DOI 10.37882/2223-2966.2024.6-2.03

РАЗРАБОТКА ОНЛАЙН-КОМПИЛЯТОРА PYTHON В ЦЕЛЯХ ПОВЫШЕНИЯ КАЧЕСТВА ОНЛАЙН ОБРАЗОВАНИЯ

DEVELOPMENT OF AN ONLINE PYTHON COMPILER IN ORDER TO IMPROVE THE QUALITY OF ONLINE EDUCATION

N. Azarenko
K. Magankov
N. Ryabtsev

Summary. The article describes the process of developing a compiler in the Python programming language with the aim of improving the quality of online education. The advantages of the chosen technology stack are identified. The process of software implementation is outlined.

Keywords: online education, software development, Python programming language, compiler.

Азаренко Наталья Юрьевна
кандидат экономических наук, доцент,
ФГБОУ ВО «Брянский государственный инженерно-технологический университет»
salovanat@mail.ru
Маганков Кирилл Сергеевич
ФГБОУ ВО «Брянский государственный инженерно-технологический университет»
kirill.magankov@gmail.com
Рябцев Николай Павлович
ФГБОУ ВО «Брянский государственный инженерно-технологический университет»
lolofmeistahhz@gmail.com

Аннотация. В статье описывается процесс разработки компилятора на ЯП Python с целью повышения качества образования в онлайн формате. Выявлены преимущества выбранного стека технологий. Расписан процесс реализации программного обеспечения.

Ключевые слова: онлайн образование, разработка ПО, ЯП Python, компилятор.

Введение

В течение последних годов образовательная сфера перетерпела значительные изменения, появилось множество новшеств, одним из которых является внедрение технологий дистанционного образования, как одного из ключевых элементов предоставления возможности обучения.

Такой вид образования быстро закрепился в образовательной сфере и стал набирать обороты, что вызвано его уникальностями — например такой формат предоставляет возможности для саморазвития и обучения в любом месте и в любое время.

Использование дистанционных обучающих технологий (ДОТ) — предполагает внедрение информационные и коммуникационных систем и технологий в процесс обучения. Данный метод позволяет студентам получать навыки и знания при помощи различных, специализированных платформ на которых преподаватели могу размещать свои курсы, вебинары, проводить лекции и многое другое, а самое главное — всё это происходит, не выходя из дома [3].

Таким образом одним из ключевых преимуществ дистанционного образования является его гибкость, что обуславливается возможностью студентов выбирать удобное время для проведения и посещения занятий, позволяет им самим планировать свой график обучения и не ограничивает их во времени.

Исходя из этого образование становится доступным для всех — независимо от географического расположения, возраста или семейного положения [5].

Кроме того, электронное образование предполагает использование интерактивных технологий в рамках обучения студентов, что позволяет им использовать различные мультимедийные материалы, быть на связи с преподавателем при помощи различных средств связи, например видеоконференцсвязь (ВКС) — имеет большой спрос и популярность в ходе проведения лекционных занятий, семинаров и т.д. А если к такого роду занятий интегрировать онлайн-платформу с различными практическими заданиями и материалами по теме, то эффект обучения значительно возрастет.

Одной из крупнейших платформ в РФ, предоставляющей услуги по размещению и организации онлайн-школ и курсов, является платформа «GetCourse» [https://getcourse.ru/]. Возрастающая популярность курсов по программированию, в частности по изучению ЯП Python, привела к возникновению необходимости разработки онлайн компилятора.

В образовательных материалах содержатся текстовые блоки, содержащие код на ЯП Python и результат

его интерпретации. Однако, наличие интерпретатора в онлайн-среде упрощает процесс обучения. Студенту не требуется стороннее ПО и IDE для выполнения кода, что экономит время на перенос кода и способствует повышению концентрации обучающегося.

Компилятор кода на платформе онлайн-образования играет важную роль в обучении программированию и другим техническим дисциплинам. Перечислим ключевые аспекты его актуальности и необходимости:

1. Возможность непосредственного исполнения кода: компилятор позволяет студентам немедленно видеть результаты написанного кода, способствуя улучшению понимания принципа работы кода.
2. Проверка кода на правильность: компилятор автоматически проверяет код на соответствие заданному синтаксису, что особенно полезно для начинающих, помогая избегать механических ошибок.
3. Удобство и доступность: при наличии компилятора кода на платформе онлайн-образования, студентам не нужно устанавливать и настраивать собственное программное обеспечение.
4. Совместная работа и обратная связь: компилятор позволяет вести обсуждения, делиться кодом, задавать вопросы и получать обратную связь от других участников курса, способствуя более полному пониманию материала.

