



Бутаков РА

Структура курса языка Python

1. Обзор языка (Интерпретируемые языки и компилируемые. Разные варианты запуска программ на Питон, виртуальное окружение, точка входа в программу.)
2. Встроенные типы данных (Переменные, основные типы данных и структуры данных)
3. Управляющие конструкции (итерируемые объекты)
4. Строки и регулярные выражения
5. Функции (Встроенные функции Python. Процедурное программирование. Понятие функции)
6. Функциональное программирование (Функция как объект. Лямбда-выражения. Генераторы. Итераторы. Области видимости. Замыкания. Декораторы)
7. Объектно-ориентированное программирование
8. Отладка, тестирование и документация
9. Анализ данных (Pandas)
10. Модули и библиотеки (библиотека openpyxl - excel, библиотека matplotlib - графики png, библиотека - reportlab pdf)
11. Многопоточность (threading, multiprocessing)
12. Удаленный вызов процедур (Удаленное взаимодействие процессов и потоков. Клиент-серверное взаимодействие. Сокеты.)
13. Базы данных
14. Программный интерфейс приложения (API)
15. Веб-фреймворк Django

Что такое Python

Python — это язык программирования, который широко используется в интернет-приложениях, разработке программного обеспечения, науке о данных и машинном обучении (ML). Разработчики используют Python, потому что он эффективен, прост в изучении и работает на разных платформах. Программы на языке Python можно скачать бесплатно, они совместимы со всеми типами систем и повышают скорость разработки.

В чем заключаются преимущества языка Python?

- Разработчики могут легко читать и понимать программы на Python, поскольку язык имеет базовый синтаксис, похожий на синтаксис английского.
- Python помогает разработчикам быть более продуктивными, поскольку они могут писать программы на Python, используя меньше строк кода, чем в других языках.
- Python имеет большую стандартную библиотеку, содержащую многократно используемые коды практически для любой задачи. В результате разработчикам не требуется писать код с нуля.
- Разработчики могут легко сочетать Python с другими популярными языками программирования: Java, C и C++.
- Активное сообщество Python состоит из миллионов поддерживающих разработчиков со всего мира. При возникновении проблем сообщество поможет в их решении.
- Кроме того, в Интернете доступно множество полезных ресурсов для изучения Python. Например, вы можете легко найти видеоролики, учебные пособия, документацию и руководства для разработчиков.
- Python можно переносить на различные операционные системы: Windows, macOS, Linux и Unix.

Где применяется Python?

Язык Python имеет несколько стандартных примеров использования при разработке приложений, в числе которых:

- Веб-разработка на стороне сервера
- Автоматизация с помощью скриптов Python
- Наука о данных и машинное обучение
- Разработка программного обеспечения
- Автоматизация тестирования программного обеспечения

Каковы особенности Python?

Интерпретируемый язык

Python является интерпретируемым языком, то есть он выполняет код построчно. Если в коде программы присутствуют ошибки, она перестает работать. Это позволяет программистам быстро найти ошибки в коде.

Простой в использовании язык

Python использует слова, подобные словам английского языка. В отличие от других языков программирования, в Python не используются фигурные скобки. Вместо них применяется отступ.

Язык с динамической типизацией

Программистам не нужно объявлять типы переменных при написании кода, потому что Python определяет их во время выполнения. Эта функция позволяет писать программы на Python значительно быстрее.

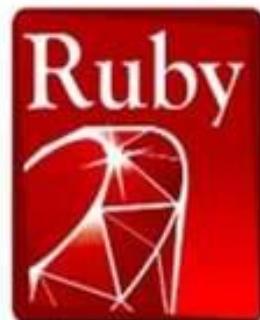
Язык высокого уровня

Python ближе к естественным языкам, чем ряд других языков программирования. Благодаря этому программистам не нужно беспокоиться о его базовой функциональности, например об архитектуре и управлении памятью.

Объектно-ориентированный язык

Python рассматривает все элементы как объекты, но также поддерживает другие типы программирования (например, структурное и функциональное программирование).

Краткая история языков программирования



C#



C++



Objective-C



Perl



=GO

JavaScript



THE
PROGRAMMING
LANGUAGE



1940-50-е годы: от Ассемблера к Fortran и AGOL

- В конце сороковых - начале пятидесятых стал применяться **Ассемблер**. В нем для обозначения объектов и команд использовались буквы или укороченные слова (например, add вместо 000010). Это был язык низкого уровня, то есть созданный для конкретного типа процессора. Он значительно упростил труд программистов.
- Следом появились так называемые языки высокого уровня (или машинонезависимые). Первым из них стал **Shortcode** (1949). В 1950 Уильям Шмитт адаптировал его для компьютера **UNIVAC**.
- С 1954 по 1957 в IBM под руководством Джона Бэкуса (1924-2007) был разработан знаменитый **Fortran** (от formula translator). Он использовался для технических расчетов и научных задач. **Fortran** стал первым относительно массовым языком программирования - к 1963 году существовало уже 40 компиляторов для различных машин.
- В 1967 году был создан компилятор для **ЭВМ "Минск-2"**, в 1968 - для **БЭСМ-6**. Язык используется до сих пор - в основном, для сложных вычислений.
- **ALGOL** (1958), "дедушка" **Java, Pascal** и **C++**. Как и **Fortran**, **ALGOL** распространился по миру, но в Европе и СССР был встречен гораздо теплее, чем в США. Язык отличался блочной структурой, что для того времени было прогрессивным решением, а затем стало стандартом.

