.
	Учащиеся практикуются осознанно применять критерии выбора свойств для их дальнейшей практической реализации в качестве элемента программного решения.
	Пример исследования визуальных свойств символов.
	Оцените визуальные свойства символов: форма, яркость, контрастность и легкость восприятия. Рассмотрите, какие символы могут хорошо вписываться в программное решение, обеспечивая четкую структуру и легкость понимания.
	Критерии визуальных свойств символов:
	▪ Четкость формы – символы должны иметь четко выраженные формы без излишней сложности.
	▪ Яркость и контрастность – выбирайте символы с хорошей видимостью и контрастом, чтобы обеспечить легкость восприятия.
	▪ Легкость восприятия – удостоверьтесь, что выбранные символы легко воспринимаются в контексте именно вашего программного решения и не вызывают путаницы.
	Пример изучения понятия.
	Параметр **flush=True** представляет собой параметр функции **print**() в языке программирования Python. Этот параметр указывает на необходимость принудительной выгрузки (очистки) буфера вывода, что означает, что данные будут сразу же отправлены в поток вывода, а не оставлены в буфере до завершения программы или до того момента, когда буфер заполнится. Использование **flush=True** обеспечивает мгновенный вывод данных без задержек, что может быть полезно, например, при создании анимаций в реальном времени или в других случаях, когда требуется немедленный вывод текста.
	Параметр **end=""** указывает, что при окончании вывода строки **print**() не должен добавлять символ новой строки (по умолчанию **end="\n"**). В результате, следующий вывод будет продолжаться с той же строки, где закончился предыдущий вывод.
	Пример условия задания на анимацию текста.
	Бегущая строка с символом анимации. Ваша задача — создать программу для эффектной бегущей строки с использованием символа анимации. Пользователь вводит текст, который будет отображаться статично, и один символ, который будет "бежать" рядом с текстом, создавая анимацию. Программа должна обеспечивать непрерывное выполнение анимации с заданным интервалом между кадрами.
	Пример кода решения задания на анимацию текста.
	`import time`
	`# Ввод текста пользователя`
	`text = input("Введите текст для анимации: ")`
	`# Ввод символа анимации пользователя`
	`animation_symbol = input("Введите символ для анимации: ")`
	`interval = 0.1  # Интервал между кадрами анимации`
	`while True:  # Бесконечный цикл анимации`
	`  print(f"{text} {animation_symbol}", end="", flush=True)`
	`  time.sleep(interval)`

Т.23. **Рандомизация выбора числа.**

1	Поищите информацию о том, как работают функции **chr(), ord()** и что такое «шифр Цезаря». Создайте собственное задание «**Шифр Цезаря**: математическая интрига». Цель задания – создать программу, которая позволяет пользователю ввести текст

	сообщения. Размер сдвига (ключ) создается с помощью функции **random**, работающей в диапазоне целых чисел от 1 до 7. Включите в условие задания описание, что собой представляет шифр Цезаря, а также информацию о том, что регистр букв должен сохраняться и необходимо указать пользователю, что символ – это буква латинского алфавита. В идеале программа должна зашифровать введенное пользователем сообщение с использованием рэндомного размера сдвига и вывести результат.

2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является знакомство с классическими задачами программирования и расширение области полученных знаний по теме. Учащиеся также практикуются анализировать и редактировать «под себя» уже существующее программное решение
	Пример описания функций.
	Функция chr принимает целочисленное значение (код символа в Unicode) и возвращает соответствующий символ. Синтаксис: chr(числовое_значение).
	Пример: chr(65) вернет строку 'A', так как 65 - это код символа 'A' в Unicode.
	Функция ord принимает символ и возвращает его числовое значение в Unicode. Синтаксис: ord('символ').
	Пример: ord('A') вернет числовое значение 65, так как 'A' имеет код 65 в Unicode.
	Эти функции часто используются в работе с символами и строками, особенно в контексте шифрования, обработки текста и работе с символьными данными.
	Пример условия задания.
	Секретный шифр с переменным сдвигом.
	Шифр Цезаря – это метод шифрования, в котором каждая буква текста заменяется на букву с фиксированным сдвигом в алфавите. Ваша задача - создать программу для шифрования текста с использованием шифра Цезаря.
	Требования:
	Пользователь вводит текст для шифрования (только латиница).
	Программа генерирует случайный размер сдвига от 1 до 7.
	Введенный текст приводится к нижнему регистру.
	Производится шифрование текста с использованием шифра Цезаря и случайного сдвига.
	Результат выводится на экран вместе с размером сдвига.
	Код решения может иметь такой вид.
	import random
	# Ввод текста от пользователя (только латиница)
	user_text = input("Введите текст для шифрования (латиница): ")
	# Приведение текста к нижнему регистру
	user_text = user_text.lower()
	# Генерация случайного размера сдвига от 1 до 7
	shift = random.randint(1, 7)
	# Шифрование текста
	encrypted_text = ""
	for char in user_text:
	  if char.isalpha():
	  # Преобразование символа с учетом регистра
	  shifted_char = chr((ord(char) + shift - ord('a')) % 26 + ord('a'))
	  encrypted_text += shifted_char
	  else:
	  encrypted_text += char
	# Вывод результата
	print("Зашифрованный текст: ", encrypted_text)

	print("Размер сдвига: ", encrypted_text)

