**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 итогового проекта при реализации программы дополнительного общеобразовательного образования

**«Python для начинающих»**

г. Карабулак

2026

# ВВЕДЕНИЕ

Настоящие методические рекомендации разработаны для организации и проведения итогового проекта в рамках дополнительной общеобразовательной программы «Python для начинающих». Документ предназначен для преподавателей курса и содержит практические указания по подготовке, сопровождению и оцениванию проектной работы обучающихся.

**Цель документа**

Основная цель — создать единое методическое пространство для эффективной организации завершающего этапа курса, на котором обучающийся создаёт самостоятельный программный продукт на языке Python, демонстрируя освоенные навыки.

**Рекомендации решают следующие задачи:**

– Дать преподавателям чёткие ориентиры по сопровождению проектной работы на финальном занятии курса.

– Объяснить обучающимся требования к проекту доступным языком: что нужно создать, как оформить и как защитить.

– Стандартизировать критерии оценивания итогового проекта.

– Сформировать у обучающихся опыт полного цикла разработки: идея > код > демонстрация.

– Создать положительное завершение курса: обучающийся уходит с реальным проектом и уверенностью в своих силах.

**Структура документа**

Рекомендации включают три раздела, охватывающих все этапы проектной работы:

– Раздел 1. Рекомендации для преподавателя: цели, варианты проектов, методические советы.

– Раздел 2. Рекомендации для обучающихся: как выбрать тему, как создать проект, алгоритм работы.

– Раздел 3. Пример задания, образец и критерии оценивания.

# РЕКОМЕНДАЦИИ ДЛЯ ПРЕПОДАВАТЕЛЯ

**Общее описание итогового проекта**

Итоговый проект — завершающее занятие курса. Обучающийся самостоятельно создаёт программу на Python, используя несколько инструментов и понятий, изученных за курс. Это не просто «ещё одна практическая работа» — это возможность для ребёнка продемонстрировать всё, чему он научился, получить настоящий результат и почувствовать гордость от завершения курса.

**Цели итогового проекта:**

– Обобщить и продемонстрировать ключевые навыки, освоенные за 48 занятий курса.

– Сформировать опыт самостоятельной разработки: от идеи до работающей программы.

– Развить умение объяснять написанный код: что делает программа, как устроена, зачем выбраны те или иные решения.

– Создать положительное завершение курса: обучающийся уходит с реальным проектом, которым гордится.

**Варианты итогового проекта (на выбор обучающегося):**

Преподаватель предлагает не менее трёх вариантов на выбор. Обучающийся выбирает тот, который соответствует его интересам и уровню. Все варианты равнозначны по сложности и требуют применения нескольких изученных инструментов.

**Вариант 1 — «Текстовая игра»**

Обучающийся создаёт текстовую приключенческую игру с использованием условных операторов, циклов и функций. Игра должна иметь как минимум три ветки сюжета, счётчик очков или жизней и финальное сообщение победителю.

**Вариант 2 — «Умная программа-помощник»**

Обучающийся разрабатывает программу-помощника: калькулятор с историей вычислений, генератор паролей, список задач с добавлением и удалением пунктов, телефонный справочник или аналогичное приложение. Программа должна использовать словари или списки, функции и ввод данных от пользователя.

**Вариант 3 — «Графический проект с turtle»**

Обучающийся создаёт графическую программу с модулем turtle: рисунок, орнамент, мини-игра с перемещением персонажа или интерактивная визуализация. Проект должен включать циклы, функции и цветовое оформление.

**Вариант 4 — «Свой проект» (по согласованию с преподавателем)**

Обучающийся предлагает собственную идею проекта. Преподаватель оценивает реалистичность идеи и помогает сформулировать требования. Проект должен содержать минимум три изученных концепции Python.

**Планируемые результаты:**

– Обучающийся самостоятельно создаёт работающую программу на Python.

– Программа включает минимум 3 концепции из курса: переменные, условия, циклы, функции, списки/словари, модули, ООП.

– Обучающийся может объяснить логику кода: что делает каждый блок, почему выбрана та или иная структура.

– Файл проекта сохранён с понятным именем в рабочей папке.

**Методические рекомендации для преподавателя:**

– Сообщите тему проекта заранее — на предпоследнем занятии, чтобы обучающийся успел обдумать идею.

