Лекция 1. Понятие языка программирования. Введение в язык программирования

Перечень вопросов

[**Введение.** 2](#_Toc167788326)

[**1.** **История создания и область применения языка программирования Python** 4](#_Toc167788327)

[**2.** **Методология программирования** 7](#_Toc167788328)

[**3.** **Среда разработки (IDE). Виды и работа с IDE** 11](#_Toc167788329)

[**3.1.** **Python IDLE** 11](#_Toc167788330)

[**3.2.** **JetBrains PyCharm** 13](#_Toc167788331)

[**3.3.** **Visual Studio Code** 14](#_Toc167788332)

[**4.** **Структура программ** 16](#_Toc167788333)

[**5.** **Простой ввод-вывод. Комментарии** 21](#_Toc167788334)

[**6.** **Работа со стандартными библиотеками** 23](#_Toc167788335)

[**Заключение** 25](#_Toc167788336)

[**Список литературы** 25](#_Toc167788337)

**Понятие языка программирования. Введение в язык программирования**

**Цели:** формирование знаний по вопросам

**План лекции:**Введение.1. История создания и область применения языка программирования Python.2. Методология программирования.  
3. Среда разработки (IDE). Виды и работа с IDE.  
4. Структура программ.  
5. Простой ввод-вывод. Комментарии.  
6. Работа со стандартными библиотеками.Заключение.Список литературы.

# **Введение.**

Язык программирования Python является одним из самых популярных и востребованных инструментов в современном мире разработки программного обеспечения. С момента своего создания в начале 1990-х годов Python прошел долгий путь, превратившись из эксперимента в полноценный, мощный и гибкий язык программирования, который используется в самых различных областях, начиная от веб-разработки и заканчивая научными исследованиями и искусственным интеллектом.

История создания Python неразрывно связана с именем его создателя, Гвидо ван Россума, который стремился создать язык, способный упростить процесс написания и чтения кода, сделать его более интуитивным и понятным для разработчиков. С тех пор Python завоевал признание за свою простоту, читаемость и обширные возможности, став выбором миллионов программистов по всему миру.

Методология программирования на Python включает в себя несколько ключевых аспектов, которые делают этот язык уникальным и эффективным. Python поддерживает как процедурное, так и объектно-ориентированное программирование, а также имеет мощные средства для функционального программирования. Эти методологии позволяют разработчикам писать чистый, понятный и поддерживаемый код, что особенно важно при работе над крупными проектами.

Для работы с Python существует множество интегрированных сред разработки (IDE), каждая из которых предлагает свои уникальные возможности и инструменты для повышения продуктивности. Среди самых популярных можно выделить PyCharm, Visual Studio Code и Jupyter Notebook, которые предоставляют разработчикам широкий спектр функций для написания, тестирования и отладки кода.

Структура программы на Python отличается своей простотой и логичностью. Код на Python легко читается благодаря использованию отступов для определения блоков кода, что способствует созданию чистой и поддерживаемой архитектуры приложений. Простота синтаксиса и мощные стандартные библиотеки позволяют быстро создавать эффективные и надежные программы.

Одним из основных аспектов любого языка программирования является возможность взаимодействия с пользователем. Ввод и вывод данных на Python реализуются с помощью простых и интуитивно понятных функций, таких как input() и print(). Эти функции обеспечивают гибкость и простоту работы с данными, что является важным элементом при создании приложений.

Комментарии кода играют важную роль в разработке программного обеспечения, обеспечивая возможность документирования и пояснения сложных участков кода. Python предоставляет разработчикам удобные средства для создания комментариев, которые помогают поддерживать и улучшать качество кода на протяжении всего жизненного цикла проекта.

В этой статье мы подробно рассмотрим каждый из упомянутых аспектов, чтобы дать читателям полное представление о языке программирования Python, его возможностях и преимуществах.

# **История создания и область применения языка программирования Python**

Сегодня мы начнём изучать основы алгоритмизации и программирования. Этот курс поможет вам освоить основные приёмы разработки алгоритмов и их реализации на компьютере. Но, прежде чем перейти к изучению самого курса, мы рассмотрим инструмент, который будем активно в нём применять. Программировать разработанные нами алгоритмы мы будем на языке программирования Python. Сегодня мы рассмотрим его историю, особенности и первый запуск рабочей среды.

Итак, язык программирования Python был задуман нидерландским программистом Гвидо ван Россумом (рис. 1) в 1980-х годах. Ван Россум приступил к его созданию в декабре 1989 года в Центре математики и информатики в Нидерландах. Язык Python был задуман как потомок языка программирования ABC. В отличие от своего предка, Python должен был получить возможность обрабатывать исключения и взаимодействовать с открытой операционной системой Amoeba, разработанной в 1983 году в свободном университете Амстердама. В 1991 году ван Россум опубликовал код своего языка программирования. Он начал набирать популярность в интернете. В 1994 году, благодаря росту числа пользователей, была создана группа comp.lang.python – основной форум языка Python.



Рис. 1. Гвидо ван Россум

Язык Python был назван так не в честь змеи. На момент создания языка, Гвидо ван Россум был увлечён просмотром комедийного сериала «Воздушный цирк Монти Пайтона». В честь главного героя этого сериала, Монти Пайтона, язык и получил своё название.

Сейчас Python – это активно развивающийся высокоуровневый многоцелевой язык программирования. Он поддерживает несколько, наиболее популярных сейчас, парадигм программирования, таких как структурное, объектно-ориентированное, функциональное программирование и другие. Популярности языка способствует то, что он соответствует стандартам Американского национального института стандартов и Международной организации по стандартизации. Язык Python входит в пятёрку наиболее популярных языков программирования по итогам 2016 года. Эталонная реализация Pythonа – это интерпретатор CPython, он совместим с большинством активно используемых на сегодняшний день платформ. Этот интерпретатор распространяется по свободной лицензии Python Software Foundation License.

Гвидо ван Россум и по сей день является активным участником разработки языка Python. Среди разработчиков он известен как «Великодушный пожизненный диктатор проекта». Это означает, что он продолжает наблюдать за разработкой языка и принимает окончательные решения по его развитию, когда это необходимо.

Почему же язык Python получил такую популярность? Начнём с того, что Python – это многоцелевой язык. Его можно одинаково хорошо использовать для разработки любых программ и их тестирования. Так, например, компания Google широко использует язык Python для своей поисковой системы. Большая часть популярного видеохостинга YouTube была написана на языке Python. Также язык Python применяется в анимационной графике, научных вычислениях и тестировании аппаратного обеспечения.

Так как для написания программ нужно меньше кода – они разрабатываются быстрее и достаточно легко читаются. Язык Python получился настолько простым в использовании, что его начали использовать для обучения студентов программированию, хотя изначально такой цели не ставилось.

Большая часть программ на языке Python выполняется без изменений, независимо от того, на какой платформе они используются. То есть, в большинстве случаев программа, написанная на языке Python, будет одинаково правильно работать на различных операционных системах, например, в Windows, Linux и Mac OS. Программы на языке Python легко интегрируются с компонентами, написанными на других языках программирования, таких как C, C++ и Java.

В стандартных библиотеках языка Python изначально реализовано множество возможностей, которые могут быть полезны при написании самых разных программ. Кроме расширений, язык Python также поддерживает библиотеки, написанные другими разработчиками, что ещё больше увеличивает доступную функциональность.

Итак, рассмотрим, как начать использовать Python. Как говорилось ранее, он распространяется по свободной лицензии. Поэтому для того, чтобы установить язык Python на компьютер, достаточно бесплатно скачать актуальную версию языка с сайта разработчиков python.org. При изучении этого курса, примеры программ будут писаться на языке Python версии 3.12.

# **Методология программирования**

Методология программирования — это совокупность идей, понятий, принципов, способов и средств, определяющая стиль написания, отладки и сопровождения программ.

Парадигма программирования — это совокупность идей и понятий, определяющая стиль написания программ. Парадигма, в первую очередь, определяется базовой программной единицей и самим принципом конструирования программы.

Методология программирования ≡ парадигма программирования.

Парадигмы программирования определяются наборами инструментов, а именно, языком программирования и используемыми библиотеками, и системами программирования.

В качестве парадигм – единиц выступают:

* определения (декларативное программирование),
* предикаты ([логическое программирование](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)),
* функции (функциональное программирование),
* действие (императивное [программирование](https://www.tadviser.ru/index.php/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)),
* правило (продукционное программирование),
* диаграмма переходов (автоматное программирование) и др.

**Декларативное программирование**

В работе над задачами мы сначала представляем себе нужный результат, а уже потом придумываем способ этого результата достичь. Этот подход используется и в программировании. Иногда программисту достаточно описать результат, а способы его достичь компьютер придумывает сам. Декларативное программирование — это и есть описание конечного результата.