Таким образом, разработка онлайн компилятора для ЯП Python становится не только необходимой, но и актуальной задачей в рамках развития электронного образования и повышения эффективности обучения программированию.

Методология

Целью статьи исследования является — рассмотрение процесса создания онлайн компилятора кода для ЯП Python, как одного из важнейших механизмов для дистанционного обучения на онлайн платформах.

В исследовании используются следующие технологии:

— Изучение и анализ требований пользователей к онлайн-компилятору;
— Изучение возможностей платформы Get Course;
— Обзор и анализ существующих решений;
— Проектирование и разработка прототипа компилятора;
— Тестирование и оценка.

Для разработки REST API использован фреймворк fastAPI, преимуществами которого являются:

— Помощь при создании api и автоматическая автоматизации его документации посредством Swagger и ReDoc;
— Быстродействие по сравнению с другими платформами Python;
— Высокая гибкость за счёт отсутствия какой-либо стандартизации.

Для управления потоками ввода/вывода в режиме реального времени использовалась библиотека socket.io, преимуществами которой являются:

— В отличие от веб-сокетов, Socket.IO позволяет отправлять сообщения всем подключенным клиентам.

Например, при разработке чата необходимо уведомлять всех пользователей о подключении нового пользователя. Это реализуется с помощью одной операции. При использовании веб-сокетов, для реализации подобной задачи потребовался бы список подключенных клиентов и отправка сообщений по одному;

— В веб-сокетах сложно использовать проксирование и балансировщики нагрузки. Socket.IO поддерживает эти технологии «из коробки»;
— Socket.IO поддерживает постепенную (изящную) деградацию;
— Socket.IO поддерживает автоматическое переподключение при разрыве соединения;

Для связи функционала со стороны сервера (API) и его привязки к UI — использовался ЯП JS, а также методология DOM. Его преимущества:

— Незаменим в веб-разработке;
— Обладает достаточно высокой скоростью работы и производительностью;
— Имеет развитую инфраструктуру с большим разнообразием фреймворков и библиотек;
— Наличие собственной мощной экосистемы (инфраструктуры).

Основным инструментом работы и динамических изменений на странице является DOM (Document Object Model) — объектная модель, используемая для XML/HTML-документов.

Согласно DOM-модели, документ является иерархией. Каждый HTML-тег образует отдельный элемент-узел, каждый фрагмент текста — текстовый элемент, и т.п. Проще говоря, DOM — это представление документа в виде дерева тегов. Это дерево образуется за счет вложенной структуры тегов плюс текстовые фрагменты страницы, каждый из которых образует отдельный узел [9].

Так же, выбор JS — обоснован функционалом самой платформы «GetCourse» — которая среди встраиваемых блоков с кодом поддерживает только HTML, CSS, JS.

Результаты

В ходе разработки онлайн-компилятора python — было принято решение разработать клиент-серверное приложение:

— Client-Side — представляет собой сам интерфейс, который служит для отправки введённого кода на сервер, с целью его последующей обработки и вывода результата, в основе стека лежит стандартный инструментарий веб-разработчика HTML+CSS+JS;

— Server-Side — представляет собой REST (Representational State Transfer) API, которое служит для отработки получаемого кода и возвращения результата его компиляции, в основе стека лежит библиотека для python FastAPI, а также Socket.io — для передачи потоков ввода-вывода, например в случае и интерактивным вводом значения переменной [1];

Реализация серверной части

Для реализации задачи необходимо было разработать сервер, способный выполнять Python— код и возвращать полученный результат. Так как задача предполагала интерактивный ввод данных, для реализации функции input(), то также требовалось использовать некий механизм, способный на создание двухсторонних соединений, и свободную передачу данных.

За основу компилятора был взят встроенный в систему linux базовый интерпретатор, но т.к. данный интерпретатор не обязательно будет на сервере хостинга, то также было принято решение создать docker-container для корректного размещения и развертывания на сервере.

После чего для корректной работы сервера было реализовано несколько классов — ProcessManager, Process, Config, CodeRunner, подробнее о каждом из них:

1. ProcessManager позволяет управлять процессами, предоставляет интерфейс для их уничтожения;
2. Process — класс, который по сути является оберткой вокруг процесса, позволяет асинхронно читать ввод, и вводить данные в процесс. Он использует библиотеку ptyprocess, которая позволяет легко взаимодействовать с процессом посредством создания псевдо-терминала.
3. Config — класс, который является местом хранения константных значений, которые необходимы для задания определенных параметров при написании сервера, например для установки значения истечения срока подключения при считывании интерактивного ввода и т.д.;
4. CodeRunner — класс, который представляет собой удобный интерфейс для обработки присылаемого пользователем кода и возвращения результата его работы.