– Помогайте с выбором темы: обучающиеся иногда теряются — предложите несколько конкретных вариантов с примерами.

– В день проекта не торопите: дайте достаточно времени на реализацию идеи и исправление ошибок.

– Организуйте мини-защиту в конце: обучающийся запускает программу и рассказывает о ней 5-10 минут.

– Создайте «праздничную» атмосферу — это итог большого пути: похвалите каждого за конкретный прогресс.

– Рекомендуется сохранить проект как часть «портфолио» обучающегося — напоминание о первых успехах в программировании.

# РЕКОМЕНДАЦИИ ДЛЯ ОБУЧАЮЩИХСЯ

**Как подготовиться к итоговому проекту:**

– Подумай заранее: что именно ты хочешь создать? Игру, программу-помощника, рисунок черепашкой?

– Вспомни всё, чему ты научился: переменные, условия, циклы, функции, списки, словари, модули, ООП.

– Обсуди идею с преподавателем — он поможет понять, реалистична ли задумка за одно занятие.

– Если выбрал игру — придумай сюжет, что будет происходить, какие будут вопросы или задания.

– Если выбрал программу-помощника — реши, какие данные она будет принимать и что возвращать пользователю.

– Если выбрал turtle — нарисуй на бумаге, как должна выглядеть картинка, прежде чем начать писать код.

**Что должно быть в итоговом проекте:**

– Минимум 3 концепции Python — например: функция + список + цикл, или условия + словарь + ввод от пользователя.

– Работающий код без критических ошибок: программа запускается и делает то, что задумано.

– Аккуратное оформление кода: отступы, понятные имена переменных, короткие комментарии.

– Твоя идея — пусть проект отражает то, что тебе по-настоящему интересно.

– Готовность рассказать о работе: 2–3 предложения — что сделал, какие инструменты использовал, что понравилось.

– Сохранённый файл с понятным именем (например: igra_kvest.py, kalkulator.py, uzor_turtle.py).

**Алгоритм создания итогового проекта:**

Следуй этим шагам, и у тебя всё получится:

1. Открой IDLE или другой редактор и создай новый файл (.py).
2. Напиши в самом начале комментарий: # Автор: [твоё имя], # Описание: [что делает программа].
3. Начни с основной идеи: напиши скелет программы — функции, главный цикл или базовую структуру.
4. Добавляй детали по шагам: сначала основное, потом дополнительный функционал.
5. Запускай программу после каждого добавленного блока — ищи и исправляй ошибки сразу.
6. Сохраняй файл почаще (Ctrl+S) — чтобы не потерять работу.
7. Когда основа готова — улучши: добавь понятные сообщения для пользователя, проверку ввода, красивый вывод.
8. Проверь: всё ли работает? Нет ли ошибок? Понятен ли код?
9. Сохрани финальную версию с понятным именем и покажи преподавателю.

**Полезные советы:**

– Не бойся ошибок в коде — ошибки — это нормально! Читай сообщение об ошибке, оно подсказывает, где проблема.

– Если программа не работает — попробуй добавить print() в разные места, чтобы понять, где что-то идёт не так.

– Если совсем застрял — попроси преподавателя показать направление, но сам пиши код.

– Лучше простой проект, который работает, чем сложный, который не запускается.

# ПРИМЕР ЗАДАНИЯ И КРИТЕРИИ ОЦЕНИВАНИЯ

**Пример проекта: «Текстовая игра-квест»**

Ниже приведён пример итогового проекта для варианта «Текстовая игра». Обучающийся может взять его за основу или создать собственный вариант.

***Описание задания:***

Создать текстовую игру-квест на Python. Персонаж попадает в ситуацию и делает выборы, которые влияют на исход. Игра должна работать в режиме диалога с пользователем через консоль.

***Требования к программе:***

– Приветствие и объяснение правил при запуске.
– Минимум 3 развилки (выбора из 2+ вариантов) с помощью if/elif/else.
– Счётчик очков или жизней, хранящийся в переменной.
– Минимум одна функция, определённая через def.
– Финальное сообщение с результатом — победа или поражение.
– Возможность сыграть снова (цикл while с вопросом «Сыграть ещё раз? (да/нет)»).