Решая задачу в декларативном стиле, мы описываем, как результат соотносится со входными данными. При необходимости мы уточняем детали, пока интерпретатор не получит достаточно информации, чтобы дальше действовать самостоятельно. Уточняя наши требования, мы все еще не говорим компьютеру, как именно делать работу — мы лишь точнее описываем соотношение «входов» и «выходов».

Декларативное программирование часто похоже на написание некоторой формулы или соединение отдельных операций в конвейер, который из входных данных формирует выходные. Сложные формулы мы разбиваем на более простые, сложные операции делим на цепочки операций попроще. Во время декомпозиции не приходится думать, как будет меняться состояние компьютера, как он будет хранить промежуточные данные.

Это упрощает написание и чтение кода, но усложняет сам интерпретатор или компилятор. И тем сложнее делать интерпретаторы и компиляторы, которые позволят компьютеру работать эффективно. Поэтому языки низкого уровня практически не имеют средств для декларативного программирования, ведь для таких языков главное — это эффективность. При этом далеко не каждый язык высокого уровня предоставляет средства для решения любых задач строго декларативно. Обычно речь идет лишь об элементах декларативного программирования.

**Логическое программирование**

Логическое программирование — это подход к программированию, в котором основной упор делается на логическое следование и декларативный характер описания программы.

**Принципы логического программирования**

В логическом программировании программист не описывает последовательность действий, необходимых для решения задачи, а скорее описывает знания и факты, которые должны быть учтены для решения задачи. Программа на языке логического программирования представляет собой набор правил и фактов, описывающих отношения между объектами и причинно-следственные связи.

Логическое следование — это основной принцип логического программирования. Оно позволяет получать новые факты из уже известных, а также делать выводы на основе имеющейся информации. При выполнении программы на языке логического программирования система стремится получить правильный ответ, находя логические последовательности из фактов и правил.

В логическом программировании процедуры и функции неразличимы, то есть они описывают отношения между объектами, а не последовательность действий. Обработка списков — это еще один принцип логического программирования, который позволяет программе работать с набором данных в виде списков.

**Функциональное программирование**

Функциональное программирование является одной из парадигм, поддерживаемых языком программирования Python. Основными предпосылками для полноценного функционального программирования в Python являются: функции высших порядков, развитые средства обработки списков, рекурсия, возможность организации ленивых вычислений. Элементы функционального программирования в Python могут быть полезны любому программисту, так как позволяют гармонично сочетать выразительную мощность этого подхода с другими подходами.

**Императивное программирование**

Представим, что нам нужно решить какую-то сложную задачу. Ее нельзя решить в одно действие компьютера, поэтому мы описываем последовательность шагов, которые надо выполнить.

Широкие шаги разбиваются на более мелкие до тех пор, пока каждый отдельный шаг не станет понятен интерпретатору или компилятору выбранного языка. Проще говоря, мы продумываем и описываем то, как достичь нужного результата. Такое описание четкой последовательности шагов называется императивным программированием.

Императивное программирование сопряжено с изменением состояния компьютера. Большие программы постоянно модифицируют глобальные и локальные переменные: записывают что-то на диск, посылают или принимают из сети, запускают отдельные подпрограммы и процедуры.

Такой код дает максимально возможную эффективность. Но есть и минусы: он заставляет программиста держать в уме все, что меняется при выполнении каждого конкретного шага.

Большинство языков низкого уровня предполагает кодирование только в этой парадигме — ведь компьютер только и делает, что последовательно выполняет команды на машинном языке и меняет состояние памяти. Многие языки высокого уровня тоже в основном предполагают решение задач императивно.

**Продукционное программирование**

Языки продукционного программирования (Rule-Oriented Programming Languages) - класс декларативных языков программирования, основанных на подходе к программированию, при котором программа задается совокупностью правил (продукций) без явного указания последовательности их применения. Правила содержат либо условие и действия, которые должны быть выполнены в случае истинности этого условия, либо условие и совокупность других условий, достаточных для истинности этого условия.

Д.Грис пишет[[1]](#footnote-1): "Продукционный язык — это язык программирования, предназначенный для написания синтаксических распознавателей. Программа в основном состоит из последовательности продукций. Употребление здесь этого слова надо считать довольно неудачным, так как обычно с этим словом связывают другой смысл, обозначая им правила подстановки в грамматике. Эти новые продукции лучше было бы назвать редукциями, так как они (обычно, но не всегда) используются для приведения предложений к начальному символу грамматики. Не станем, однако, изобретать новый термин и будем придерживаться слова "продукция", которое постоянно употребляется в литературе. Продукционный язык обычно входит в состав системы построения трансляторов (т.е. системы, предназначенной для реализации таких программ, как ассемблеры и компиляторы.

# **Среда разработки (IDE). Виды и работа с IDE**

Интегрированная среда разработки, ИСP (англ. integrated development environment — IDE), также единая среда разработки, ЕСР — комплекс программных средств, используемый программистами для разработки программного обеспечения (ПО).

Среда разработки включает в себя:

* текстовый редактор;
* транслятор (компилятор и/или интерпретатор);
* средства автоматизации сборки.

Иногда содержит также средства для интеграции с системами управления версиями и разнообразные инструменты для упрощения конструирования графического интерфейса пользователя. Многие современные среды разработки также включают браузер классов, инспектор объектов и диаграмму иерархии классов для использования при объектно-ориентированной разработке ПО.

Рассмотрим несколько сред разработки, позволяющих свободно работать с языком программирования Python: стандартная Python IDLE, специализированная IntelliJ IDEA PyCharm и универсальная Visual Studio Code.

## **Python IDLE**

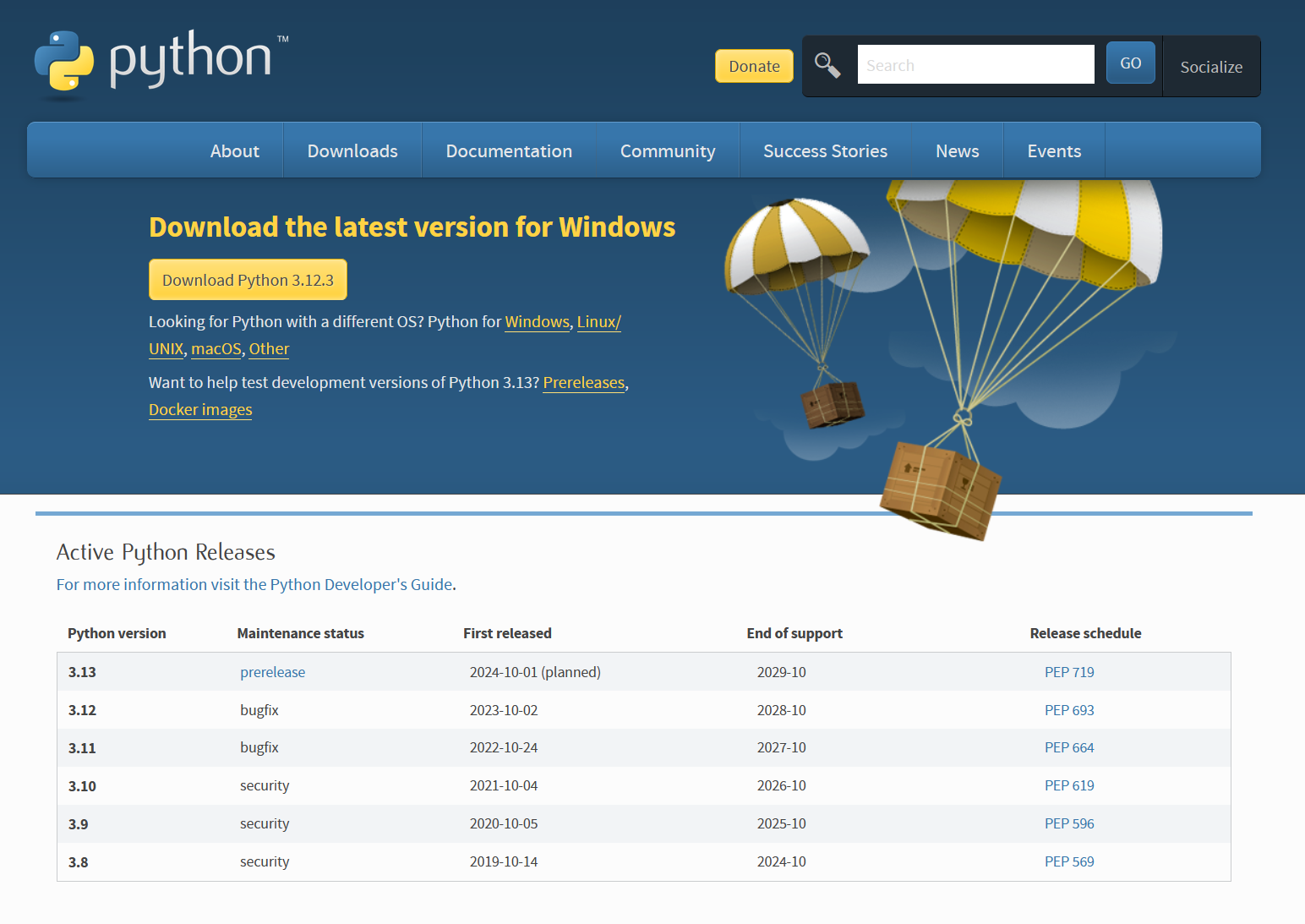


Рис. 2. Сайт python.org

После того, как вы скачали (рис. 2) и установили язык Python на компьютер, настало время запустить его рабочую среду. Рассмотрим её запуск в операционной системе Windows 10. После установки языка Python в меню «Пуск», в списке программ, появится папка «Python» с указанием версии. Из этой папки нам нужно запустить файл «IDLE». Таким образом, мы запустили графический интерфейс среды разработки языка Python.

