Общество с ограниченной ответственностью «ЦЕНТР ОБРАЗОВАТЕЛЬНЫХ ТЕХНОЛОГИЙ»
(ООО «Центр образовательных технологий»)
ИНН 0600023803 ОГРН 1260600000913
386231, Республика Ингушетия, г. Карабулак, ул. Дьякова, д. 108

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ

по выполнению внеаудиторной работы при реализации программы дополнительного общеобразовательного образования

«Python для начинающих»

г. Карабулак

2026

ВВЕДЕНИЕ

Настоящие методические рекомендации разработаны с целью систематизации работы по формированию и развитию ключевых компетенций в области программирования на языке Python. Документ представляет собой комплексное руководство, объединяющее теоретические основы, практические рекомендации и конкретные примеры организации различных видов учебной деятельности.

Цель документа

Основная цель, создать рекомендации для эффективного освоения курса программирования на языке Python. Рекомендации помогают преподавателям организовать текущий контроль, а обучающимся понять, чего от них ожидают на каждом этапе.

Рекомендации решают следующие задачи:

– Дать преподавателям чёткие ориентиры для проведения тестирования и оценки практических работ на каждом из занятий по программированию.

– Объяснить обучающимся требования к каждому виду учебной деятельности простым и понятным языком.

– Стандартизировать критерии оценивания: тест, практическая работа в Python, итоговый проект.

– Поддержать мотивацию и уверенность юных программистов через чёткие, достижимые результаты.

– Обеспечить безопасную и комфортную образовательную среду при работе в среде программирования Python.

Структура документа

Рекомендации включают два раздела, соответствующих трём ключевым видам учебной деятельности курса программирования:

– Тестирование — проверка теоретических знаний по темам курса Python.

– Практическая работа в среде разработки (IDE) — выполнение конкретного задания по программированию.

– Заключение и рекомендации.

ТЕСТИРОВАНИЕ

Рекомендации для преподавателя

Цели проведения тестирования:

– Проверить, что ребёнок усвоил ключевые понятия занятия: переменные, типы данных, циклы, условия, функции Python.

– Выявить непонимание темы до перехода к следующему занятию.

– Поддержать интерес через короткие, нестрашные задания с картинками и примерами из кода на Python.

– Создать положительный опыт проверки знаний — тест не должен пугать, он должен помогать.

Планируемые результаты:

– Обучающийся называет основные элементы среды разработки Python: консоль, редактор кода, среда разработки, файловая система.

– Различает основные конструкции Python: ввод/вывод, математические операции, строки, списки, словари, модули.

– Знает назначение ключевых конструкций: циклы (for, while) — повторение действий, условия (if, elif, else) — ветвление, переменные — хранение данных, списки и словари — структурирование данных.

– Объясняет принцип структурного программирования в Python: программа выполняется последовательно, используются функции.

– Понимает разницу между типами данных (строки, числа), знает как использовать встроенные функции.

– Умеет находить и исправлять типичные ошибки в коде на Python.

Методические рекомендации:

– Используйте тесты в игровой форме: «Найди ошибку в коде», «Что выведет программа?», карточки с функциями

– Добавляйте скриншоты кода на Python — дети лучше воспринимают визуальные вопросы: «Что сделает этот код?»

– Ограничивайте время: 8–12 минут на тест, не больше — внимание у детей короткое

– Разбирайте ошибки сразу и весело: «Почти! Давай вместе вспомним, как правильно»

Типы заданий для теста:

– Выбор одного правильного ответа из 4 вариантов

– Соответствие: конструкция Python — его действие

– «Верно/Неверно» — утверждения о работе функций и конструкций

– Визуальное задание: назови функцию или конструкцию на скриншоте кода

– Ситуационное: «Какую команду нужно добавить, чтобы цикл остановился?»

Рекомендации для обучающихся

Как подготовиться к тесту:

– Вспомни, что делали на прошлом занятии — расскажи об этом маме или другу.

– Повтори основные слова: как называются конструкции Python, что делает каждый из них.

– Если что-то забыл — попроси преподавателя объяснить ещё раз до начала теста.

– Не волнуйся: тест — это не экзамен, это просто способ проверить, что ты запомнил.

Как выполнять тест:

– Читай каждый вопрос внимательно — до самого конца.

– Если не знаешь ответ — подумай, какой вариант неправильный, и исключи его.

– Отвечай на лёгкие вопросы сначала, потом возвращайся к сложным.

