Департамент образования и науки города Москвы

Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Средняя общеобразовательная школа

СОГЛАСОВАНО
Председатель экспертного совета
по дополнительному образованию
ГАОУ ВО МГПУ
_____ /Н.П. Ходакова/
Протокол № 24 от 23 июня 2026 г.

УТВЕРЖДАЮ
И.о. ректора
ГАОУ ВО МГПУ
_____ /Е.Н. Геворкян/
«23» июня 2026 г.

Дополнительная общеразвивающая программа

«Клуб будущих чемпионов (программирование)»

(66 часов)

Уровень программы – ознакомительный
Направленность программы – техническая

Автор:
Чеснокова А.В.

Москва, 2026

Раздел 1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Олимпиадное программирование – особый вид деятельности, увлекательный и азартный. Олимпиады по программированию бывают индивидуальными и командными, бывают официальными, дающими льготы при поступлении в высшее учебное заведение. Главное отличие олимпиадного программирования от обычного – необходимость быстро анализировать многословные условия задач, выбирать из обширного набора решаемые, добиваться верного результата любыми средствами.

Для достижения успеха прежде всего необходимо умение читать задачу, увидеть в её описании путь к решению. Затем – опыт программирования, уверенное владение синтаксисом языка и инструментами отладки. Далее – знание стандартных приёмов решения задач, классических алгоритмов. И наконец, воля, внимание, стрессоустойчивость, умение работать и в команде, и самостоятельно.

1.1. Актуальность программы

Одним из популярных языков программирования является язык программирования Python. Он позволяет решать задачи на всех уровнях сложности – от пропедевтического до профессионального. Python – это современный язык программирования и средство для моделирования различных объектов. Язык этот по синтаксису предельно прост, в то же время он обладает мощными современными средствами, формирующими культуру мышления и позволяющими создавать программы лаконичные, прозрачные по структуре и эффективности. Поэтому целесообразно использовать именно этот язык при изучении основ программирования и решения олимпиадных задач на предпрофессиональном уровне.

Цель:

Сформировать знания, навыки и опыт, необходимые для участия в олимпиадах по спортивному программированию (как индивидуальных, так и командных), обеспечить организационную поддержку и руководство при участии в олимпиадах, помочь в выборе стратегии дальнейшего самосовершенствования.

Задачи:

Обучающие:

– сформировать основные алгоритмические понятия: исполнитель, алгоритм, команда, виды алгоритмов, формы представления алгоритмов;

– сформировать базовые представления о записи программ на языке программирования Python и познакомить с основными управляющими конструкциями этого языка на примере графического исполнителя «Черепаха»;

– сформировать навыки построения изображений методами черепашьей графики;

– создавать условия для развития логического и алгоритмического мышления;

– сформировать понятие процедуры, процедуры с одним и несколькими параметрами, научить оформлять и применять процедуры;

– развивать у учащихся устойчивый интерес к предмету, элементы информационной культуры;

– создавать условия для самовыражения ребёнка, развивать потребность в творческой деятельности;

– показать взаимосвязь программирования с другими сферами жизни человека.

Развивающие:

– научить установлению причинно-следственных связей;

– научить придумывать и разрабатывать идеи;

– развить алгоритмическое мышление;

– обучить основам проектной деятельности;

– выработать у учащихся навыки самостоятельной исследовательской деятельности.

Воспитательные:

– развивать коммуникативные способности и навыки межличностного общения;

– формировать навыки сотрудничества при работе в коллективе и в команде;

– формировать основы безопасности собственной жизнедеятельности и окружающих людей, необходимых при конструировании робототехнических моделей.

– воспитывать ценностное отношение к собственному труду, труду других людей и его результатам;

– научить представлять свой проект перед аудиторией.

Планируемые результаты обучения

Личностные результаты: формирование ответственного отношения к учению, готовности и способности обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию; выбору и построению дальнейшей индивидуальной траектории образования на базе ориентировки в мире профессий и профессиональных предпочтений; умения применять полученные знания на других учебных предметах и в окружающей действительности; формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками.

Метапредметные результаты: умение самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учёбе, развивать мотивы и интересы своей познавательной деятельности; умение самостоятельно планировать пути достижения целей в разработке программ, решении практических задач, работе над творческим проектом; осознанно выбирать наиболее эффективные способы решения задач; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности в процессе достижения результата; умение создавать, применять и преобразовывать знаки и символы, модели и схемы для решения учебных и познавательных задач; формирование и развитие компетентности в области использования информационно-коммуникационных технологий.

Предметные результаты: умение работать с учебным математическим текстом (анализировать, извлекать необходимую информацию при решении практических задач); сформированные представления о системе команд исполнителя, основных алгоритмических конструкциях, понятии процедуры; управление исполнителем в среде языка программирования Python; овладение приёмами решения задач на

циклические алгоритмы, вложенные циклы; моделирование графических объектов через комбинацию различных примитивов простейших фигур на языке программирования; умения моделировать реальные ситуации на языке программирования Python.

В результате обучающиеся будут

знать: этапы решения задач на компьютере, типы данных, базовые конструкции изучаемых языков программирования, инструментальные средства Python

уметь: работать в среде программирования, реализовывать построенные алгоритмы в виде программ на конкретном языке программирования, настраивать рабочую среду Python

владеть: основами алгоритмизации, возможностями современных языков и технологи программирования

Категория обучающихся: 5-8 классы (с разделением на возрастные группы: 5-6 классы и 7-8 классы)