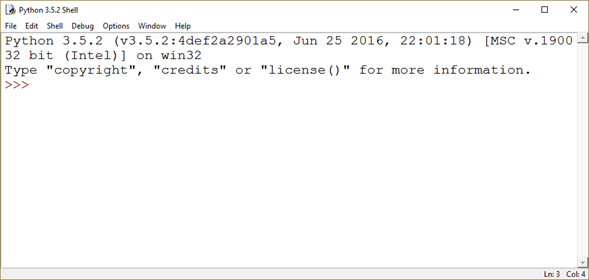


Рис. 3. Окно IDLE

Как видим (рис. 3), окно среды разработки представляет собой поле ввода текста, которое занимает большую часть площади окна, а также меню. В поле ввода текста изначально записано несколько строк, в которых указана текущая версия языка, дата её публикации, а также приглашение на ввод. Так как мы ещё не знаем ни одной команды языка, мы можем сначала использовать его как калькулятор. Для того, чтобы Python рассчитал результат математического выражения, достаточно ввести его в текстовом поле и нажать клавишу «Enter». Попробуем таким образом рассчитать, чему будет равно значение выражения (2^10 – 1) / 3. Возведение в степень обозначается «\*\*». После нажатия клавиши «Enter», на экран сразу же был выведен результат – 341 (рис. 4).

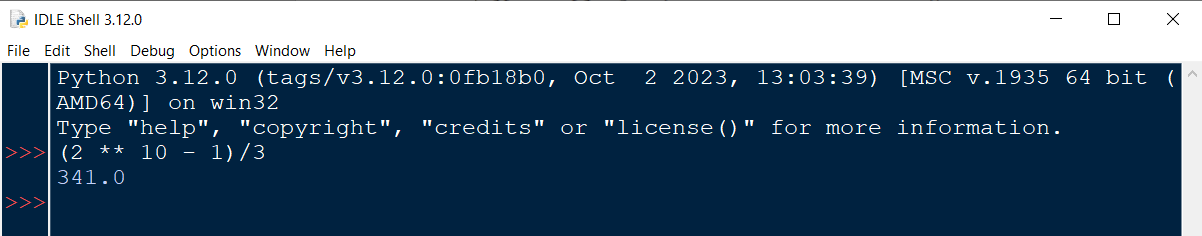


Рис. 4. Пример работы простейшей программы в окне IDLE

Значение выражения вычисляется сразу, потому что Python относится к интерпретируемым языкам программирования. То есть, записываемые на нём команды при каждом выполнении построчно переводятся в двоичный код и выполняются сразу после перевода, а среда разработки языка Python по умолчанию работает в интерактивном режиме. В ней все команды запускаются на выполнение сразу после ввода.

Однако такой метод разработки не является удобным при написании многофайловых объемных программ и проектов.

## **JetBrains PyCharm**

Более удобным с точки зрения рефакторинга и написания кода является среда разработки от компании JetBrains – PyCharm. В данной среде удобно размещена навигация по проекту, автоматическая проверка ошибок – пропущенные отступы, опечатки, вызов неопределенных переменных и функций, работа с системами контроля версий прямо из самой среды разработки (рис. 5).

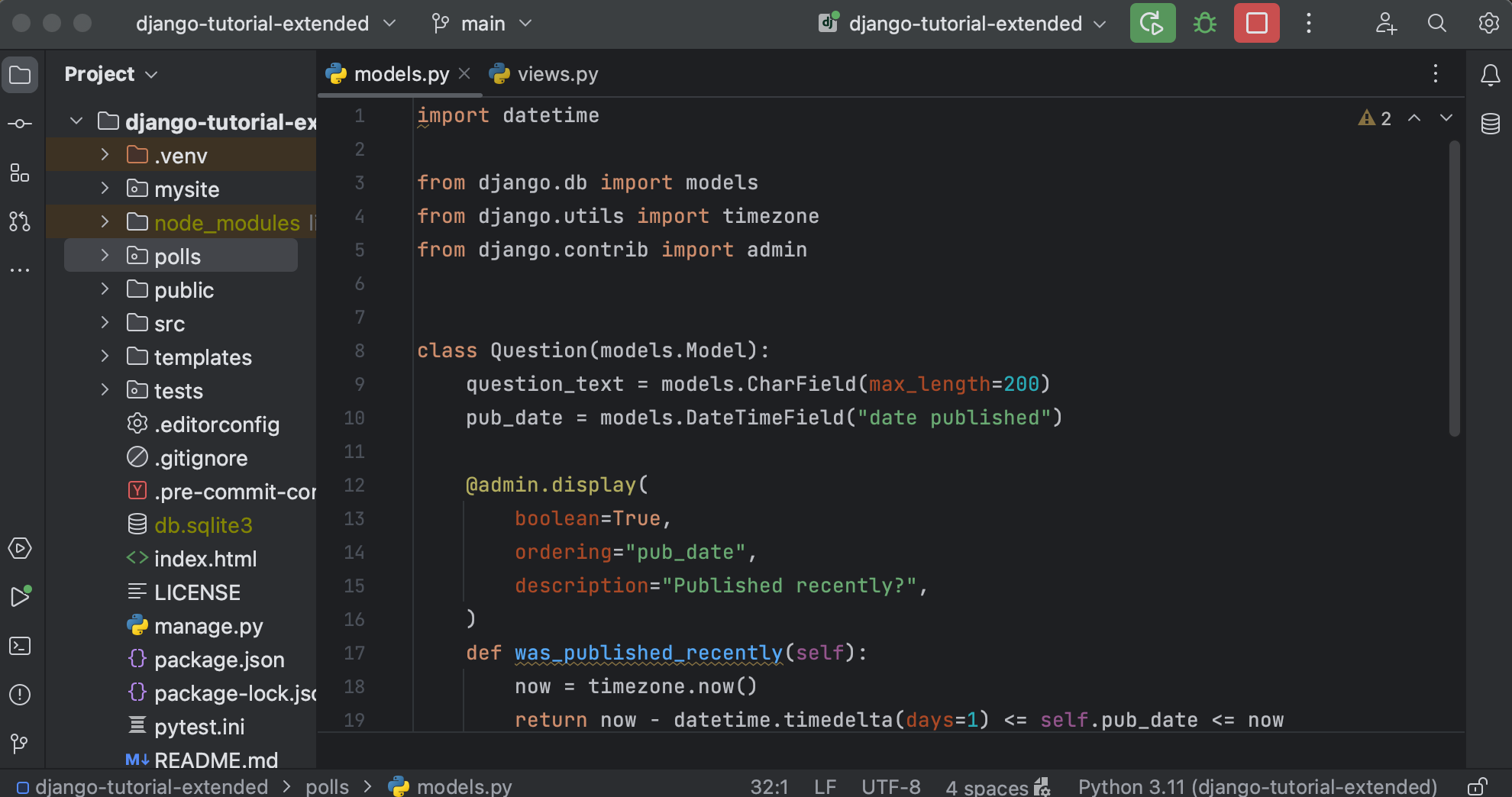


Рис. 5. Окно PyCharm

Также в PyCharm находится терминал, через который можно взаимодействовать с компьютером, не вызывая дополнительно командную строку Windows или терминал Linux. С помощью встроенного терминала осуществляется и автоматическая установка дополнительных пакетов.

Поскольку современная разработка приложений очень редко обходится без графического интерфейса, то в PyCharm присутствует возможность работать и с языками веб‑программирования: HTML, CSS, JavaScript. Это очень сильно упрощает работу с проектами, где разрабатывается веб‑интерфейс или графический пользовательский интерфейс.