Таким образом мы получил сервер, состоящий всего из одного рута (пути) — /api/python/run, после чего началась реализация интерактивного ввода данных и его обработки в режиме реального времени.

Для начала был создан SocketManager, который позволяет вызывать события на сервере и становиться их слушателем/подписчиком, т.е. на сервере создается процесс, после чего мы ждем пока клиент отправит нам сообщение, содержащее определенный статус и уникальный идентификатор процесса, после чего мы проверяем uuid создаем, с помощью библиотеки asyncio два асинхронных задания:

1. Первое на чтение потоков ввода/ошибок процесса;
2. Второе для автоматического завершения процесса, если он запущен слишком долго. Клиент может послать нам сообщение prompt с uuid процесса, и данными, эти данные мы передаем в поток ввода процесса.

На данном моменте разработка серверной части была завершена, после чего требовалось создать интерфейс способный взаимодействовать с сервером.

Реализация клиентской части

За основу был взят онлайн интерпретатор, расположенный по адресу общедоступного размещения: https://www.online-python.com.

Исходя из этого, появилась необходимость в верстке компилятора, который представляет собой 2 текстовых поля для ввода и вывода и 1 поле для отправка интерактивного ввода input(). Также для отправки ввода, выполнения кода или сохранения кода в файл, — требовались элементы управления, в данном случае — это кнопки.

После того, как все необходимые структурные элементы были готовы, началась их стилизация, для который использовалась каскадная таблица стилей (CSS) + библиотека Bootstrap, для подсветки синтаксиса кода использовалась библиотека prism.js [2,9].

По завершению стилизации — имелась необходимость функциональной привязки элементов, т.е. надо было задать функционал, осуществляемый при отправке формы, именно для этого использовался JS + SocketIO.client.

Первоначально потребовалось определить маршруты — куда посылать запросы на выполнение кода, после чего надо было инициализировать те переменные, которые при загрузке страницы будут хранить в себе ссылки на все используемые элементы (поля ввода, кнопки).

После этого — необходимо было для каждого элемента создать событие, таким образом были написаны 3 основных функции:

1. consoleGo — считывает значения из поля ввода кода, после чего передаёт данное значение в качестве аргумента в функцию executeCode, после чего производит проверку на наличие ответа со стороны сервера и заполняет соответсвующие поля содержимым ответа из функции executeCode;
2. executeCode — функция, которая принимает один параметр, после чего передаёт его значение на сторону сервера по хранимому маршруту вместе с телом запроса json, чтобы получить объект promise со стороны сервера в качестве ответа.
3. sendInput() — функция отправки интерактивного ввода пользователя, является триггером события «prompt» — для сокета на сервере, передаёт ему uuid процесса и само считанное значение;
4. saveCodeToFile() — функция сохранения введённого кода в формате .py скрипта;

В результате реализации данных функций — процесс разработки онлайн компилятора для ЯП python — был завершен.

Рис. 1. Демонстрация интерфейса и работы компилятора

Обсуждения

Исследования о построении компиляторов подтверждают разнообразие подходов и методов, используемых в этом процессе. Результаты анализа показывают, что архитектура компилятора в значительной степени зависит от окружающей среды, в которую он будет интегрирован. Это означает, что разработчики компиляторов должны учитывать особенности платформы, языка программирования и требования конечных пользователей [8,10].

Например, в среде веб-разработки акцент может быть сделан на эффективной работе с сетью и обработке запросов, в то время как в мобильной разработке важным является оптимизация под ограниченные ресурсы устройства. Кроме того, важно учитывать потребности пользователей и их ожидания от компилятора.

Бывают пользователи, которые используют компиляторы с обширным набором функций и расширенными возможностями, в то время как другие предпочитают легкие и быстрые инструменты. В процессе разработки компилятора важно не только иметь технические знания, но и понимать потребности и ожидания конечных пользователей. Гибкость и адаптивность при выборе архитектуры компилятора позволяют разработчикам создавать инструменты, которые наилучшим образом соответствуют требованиям среды и предпочтениям пользователей.

Выводы

В ходе данного исследования авторами выдвигаются следующие выводы, касаемые исследуемой области:

1. Онлайн-компилятор как важный инструмент в рамках процесса Обучения в онлайн формате;
2. Актуальность, важность, возможная масштабируемость онлайн-курсов;
3. Дистанционное обучение как мощный инструмент преподавания в нетипичных условиях, например — учеба для людей с ограниченными возможностями или в период пандемии;

В заключении, стоит отметить, что разработанный прототип онлайн компилятора python является одним из необходимых инструментов масштабирования и развития онлайн-курсов.