Форма обучения: очная

Режим занятий: 2 часа в неделю

Трудоемкость программы: 66 часов

Раздел 2. СОДЕРЖАНИЕ ПРОГРАММЫ

2.1. Учебный план

5-6 классы

№ п/п	Наименование разделов (модулей) и тем	Аудиторные учебные занятия, учебные работы			Внеаудиторная работа	Формы контроля	Всего часов
		Всего ауд.часов (ак.час)	Теоретические занятия	Практические занятия	Самостоят. работа		
1	**Модуль 1. Основные команды языка Python**	**11**	**5**	**6**			**11**
1.1	Основные команды языка Python	1	1				1
1.2	Команды перемещения черепашки	2	1	1			2
1.3	Команда повторения в Python	1		1			1
1.4	Виды углов и их построение в Python	2	1	1			2
1.5	Построение многоугольников	3	1	2			3
1.6	Построение окружностей и дуг	2	1	1			2
2	**Модуль 2. Процедуры**	**13**	**4**	**9**			**13**
2.1	Понятие процедуры	3	1	2			3
2.2	Вызов процедуры в процедуре	3	1	2			3
2.3	Процедуры с параметром	3	1	2			3
2.4	Процедуры с несколькими параметрами	4	1	3			4
3	**Модуль 3. Арифметические операции**	**11**	**5**	**6**			**11**
3.1	Команда присваивания	3	1	2			4
3.2	Арифметические операции.	4	2	2			4
3.3	Операторы вывода на экран	4	2	2			4
4	**Модуль 4. Условные и циклические конструкции**	**31**	**10**	**21**			**31**

4.1	Команда ветвления	7	2	5			7
4.2	Цикл со счётчиком	6	2	4			6
4.3	Цикл с условием	7	2	5			7
4.4	Рекурсивные процедуры	6	2	4			6
4.5	Фрактальные изображения	5	2	3			5
	Итого	**66**	**24**	**42**			**66**

7-8 классы

№ п/п	Наименование разделов (модулей) и тем	Аудиторные учебные занятия, учебные работы			Внеаудиторная работа	Формы контроля	Всего часов
		Всего ауд.часов (ак.час)	Теоретические занятия	Практические занятия	Самостоят. работа		
1	**Модуль 1. Основы программирования и Python**	**10**	**5**	**5**			**10**
1.1	Понятие алгоритма.	1	1				1
1.2	Язык программирования Python	2	1	1			2
1.3	Классификация данных	2	1	1			2
1.4	Запись выражений на языке Ptyhon	2	1	1			2
1.5	Операторы ввода, вывода и присваивания	3	1	2			3
2	**Модуль 2. Базовые алгоритмические конструкции**	**11**	**4**	**7**			**11**
2.1	Программирование линейных алгоритмов	3	1	2			3
2.2	Программирование разветвляющихся алгоритмов	3	1	2			3
2.3	Программирование циклических алгоритмов	5	2	3			5
3	**Модуль 3. Структурное программирование**	**23**	**7**	**16**			**23**
3.1	Процедуры	2	1	1			2

3.2	Одномерные массивы	7	2	5			7
3.3	Методы поиска в одномерном массиве	3	1	2			3
3.4	Методы сортировки в одномерном массиве	7	2	5			7
3.5	Двумерные массивы	4	1	3			4
4	**Модуль 4.** Работа со строками и структурами данных	**22**	**11**	**11**			**22**
4.1	Строки и символы	4	2	2			4
4.2	Структуры данных. Стек	6	3	3			6
4.3	Структуры данных. Очередь	6	3	3			6
4.4	Структуры данных. Графы	6	3	3			6
	Итого	**66**	**27**	**39**			**66**

2.2. Рабочая программа
5–6 классы

№ п/п	**Виды учебных занятий, учебных работ, объем в часах**	**Содержание**
Модуль 1. **Основные команды языка Python**		
Тема 1.1 Основные команды языка Python	Теоретическое занятие, 1 час	Основные понятия: алгоритм, исполнитель, команда, программа, система команд исполнителя, программирование
Тема 1.2 Команды перемещения черепашки	Теоретическое занятие, 1 час	Изучение основных команд перемещения черепашки, такие как forward(), backward(), right(), left(), параметры, которые можно использовать для управления движением. Участники узнают о системе координат и о том, как изменять угол поворота черепашки. В ходе занятия будет проведен анализ примеров кода, чтобы закрепить полученные знания
	Практическое занятие, 1 час	Создание простой программы, в которой черепашка будет рисовать фигуры (например, квадрат, треугольник или звезду) с использованием команд перемещения
Тема 1.3 Команда повторения в Python	Практическое занятие, 1 час	На теоретическом занятии участники познакомятся с концепцией команд повторения в Python, таких как for и while. Обучающиеся изучат как использовать эти команды для выполнения повторяющихся действий в программе, синтаксис и основные принципы работы с циклами. Участники узнают о различных вариантах применения циклов, таких как итерация по спискам и

		использование счетчиков. Также будет рассмотрена тема управления циклом с помощью операторов break и continue. В ходе занятия будут приведены примеры кода для лучшего понимания материала
Тема 1.4 Виды углов и их построение в Python	Теоретическое занятие, 1 час	Знакомство с основными видами углов (острые, прямые, тупые и развернутые) и их свойствами. Мы обсудим, как углы измеряются в градусах и радианах, а также рассмотрим методы построения углов с помощью графических библиотек в Python, таких как turtle. Участники узнают о синтаксисе и функциях, необходимых для создания графических изображений углов, и научатся применять математические формулы для вычисления величин углов.
	Практическое занятие, 1 час	Создание программы для построения различных видов углов с использованием библиотеки turtle. Задачи будут включать создание графических изображений острых, прямых и тупых углов, а также комбинирование углов для формирования сложных фигур. Участники будут работать над проектами в группах, исследуя возможности настройки цветов, толщины линий и других параметров графики
Тема 1.5 Построение многоугольников	Теоретическое занятие, 1 час	Ознакомление с основами построения многоугольников в Python. Мы обсудим, что такое многоугольники, их свойства и классификацию (выпуклые и вогнутые). Участники узнают о координатной системе и принципах работы с графическими библиотеками, такими как turtle. Мы рассмотрим алгоритмы для вычисления координат вершин многоугольников и правила их построения. Занятие также включает изучение функций для рисования многоугольников, таких как begin_fill(), end_fill() и другие, а также применение циклов для автоматизации процесса
	Практическое занятие, 2 часа	Создание программы для построения различных многоугольников с использованием библиотеки turtle. Задачи будут включать рисование простых многоугольников (треугольников, квадратов, пятиугольников) и более сложных фигур (шестигранников и восьмиугольников)
Тема 1.6 Построение окружностей и дуг	Теоретическое занятие, 1 час	Ознакомление с основами построения окружностей и дуг в Python с использованием графической библиотеки turtle. Мы обсудим математические принципы, лежащие в основе окружностей, такие как радиус, диаметр и длина окружности. Участники узнают о координатной системе и о том, как использовать функции библиотеки turtle, такие как circle(), для рисования окружностей и дуг. Также будут рассмотрены параметры функции, позволяющие управлять радиусом и углом дуги, а также методы изменения цвета и стиля линий.