Одним из недостатков программы можно отметить платное распространение версии Professional. Однако можно бесплатно воспользоваться версией Community, функционал которой несколько ограничен по сравнению с профессиональной версией, но всё равно поддерживает работу со всеми необходимыми языками, позволяет работать с системами контроля версий и удобно рефакторить написанный код.

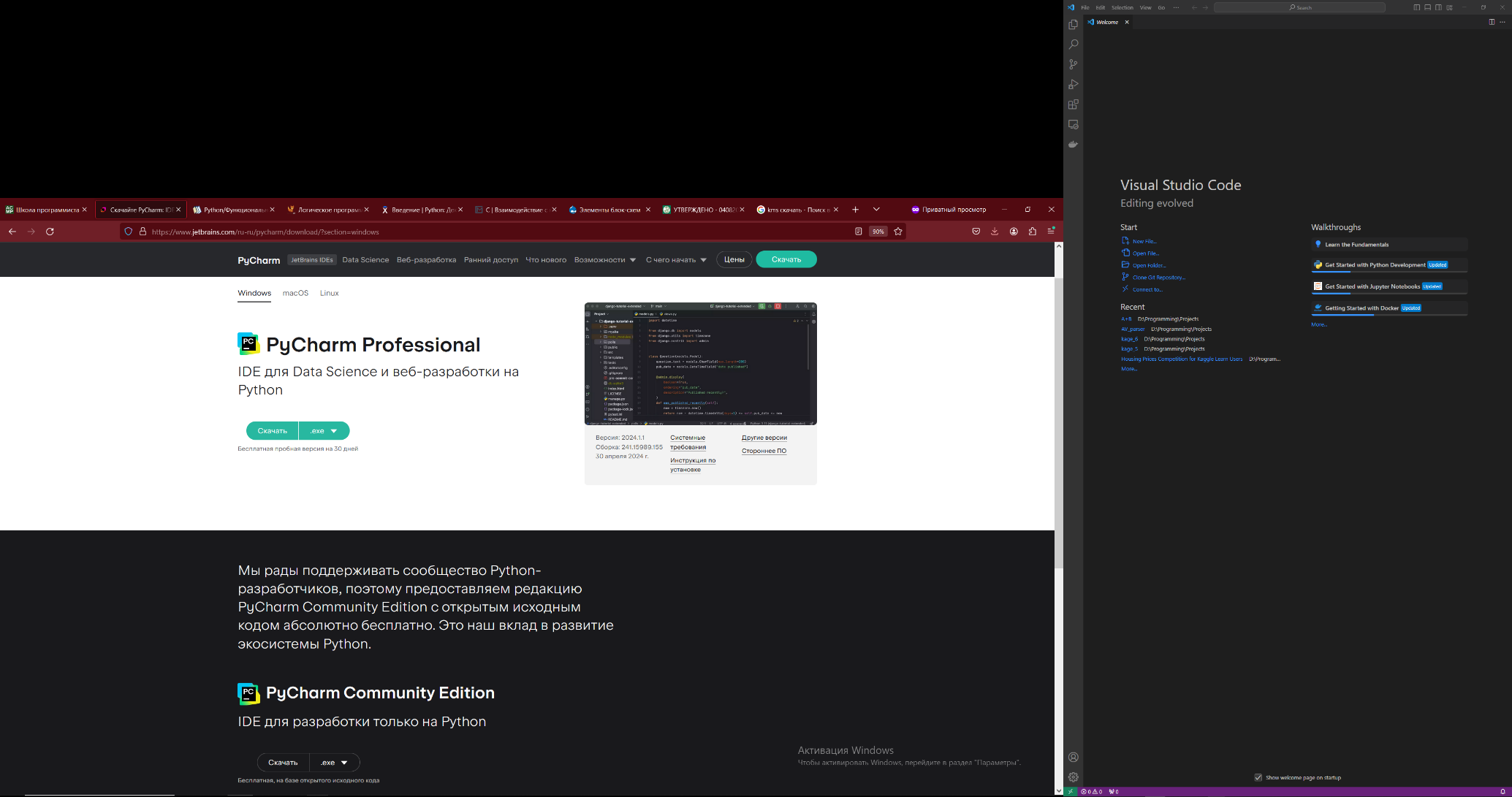


Рис. 6. Окно загрузки PyCharm на сайте JetBrains

## **Visual Studio Code**

Функционал Visual Studio Code похож на PyCharm (рис. 7), однако данная среда разработки мультиязычная – поддерживает возможность работы с большим множеством языком программирования. Отличительной особенностью VS Code является наличия менеджера дополнений – можно из магазина дополнений доустановить требуемые аддоны для работы с конкретной задачей.

С точки зрения удобства работы и требовательности к мощностям компьютера наиболее оптимальным решением является именно VS Code – он требует меньше ресурсов чем PyCharm и гораздо удобнее, чем стандартная IDLE. Распространяется данная среда разработки бесплатно, доступна для скачивания на официальном сайте (рис. 8).