Т.24. **Игра «Угадай число».**

1	Компьютерные игры представляют собой уникальный мир развлечений с разнообразными жанрами. В этом исследовании мы приглашаем вас изучить основные категории компьютерных игр и выявить их концептуальные особенности. Рассмотрите в своем исследовании следующие категории: Симуляторы – исследуйте разнообразие симуляторов на компьютере; определите, какие аспекты реальности успешно передаются в симуляторах, и какие критерии делают их уникальными в своей категории. Ролевые игры (RPG) – рассмотрите основные поджанры RPG и их особенности; выделите ролевые элементы, которые сделали некоторые игры успешными в этом жанре. Спортивные игры – изучите различные виды спортивных компьютерных игр; определите, какие аспекты реальных видов спорта успешно реализованы в этих играх. Логические игры – рассмотрите концепцию логических игр и их разновидности; определите, какие критерии делают логические игры увлекательными и популярными. Экшн и приключения – исследуйте аспекты экшн и приключенческих игр; определите, какие элементы делают эти игры захватывающими и успешными. Выводы: составьте обзор по каждой категории компьютерных игр, описав концептуальные особенности и критерии успешности в каждой из них.
2	Предложите ученикам на выбор несколько ссылок на ресурсы для поиска информации. Объясните, что целью выполнения задания является развитие умения анализировать большие информационные блоки, формировать результаты анализа в краткой форме и приводить логичную аргументацию. Пример варианта исследования. Исследование: Компьютерные игры – категории и концептуальные Особенности Компьютерные игры являются важной частью современной развлекательной индустрии. Разнообразие жанров позволяет каждому игроку найти что-то по своему вкусу. В данном исследовании мы рассмотрим основные категории компьютерных игр и их концептуальные особенности. **Симуляторы** – это жанр, стремящийся максимально приблизить виртуальный опыт к реальному. В таких играх акцент делается на точной имитации определенной деятельности или среды. Например, фермерские симуляторы позволяют игрокам взаимодействовать с аспектами фермерской жизни. Концептуальная особенность в точности воспроизведения реальности. **Ролевые игры** (RPG) – ролевые игры погружают игроков в увлекательные сюжеты и предоставляют им возможность развивать своих персонажей. В поджанре MMORPG (массовые многопользовательские онлайн-ролевые игры) игроки взаимодействуют в виртуальных мирах. Особенность заключается в эпических сюжетах и постоянном развитии персональных характеристик. **Спортивные игры** – спортивные компьютерные игры успешно передают атмосферу реальных соревнований. Они позволяют игрокам соревноваться в различных видах спорта, будь то футбол, баскетбол или автогонки. Особенность в точной имитации правил и физических характеристик реальных видов спорта. **Логические игры** – логические игры ставят перед игроком задачи, требующие размышлений и стратегического мышления. От головоломок до стратегических сражений, эти игры требуют от игрока решения сложных задач. Особенность в развитии логического мышления и способности к стратегическому планированию.

	Экшн и приключения – экшн и приключенческие игры предлагают захватывающие приключения и динамичные бои. От стрельбы от первого лица до исследования загадочных

Т.25. **Создание списка, сортировка, действия над элементами, циклический просмотр списка».**

1	Исследуйте методы сортировки списков в Python. Какие алгоритмы сортировки применяются для списков и как они работают?
2	Объясните ученикам актуальность задач, связанных с сортировкой списков. Предложите им проанализировать различные алгоритмы сортировки: сравнить эффективность и исследовать примеры использования каждого алгоритма сортировки в реальных задачах. Можно рассматривать различные алгоритмы, включая: сортировка пузырьком (Bubble Sort), сортировка выбором (Selection Sort), сортировка вставками (Insertion Sort), быстрая сортировка (Quick Sort), сортировка слиянием (Merge Sort), сортировка кучей (Heap Sort). Для получения более подробной информации о каждом алгоритме предложите ученикам обратиться к официальной документации Python (https://docs.python.org) и специализированным ресурсам по программированию, например: Stack Overflow (https://stackoverflow.com/) и GitHub (https://github.com/).

Т.26. **Работа со списками, выбор случайного элемента. Игра «Предсказание».**

1	Изучите функции поиска элементов в списке в Python. Какие методы и функции предоставляются для выполнения операций поиска?
2	Объясните ученикам актуальность задач, связанных с поиском элементов в списках. Предложите им проанализировать различные методы и функции, представленные в Python для выполнения операций поиска. Можно рассматривать: ▪ метод **index**(): позволяет найти индекс первого вхождения заданного элемента в список; ▪ метод **count**(): возвращает количество вхождений заданного элемента в список; ▪ функция **in**: используется для проверки присутствия элемента в списке; ▪ методы срезов (**slicing**): позволяют получить подсписок элементов, удовлетворяющих определенному условию; ▪ функция **filter**(): позволяет отфильтровать список по заданному критерию. В качестве выполнения задания учащиеся представляют результат исследования. Примерный план результата исследования: 1) описание особенностей каждого метода и его применение в различных сценариях; 2) сравнительный анализ эффективности и производительности каждого метода; 3) оценка преимуществ и ограничений каждого метода в различных сценариях использования; 4) рекомендации по выбору методов в зависимости от конкретных задач. Для получения более подробной информации о каждом алгоритме предложите ученикам обратиться к официальной документации Python (https://docs.python.org) и специализированным ресурсам по программированию, например: Stack Overflow (https://stackoverflow.com/) и GitHub (https://github.com/).