	Практическое занятие, 1 часа	Создание программы для рисования окружностей и дуг. Задачи будут включать построение различных окружностей с заданными радиусами, создание цветных дуг, а также комбинирование окружностей для создания сложных узоров и фигур. Участники смогут экспериментировать с параметрами функции circle(), изменяя цвет, толщину линий и заполняя фигуры
Модуль 2. Процедуры		
Тема 2.1 Понятие процедуры	Теоретическое занятие, 1 час	Знакомство с понятием процедуры в Python, ее определением и назначением. Мы обсудим, что такое функции и процедуры, в чем их отличия, а также рассмотрим синтаксис определения и вызова процедур. Участники узнают о параметрах и аргументах, передаваемых в процедуры, а также о возвращаемых значениях. Будет проведен анализ примеров использования процедур для организации кода, повышения его читаемости и повторного использования
	Практическое занятие, 2 часа	Составление собственных процедур в Python. Задачи будут включать написание простых процедур для выполнения различных операций, таких как вычисление факториала, нахождение суммы чисел в списке или создание функций для работы с текстом. Участники смогут экспериментировать с параметрами и возвращаемыми значениями, а также научатся вызывать свои процедуры в различных частях программы
Тема 2.2 Вызов процедуры в процедуре	Теоретическое занятие, 1 час	Знакомство с концепцией декомпозиции задачи на подзадачи. Объясняется, как использование нескольких небольших функций вместо одной большой улучшает читаемость и упрощает отладку кода. Рассматривается синтаксис вызова одной функции внутри тела другой на примерах. Особое внимание уделяется порядку выполнения инструкций (потоку управления), механизму передачи аргументов во вложенный вызов и возврату значений. Учащиеся изучают, как результат работы внутренней функции становится данными для внешней, формируя цепочку вычислений
	Практическое занятие, 2 часа	Практическая работа направлена на закрепление навыка создания многоуровневых программных модулей. Учащимся предлагается решить комплексную задачу (например, расчёт стоимости заказа с учётом скидок и доставки или построение сложной геометрической фигуры). Задача разбивается на несколько этапов, каждый из которых реализуется в виде отдельной функции. Главная цель — спроектировать архитектуру решения так, чтобы основная управляющая функция вызывала вспомогательные, а те, в свою очередь, при необходимости могли вызывать другие. В ходе работы ученики учатся тестировать отдельные модули, отслеживать порядок их выполнения и анализировать итоговый результат работы всей системы
	Теоретическое занятие, 1 час	Знакомство с понятием параметра как способа передачи данных внутрь функции. Объясняется разница между

Тема 2.3 Процедуры с параметром		формальными параметрами (в определении функции) и фактическими аргументами (при вызове). Рассматриваются позиционные параметры, передача значений по значению для неизменяемых типов и важность порядка аргументов. Вводится понятие области видимости переменных и ключевое слово return для возврата результата из процедуры. На примерах показывается, как использование параметров делает код гибким, переиспользуемым и избавляет от дублирования
	Практическое занятие, 2 часа	Практическая работа на закрепление навыка проектирования и использования функций с параметрами. Учащимся предлагается решить серию учебных задач, требующих создания обобщённых решений. Например, написание функций для вычисления площади прямоугольника (с параметрами длины и ширины), форматированного вывода текста или проверки числа на чётность
Тема 2.4 Процедуры с несколькими параметрами	Теоретическое занятие, 1 час	Изучение синтаксиса создания функций, принимающих более одного аргумента. Рассматриваются различные способы передачи данных: - Позиционные аргументы: передача значений в строгом соответствии с порядком параметров в определении функции. - Именованные аргументы (*keyword arguments*): передача значений путём указания имени параметра, что позволяет менять их порядок при вызове. - Значения по умолчанию: установка стандартных значений для параметров, которые позволяют вызывать функцию без их явной передачи.
	Практическое занятие, 3 часа	Организация процедуры с несколькими параметрами
Модуль 3. Арифметические операции		
Тема 3.1 Команда присваивания	Теоретическое занятие, 1 час	Знакомство с фундаментальным механизмом языка программирования Python — командой присваивания. В доступной форме объясняется, как компьютер хранит данные в памяти и как переменные служат именами для этих данных. Рассматривается синтаксис оператора =, правила создания имён переменных (идентификаторов), а также ключевые различия между операцией присваивания («запомнить значение») и математическим равенством. Особое внимание уделяется динамической типизации в Python и тому, как одна и та же переменная может хранить значения разных типов
	Практическое занятие, 2 часа	Организация программ с переменными
Тема 3.2 Арифметические операции	Теоретическое занятие, 2 часа	Рассматривается стандартный набор операций: сложение, вычитание, умножение, деление (обычное и целочисленное), а также возведение в степень и нахождение остатка от деления. Особое внимание уделяется приоритету выполнения операций и использованию скобок для его изменения