Данный тезис обуславливается возможностями доступа к лицензионному программному обеспечению, а также тем, что в ходе взаимодействия с онлайн-площадкой, предоставляющей возможность дистанционного обучения, имеющей онлайн-компилятор — процесс вовлечения студента больше. Это обуславливается принципом удержания пользователя, т.е. возможность решать задания там же, где они были получены — вовлекает пользователя в процесс сильнее, нежели процесс решения задач в одной среде, а процесс получения и проверки полученных в ходе усвоения материалов — в другой среде.

Ссылка на репозиторий с кодом онлайн-компилятора: https://github.com/Lolofmeistahhz/python_online_compiler

ЛИТЕРАТУРА

1. Арно Лоре. Проектирование веб-API. — М: ДМК Пресс. 2020. 213 с.
2. Беликова С.А. Основы HTML и CSS: проектирование и дизайн веб-сайтов: учебное пособие по курсу «Web-разработка». 2020. 132 с.
3. Благасарян Л.Г., Марукян Л.Р. Эффективность использования онлайн(виртуальных) платформ в процессе обучения. 2023. URL: https://cyberleninka.ru/article/n/effektivnost-ispolsovaniya-onlayn-virtualnyh-platform-v-protsesse-obucheniya/viewer (дата обращения: 27.12.2023);
4. Кондратьева, Е.Е. Актуальность использования образовательной интернет-платформы в процессе развития интеллектуальных способностей младших школьников / Е.Е. Кондратьева. — Текст: непосредственный // Молодой ученый. 2023. № 39 (486). С. 177–179. URL: https://moluch.ru/archive/486/106182/ (дата обращения: 29.12.2023);
5. Ламонина Л.В., Смирнова О.Б. Об использовании цифровых онлайн-технологий в дистанционном обучении. 2020. URL: https://cyberleninka.ru/article/n/ob-ispolzovanii-tsifrovyh-onlayn-tehnologiy-v-distantsionnom-obuchenii (дата обращения: 27.01.2024);
6. Лутц Марк. Изучаем Python. — М: Эксмо. 2020. 455 с.
7. Макфарланд Дэвид. JavaScript и jQuery. Исчерпывающие руководство. М: Эксмо.2021. 111 с.
8. Павел Соломатин. JIT-компилятор Python в 300 строк. 30.06.2022. URL: https://habr.com/ru/articles/674206/ (дата обращения: 21.01.2024).
9. Флэнаган Дэвид. JavaScript. Полное руководство: справочник по самому популярному языку программирования. М: Диалектика. 2021. 431 с.
10. Bergstra J. et al. Theano: A CPU and GPU math compiler in Python //Proc. 9th python in science conf. — 2010. — Т. 1. — С. 3–10.
11. Chen J. et al. Compiler test-program generation via memoized configuration search //Proc. 45th International Conference on Software Engineering. 2023.
12. Digital Blackboard. Building an online Python compiler using the MERN-stack. 23.01.2023. URL: https://digitalblackboard.io/posts/online-compiler/ (дата обращения: 07.01.2024).
13. Jovanović N. et al. ComVIS—Interactive simulation environment for compiler learning //Computer Applications in Engineering Education. 2022. Т. 30. №. 1. С. 275–291.
14. Kaya M., Özel S.A. Integrating an online compiler and a plagiarism detection tool into the Moodle distance education system for easy assessment of programming assignments //Computer Applications in Engineering Education. 2015. Т. 23. №. 3. С. 363–373.
15. Kovtaniuk M.S. Online compiler «Replit» usage during the study of the Programming discipline //Publishing House «Baltija Publishing». 2023.
16. Piwek P., Savage S. Challenges with learning to program and problem solve: An analysis of student online discussions //Proceedings of the 51st ACM Technical Symposium on Computer Science Education. 2020. С. 494–499.
17. Salama R., Uzunboylu H., Alkaddah B. Distance learning system, learning programming languages by using mobile applications //New Trends and Issues Proceedings on Humanities and Social Sciences. 2020. Т. 7. №. 2. С. 23–47.
18. Sinanaj L. et al. A comparison between online compilers: A case study //2022 11th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2022. С. 1–6.
19. Steingartner W. Compiler module of abstract machine code for formal semantics course //2021 IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI). IEEE, 2021. С. 000193–000200.
20. Watanobe Y. et al. Next-generation programming learning platform: Architecture and challenges //SHS Web of Conferences. EDP Sciences, 2020. Т. 77. С. 01004.