Т.27. **Сложные списки и методы работы с ними.**

1	Проведите сравнительный анализ различных методов итерации по списку в Python, таких как цикл **for**, **while** и функции **map**(), **filter**() и т. д. Какие методы наиболее удобны и эффективны для различных сценариев использования?
2	Предложите ученикам провести сравнительный анализ различных методов итерации по списку в Python. Объясните актуальность и значимость этой темы. В аналитическом отчете рекомендуйте придерживаться следующего плана: ▪ обзор основных методов итерации по списку; ▪ описание особенностей каждого метода и его применение в различных сценариях; ▪ сравнительный анализ эффективности и удобства каждого метода для различных задач; ▪ оценка преимуществ и ограничений каждого метода в различных сценариях использования; ▪ выводы о наиболее удобных и эффективных методах итерации по списку в Python; ▪ рекомендации по выбору методов в зависимости от конкретных задач. Для получения более подробной информации о каждом алгоритме предложите ученикам обратиться к официальной документации Python (https://docs.python.org) и специализированным ресурсам по программированию, например: Stack Overflow (https://stackoverflow.com/) и GitHub (https://github.com/). Руководства и учебники по программированию на Python, доступные онлайн.

Т.28. **Работа со списками, перемешивание элементов. Игра «Угадай столицу».**

1	Исследуйте различные методы перемешивания элементов списка в Python, такие как использование функции **random.shuffle()**, алгоритм **Фишера — Йетса — Дурбина** и **рекурсивный** метод. Опишите преимущества и недостатки каждого метода, а также эффективность в зависимости от размера списка.
2	Предложите ученикам провести сравнительный анализ различных методов перемешивания списков в Python. Объясните актуальность и значимость этой темы. В проведении анализа рекомендуйте придерживаться следующего плана: 1. В начале исследования предоставить описание каждого из методов перемешивания элементов списка: функция random.shuffle(), алгоритм Фишера — Йетса — Дурбина и рекурсивный метод. 2. Провести сравнительный анализ этих методов, описав их преимущества и недостатки: ▪ преимущества и недостатки функции random.shuffle(); ▪ преимущества и недостатки алгоритма Фишера — Йетса — Дурбина; ▪ преимущества и недостатки рекурсивного метода. 3. Проанализировать эффективность каждого метода в зависимости от размера списка: ▪ как изменяется время выполнения каждого метода при различных размерах списка; ▪ определить, при каких размерах списка каждый метод становится наиболее эффективным.

	Завершить анализ общим выводом о том, какой метод наиболее подходит для конкретных сценариев использования, учитывая их преимущества, недостатки и эффективность.

Т.29. **Множества и методы работы с ними.**

1	Сравните различные методы создания множеств в Python, такие как использование фигурных скобок, функции **set**() и генераторов множеств. Опишите преимущества и недостатки каждого метода.
2	Предложите ученикам провести сравнительный анализ различных методов создания множества в Python. Объясните актуальность и значимость этой темы. В проведении анализа рекомендуйте придерживаться следующего плана: 1. описание каждого из методов создания множеств в Python: ▪ использование фигурных скобок: {}. ▪ использование функции set(). ▪ использование генераторов множеств. 2. проведение сравнительного анализа этих методов, описание их преимущества и недостатков: ▪ преимущества и недостатки создания множеств с помощью фигурных скобок. ▪ преимущества и недостатки использования функции set(). ▪ преимущества и недостатки генераторов множеств. 3. описание особенностей каждого метода: ▪ удобство использования; ▪ читаемость кода; ▪ производительность. 4. общий вывод о том, в каких ситуациях каждый метод является наиболее подходящим и эффективным, учитывая преимущества и недостатки.

Т.30. **Игра «Быки и коровы». Формирование данных.**

1	Исследуйте различные операции над множествами в Python, такие как объединение, пересечение, разность и симметрическая разность. Сравните эффективность выполнения этих операций для малых и больших множеств.
2	Предложите ученикам провести исследование продуктивности операций над множествами в Python. Объясните актуальность и значимость этой темы. В проведении исследования рекомендуйте учащимся придерживаться следующего плана: • предоставить описание каждой из операций над множествами в Python: объединение (union), пересечение (intersection), разность (difference) и симметрическая разность (symmetric difference); • приступить к сравнительному анализу эффективности выполнения этих операций для малых и больших множеств: – сравнить время выполнения каждой операции для малых множеств (несколько элементов) и оценить производительность;

	– провести аналогичные измерения для больших множеств (с большим количеством элементов) и проанализировать результаты; – определить, какие операции оказываются более эффективными для работы с малыми множествами, а какие - для больших (определить преимущества каждой операции в зависимости от размера множества, сделать выводы о том, какие операции наиболее подходят для различных сценариев использования и размеров данных); • сделать общий вывод о том, какие операции над множествами следуем предпочитать в различных ситуациях, учитывая их эффективность и производительность при работе с малыми и большими объемами данных.