	Практическое занятие, 2 часа	Организация программ с арифметическими операциями
Тема 3.3 Операторы вывода на экран	Теоретическое занятие, 2 часа	Знакомство с главной функцией для вывода информации на экран — `print()`. Рассматривается её базовый синтаксис, а также ключевые параметры (`sep` и `end`) для управления форматированием выводимых данных. Объясняется, как функция автоматически преобразует различные типы данных (строки, числа) в текстовый вид и выводит их в консоль
	Практическое занятие, 2 часа	Организация программ с операторами вывода на экран
Модуль 4 Условные и циклические конструкции		
Тема 4.1 Команда ветвления	Теоретическое занятие, 2 часа	Знакомство с логикой ветвления, которая позволяет программе принимать решения. Рассматривается синтаксис конструкции if, а также её расширенные варианты — elif (иначе если) и else. Объясняется роль условий, как составных, так и вложенных, и приводятся жизненные примеры, где программа должна выполнять разные действия в зависимости от входных данных
	Практическое занятие, 5 часов	Организация программ с командой ветвления
Тема 4.2 Цикл со счётчиком	Теоретическое занятие, 2 часа	Знакомство с циклом for — инструментом для выполнения блока кода заданное количество раз. Рассматривается синтаксис цикла, работа функции range() для генерации последовательностей чисел (включая параметры *start*, *stop* и *step*) и ключевое слово break для досрочного выхода из цикла. Объясняется, как использовать переменную-счётчик внутри тела цикла
	Практическое занятие, 4 часа	Организация программ с циклом со счётчиком
Тема 4.3 Цикл с условием	Теоретическое занятие, 3 часа	Знакомство с циклом while, который выполняет блок кода до тех пор, пока заданное условие остаётся истинным. Рассматривается синтаксис конструкции, важность изменения переменных внутри цикла для предотвращения зацикливания и использование операторов break (для досрочного выхода) и continue (для пропуска текущей итерации). Проводится сравнение циклов for и while и даются примеры задач, где применение цикла с условием наиболее уместно
	Практическое занятие, 5 часов	Организация программ с циклом с условием
Тема 4.4 Рекурсивные процедуры	Теоретическое занятие, 2 часа	Знакомство с концепцией рекурсии — процессом, когда функция вызывает саму себя для решения задачи. Рассматриваются два ключевых компонента рекурсивной функции: базовый случай (условие остановки) и шаг рекурсии (переход к более простой задаче). Объясняется принцип работы системного стека вызовов и опасность бесконечной рекурсии, приводящей к переполнению стека

	Практическое занятие, 4 часа	Организация рекурсивных процедур
Тема 4.5 Фрактальные изображения	Теоретическое занятие, 2 часа	Знакомство с понятием фрактала — геометрической фигуры, обладающей свойством самоподобия (целое имеет ту же форму, что и его части). Рассматриваются классические примеры: треугольник Серпинского, множество Кантора и кривая Коха. Объясняется математическая основа их построения и рекурсивная природа алгоритмов, которые используются для их генерации
	Практическое занятие, 3 часа	Построение фрактальные изображений и фракталов

7-8 классы

№ п/п	**Виды учебных занятий, учебных работ, объем в часах**	**Содержание**
Модуль 1. Основы программирования и Python		
Тема 1.1 Понятие алгоритма	Теоретическое занятие, 1 час	Знакомство с основами алгоритмизации и определением алгоритма как последовательности действий, направленных на решение задачи
Тема 1.2 Язык программирования Python	Теоретическое занятие, 1 час	Рассматриваются ключевые особенности языка: его интерпретируемость, динамическая типизация и кроссплатформенность. Основное внимание уделяется синтаксису — правилам написания кода. Учащиеся узнают о понятиях «оператор», «идентификатор», «переменная» и изучают правила их именования. Демонстрируется среда разработки
	Практическое занятие, 1 час	Занятие направлено на закрепление теоретических знаний на практике. Учащиеся самостоятельно настраивают рабочее окружение и пишут свой первый код
Тема 1.3 Классификация данных	Теоретическое занятие, 1 час	Учащиеся знакомятся с типами задач классификации (бинарная, многоклассовая), основными метриками оценки качества моделей (точность, полнота, F1-мера, матрица ошибок) и принципами работы популярных алгоритмов (логистическая регрессия, деревья решений, метод k-ближайших соседей). Особое внимание уделяется этапам подготовки данных: кодированию категориальных признаков, масштабированию и разбиению выборок на обучающую и тестовую. В качестве основного инструмента используется библиотека *scikit-learn*
	Практическое занятие, 1 час	В ходе практической работы участники закрепляют полученные знания, решая задачу классификации на реальном или синтетическом датасете. Под руководством преподавателя они выполняют полный цикл работы: загрузка и первичный анализ данных, предобработка (обработка пропусков, кодирование

		признаков), разделение выборки и обучение модели. Практическая часть включает настройку гиперпараметров, оценку качества модели на тестовых данных с помощью метрик и визуализацию результатов
Тема 1.4 Запись выражений на языке Ptyhon	Теоретическое занятие, 1 час	Учащиеся изучают синтаксис арифметических, логических и строковых операций, а также правила приоритета и ассоциативности операторов. Особое внимание уделяется особенностям языка: динамической типизации, строгим правилам отступов (пробелов) для определения блоков кода, записи многострочных выражений с помощью символа обратной косой черты (\\\) и круглых скобок. Также изучаются работа с переменными, литералами различных типов и основы функционального вызова
	Практическое занятие, 1 час	В ходе практической работы участники закрепляют теоретические знания, выполняя задания по написанию, анализу и исправлению выражений. Практика включает перевод математических формул на язык Python, построение сложных логических условий и конкатенацию строк. Учащиеся учатся использовать отладчик для пошагового выполнения кода, анализировать ошибки (синтаксические и логические) и применять инструменты *PEP 8* для проверки стиля
Тема 1.5 Операторы ввода, вывода и присваивания	Теоретическое занятие, 1 час	На занятии рассматриваются ключевые механизмы взаимодействия программы с данными и пользователем. Учащиеся изучают оператор присваивания (=) и его роль в работе с переменными, включая динамическую типизацию. Подробно разбираются функции вывода информации на экран (print()) с использованием форматирования строк (f-строки, метод .format()) и функции ввода данных от пользователя (input()). Особое внимание уделяется преобразованию типов данных (приведение строк к числам с помощью int() и float()) для выполнения вычислений
	Практическое занятие, 2 часа	В ходе практической работы участники закрепляют знания, создавая простые консольные приложения. Под руководством преподавателя они реализуют программы, которые запрашивают у пользователя данные (имя, числа для расчёта), выполняют базовые арифметические операции и выводят отформатированный результат. Практика включает решение задач на обмен значений переменных, построение диалогов с пользователем и обработку введённых данных, что формирует базовые навыки для создания интерактивного программного кода
Модуль 2. Базовые алгоритмические конструкции		
Тема 2.1 Программирование линейных алгоритмов	Теоретическое занятие, 1 час	На занятии рассматривается понятие алгоритма и его линейной структуры, где все действия выполняются строго последовательно, одно за другим. Учащиеся изучают базовый синтаксис для построения таких программ: операторы присваивания, ввода и вывода. Особое внимание уделяется этапам решения задачи: от математической постановки и составления блок-схемы