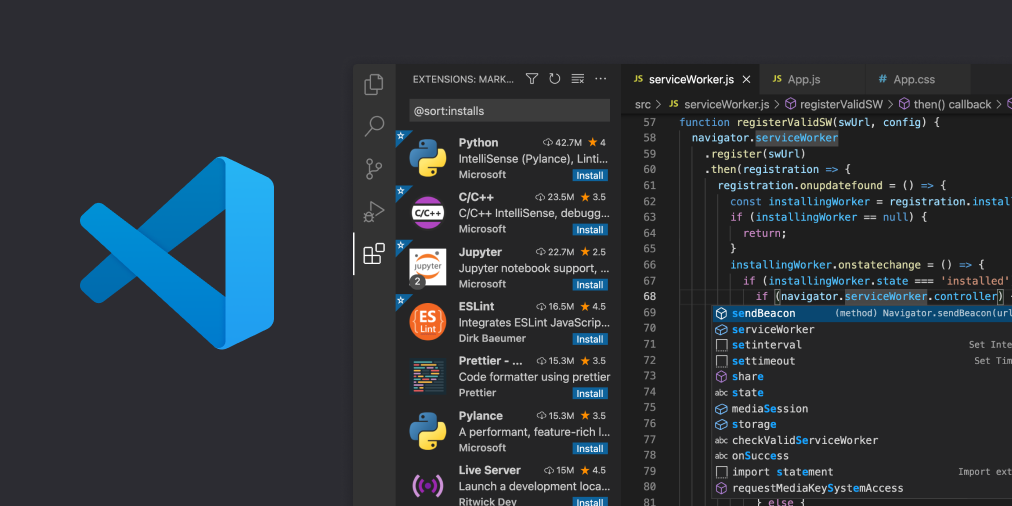


Рис. 7. Окно VS Code

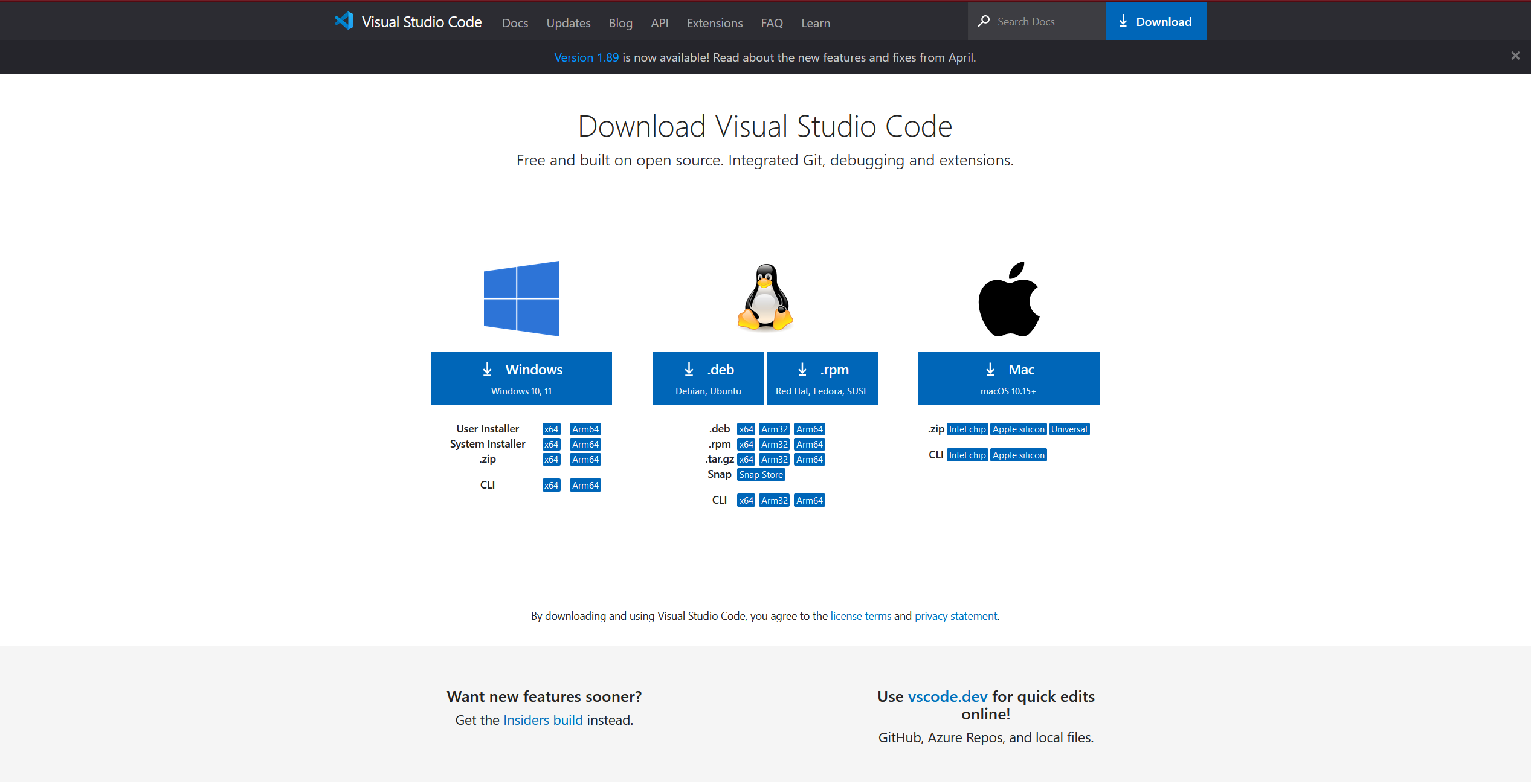


Рис. 8. Окно загрузки VS Code с официального сайта

Все рассмотренные среды разработки являются кроссплатформенными и поддерживаются всеми наиболее распространенными операционными системами.

# **Структура программ**

Для написания кода любой программы необходимо составить алгоритм, согласно которому программа будет работать. А для визуализации любого алгоритма используют блок‑схемы.

**Алгоритм** — это последовательность шагов, ведущая к решению задачи.

Любой алгоритм нужно наделять определенными свойствами. Наиболее важную роль играют:

* Дискретность. Общая задача разделяется на отдельные, последовательно выполняемые шаги. Это, как правило, простейшие действия, порядок которых строго определен. Каждый шаг представляет собой инструкцию или команду, выполнение которой должно начинаться только тогда, когда выполнена предыдущая команда в этой последовательности.
* Конечность. Количество шагов должно быть конечным. Иными словами, результат необходимо получить по завершении строго определенного числа команд в алгоритме.
* Понятность. Все шаги необходимо формулировать так, чтобы исполнитель их полностью понимал. То есть алгоритм должен состоять только из команд, входящих в систему компетенций данного человека.
* Детерминированность. Каждую команду, а также порядок выполнения всех команд необходимо предварительно четко и однозначно определить. При этом на результат выполнения каждого шага не должна влиять никакая сторонняя информация. Необходимо выстроить команды так, чтобы конечная цель алгоритма обеспечивалась только формальным выполнением четких и последовательных инструкций. Исполнителю при этом не нужно вникать в смысл команд. Благодаря такому подходу итоговый результат будет всегда одинаковым и предсказуемым независимо от того, кто выполняет эту работу.
* Массовость. Один алгоритм призван решать сразу комплекс однотипных задач, определяемый заданным диапазоном входной информации.

Алгоритмы могут быть представлены в нескольких формах:

* Текстовая запись. Команды записываются обычным текстом на каком-либо языке. Порядок выполнения определяется нумерацией. Действия описываются произвольно и максимально конкретно.
* Блок–схема. Команды отображаются графически, в виде блок-схемы из геометрических фигур.
* Алгоритмические языки. Алгоритм строится с использованием специального искусственного языка, состоящего из определенной системы обозначений.
* Псевдокод. Обычный язык комбинируется с алгоритмическим. Последний берется за основу для описания базовых структур алгоритма.

**Блок-схема** — это схематическое изображение, иллюстрирующее последовательность операторов, которые необходимо выполнить для достижения результата.

Блок-схемы широко используются при написании программ, так как они:

* Гораздо проще для понимания, чем запись в виде команд.
* Упрощают процесс отладки.
* Позволяют составить эффективную программную документацию.
* Облегчают процесс демонстрации и обсуждения программы.

Существует несколько видов блок-схем алгоритма, которые отличаются в числе прочего сложностью исполнения, начиная от простейших и нарисованных от руки эскизов, заканчивая сложными, спроектированными на компьютере диаграммами. Блок-схемы, учитывая многообразие всех вариаций, могут быть использованы в самых разных областях жизнедеятельности и, соответственно, по-разному называться.

Так, в зависимости от отрасли, встречаются схемы процессов, функциональные блок-схемы, модели и нотации бизнес-процессов, схемы технологических процессов. Все это имеет тесную связь с другими популярными разновидностями схем (например, с некоторыми диаграммами).

Каждая блок-схема состоит из элементов (рис. 9). Обязательными элементами любой блок-схемы являются начало и окончание алгоритма.



Рис. 9. Элементы блок-схемы

**Правила составления блок-схемы**

Чтобы составить блок-схему алгоритма грамотно, необходимо следовать приведенным ниже принципам.

* Начало и конец схемы обязательно ограничиваются соответствующими блоками в одном экземпляре.
* Начальный блок должен быть соединен с конечным линиями связи.
* Линии потока необходимо рисовать из всех блоков, кроме конечного.
* Все блоки нумеруются по порядку слева направо и сверху вниз. Номера ставятся в верхнем левом углу с разрывом начертания.
* Между всеми блоками обеспечивается взаимная связь через линии, определяющие последовательность выполнения команд. Движение потока в обратном порядке от принятого по умолчанию обязательно обозначается стрелками.
* Используемые в схеме линии могут быть входящими или выходящими. Это разделение относительное. Для одного линия, выходящая из одного блока, для другого уже будет являться входящей.
* Начальный блок имеет лишь выходящие линии потока. Соответственно, в конечный блок линии могут только входить.
* Поскольку движение потока идет сверху вниз, входящие линии принято изображать сверху от блока, а выходящие — снизу. Это в целом упрощает чтение блок-схемы.
* Линии потока могут обрываться. При этом места разрывов необходимо помечать специальными соединительными элементами.
* Чтобы блок-схема легче читалась, допускается описательную часть выносить в комментарии.

Рассмотрим простую задачу по началу работы за компьютером:

* включить системный блок и монитор в розетку;
* нажать кнопку включения на системном блоке и мониторе;
* дождаться загрузки операционной системы;
* ввести пароль;
* начать работать.

Теперь представим этот набор действий в виде блок‑схемы (рис. 10):

Начало алгоритма

Включить системный блок и монитор в розетку

Конец алгоритма

Нажать кнопку включения на системном блоке и мониторе

Дождаться загрузки операционной системы

Начать работать

Ввести пароль пользователя

Рис. 10. Блок-схема процесса начала работы за компьютером

В ходе изучения дисциплины блок‑схемы будут применяться для изучения всех тем и разделов, также будет увеличиваться сложность и размер блок‑схем.