Т.31. **Игра «Быки и коровы». Обработка данных.**

1	Исследуйте различные методы обновления множества в Python, такие как методы **add**(), **update**(), **remove**() и **discard**(). Сравните их эффективность при добавлении, удалении и обновлении элементов множества.
2	Предложите ученикам провести исследование различных методов обновления множеств в Python. Объясните актуальность и значимость этой темы. В проведении исследования рекомендуйте учащимся придерживаться следующего плана: ▪ описание каждого из методов обновления множеств: add(), update(), remove() и discard(). ▪ сравнительный анализ эффективности каждого метода при различных операциях: – сравнение времени выполнения каждого метода при добавлении новых элементов в множество; – оценка производительности методов при удалении элементов из множества; – сравнение эффективности методов при обновлении элементов множества. ▪ анализ результатов и выводы: – какие методы наиболее эффективны для различных операций с множеством; – преимущества и недостатки каждого метода в зависимости от конкретной ситуации использования. ▪ общий вывод о том, какие методы обновления множеств в Python следует предпочитать в различных сценариях работы с данными.

Т.32. **Создание словаря, работа с его элементами.**

1	Проведите анализ методов добавления и удаления элементов из словаря в Python, таких как методы **update**(), **pop**(), **popitem**() и **del**. Сравните их производительность и определите наиболее эффективные методы для каждой операции.
2	Предложите ученикам провести исследование различных методов добавления и удаления элементов из словаря в Python. Объясните актуальность и значимость этой темы. Рекомендуйте учащимся структурировать свое решение следующим образом: 1. Описание методов: ▪ **Метод update**(): этот метод используется для добавления элементов из одного словаря в другой. Если ключ уже существует, его значение будет заменено.

	▪ **Метод pop**(): удаляет элемент по указанному ключу из словаря и возвращает его значение. Если ключ не найден, возникает ошибка KeyError. ▪ **Метод popitem()**: удаляет произвольный элемент из словаря. Если словарь пуст, возникает ошибка KeyError. ▪ **Оператор del**: удаляет элемент из словаря по указанному ключу. Если ключ не найден, возникает ошибка KeyError. 2. Анализ производительности: ▪ для каждого метода оцените время выполнения операции добавления и удаления элементов из словаря; ▪ оцените измерения на различных объемах данных, чтобы сделать вывод о производительности методов при различных размерах словаря. 3. Сравнительный анализ: ▪ сравните время выполнения каждого метода при добавлении и удалении элементов из словаря; ▪ определите, какие методы являются наиболее эффективными для каждой операции (добавление и удаление) и при каких условиях. 4. Общий вывод: сделайте заключение о том, какие методы добавления и удаления элементов из словаря являются наиболее эффективными в различных сценариях использования, предложите свои рекомендации по выбору методов в зависимости от типа задачи и объема данных.

Т.33. **Работа со словарями. Проект «Телефонный справочник».**

1	Исследуйте методы итерации по элементам словаря в Python, такие как использование цикла **for**, методов **keys**(), **values**() и **items**(). Оцените производительность каждого метода и их удобство использования.
2	Предложите ученикам провести исследование различных методов итерации по элементам словаря в Python. Объясните актуальность и значимость этой темы. Рекомендуйте учащимся структурировать свое решение следующим образом: 1. Описание методов: ▪ Цикл **for**: итерируется по ключам словаря. Прост в использовании и понимании. ▪ Метод **keys**(): возвращает представление всех ключей в словаре. ▪ Метод **values**(): возвращает представление всех значений в словаре. ▪ Метод **items**(): возвращает представление всех пар ключ-значение в словаре 2. Оценка производительности и удобства использования: ▪ для каждого метода оцените время выполнения итерации по словарю; ▪ оцените удобство использования каждого метода в различных сценариях. 3. Сравнительный анализ: ▪ сравните производительность каждого метода итерации по элементам словаря; ▪ определите, какой метод является наиболее эффективным с точки зрения производительности; ▪ сделайте выводы о том, какой метод наиболее удобен и эффективен в различных сценариях использования. Общий вывод: сделайте заключение о преимуществах и недостатках каждого метода итерации по элементам словаря; предложите рекомендации по выбору

	метода в зависимости от конкретной задачи и требуемой производительности.

Т.34. **Массивы и способы работы с ними.**

1	Проведите анализ различных методов создания массивов в Python, таких как использование списков, массивов библиотеки **NumPy** и генераторов массивов. Сравните их производительность и удобство использования.
2	Предложите ученикам провести исследование различных методов создания массивов в Python. Объясните актуальность и значимость этой темы. Рекомендуйте учащимся структурировать свое решение следующим образом: 1. Описание методов создания массивов. ✦ Использование списков: в Python списки могут быть использованы для хранения последовательностей элементов любых типов данных, включая числа. Списки создаются с помощью квадратных скобок [] и могут содержать любое количество элементов. ✦ Массивы библиотеки NumPy: NumPy — это библиотека Python для научных вычислений, которая предоставляет множество функций для работы с многомерными массивами. Массивы NumPy создаются с помощью функции numpy.array() или других функций для создания специфических типов массивов. ✦ Генераторы массивов: в Python существует возможность создавать массивы с помощью генераторов списков или генераторов массивов в библиотеке NumPy. Генераторы массивов представляют собой компактный и эффективный способ создания массивов на основе определенного шаблона или правила. 2. Измерение производительности. ✦ Для каждого метода создания массивов проведите измерения времени выполнения. ✦ Создайте массивы различных размеров и типов данных для оценки производительности. 3. Оценка удобства использования. ✦ Оцените удобство использования каждого метода с точки зрения читаемости кода, удобства написания и понимания. ✦ Учитывайте также возможность работы с различными типами данных и сложность создания специализированных массивов. 4. Сравнительный анализ. ✦ Сравните результаты производительности каждого метода создания массивов. ✦ Сделайте выводы о том, какой метод является наиболее эффективным и удобным в различных сценариях использования. **Общий вывод:** сделайте общие выводы о преимуществах и недостатках каждого метода создания массивов, и предложите рекомендации по выбору метода в зависимости от конкретной задачи и требуемой производительности.