– Если совсем не знаешь — попробуй угадать: иногда интуиция помогает!

– Проверь все ответы перед тем, как сдать работу.

Полезные подсказки:

– Переменная = ячейка для данных. Функция = набор команд. Консоль = экран для вывода результата.

– Цикл while — повторяет пока условие истинно. Цикл for — перебирает элементы. Условие if...else — выполняется при выполнении условия.

– Команда print() — выводит текст. Команда input() — получает данные от пользователя.

– Переменная — это ячейка для хранения числа или текста. Команда x = x + 1 — увеличивает значение переменной.

– Списки — это когда одна переменная хранит много значений. К каждому элементу можно обратиться по индексу!

Пример задания и образец выполнения

Темы: Основы Python. Переменные и типы данных. Циклы, условия, списки

Тестовые задания:

1. Что делает функция print() в Python?

a. Выводит информацию на экран
b. Удаляет файл
c. Считывает данные с клавиатуры
d. Останавливает программу

2. Какая функция используется для ввода данных от пользователя?

a. print()
b. input()
c. type()
d. int()

3. Сколько раз выполнится цикл for i in range(5): ?

a. 1 раз
b. 5 раз
c. 10 раз
d. Бесконечно

4. Что такое переменная в Python?

a. Встроенная функция
b. Ячейка памяти для хранения данных
c. Специальный модуль
d. Команда для остановки цикла

5. Верно ли утверждение: «Цикл while работает, пока его условие истинно»?

e. Верно
f. Неверно

Образец выполнения:

a. Выводит информацию на экран
b. input()
c. 5 раз
d. Ячейка памяти для хранения данных
e. Верно

Критерии оценивания

Оценка	**Правильных ответов**	**Характеристика выполнения**
5	90–100%	Все основные понятия усвоены, допущена не более 1 ошибки. Отличный результат!
4	75–89%	Большинство тем усвоено, допущено 2–3 ошибки. Молодец, почти всё правильно!
3	60–74%	Больше половины верно, но некоторые темы требуют повторения
2	менее 60%	Нужно повторить материал — преподаватель поможет разобраться

Примечания:

– Тест проводится после завершения каждого раздела или по завершении темы.
– Вопросы оцениваются по сути ответа, а не точной формулировке.
– После теста рекомендуется короткое обсуждение: «Что было сложнее всего?».

ПРАКТИЧЕСКАЯ РАБОТА ЗА КОМПЬЮТЕРОМ

Рекомендации для преподавателя

Практическая работа — основной вид деятельности курса. На каждом из занятий обучающийся выполняет конкретное задание в среде разработки (IDE): пишет код, работает с переменными, создает функции и классы, решает алгоритмические задачи. Практика закрепляет теорию и даёт ощущение реального результата — это ключевой мотиватор для юных программистов.

Цели практической работы:

– Освоить практические навыки работы с конкретным типом данных или конструкцией в Python.

– Закрепить теоретический материал через выполнение реальной задачи.

– Развить логическое и алгоритмическое мышление через составление кода.

– Воспитать аккуратность при написании кода и организации структуры проекта.

– Сформировать привычку сохранять проект и делиться ссылкой через систему контроля версий или файлообменник.

Перечень практических работ:

– Занятие 1-2: Введение в программирование, установка языка и запуск первой программы

– Занятие 3-4: Вывод информации на экран: функция print()

– Занятие 5-6: Переменные: как программа запоминает данные

– Занятие 7-8: Числа в Python: целые и дробные

– Занятие 9-10: Простые математические операции (сложение, вычитание, умножение, деление)

– Занятие 11-12: Работа с текстом: знакомство со строками

– Занятие 13-14: Ввод данных от пользователя: функция input()

– Занятие 15-16: Преобразование данных: как сделать из текста число и наоборот

– Занятие 17-18: Калькулятор с помощью Python

– Занятие 19-20: Ошибки в коде: как их читать и исправлять

– Занятие 21-22: Логика в коде: Истина (True) и Ложь (False)

– Занятие 23-24: Операторы сравнения: больше, меньше, равно

– Занятие 25-26: Условный оператор if

– Занятие 27-28: Альтернативные пути: блоки else и elif

– Занятия 29-48: Циклы, списки, функции, модули, ООП и итоговый проект

Методические рекомендации:

– Показывайте каждый шаг на экране — маленьким детям нужна наглядная пошаговая демонстрация.