# **Простой ввод-вывод. Комментарии**

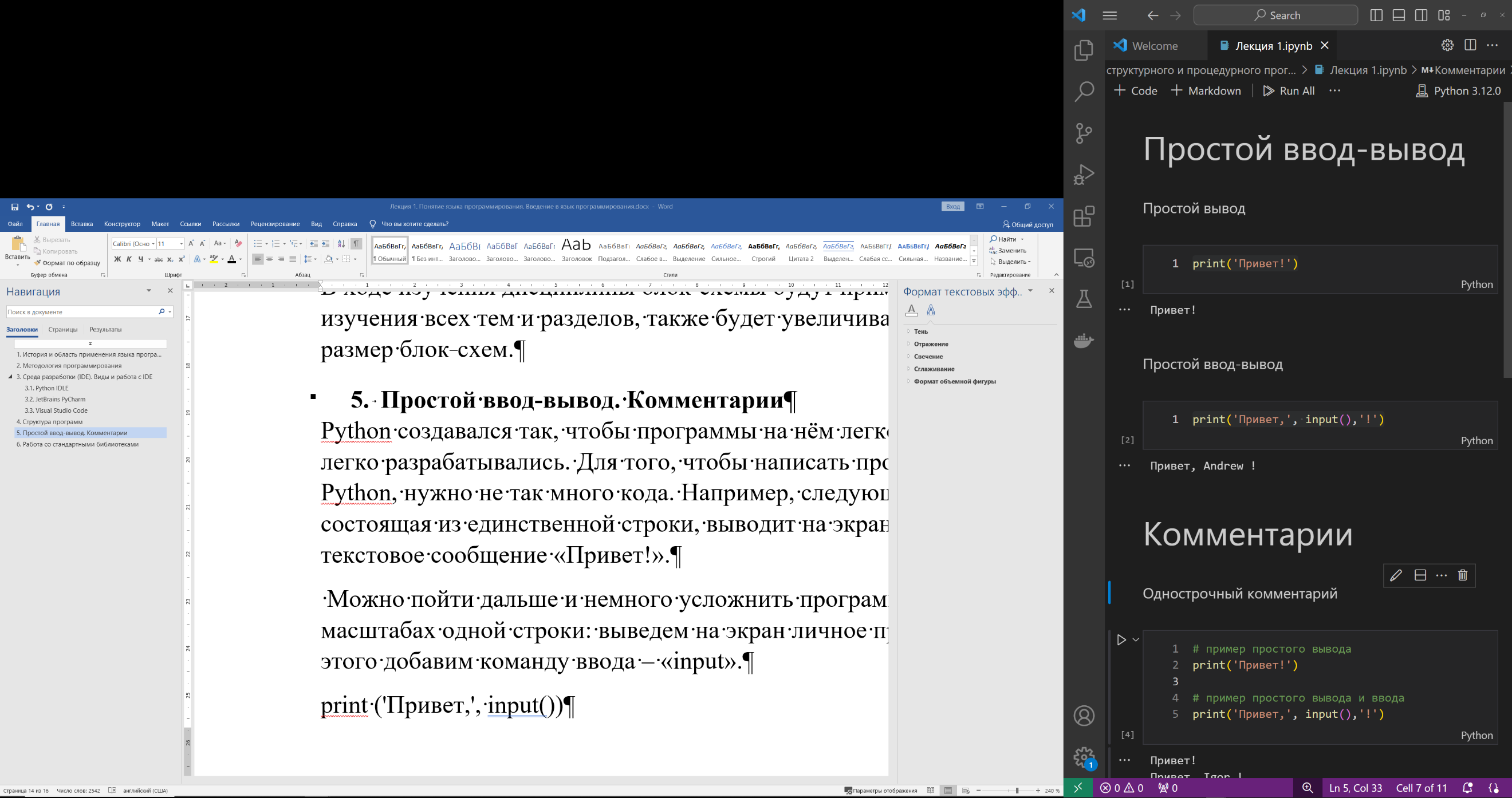
Python создавался так, чтобы программы на нём легко читались и также легко разрабатывались. Для того, чтобы написать программу на языке Python, нужно не так много кода. Например, следующая программа, состоящая из единственной строки (рис. 11), выводит на экран компьютера текстовое сообщение «Привет!».

Рис. 11. Пример простого вывода

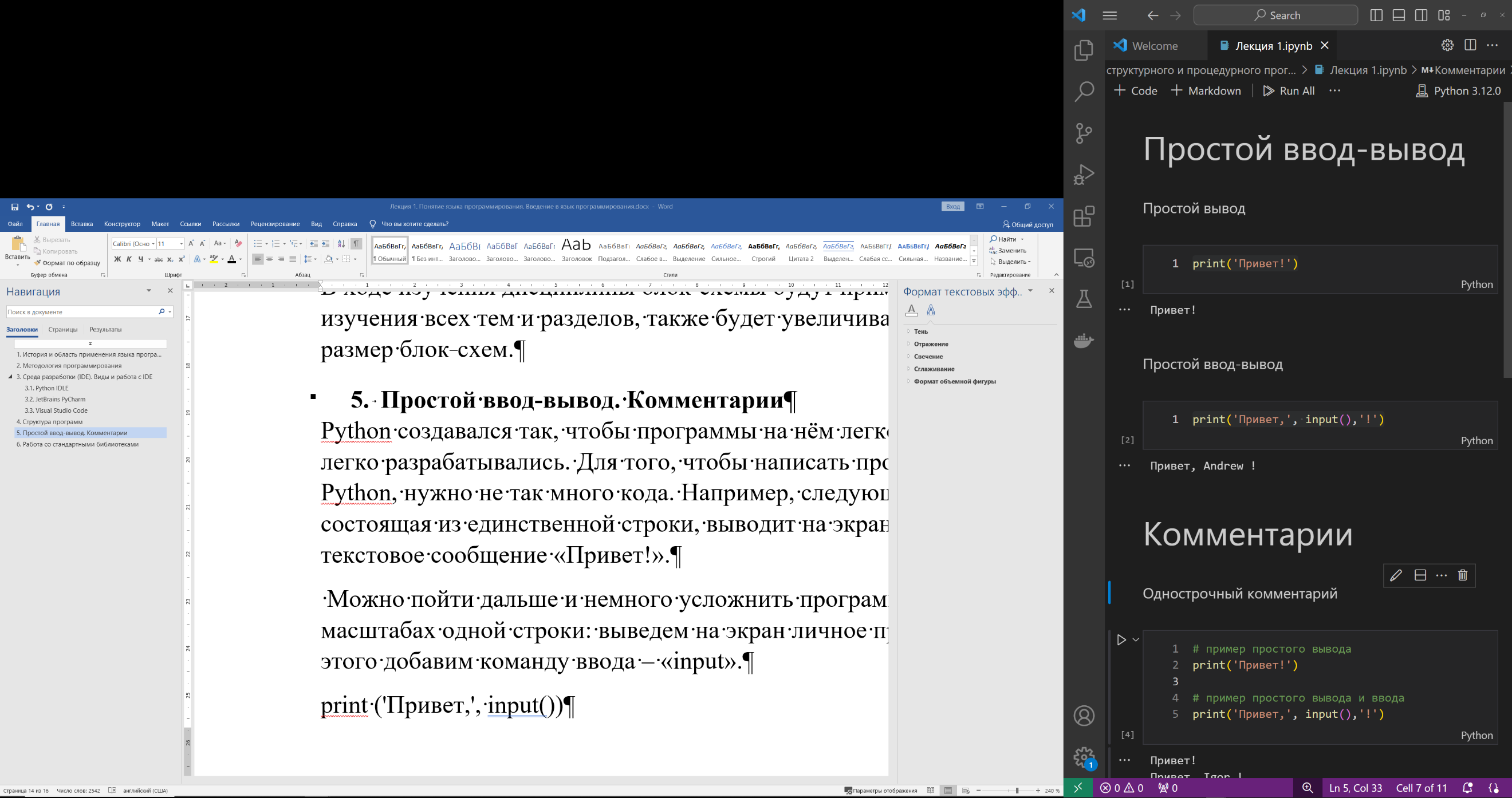
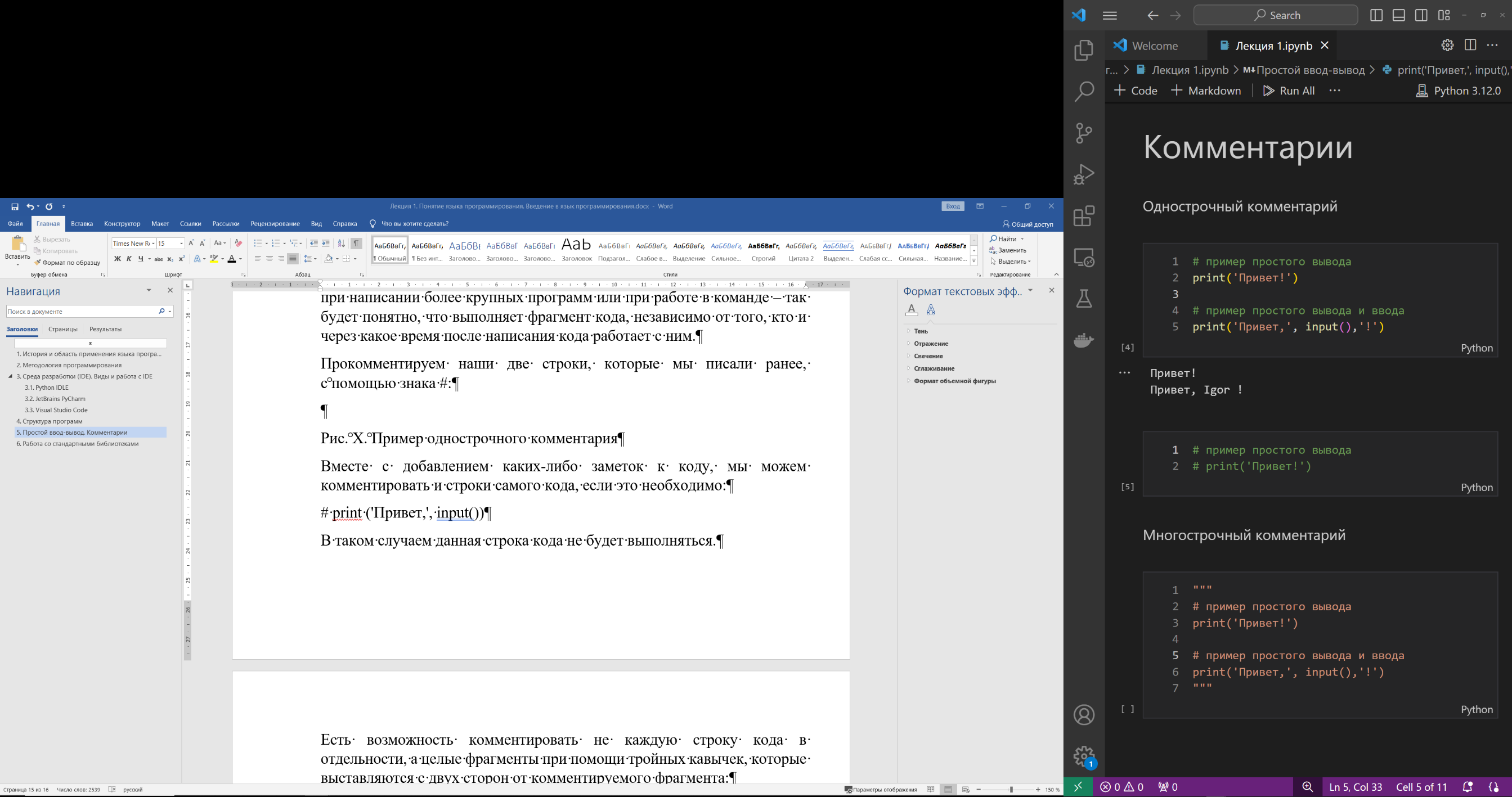
Можно пойти дальше и немного усложнить программу, но все еще в масштабах одной строки: выведем на экран личное приветствие (рис. 12). Для этого добавим команду ввода – «input».