Т.35. **Типовые алгоритмы обработки числовых массивов.**

1	Проведите анализ различных методов поиска элементов в числовом массиве в Python, таких как использование циклов, методов **index**() и **searchsorted**(). Оцените их производительность и эффективность при работе с различными объемами данных и типами поиска.
2	Предложите ученикам провести исследование различных методов поиска элементов в числовом массиве в Python. Объясните актуальность и значимость этой темы. Рекомендуйте учащимся структурировать свое решение следующим образом: 1. Описание методов поиска элементов в массиве. ✦ Использование циклов: поиск элементов в массиве с использованием циклов осуществляется путем итерации по каждому элементу массива и проверки условия на совпадение. ✦ Метод **index**(): этот метод используется для поиска индекса первого вхождения указанного элемента в массиве. Если элемент не найден, вызывается исключение **ValueError**. ✦ Метод **searchsorted**(): этот метод используется для поиска элемента в отсортированном массиве. Он возвращает индекс вставки элемента в массив таким образом, чтобы массив оставался отсортированным. 2. Оценивание производительности и эффективности. ✦ Для каждого метода поиска оцените время выполнения на различных объемах данных. ✦ Оцените эффективность каждого метода для разных типов поиска, таких как поиск конкретного значения, поиск минимального или максимального значения и т. д. 3. Сравнительный анализ. ✦ Сравните результаты производительности и эффективности каждого метода поиска. ✦ Определите, какой метод является наиболее быстрым и эффективным для каждого типа поиска и для разных объемов данных. **Общий вывод:** сделайте общие выводы про преимущества и недостатки каждого из рассмотренных методов поиска элементов, предложите свои рекомендации по выбору метода в зависимости от типа поиска, объема данных и с учетом ожидаемой производительности.

Т.36. **Двумерные массивы. Основные алгоритмы обработки двумерных массивов.**

1	Изучите возможности работы со списками с использованием библиотеки **NumPy** в Python. Какие дополнительные функции и методы предоставляет **NumPy** для работы с данными в виде списков?
2	Объясните актуальность и значимость этой темы. Предложите учащимся провести исследование возможностей работы со списками с использованием библиотеки **NumPy** в Python. В исследовании учащиеся могут использовать следующий план: ▪ краткое описание библиотеки NumPy и её роли в анализе данных; ▪ основные преимущества использования NumPy для работы со списками; ▪ основные функции и методы NumPy;

	▪ примеры использования основных функций и методов NumPy для различных задач обработки данных; ▪ выводы о преимуществах и перспективах использования библиотеки NumPy для работы со списками в Python. Для получения более подробной информации предложите обратиться к следующим ресурсам: – Официальный сайт библиотеки NumPy: https://numpy.org/ – Документация NumPy: https://numpy.org/doc/ – Руководства и учебники по использованию NumPy, доступные онлайн. – Примеры кода и обсуждения на форумах для программистов, таких как Stack Overflow (https://stackoverflow.com/).

Т.37. **Пользовательские функции и методы работы с ними.**

1	Исследуйте методы работы с параметрами функции в Python, такие как позиционные аргументы, именованные аргументы и аргументы по умолчанию. Сравните их применимость в различных сценариях использования.
2	Объясните актуальность и значимость этой темы. Предложите учащимся провести исследование возможностей различных методов работы с параметрами функции в Python. Пример исследования по этому заданию может выглядеть следующим образом. ▪ **Позиционные** аргументы – определяются в порядке их указания при вызове функции. Используются, когда количество и тип аргументов известны заранее, и порядок их передачи важен. *Пример*: def greet(name, greeting): *Вызов*: greet("Alice", "Hi") ▪ **Именованные** аргументы – представляют собой пары "имя = значение", указанные при вызове функции. Используются для явного указания значений аргументов и улучшения читаемости кода, позволяют изменять порядок передачи аргументов без нарушения работы функции. *Пример*: def greet_named(name="Anonymous", greeting="Hello"): *Вызов*: greet_named(name="Bob", greeting="Hey") ▪ Аргументы **по умолчанию** – имеют значение, которое будет использоваться, если аргумент не передан при вызове функции. Используются для установки значений, которые часто используются и редко изменяются, позволяют сделать функцию более гибкой, поскольку могут быть вызваны без указания всех аргументов. *Пример*: def greet_default(name="Anonymous", greeting="Hello"): *Вызов*: greet_default() **Сравнение**: – Позиционные аргументы обеспечивают простоту использования и являются наиболее распространенным методом передачи аргументов. – Именованные аргументы улучшают читаемость кода и позволяют изменять порядок передачи аргументов без изменения самой функции. – Аргументы по умолчанию делают функцию более гибкой, позволяя вызывать её без указания всех аргументов, и уменьшают количество необязательных аргументов в вызове функции. Однако, следует быть осторожными при

	использовании аргументов по умолчанию, чтобы избежать неявных побочных эффектов при изменении значений по умолчанию.