		до записи готового кода. Разбираются правила именования переменных, форматирования вывода и типичные ошибки при написании последовательных инструкций
	Практическое занятие, 2 часа	В ходе практической работы участники закрепляют знания, реализуя программы для решения прикладных задач, описываемых линейным алгоритмом. Под руководством преподавателя они составляют математическую модель, проектируют алгоритм в виде блок-схемы и переводят его в код на Python. Практика включает написание программ для вычисления физических величин, финансовых расчётов (например, скидки или налогов) и решения математических выражений, что формирует навык формализации задачи и создания готового программного продукта
Тема 2.2 Программирование разветвляющихся алгоритмов	Теоретическое занятие, 1 час	На занятии изучаются основы создания программ, логика которых зависит от входных данных. Учащиеся знакомятся с понятием разветвляющегося алгоритма и его представлением в виде блок-схем. Рассматривается синтаксис условных операторов Python: простая конструкция `if`, альтернативный выбор `if-else` и множественное ветвление с помощью `elif`. Особое внимание уделяется составлению и вычислению логических выражений с использованием операторов сравнения (`>`, `<`, `==`) и логических связок (`and`, `or`, `not`)
	Практическое занятие, 2 часа	В ходе практической работы участники закрепляют знания, создавая интерактивные программы. Под руководством преподавателя они решают прикладные задачи, требующие проверки условий: валидация введённых данных, определение категории по заданным критериям, а также реализация простого консольного меню с выбором действий. Практика включает написание вложенных условных операторов для обработки сложных сценариев и отладку кода для обеспечения корректной работы программы во всех возможных ветвях
Тема 2.3 Программирование циклических алгоритмов	Теоретическое занятие, 1 час	Учащиеся знакомятся с понятием циклического алгоритма, его блок-схемным представлением и жизненным циклом (инициализация, проверка условия, выполнение тела, обновление счётчика). Рассматриваются два основных типа циклов в Python: 1. Цикл с параметром for — для перебора элементов последовательности (строк, списков, словарей) и работы с функцией range(). 2. Цикл с предусловием while — для выполнения действий, пока заданное условие остаётся истинным. Особое внимание уделяется операторам управления исполнением цикла: break (для досрочного прерывания) и continue (для перехода к следующей итерации), а также концепции вложенных циклов. Вводится понятие

		бесконечного цикла и рассматриваются типичные ошибки, приводящие к его возникновению
	Практическое занятие, 3 часа	Цикл for: обработка данных и числовые ряды Учащиеся осваивают цикл for для работы с последовательностями (списками, строками) и функцией range(). Практические задачи включают: вычисление сумм, произведений и средних арифметических значений; поиск элементов, удовлетворяющих условию (например, нахождение максимального/минимального числа); формирование новых списков на основе существующих. Особое внимание уделяется форматированному выводу результатов
		Цикл while: алгоритмы с неизвестным числом шагов На занятии изучается применение цикла while для решения задач, где количество итераций зависит от выполнения условия. Учащиеся реализуют валидацию пользовательского ввода (программы, которые запрашивают данные до тех пор, пока не получат корректное значение). Решаются задачи на приближённые вычисления, например, нахождение суммы числового ряда с заданной точностью
		Вложенные циклы и управление исполнением Занятие посвящено решению более сложных задач с использованием вложенных циклов для генерации таблиц и обработки многомерных данных (например, вывод таблицы умножения). Учащиеся на практике осваивают операторы управления циклом break (для досрочного выхода) и continue (для пропуска итерации), а также учатся избегать и отлаживать бесконечные циклы
Модуль 3. Структурное программирование		
Тема 3.1 Процедуры	Теоретическое занятие, 1 час	Знакомство с понятием функции как именованного блока кода для решения конкретной задачи. Объяснить синтаксис создания (оператор def), правила передачи аргументов и механизм возврата значений (return)
	Практическое занятие, 1 час	Закрепление теоретических знаний на практике через написание, отладку и тестирование собственных функций для решения прикладных задач
Тема 3.2 Одномерные массивы	Теоретическое занятие, 2 часа	Понятие массива: зачем нужны массивы, их преимущества перед использованием отдельных переменных. Создание и инициализация: создание пустых и заполненных списков. Литеральный синтаксис [] и функция list(). Индексация и срезы (*slicing*): Доступ к элементам по индексу (положительному и отрицательному). Извлечение подпоследовательностей с помощью срезов. Основные методы и операции: Изменение списков (append(), insert(), remove(), pop()), определение длины (len()), конкатенация и повторение. * Итерация по массиву: Цикл for для последовательного доступа ко всем элементам