Рис. 12. Пример простого ввода‑вывода

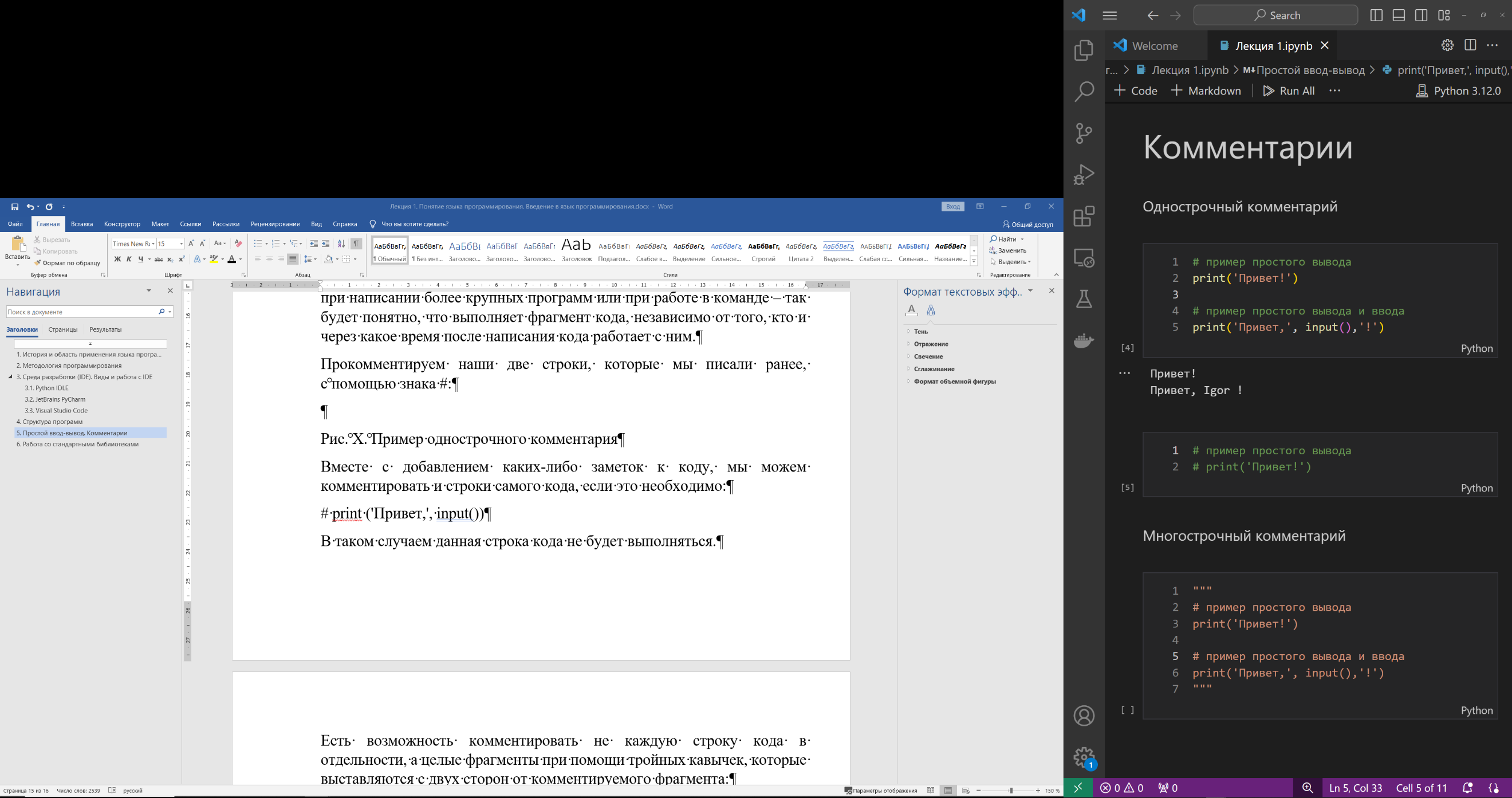
Этими двумя строками мы получили опыт использования двух простейших команд в Python: ввод и вывод. Для их применения не требуется подключения сторонних библиотек – они встроены по умолчанию.

Еще один простейший элемент программирования – комментарии. **Комментарии** — поясняющие строки в программном коде, которые позволяют понять смысл написанного. Они пишутся для людей, но игнорируются компиляторами и интерпретаторами. При первом взгляде может показаться, что комментарии не нужны, но правильнее будет сказать, что они не обязательны, но очень важны. Любая программа будет выполняться без ошибок, даже если в ней не будет содержаться ни одного комментария. Однако правилом хорошего тона считается повсеместное использование комментариев в коде. Это очень полезно при написании более крупных программ или при работе в команде – так будет понятно, что выполняет фрагмент кода, независимо от того, кто и через какое время после написания кода работает с ним.

Прокомментируем наши две строки, которые мы писали ранее, с помощью знака # (рис. 13):