Т.38. **Разбиение задачи на подзадачи с использованием функций. Проект «Загадки». Рекурсия.**

1	Исследуйте различные способы определения пользовательских функций в Python, такие как использование ключевого слова **def**, **лямбда-выражений** и **функций-генераторов**. Сравните их применимость и эффективность в различных сценариях использования.
2	Объясните актуальность и значимость этой темы. Предложите учащимся провести исследование возможностей различных методов работы с параметрами функции в Python. Пример исследования по этому заданию может выглядеть следующим образом. **Способы определения пользовательский функций.** ▪ Использование ключевого слова **def**: ключевое слово **def** используется для определения именованных функций. Подходит для определения функций с произвольным числом операторов и возможностью многократного вызова, обеспечивает более читаемый и структурированный код благодаря явному указанию имени функции. ▪ **Лямбда-выражения**: позволяют определить анонимные функции в одной строке кода без использования ключевого слова **def**. Подходят для определения простых функций, которые выполняют одну операцию или преобразование данных, удобны в случаях, когда функция используется в качестве аргумента для другой функции. ▪ **Функции-генераторы**: используют ключевое слово **yield** для возврата значений во время выполнения функции, что делает их итерируемыми. Подходят для обработки больших объемов данных и ленивой загрузки результатов, позволяют экономить память за счет генерации значений по мере необходимости. **Сравнение**: – Использование ключевого слова def предпочтительно для определения именованных функций с несколькими операторами и повторным вызовом. – Лямбда-выражения удобны для создания простых функций в одной строке кода, особенно при использовании встроенных функций, таких как map(), filter() и sorted(). Функции-генераторы предпочтительны для обработки больших объемов данных, где требуется ленивая загрузка результатов и экономия памяти. Они особенно полезны при работе с бесконечными последовательностями или потоками данных

Т.39. **Алгоритм «Мои результаты». Проведение итоговой викторины.**

1	Исследуйте методы оптимизации пользовательских функций в Python, такие как использование декораторов **@lru_cache** и **@jit**. Оцените их влияние на производительность функций. *Декоратор в Python – это особый вид функции, которая принимает другую функцию в качестве аргумента и возвращает новую функцию. Он позволяет изменять

	поведение функции или добавлять к ней дополнительную функциональность, не изменяя её исходного кода.
2	Объясните актуальность и значимость этой темы. Предложите учащимся провести самостоятельное исследование методов оптимизации пользовательских функций в Python. Пример исследования по этому заданию может выглядеть следующим образом. Декораторы @lru_cache и @jit: **@lru_cache** – это декоратор из стандартной библиотеки Python **functools**, который предназначен для кеширования результатов вызова функции с определенным набором аргументов. Он использует стратегию "Least Recently Used" (LRU), чтобы автоматически удалять наименее недавно использованные элементы из кеша, когда он достигает предельного размера. *Применение*: @lru_cache применяется, когда нужно повысить производительность функций, которые часто вызываются с одними и теми же аргументами. Он хорошо подходит для функций с долгим временем выполнения или с вычислительно сложными операциями, которые можно кешировать. **@jit** – это декоратор из библиотеки **numba**, который компилирует функцию в машинный код для ускорения её выполнения. JIT (Just-In-Time) компиляция происходит во время выполнения программы, что позволяет оптимизировать скорость выполнения функций. *Применение*: @jit применяется для оптимизации скорости выполнения функций, особенно тех, которые содержат циклы или выполняют вычисления над массивами данных. Он может быть полезен при работе с большими объемами данных в численных вычислениях. Сравнение: – @lru_cache используется для кеширования результатов вызова функции с определенным набором аргументов, что уменьшает время выполнения функции при повторных вызовах с теми же аргументами. – @jit используется для компиляции функций в машинный код, что позволяет значительно ускорить выполнение функции, особенно при работе с числовыми вычислениями. @lru_cache применяется для кеширования результатов и повышения производительности функций, а @jit - для оптимизации скорости выполнения функций за счет компиляции в машинный код

Т.40. **Модуль turtle: модель RGB, понятие объект, основные команды управления.**

1	Найдите и изучите различные способы создания градиентов с помощью модуля **turtle**. Предложите свою собственную модель создания градиента, используя методы модуля **turtle**.
2	Объясните актуальность и значимость визуализации данных в работе с модулем. Предложите учащимся провести самостоятельное исследование о возможностях использования градиентных заливок. Пример представленной модели создания градиента. Для создания градиента с помощью модуля **turtle** мы можем использовать комбинацию различных цветов и плавно изменять их от одного к другому. Например,

	для создания градиента от красного к желтому, мы можем использовать следующий алгоритм: 1. Определим два цвета - красный (255, 0, 0) и желтый (255, 255, 0). 2. Создадим черепаху и установим ее в начальную позицию. 3. Установим ширину пера равной ширине экрана, чтобы закрасить его полностью. 4. Начнем рисование градиента, двигая черепаху по горизонтальной оси. 5. На каждом шаге будем изменять цвет пера так, чтобы он плавно переходил от красного к желтому. 6. После достижения конечной позиции, остановим черепаху. Заключение: это пример простого градиента от красного к желтому по горизонтальной оси. Можно изменять параметры, такие как ширина экрана, цвета и направление градиента, чтобы создавать свои уникальные эффекты.