1960-70-е: от структурного программирования к объектно-ориентированному

- Первая "звезда" шестидесятых - **BASIC** (он же Beginner's All-purpose Symbolic Instruction Code, универсальный код символьических инструкций для начинающих), созданный Томасом Курцем и Джоном Кемени, профессорами Дартмутского колледжа в 1964 году.
- Первые ООП-языки - это **Simula** (1967) и основанный на нем **Smalltalk** (1972). Последний привнес в программирование множество идей и концептов, актуальных по сей день: взаимодействие между элементами через сообщения, возможность редактировать код "на ходу" и динамическая типизация. Logitech и ряд других компаний и сейчас используют Smalltalk для отдельных операций.
- В 1972 появились **C** и **SQL** (первоначально назывался **SEQUEL**). **C (прапородитель C#, Java, Python и других)**, был создан Деннисом Ричи, сотрудником Bell Telephone Laboratories для работы с Unix. **SQL** - детище специалистов IBM Бойса и Чемберлена, ориентирован на работу с информацией из баз данных. Оба до сих пор используются.

1980-1990-е: мультипарадигмальность, визуальные языки, Интернет

- В начале 1980-х сотрудник Bell Labs Бьерн Страуструп решил улучшить язык **C** и добавил к нему ряд возможностей **Simula** (1967). Так появился **C++**, объединивший в себе черты объектно-ориентированных и системных языков. Страуструп внедрил в C возможность работать с объектами и классами, строгую проверку типов, аргументы по умолчанию и т.д. Первые версии языка (1980) назывались "Си с классами", а наименование C++ стало использоваться в 1985 году с выходом очередной версии.
- Еще одной важной вехой стало появление объектно-ориентированного языка программирования **Ada** (1980/81), названного в честь Ады Лавлейс. Его разработала команда Жана Ишбия по заказу Министерства обороны США. В основу языка легли **Pascal (1970)** и **Algol (1960)**, однако присущий им синтаксис был упорядочен. "Ада" создавался для военных и смежных задач - прежде всего для систем управления кораблями и самолетами.
- **Perl (1987)**, разработанный для редактирования текстов, в наши дни применяется в самых разных целях - от системного администрирования до работы с базами данных.

1980-1990-е: мультипарадигмальность, визуальные языки, Интернет

- **Python** (1991), созданный голландцем Гвидо ван Россумом, был назван в честь знаменитой комедийной группы из Великобритании "Монти Пайтон". Сейчас это универсальный язык, широко известный и удобный. Ruby Юкихиро Мацумото тоже актуален по сей день и используется для веб-приложений.
- В 1995 был выпущен **PHP** - С-подобный язык для разработки интернет-страниц и веб-приложений (активно применяется и сейчас, в том числе WordPress и Wikipedia)
- В том же году вышла первая версия популярного **JavaScript** Брэндана Эйха, также применяющегося для динамической веб-разработки, браузеров и виджетов.
- Джеймс Гослинг выпустил один из самых популярных языков наших дней **Java** (1995), который не стоит путать с JavaScript.. Сейчас он часто используется для создания Android-приложений и веб-сервисов. И конкурирует с **Kotlin** (2011), о котором мы еще расскажем.

От нулевых до наших дней

- Первый "хит" нового тысячелетия - это **C#** (2001), разработанный в Microsoft Андерсоном Хейлсбергом, создателем **Delphi**. Это С-подобный язык, синтаксически близкий к **C++** и **Java**, взявший многое от Delphi, Modula и Smalltalk. C# активно используется самой Microsoft, применяется при разработке игр на Unity и веб-разработке.
- 2003 свет увидел **Scala**, созданный в Швейцарии под руководством Мартина Одерски. Его "фишкой" стала масштабируемость, а также объединение объектно-ориентированного и функционального программирования. Ближайшие "родственники" языка - это Java и C++.
- "Ровесник" **Scala** - язык **Groovy**, созданный для платформы **Java**. Синтаксически он близок Java, но имеет ряд отличий.
- В нулевые и десятие набирала силу **Google**, не оставшаяся в стороне от большой игры - в ноябре 2009 компания представила язык **Go** (также известен как Golang - не путать с языком Go!). Продукт Google разрабатывался как относительно простая замена **C** и **C++**, которая сможет эффективно работать на многоядерных процессорах и распределенных системах. Go не стал "прорывом", но вошел в число популярных современных языков

От нулевых до наших дней

- Российские программисты тоже внесли вклад в развитие языков нулевых-десятых. С 2010 по 2011 компанией **JetBrains** (под руководством Андрея Бреслава) был разработан **Kotlin**. Это объектно-ориентированный язык, функционирующий на базе **Java Virtual Machine**, создававшийся как более лаконичная и простая альтернатива **Java** и **Scala**. Как и **Java**, он часто используется **Android-разработчиками**.
- **Swift** (2014) создавался **Apple** как более простая и эффективная замена **C**, **C++** и **Objective-C**. Это универсальный язык, на котором прежде всего пишут продукты для **macOS**, **iOS** и **других систем корпорации**. Иногда может использоваться для поддержки сайтов и веб-приложений.