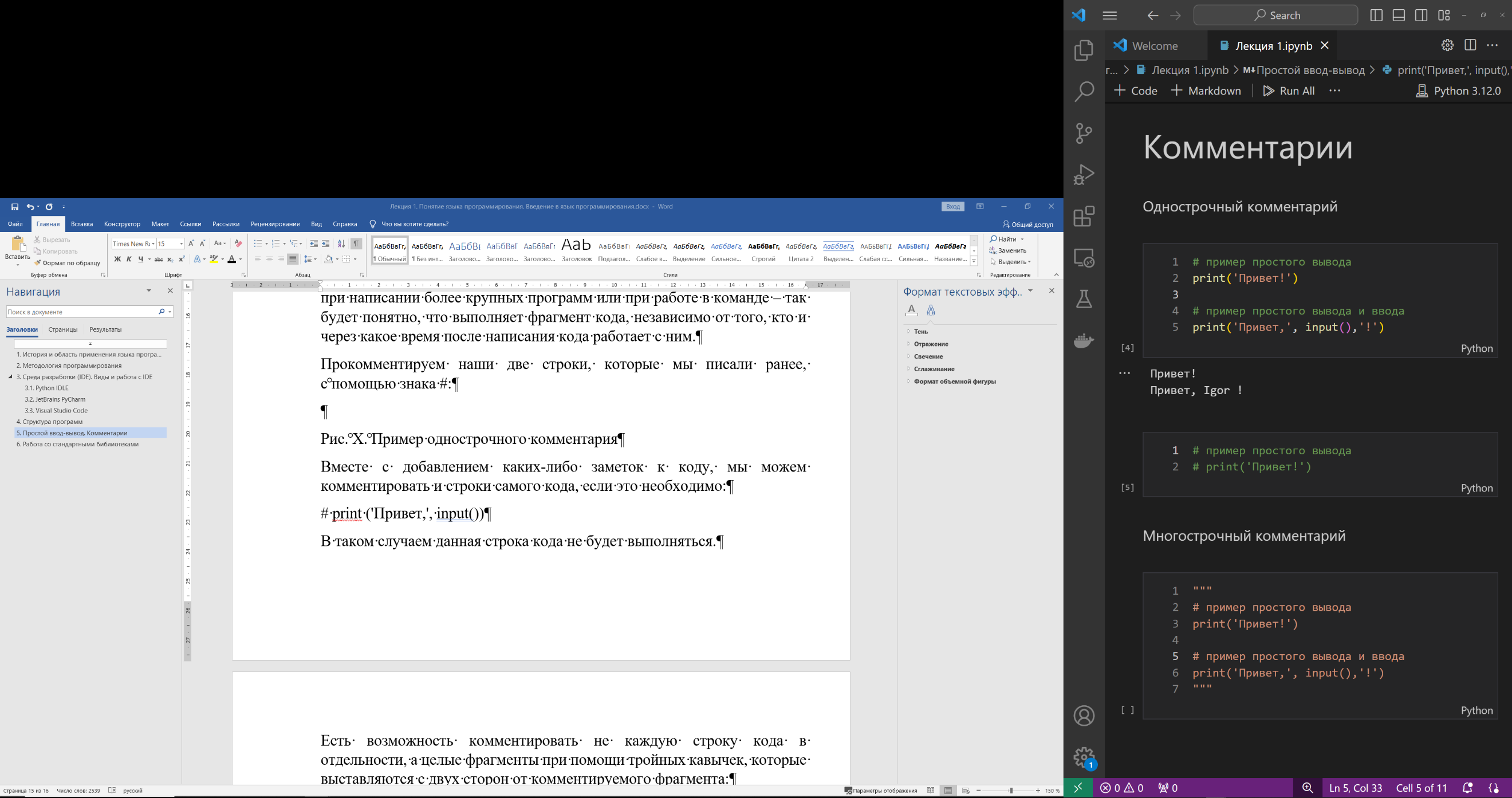
Рис. 13. Пример однострочного комментария

Вместе с добавлением каких-либо заметок к коду, мы можем закомментировать и строки самого кода, если это необходимо (рис. 14):

Рис. 14. Пример комментария строки кода

В таком случаем данная строка кода не будет выполняться.

Есть возможность комментировать не каждую строку кода в отдельности, а целые фрагменты при помощи тройных кавычек, которые выставляются с двух сторон от комментируемого фрагмента (рис. 15):

Рис. 15. Пример комментария строки кода

# **Работа со стандартными библиотеками**

**Модули в Python**

Модуль — это файл, содержащий определения функций, классов и переменных, а также исполняемый код. Модули в Python обычно имеют расширение .py. Они позволяют структурировать код и делить его на мелкие, легко управляемые части.

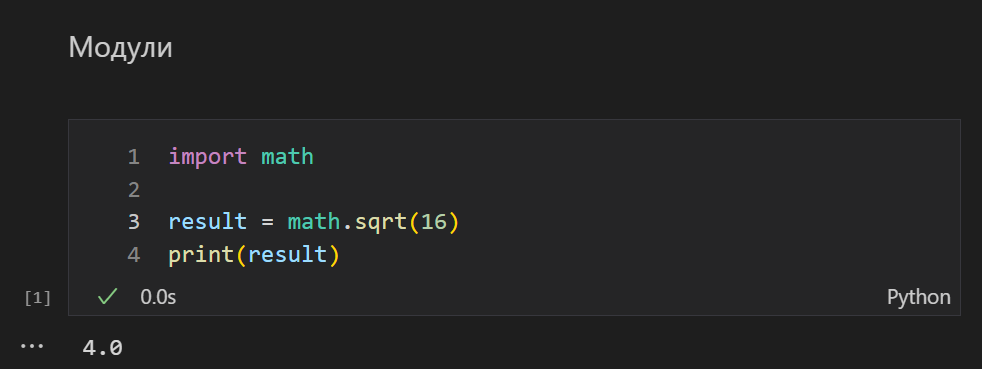
Чтобы использовать модуль, его нужно импортировать с помощью инструкции import. Например, импортируем модуль math, который предоставляет математические функции (рис. 16):

Рис. 16. Пример подключения модуля

**Библиотеки в Python**

Библиотека — это набор связанных модулей, объединенных вместе для решения определенной задачи или предоставления определенного функционала. Библиотеки можно представить как расширение базового функционала Python, которое помогает разработчикам решать различные задачи без необходимости писать все с нуля.

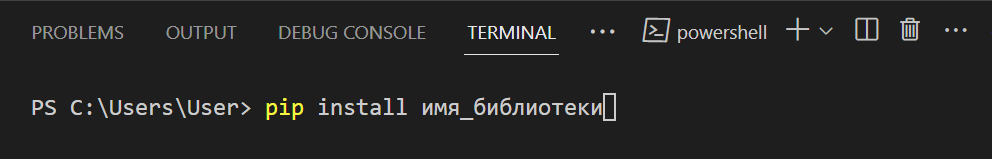
Python имеет множество встроенных библиотек, таких как os, sys, math и других. Кроме того, существует огромное количество сторонних библиотек, доступных для установки и использования. Чтобы установить стороннюю библиотеку, можно использовать инструмент pip из терминала среды разработки или командной строки / powershell (рис. 17):

Рис. 17. Команда для установки библиотеки

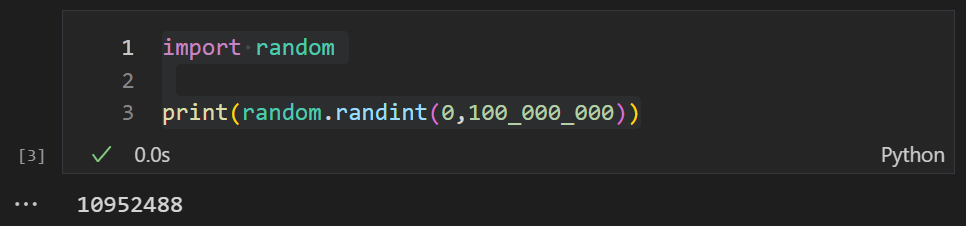
Для импорта библиотеки используется та же инструкция import, что и для модулей. Например, импортируем библиотеку random для генерации псевдослучайных чисел (рис. 18):

Рис. 18. Пример импорта библиотеки

Для того, чтобы использовать конкретный модуль из библиотеки необходимо писать имя библиотеки, а после точки указывать имя модуля. В то же время, в Python поддерживается возможность импортировать конкретный модуль из конкретной библиотекой, не импортируя целую библиотеку (рис. 19):

Рис. 19. Пример подключения модуля из библиотеки

В текущем примере названия библиотек и модулей достаточно короткие, однако встречаются названия библиотек и модулей большой длины, что затрудняет чтение кода при их применении. Для сокращения названий библиотек и модулей используется команда as:

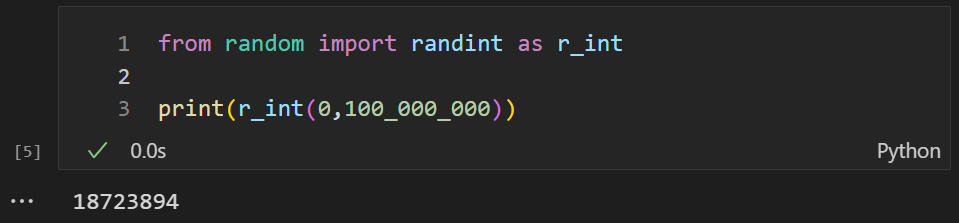


Рис. 20. Пример подключения модуля из библиотеки с изменением имени

# **Заключение**

Язык программирования Python, благодаря своей истории, гибкости и простоте, занял важное место в арсенале современных разработчиков. Его многообразие методологий, поддержка различных парадигм программирования и мощные инструменты делают Python идеальным выбором для самых разнообразных задач — от веб-разработки до научных вычислений и искусственного интеллекта.

В заключение можно сказать, что Python продолжает развиваться и адаптироваться к новым вызовам и потребностям индустрии, оставаясь одним из лидеров в мире разработки программного обеспечения.

# **Список литературы**

1. https://videouroki.net/video/01-vvedenie-v-yazyk-programmirovaniya-python-ego-istoriya-pervyj-zapusk-rabochej-sredy.html

2. https://younglinux.info/python/input

3. https://proglib.io/p/framework-or-library

4. https://habr.com/ru/articles/31180/

5. Грис Д. Конструирование компиляторов для цифровых вычислительных машин. - М.: Мир, 1975. Стр. 367-376, гл.16.

1. [5, с.131] [↑](#footnote-ref-1)