Т.41. **Модуль turtle: рисование замкнутых многоугольников, заливка фигур.**

1	Исследуйте способы создания трехмерных многогранников, таких как кубы, призмы и пирамиды, с использованием модуля **turtle**. Найдите информацию о том, как реализовать масштабирование трехмерных объектов с помощью модуля **turtle.**
2	Объясните актуальность и значимость визуализации данных в работе с модулем. Предложите учащимся провести самостоятельное исследование о возможностях создания и масштабирования трехмерных многогранников. Пример представленной модели создания градиента. Рассмотрим алгоритм работы функции по созданию трехмерных многогранников на примере функции **draw_cube**. Эта функция позволяет черепахе рисовать куб на экране. В качестве аргументов она принимает черепаху для рисования **t** и размер стороны куба **size**. *Алгоритм* работы функции: – Черепаха рисует каждую сторону куба, поворачивая направо на 90 градусов после каждого рисования, чтобы создать прямоугольник. – После рисования четырех прямоугольников для основания куба, черепаха поднимается и перемещается на верхнюю сторону куба, начиная с вершины одного из углов. – Затем черепаха опускается и рисует четыре диагональные линии, соединяющие вершины основания и верха куба. – Это позволяет создать трехмерный эффект куба на экране. Рассмотрим алгоритм работы функции масштабирования **scale_object**. Данная функция предназначена для масштабирования объектов, нарисованных черепахой. Она принимает черепаху **t**, объект которой будет масштабирован, и масштабный коэффициент **scale_factor**. Принцип работы функции заключается в изменении размера объекта черепахи. Коэффициент **scale_factor** определяет, на сколько раз увеличится размер объекта. Например, если scale_factor = 2, объект увеличится вдвое. **Другие варианты** масштабирования: – Масштабирование по осям: можно применить различные масштабные коэффициенты к разным осям, чтобы изменить форму объекта.

	– Отзеркаливание: можно отражать объект относительно определенной оси, применяя отрицательные масштабные коэффициенты к соответствующим осям. – Произвольное масштабирование: можно комбинировать масштабные коэффициенты по разным осям для создания произвольных эффектов масштабирования. Эти инструменты позволяют экспериментировать с различными методами масштабирования и создавать разнообразные трехмерные модели с использованием модуля **turtle**

Т.42. **Модуль turtle: рисование полных и неполных окружностей.**

1	Исследуйте возможности использования модуля **turtle** для создания образовательных игр. Предложите свой собственный план для образовательной игры, которую можно создать с помощью модуля **turtle**.
2	Объясните учащимся насколько важно понимать аспекты практического применения знаний в области программирования. Предложите провести самостоятельно процесс творческого планирования для создания образовательной игры с использованием возможностей модуля turtle. Пример варианта решения. – Изучение принципов создания игр: основные принципы создания образовательных игр, такие как выбор темы, определение целей и задач игры, создание уровней сложности и схем взаимодействия с пользователем. – Поиск вдохновения и идей: обратить внимание на различные области образования и проанализировать игры в этих областях. Выбрать область для создания своей собственной образовательной игры (например, геометрия). – Разработка концепции игры: на основе выбранной области решаем определяющую концепцию своей игры. Например, это может быть игра, в которой игроку нужно решать геометрические задачи по теме «Окружность и круг», перемещая черепаху по экрану и рисуя фигуры. – Создание прототипа: на этом этапе с помощью модуля turtle создаем прототип своей игры. Определяем основные элементы игры: интерфейс пользователя, игровые объекты и т.д. Одна из идей – игра «Учим геометрию». Игроку предлагается решать различные геометрические задачи по теме «Окружность и круг»: нахождение площади и периметра фигур, построение геометрических форм с учетом подобия и пропорций, угадывание геометрических терминов. К примеру, для нахождения площади круга игроку нужно переместить черепаху и нарисовать круг нужного размера. Чем быстрее и точнее игрок решит задачу, тем больше очков он получит. Игра может содержать несколько уровней сложности в зависимости от уровня знаний. Вывод: таким образом, создание игры на основе данной идеи с использованием модуля turtle позволит игрокам не только развлечься, но и улучшить свои навыки в области геометрии.

Т.43. **Модуль turtle: рисование полных и неполных окружностей.**

1	Изучите различные методы рисования сложных циклических фигур, таких как спирали, круги и петли, используя модуль **turtle**. Какие факторы влияют на форму и размер таких фигур.
2	Объясните значимость возможностей модуля turtle в создании сложных циклических фигур. Предложите самостоятельно разобраться в методах их рисования и подготовить сообщение на эту тему. Примерный план самостоятельного исследования. 1. Исследование доступных методов: изучить различные подходы к рисованию сложных циклических фигур с помощью модуля turtle (обратиться к руководству по использованию turtle и ресурсам онлайн-сообщества для изучения различных техник и приемов). 2. Эксперименты с параметрами: провести эксперименты с различными параметрами для понимания того, какие факторы влияют на форму и размер циклических фигур. Можно включать в эксперименты изменение шага, угла поворота, настройку скорости черепахи. 3. Изучение влияния параметров: проанализировать как изменения влияют на форму и размер циклических фигур. Например, увеличение шага при рисовании спирали может изменить ее радиус и количество оборотов, а изменение угла поворота при рисовании круга может изменить его диаметр и форму. 4. Эксперименты с комбинациями параметров: проанализировать эксперименты с различными комбинациями параметров, которые позволяют создать более сложные и интересные циклические фигуры. Например, комбинации разных шагов и углов поворота для создания уникальных визуальных эффектов. 5. Анализ результатов: проанализировать все результаты экспериментов и сделать выводы о том, какие факторы наиболее сильно влияют на форму и размер циклических фигур.