***Пример структуры кода:***

```
# Автор: Ислам, # Описание: Текстовый квест «Тайна старого замка»
def start_game():
    print("Ты стоишь у старого замка. Войти или уйти?")
    choice = input("1 - Войти  2 - Уйти: ")
    if choice == "1":
        room_one()
    else:
        print("Ты ушёл. Игра окончена.")
```

***Признаки хорошо выполненного проекта:***

– Программа запускается и работает без ошибок.
– Все ветки сюжета реализованы и приводят к завершению.
– Код написан самостоятельно, с понятными именами переменных.
– Обучающийся может объяснить логику: «Здесь я использовал функцию, потому что...»
– Файл сохранён с понятным именем.

**Рекомендации по подготовке к мини-защите**

***Для преподавателя:***

– Сообщите об итоговом проекте на занятии 47, дайте время обдумать идею.
– В начале занятия 48 отведите 5 минут на уточнение идеи у каждого обучающегося.
– Во время работы — направляйте, но не пишите код за обучающегося.
– В конце занятия проведите мини-защиты: каждый запускает программу и рассказывает о ней 2–3 минуты.
– Завершите курс торжественно: отметьте прогресс каждого, вручите виртуальный «диплом программиста».

***Для обучающихся:***

– Подготовь короткий рассказ о проекте: что делает программа, какие инструменты Python ты использовал.

– Перед защитой запусти программу и убедись, что она работает.

– Не бойся говорить о трудностях: «Я долго не мог разобраться с циклом, но потом понял...» — это тоже достижение.

– Покажи самый интересный момент в коде — то, чем гордишься больше всего.

**Критерии оценивания итогового проекта**

| Оценка | Критерий | Характеристика выполнения |
|---|---|---|
| **5** | **Отлично** | Проект полностью завершён и работает без ошибок. Использовано 3+ изученных инструмента (переменные, функции, циклы, структуры данных и т.д.). Код аккуратно оформлен, обучающийся уверенно демонстрирует и объясняет решение. |
| **4** | **Хорошо** | Проект выполнен с 1–2 незначительными недочётами: небольшие ошибки в оформлении кода или неполное объяснение. Основной функционал работает, использовано 2+ инструмента. |
| **3** | **Удовлетворительно** | Проект выполнен частично: реализована основная идея, но не все функции работают корректно. Рассказ об идее поверхностный. Минимум 1 изученный инструмент применён самостоятельно. |
| **2** | **Неудовлетворительно** | Проект не завершён или выполнен целиком с помощью преподавателя без самостоятельных решений. Обучающийся не может объяснить собственный код. |

***Особые отметки преподавателя:***

– Рекомендуется дополнительно отмечать: оригинальность идеи, трудолюбие, помощь другим обучающимся.

– Итоговый проект желательно сохранить как «портфолио» обучающегося — напоминание о первых успехах в программировании.

– По возможности организуйте мини-выставку работ или поделитесь проектами с родителями.

# ЗАКЛЮЧЕНИЕ

## Общие рекомендации для преподавателя:

– Создавайте атмосферу открытия: каждый запуск рабочей программы — это «я сделал это!». Небольшой успех — огромная мотивация.

– Используйте игровые форматы: соревнование «чей персонаж пройдёт дальше», командная отладка кода.

– Темп обучения у всех разный: не торопите тех, кто медленнее, поддерживайте тех, кто опережает.

– Напоминайте, что профессиональные программисты тоже ищут ошибки и пользуются документацией — это нормально.

– Родители — ваши союзники: давайте простые задания («покажи маме, что делает твоя программа») — это закрепляет навык и создаёт гордость.

– Фиксируйте прогресс: папка с проектами за курс — лучший показатель роста.

– Завершайте курс словами: «Сегодня ты стал настоящим программистом» — это формирует уверенность в себе.

## Общие рекомендации для обучающихся:

– Практикуйся дома: попробуй улучшить свой проект после курса — добавь новые уровни, команды или рисунки.

– Python — это только начало! После этого курса ты готов изучать более сложные темы: веб-разработку, данные, игры.

– Береги свои проекты: это твои первые программы — через год ты будешь с гордостью показывать их друзьям.

– Продолжай программировать: создавай маленькие программы для себя — для учёбы, для игры, просто ради интереса.