– Используйте правило «покажи — попробуй сам»: сначала демо, потом ребёнок повторяет.

– Поощряйте за самостоятельные попытки, даже если результат неидеален.

– Для знакомства с интерфейсом — используйте режим пошагового выполнения (подсветка синтаксиса).

– Напоминайте о правиле сохранения: «Сохранить сейчас» после каждого важного изменения.

– Заканчивайте каждое занятие «показом» работы: ребёнок гордится результатом.

Рекомендации для обучающихся

Правила работы за компьютером:

– Садись прямо: спина ровная, ноги на полу, экран на уровне глаз.
– Пиши код аккуратно: соблюдай отступы и правильный синтаксис.
– Не ешь и не пей рядом с компьютером — это может его сломать.
– Перед началом работы убедись, что знаешь, какой результат должен получиться.
– Нажимай Ctrl+S почаще — чтобы не потерять написанный код.
– Если что-то пошло не так — не паникуй, позови преподавателя.

Алгоритм выполнения практической работы:

– Послушай объяснение преподавателя и посмотри, как он делает.
– Повтори шаги самостоятельно — не торопись, делай аккуратно.
– Если забыл шаг — можно подсмотреть в инструкцию или спросить.
– Сохрани готовую работу в нужную папку с правильным именем файла.
– Покажи результат преподавателю и расскажи, что ты сделал.

Полезные горячие клавиши (запомни!):

– Ctrl+S — сохранить файл.
– Ctrl+Z — отменить последнее действие (если ошибся!).
– Ctrl+C — скопировать, Ctrl+V — вставить.
– Ctrl+P — распечатать.
– Delete — удалить выделенный объект.
– Win — открыть меню Пуск.

Пример задания и образец выполнения

Тема: «Создание калькулятора в Python. Ввод/вывод, переменные, математика»

Цель работы: создать в Python программу-калькулятор, которая запрашивает у пользователя два числа и выполняет над ними базовые математические операции.

Задание:

– Открой среду разработки и создай новый файл
– Напиши код для запроса первого и второго числа с помощью input()
– Преобразуй введенные данные в числа с помощью int() или float()
– Напиши код для вывода суммы, разности, произведения и частного этих чисел с помощью print()
– Сохрани файл с кодом с названием «calculator_[имя].py»

Критерии оценивания

Оценка	Критерии	Характеристика выполнения
5	Отлично	Задание выполнено полностью: переменные созданы, вычисления верны, проект сохранён. Ребёнок уверенно объяснил логику программы

Оценка	Критерии	Характеристика выполнения
4	Хорошо	Задание выполнено с 1–2 незначительными недочётами в логике или оформлении кода
3	Удовл.	Задание выполнено частично: основные команды есть, но программа работает не полностью
2	Неудовл.	Задание не выполнено или выполнено с помощью преподавателя полностью, без самостоятельной сборки кода

Примечания:

– Частичная помощь преподавателя допустима — оценивается степень самостоятельности

– Если ребёнок старался, но не успел — дайте дополнительное время или оцените частичный результат

ЗАКЛЮЧЕНИЕ

Общие рекомендации для преподавателя:

– **Создавайте атмосферу открытия**: каждый урок — это «ого, я теперь умею это!». Маленький успех — огромная мотивация

– **Используйте игровые форматы**: квизы, соревнования «кто быстрее сохранит файл», командные задания

– **Темп обучения у всех разный** — не торопите тех, кто медленнее, поддерживайте тех, кто опережает

– **Возвращайтесь к принципам алгоритмического мышления на каждом занятии** — это не занудство, это защита ребёнка

– **Родители — ваши союзники**: давайте простые домашние задания («покажи маме свой проект в Python») — это закрепляет навык

– **Фиксируйте прогресс**: папка с работами ребёнка за курс — лучший показатель роста

– **Завершайте каждое занятие «итогом дня»:** «Сегодня ты научился...» — это формирует осознанность

Общие рекомендации для обучающихся:

– **Практикуйся дома**: открой Python и доработай свой проект — добавь новые функции или улучши алгоритм

– **Экспериментируй смело**: в Python нельзя «сломать» проект — ты всегда можешь отменить действие или начать заново

– **Python — это мощный язык!** В будущем ты сможешь создавать на нем сложные программы и сайты

– **Береги аккаунт**: не делись паролем ни с кем — в нём хранятся все твои проекты