	Практическое занятие, 5 часов	Закрепление навыков работы со списками через решение прикладных задач, требующих обработки наборов однотипных данных. Практические задания: 1. Базовые операции: Написание функций для нахождения суммы, среднего арифметического, максимального и минимального значений в числовом списке. 2. Фильтрация данных: Создание нового списка на основе условий (например, оставить только чётные числа или строки определённой длины). 3. Преобразование данных: Применение некоторой операции к каждому элементу списка (например, возведение всех чисел в квадрат). 4. Решение комплексной задачи: Реализация простой программы, например, ведения списка дел (*To-Do list*) или анализа температурных показателей за неделю, где требуется комбинировать несколько операций над массивом
Тема 3.3 Методы поиска в одномерном массиве	Теоретическое занятие, 1 час	Изучение и классификация основных алгоритмов поиска элемента в неупорядоченном и упорядоченном одномерных массивах (списках), а также проанализировать их эффективность. Линейный поиск, бинарный поиск. Сравнение методов, выбор подходящего алгоритма в зависимости от исходных дан
	Практическое занятие, 2 часа	Закрепление теоретических знаний путём реализации линейного и бинарного поисков на языке Python и провести их сравнительный анализ на практике. 1. Реализация линейного поиска: Написание функции для нахождения индекса первого вхождения заданного значения в произвольном списке. 2. Реализация бинарного поиска: Создание функции для поиска элемента в уже отсортированном списке с использованием цикла или рекурсии. 3. Сравнительный анализ: Проведение экспериментов для замера времени выполнения обоих алгоритмов на больших объёмах данных (например, поиск элемента в списке из миллиона чисел) для наглядной демонстрации разницы в их производительности. 4. Решение прикладной задачи: Разработка программы, которая использует один из изученных методов для решения практической задачи (например, проверка наличия слова в словаре или товара в каталоге)
Тема 3.4 Методы сортировки в одномерном массиве	Теоретическое занятие, 2 часа	Понятие сортировки: зачем нужна сортировка данных. Оценка эффективности алгоритмов через временную (*О-нотация*) и пространственную сложность. Простые квадратичные алгоритмы: *Сортировка пузырьком (Bubble Sort):* принцип попарного сравнения и «всплытия» максимальных элементов.

		Сортировка выбором (Selection Sort): поиск минимального элемента и его постановка на правильную позицию. *Сортировка вставками (Insertion Sort):* построение отсортированной последовательности по одному элементу. Эффективные алгоритмы «разделяй и властвуй»: *Быстрая сортировка (Quicksort):* выбор опорного элемента и разделение массива на части. *Сортировка слиянием (Merge Sort):* рекурсивное деление массива пополам и последующее слияние отсортированных частей
	Практическое занятие, 5 часов	1. Реализация базовых алгоритмов: Написание функций для сортировки пузырьком и сортировки вставками. Визуализация процесса сортировки на небольших примерах. 2. Реализация эффективной сортировки: Создание функции для быстрой сортировки (с использованием рекурсии или дополнительной памяти). 3. Экспериментальное сравнение: Замер времени выполнения реализованных алгоритмов на массивах разной длины и с разным порядком исходных данных (случайный, обратный). Построение графиков зависимости времени от размера входных данных. 4. Использование встроенных средств: Демонстрация работы стандартного метода list.sort() и обсуждение причин его высокой производительности
Тема 3.5 Двумерные массивы	Теоретическое занятие, 1 час	Понятие двумерного массива: Определение, аналогия с таблицей или шахматной доской. Обоснование необходимости использования вложенных списков (list of lists) в Python. Создание и инициализация: Способы создания пустых и заполненных матриц фиксированного размера. Индексация элементов: Доступ к элементу по паре индексов [строка][столбец]. Особенности изменения отдельных ячеек. Обход матрицы: Использование вложенных циклов for для последовательного перебора всех элементов, а также для обработки по строкам или по столбцам. Практические примеры применения: Изображения (пиксели), таблицы данных, игровые поля (*например, поле для "крестиков-ноликов"*)
	Практическое занятие, 3 часа	1. Базовые операции: Создание квадратной матрицы N×N, её заполнение числами от пользователя или случайным образом, вывод на экран в виде таблицы. 2. Вычисления по строкам и столбцам: Написание функций для нахождения сумм элементов в каждой строке и в каждом столбце. 3. Работа с диагоналями: Реализация функций для вычисления суммы элементов на главной и побочной диагоналях матрицы. 4. Решение прикладной задачи: Разработка программы для проверки, является ли введённая пользователем

		матрица *«магическим квадратом»* (то есть равны ли между собой суммы всех строк, столбцов и обеих диагоналей)
Модуль 4. Работа со строками и структурами данных		
Тема 4.1 Строки и символы	Теоретическое занятие, 2 часа	Создание строк: Использование одинарных, двойных и тройных кавычек. Понятие экранирования символов (backslash). * Неизменяемость (*immutability*): Объяснение того, что строка после создания не может быть изменена; любые операции создают новую строку. * Индексация и срезы: Доступ к отдельному символу по индексу. Извлечение подстроки с помощью среза ([start:stop:step]). * Базовые методы: Обзор наиболее часто используемых методов для преобразования регистра (upper(), lower()), поиска (find()), замены (replace()) и проверки свойств (isdigit(), isalpha()). * Форматирование строк: Введение в f-строки как современный способ вставки значений переменных в текст
	Практическое занятие, 2 часа	1. Работа со срезами: Написание функций для решения задач: вывод строки задом наперёд, получение каждого второго символа, вырезка слова из предложения по заданным индексам. 2. Анализ данных: Создание программы, которая подсчитывает количество гласных/согласных букв во введённой строке или проверяет, является ли фраза палиндромом. 3. Парсинг строк: Разбор строки, содержащей данные в определённом формате (например, *«ФИО»*, *«город, страна»* или *CSV-запись*), на составные части с использованием методов split() и strip(). 4. Комплексная задача: Реализация простого шифра (например, шифра Цезаря), где требуется пройтись циклом по каждому символу строки, изменить его код и собрать новую зашифрованную строку
Тема 4.2 Структуры данных. Стек	Теоретическое занятие, 3 часа	Определение стека: Понятие структуры данных «стек» и принципа *LIFO (Last-In, First-Out — последним пришёл, первым ушёл)*. Базовые операции: push (добавление элемента на вершину) и pop (удаление элемента с вершины). Вспомогательные операции: peek (просмотр верхнего элемента без удаления) и проверка на пустоту (is_empty). Реализация в Python: Два основных способа реализации стека: 1. На основе списка (*list*): использование методов append() для push и pop(). 2. На основе модуля collections.deque: для создания более эффективной реализации, особенно при большом количестве элементов. Практические примеры из жизни: стопка тарелок, история посещений страниц в браузере («назад»), механизм вызова функций (стек вызовов)

