План изучения Python для школьника 9–11-го класса.

Учтены:

- ограниченный объём времени (учёба + подготовка к экзаменам);
- интерес к реальным проектам и «видимому» результату;
- необходимость сочетать теорию с практикой без перегрузки.

Принципы обучения

1. **20–40 минут в день** — лучше регулярно, чем редко по 4 часа.
2. **Каждый этап — мини-проект** (чтобы видеть результат).
3. **Минимум «сухой» теории** — только то, что нужно для задачи.
4. **Использование школьных знаний** (математика, логика, английский).
5. **Портфолио из 5–7 проектов** к концу обучения.

Этап 1. Первые шаги (2–4 недели)

Цель: написать первую программу, понять, как запускается код.

Что изучить:

- установка Python и VS Code/PyCharm Community;
- первая программа: `print("Привет, мир!")`;
- переменные и типы данных (int, float, str, bool);
- ввод с клавиатуры: `input()`;
- арифметические операции.

Практика:

- калькулятор для расчёта среднего балла по предметам;
- программа, которая спрашивает имя и выводит приветствие.

Ресурсы:

- официальный туториал Python (раздел «Beginners»);
- видео «Python для школьников» (каналы «Хауди Хо», «Поколение Python»).

Этап 2. Логика и циклы (3–5 недель)

Цель: научиться управлять ходом программы.

Что изучить:

- условия `if-elif-else`;
- циклы `for` и `while`;
- операторы сравнения и логические операторы (`and`, `or`, `not`).

Практика:

- игра «Угадай число» (компьютер загадывает, игрок угадывает);

- программа для расчёта сложных процентов по вкладу;
- вывод таблицы умножения с помощью цикла.

Совет: рисуйте блок-схемы перед кодом — это связывает логику и программирование.

Этап 3. Функции и списки (4–6 недель)

Цель: писать модульный код и работать с коллекциями данных.

Что изучить:

- функции (def, аргументы, return);
- списки, методы append(), remove(), индексация;
- строки и методы работы с ними;
- основы отладки (чтение ошибок, print() для проверки).

Практика:

- список дел (To-Do List) с добавлением и удалением задач;
- программа для анализа оценок (среднее, максимум, минимум);
- шифр Цезаря (сдвиг букв в строке).

Ресурс: курс «Python: основы и применение» на Stepik (русскоязычный, с практикой).

Этап 4. Файлы и ошибки (3–4 недели)

Цель: сохранять данные и обрабатывать исключения.

Что изучить:

- работа с файлами (open(), read(), write());
- обработка ошибок (try-except);
- формат CSV (простейшая работа с таблицами).

Практика:

- дневник оценок, который сохраняет данные в файл;
- программа для поиска слов в тексте (например, в сочинении).

Совет: начните вести *журнал ошибок* — записывайте, что сломалось и как починили.

Этап 5. Первые «полезные» проекты (6–8 недель)

Цель: применить знания к реальным задачам.

Выбор проектов (сделайте 2–3):

1. **Калькулятор ИПК** (индивидуальный пенсионный коэффициент) — использует математику из обществознания.
2. **Анализатор текста** — считает слова, предложения, частотные слова в тексте (связь с русским/литературой).

3. **Таймер для Pomodoro** — помогает учиться по методу интервалов.
4. **Бот для напоминаний** (через time или schedule).
5. **Визуализация данных** — график роста растений (используйте matplotlib).

Что освоите:

- структурирование кода;
- работу с внешними модулями;
- пользовательский ввод и вывод.

Этап 6. Веб и API (4–6 недель, по желанию)

Цель: понять, как программы общаются с интернетом.

Что изучить:

- библиотека requests (запросы к сайтам);
- JSON-данные;
- простейший парсинг (извлечение информации с веб-страницы).

Практика:

- программа, которая показывает погоду (через открытый API);
- бот, который присылает факты о космосе (API NASA);
- парсер новостей с образовательного сайта.

Ресурс: документация requests, туториалы на Real Python.

Этап 7. Подготовка к олимпиадам/проектам (по желанию)

Если интересно соревновательное программирование:

- сайт **Codeforces** (раздел «Div. 4»);
- книга «Программирование: теоремы и задачи» (А. Шень);
- разбор задач с муниципальных этапов ВсОШ по информатике.

Если интересует прикладное программирование:

- создайте **портфолио на GitHub** (5–7 проектов с описанием);
- попробуйте **Flask** (простейший веб-сайт с калькулятором);
- изучите **Git** (базовые команды: commit, push, pull).

Как организовать процесс

1. **Расписание:**
    - 3–4 раза в неделю по 30–40 минут;
    - 1 раз в неделю — «проектный день» (1,5 часа на новый проект).
2. **Фиксация прогресса:**

- тетрадь/Notion/Google Docs — темы, ошибки, идеи;
- GitHub — код проектов.

3. **Поддержка:**
   - школьные кружки по программированию;
   - Telegram-чаты по Python для школьников;
   - форумы (Stack Overflow на английском — с переводчиком).
4. **Мотивация:**
   - показывайте проекты друзьям/учителям;
   - участвуйте в школьных хакатонах;
   - ставьте маленькие цели («Сегодня добавлю функцию в калькулятор»).

Что НЕ делать

- не пытайтесь выучить всё сразу;
- не копируйте код без понимания (лучше 10 строк своего, чем 100 чужих);
- не бойтесь ошибок — они часть обучения;
- не сравнивайте себя с другими (у всех свой темп).

Итоговый «чек-лист» к 11-му классу

К концу обучения у вас будет:

- 5–7 рабочих проектов в портфолио;
- понимание основ синтаксиса и логики Python;
- опыт работы с файлами и внешними API;
- навык отладки и поиска решений;
- аккаунт на GitHub с чистым кодом;
- уверенность, чтобы продолжить в вузе или на курсах.

Главное: программирование — это творчество. Пробуйте, ошибайтесь, исправляйте — и у вас всё получится!

Источники:

1. https://clubpixel.ru/sozdanie-igr-na-pajton/tpost/ghtkknrah1-12-besplatnih-urokov-po-python-dlya-shko
2. https://blog.pixel.study/python-dlya-detej/uroki-python-dlya-podrostkov-kak-napisat-pervyj-kod
3. https://mmx2024.ucoz.net/programming_for_beginners.html