Т.44. **Модуль turtle: перемещение объекта по координатам, работа с костюмами.**

1	Изучите возможности использования физического моделирования с помощью черепахи для создания симуляций. Какие законы физики можно учесть при создании таких программ?
2	Объясните учащимся насколько разнообразны области практического применения знаний в области программирования. Предложите провести самостоятельное планирование для создания программы физического моделирования с помощью модуля turtle. Пример планирования. – Изучение физических законов: изучим основные законы физики, которые можно учесть при создании симуляций с помощью черепахи. Это может включать законы движения Ньютона, законы сохранения энергии и импульса, законы термодинамики и другие.

	– Поиск примеров и идей: найдем примеры существующих симуляций, созданных с использованием модуля черепахи, а также идеи для новых симуляций. Это может быть моделирование движения тел и коллизий, гравитации, волн и звуковых волн, теплопередачи и т. д. – Разработка симуляции: на основе полученных знаний и идей разработаем собственную симуляцию с использованием черепахи. Определим физические законы и параметры, которые будут использоваться в симуляции, и реализуем их с помощью соответствующего кода. – Тестирование и доработка: после создания симуляции проведем тестирование ее работы и доработаем по необходимости. Мы можем изменять параметры симуляции и проверять их воздействие на результаты, а также улучшать визуализацию симуляции. – Обратная связь и анализ: соберем обратную связь от пользователей и проанализируем результаты тестирования симуляции. Будем учитывать комментарии и предложения для дальнейшего улучшения симуляции. Заключение: путем изучения, разработки и тестирования можно создать симуляцию с использованием модуля turtle, учитывая при этом основные законы физики.

Т.45. **Модуль turtle: обработка событий «мыши» для управления анимацией.**

1	Найдите информацию о том, как добавить музыкальное сопровождение к анимации, созданной с помощью модуля turtle
2	Объясните учащимся насколько разнообразны области практического применения знаний в области программирования. Предложите провести самостоятельное исследование для расширения возможностей работы с модулем turtle. Пример варианта решения. Для добавления музыкального сопровождения к анимации, созданной с помощью модуля turtle, и синхронизации движения черепахи с музыкальным ритмом, можно использовать следующий подход: – Импортировать модуль turtle в программу, чтобы получить доступ к функциям черепахи. – Импортировать модуль pygame, который позволяет воспроизводить звуковые файлы. – Создать объект черепахи и настроить его параметры, такие как цвет, размер и скорость движения. – Загрузить музыкальный файл с помощью функции pygame.mixer.music.load(), указав путь к файлу. – Установить ритмический таймер с помощью функции pygame.time.set_timer(), чтобы черепаха двигалась синхронно с музыкой. – В цикле программы, который обновляется с определенной частотой, вызывать функцию pygame.mixer.music.play() для воспроизведения музыки и функции черепахи для ее движения. – При необходимости можно настроить другие параметры музыки, такие как громкость или повторение.

	Приведенный подход позволит добавить музыкальное сопровождение к анимации черепахи и синхронизировать ее движение с музыкальным ритмом.

Т.46. **Модуль turtle: обработка событий клавиатуры для управления анимацией.**

1	Найдите и изучите различные библиотеки или пакеты, которые расширяют возможности модуля turtle. Предложите свою собственную идею для библиотеки или пакета, который можно было бы создать для расширения возможностей модуля turtle.
2	Обсудите с учащимися какие преимущества дает возможность расширения возможностей языка программирования за счет использования различных библиотек и пакетов. Как пример можно предложить следующий вариант ответа: При изучении различных библиотек или пакетов можно обратить внимание на библиотеку "turtle" в Python и ее возможности для создания анимаций и графики. Собственная идея для библиотеки или пакета, который можно было бы создать для расширения возможностей модуля turtle, включает в себя разработку модуля "turtleextras", который предоставлял бы дополнительные функции и возможности для создания интерактивной и анимационной графики. Этот модуль мог бы включать в себя: 1. Дополнительные графические примитивы, такие как эллипсы, многоугольники, спирали и другие. 2. Возможность работы с аудио и видео файлами для создания мультимедийных анимаций. 3. Возможности для работы с 3D-графикой и создания трехмерных объектов. 4. Интеграцию с сенсорными устройствами или виртуальной реальностью для создания интерактивных анимаций. 5. Поддержку различных алгоритмов и эффектов для улучшения визуального восприятия анимаций. Такой модуль позволил бы разработчикам и художникам создавать более сложные и интересные анимации, расширяя возможности модуля turtle и делая процесс создания графики более увлекательным.