	Практическое занятие, 3 часа	1. Создание класса Stack: Реализация собственного класса Stack с методами push, pop, peek и is_empty на основе стандартного списка Python. 2. Решение задачи «Баланс скобок»: Написание программы, которая проверяет правильность расстановки круглых, квадратных или фигурных скобок в математическом выражении, используя стек для хранения открывающих скобок. 3. Обход графа в глубину (DFS): Демонстрация применения стека (явного или неявного через рекурсию) для обхода узлов графа или дерева вглубь. 4. Сравнение реализаций *(для продвинутых групп)*: Проведение эксперимента по оценке производительности собственной реализации на списке и на базе deque при выполнении большого количества операций push и pop
Тема 4.3 Структуры данных. Очередь	Теоретическое занятие, 3 часа	Определение очереди: Понятие структуры данных «очередь» и принципа *FIFO (First-In, First- — первым пришёл, первым ушёл)*. Базовые операции: enqueue (добавление элемента в конец очереди) и dequeue (удаление элемента из начала очереди). Вспомогательные операции: просмотр первого элемента (peek) и проверка на пустоту (is_empty). Реализация в Python: Неэффективная реализация: использование стандартного списка (*list*) для удаления элемента из начала (pop(0)), что приводит к высокой вычислительной сложности. Эффективная реализация: использование модуля collections.deque, который обеспечивает быстрые добавления и удаления с обоих концов. Практические примеры из жизни
	Практическое занятие, 3 часа	1. Создание класса Queue: Реализация собственного класса Queue на основе collections.deque с методами enqueue, dequeue, peek и is_empty. 2. Моделирование процесса печати: Написание программы-симулятора печати. Задачи (документы) добавляются в очередь, а «принтер» поочерёдно обрабатывает их, извлекая из начала очереди. 3. Решение задачи «Горячая линия»: Создание симуляции колл-центра, где входящие звонки помещаются в очередь и распределяются между операторами в порядке поступления. 4. Сравнительный анализ *(для продвинутых групп)*: Проведение эксперимента, наглядно показывающего разницу в производительности между реализацией очереди на списке и на базе deque при большом количестве операций
Тема 4.4 Структуры данных. Графы	Теоретическое занятие, 3 часа	Определение графа, вершины (узлы) и рёбра (связи). Типы графов: ориентированные и неориентированные, взвешенные и невзвешенные. Способы представления в памяти:

		Список смежности (Adjacency List): наиболее распространённый способ, реализуемый с помощью словаря (dict), где ключ — это вершина, а значение — список соседних вершин. *Матрица смежности (Adjacency Matrix)*: двумерный массив (список списков), где ячейка [i][j] указывает на наличие ребра между вершинами i и j. Обход графа (*Traversal*): введение в фундаментальные алгоритмы поиска в глубину (*DFS*) и в ширину (*BFS*), их логика и различия. Понятие пути в графе. Практические примеры применения: социальные сети (друзья), карты дорог (города и маршруты), веб-страницы (ссылки)
	Практическое занятие, 3 часа	1. Создание класса Graph: Реализация класса неориентированного невзвешенного графа на основе списка смежности. Методы для добавления вершин и рёбер. 2. Визуализация графа: Написание простой функции для вывода текстового представления графа, чтобы наглядно видеть его структуру. 3. Реализация BFS: Создание метода для обхода графа в ширину, который выводит все достижимые из заданной вершины узлы. 4. Решение практической задачи: Модификация алгоритма BFS для нахождения кратчайшего пути (в виде последовательности вершин) между двумя заданными узлами в невзвешенном графе *(например, найти кратчайшую цепочку знакомств между двумя людьми)*

Раздел 3. ФОРМЫ АТТЕСТАЦИИ И ОЦЕНОЧНЫЕ МАТЕРИАЛЫ

Аттестация по общеразвивающей программе не предусмотрена.

Раздел 4. ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИЕ УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ

4.1. Литература

Основная:

1. Афанасьев В.В. Python для детей. Играем, чтобы программировать. – М.: Бомбора, 2026.

2. Бен Стивенсон. Python. Сборник упражнений. – М.: ДМК Пресс, 2021

3. Босова А.Ю., Аквилянов Н.А. Черепашья графика. Введение в программирование на языке Python. 5–7 классы. – М.: Просвещение, 2025

4. Бриггс, Джейсон. Python для детей. Самоучитель по программированию / Джейсон Бриггс. пер. с англ. Станислава Ломакина. [науч. Ред. Д. Абрамова]. – М. : Манн, Иванов и Фебер, 2022.

5. Васильев А.Н. Программирование на Python в примерах и задачах. – М.: Бомбора, 2020

6. Маран М.М. Программирование. Сборник задач. – Спб.: Лань, 2021

7. Мэтиз Эрик. Изучаем Python: программирование игр, визуализация данных, веб-приложения. 3-е изд. дополненное и переработанное. – Спб.: Питер, 2025.

8. Объектно-ориентированное программирование на Python с нуля. – М.: АСТ, 2026.

9. Романо Фабрицио, Крюгер Генрих. Весь Python. Самое актуальное и исчерпывающее руководство. – М.: Спринт БУК, 2026

Дополнительная:

1. Сорокина Т. Е. Методика раннего общедоступного программирования в основной образовательной программе. Сборник научных трудов XI Международной научно-практической конференции «Современные информационные технологии и ИТ-образование». 2016.

Интернет-ресурсы:

1. Язык программирования Python [Электронный ресурс]. Режим доступа: https://www.python.org/ (дата обращения: 27.05.2026).

2. Черепашья графика [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/turtle.html (дата обращения: 26.05.2026).

4.2. Материально-технические условия реализации программы

№	виды оборудования
1.	Аудитория
2.	Компьютеры
3.	Проектор
4.	Интерактивная доска (проекционный экран)
5.	Интегрированная среда разработки и обучения на языке Python

Утверждено на педагогическом совете Средней общеобразовательной школы

Протокол № 15 от «08» июня 2026 г.

Зам. директора _____ / Омарова С.О. /

Экспертное заключение
на дополнительную общеразвивающую программу

Наименование программы, кол-во часов:
«Клуб будущих чемпионов (программирование)» (66 часов)

	Критерии экспертизы и вопросы, подлежащие рассмотрению	Экспертная оценка Да/Нет	Примечание эксперта
А. Экспертиза оформления материалов программы			
1.	Наименование программы на титульном листе совпадает с наименованием в тексте	Да	
Б. Соответствие основным нормативным требованиям к структуре, объему и оформлению программы:			
1.	**Экспертиза раздела 1 «Пояснительная записка»**		
1.1.	Отражена актуальность программы	Да	
1.2.	Наименование программы соответствует ее направленности	Да	
1.3.	Сформулирована цель и задачи программы	Да	
1.4.	Представлена возрастная категория обучающихся	Да	
1.5.	Форма обучения способствует достижению планируемых результатов	Да	
1.6.	Срок обучения, режим обучения способствуют достижению планируемых результатов	Да	
1.7.	Цели, задачи, планируемые результаты, сроки и режим обучения соответствуют уровню программы (ознакомительный, базовый, углубленный)	Да	
2.	**Экспертиза раздела 2 «Содержание программы»**		
2.1.	Представлен учебный (тематический) план программы	Да	
2.2.	Имеется рабочая программа	Да	
2.3.	В программе кратко раскрыто содержание тем, указаны виды учебных занятий и учебных работ, срок их освоения	Да	
3.	**Экспертиза раздела 3 «Форма аттестации и оценочные материалы» на наличие пунктов раздела**		
3.1.	Описаны вид аттестации, формы контроля, вид оценочных материалов итоговой (при наличии) и промежуточной (при наличии) аттестации	Да	

3.2.	Описаны оценочные средства контроля (контрольно-измерительные материалы)	Да	
4.	**Экспертиза раздела 4 «Организационно-педагогические условия реализации программы»**		
4.1.	Учебно-методическое и информационное обеспечение программы соответствует всем видам учебной деятельности, предусмотренным программой	Да	
4.2.	Перечень рекомендуемой основной и дополнительной *(при наличии)* литературы содержит современные и общедоступные источники. Перечень основной литературы должен содержать источники последних 5 лет	Да	
4.3.	Перечисленные Интернет-ресурсы достоверны (*при наличии*)[1]	Да	
4.4.	Указанное материально-техническое обеспечение программы соответствует направленности и содержанию программы	Да	
4.5.	Соблюдение требований к оформлению программы	Да	

ИТОГОВОЕ ЗАКЛЮЧЕНИЕ

Программа рекомендована к реализации в образовательном процессе

Должность (с указанием места работы)
к.п.н., доцент
кафедра педагогических технологий
непрерывного образования
Институт непрерывного
образования ГАОУ ВО МГПУ

(подпись)

Пичугин С.С.
(Ф.И.О.)

[1] Могут не указываться авторами программы. В этом случае ставится прочерк

Экспертное заключение
на дополнительную общеразвивающую программу

Наименование программы, кол-во часов:
«Клуб будущих чемпионов (программирование)» (66 часов)

	Критерии экспертизы и вопросы, подлежащие рассмотрению	Экспертная оценка Да/Нет	Примечание эксперта
А. Экспертиза оформления материалов программы			
1.	Наименование программы на титульном листе совпадает с наименованием в тексте	Да	
Б. Соответствие основным нормативным требованиям к структуре, объему и оформлению программы:			
1.	**Экспертиза раздела 1 «Пояснительная записка»**		
1.1.	Отражена актуальность программы	Да	
1.2.	Наименование программы соответствует ее направленности	Да	
1.3.	Сформулирована цель и задачи программы	Да	
1.4.	Представлена возрастная категория обучающихся	Да	
1.5.	Форма обучения способствует достижению планируемых результатов	Да	
1.6.	Срок обучения, режим обучения способствуют достижению планируемых результатов	Да	
1.7.	Цели, задачи, планируемые результаты, сроки и режим обучения соответствуют уровню программы (ознакомительный, базовый, углубленный)	Да	
2.	**Экспертиза раздела 2 «Содержание программы»**		
2.1.	Представлен учебный (тематический) план программы	Да	
2.2.	Имеется рабочая программа	Да	
2.3.	В программе кратко раскрыто содержание тем, указаны виды учебных занятий и учебных работ, срок их освоения	Да	
3.	**Экспертиза раздела 3 «Форма аттестации и оценочные материалы» на наличие пунктов раздела**		
3.1.	Описаны вид аттестации, формы контроля, вид оценочных материалов итоговой (при наличии) и промежуточной (при наличии) аттестации	Да	

3.2.	Описаны оценочные средства контроля (контрольно-измерительные материалы)	Да	
4.	**Экспертиза раздела 4 «Организационно-педагогические условия реализации программы»**		
4.1.	Учебно-методическое и информационное обеспечение программы соответствует всем видам учебной деятельности, предусмотренным программой	Да	
4.2.	Перечень рекомендуемой основной и дополнительной *(при наличии)* литературы содержит современные и общедоступные источники. Перечень основной литературы должен содержать источники последних 5 лет	Да	
4.3.	Перечисленные Интернет-ресурсы достоверны (*при наличии*)[1]	Да	
4.4.	Указанное материально-техническое обеспечение программы соответствует направленности и содержанию программы	Да	
4.5.	Соблюдение требований к оформлению программы	Да	

ИТОГОВОЕ ЗАКЛЮЧЕНИЕ

Программа рекомендована к реализации в образовательном процессе

А.Е. Васильева
ФИО эксперта

/ _____ /
Подпись

[1] Могут не указываться авторами программы. В этом случае ставится прочерк