Классификация ЯП

- Низкоуровневые языки
 - Высокоуровневые языки
-
- Компилируемые
 - Интерпретируемые
-
- Алгоритмические и Языки описания данных
 - Универсальные и специализированные
-
- Структурные языки
 - Объектно-Ориентированные языки

Низкоуровневые и Высокоуровневые

Низкоуровневые языки

Языки программирования низкого уровня (низкоуровневые) традиционно появились первыми и в последующем стали базисом для развития всей ИТ индустрии. Они так называются потому, что в своих командах обращаются фактически напрямую к железу компьютера, его микропроцессору. Каждый процессор способен воспринимать определенный набор понятных только ему команд, поэтому к каждой модели немногочисленных устройств, в то далекое время, создавались свои языки. Изначально такие языки имели минимальный набор команд, и, в отличие от высокоуровневых, не имели такого большого количества абстрактных классов, разнообразного синтаксиса.

Языки программирование низкого уровня, либо имеющие их возможности, активно применяются также и сейчас. Ведь только благодаря им возможно писать драйвера для компьютерного железа и подключаемых периферийных устройств, создавать операционные системы и ядра прошивок, а также решать многие другие важнейшие задачи.

Ассемблер используют разработчики микроконтроллеров и драйверов — те, кто работает с железом.

Язык ассемблера (Assembly, или ASM) — это язык программирования, который используют, чтобы написать программы для аппаратных платформ или архитектур.

В отличие от языков программирования высокого уровня, например Python или Java, язык ассемблера обеспечивает прямое представление инструкций машинного кода. Поэтому язык ассемблера называют языком низкого уровня: он ближе к двоичному коду, который понимает компьютер.

Программы на языке ассемблера обычно пишут с комбинациями текстовой мнемоники и числовых кодов, известных как коды операций. Это инструкции: их выполняет процессор. Программы непросто писать и отлаживать из-за их низкоуровневой природы. Зато они дают больший контроль над аппаратным обеспечением компьютера и могут быть более эффективными, чем программы на языках высокого уровня.

Где используют язык ассемблера

Сейчас используют множество различных языков ассемблера, каждый из которых предназначен для конкретной аппаратной платформы или архитектуры.

Примеры:

- x86 для ПК на базе Intel;**
- ARM для мобильных устройств и встроенных систем;**
- MIPS для некоторых встроенных систем и академического использования.**

Ассемблер нужен в областях, где требуется низкоуровневое системное программирование или аппаратное управление:

- Разработка операционной системы.** Язык ассемблера используют при разработке ОС и драйверов устройств, для которых нужен прямой доступ к аппаратным компонентам.
- Разработка вредоносных программ.** Хакеры создают на ассемблере вирусы.

Ассемблер используют в тех областях, где критически важны производительность и аппаратный контроль. Там, где другие языки программирования высокого уровня не отвечают конкретным требованиям приложения.

Высокоуровневые языки

Высокоуровневый язык программирования — это еще более очеловеченный язык. Такие языки созданы «для людей», чтобы людям было легче понимать, что написано в коде. Такие языки вообще не понимает компьютер, а для того, чтобы их понял компьютер, применяются специальные программы — компиляторы и интерпретаторы, которые «конвертируют» высокоуровневый язык программирования в понятный для компьютера набор символов.

Программный код, написанный на высокоуровневом языке, легко читается и понимается разработчиками. Программный код, написанный на низкоуровневом языке, читается и понимается разработчиками намного сложнее. Поэтому «прийти» в низкоуровневый язык новичком намного сложнее, чем в высокоуровневый.

Компилируемые и Интерпретируемые

Языки программирования в общем подходе делятся на два класса — компилируемые и интерпретируемые. Стоит отметить, что эта классификация языков программирования на компилируемые и интерпретируемые, является весьма условной, поскольку для любого языка программирования может быть создан как компилятор, так и интерпретатор.

Мы полагаемся на такие инструменты, как компиляция и интерпретация, чтобы преобразовать наш код в форму, понятную компьютеру. Код может быть исполнен нативно, в операционной системе после конвертации в машинный (путём компиляции), или же исполняться построчно другой программой, которая делает это вместо ОС (интерпретатор).

Компилируемые языки

Программа на компилируемом языке при помощи специальной программы компилятора преобразуется (компилируется) в набор инструкций для данного типа процессора (машинный код) и далее записывается в исполняемый файл, который может быть запущен на выполнение как отдельная программа. Другими словами, компилятор переводит программу с языка высокого уровня на низкоуровневый язык, понятный процессору сразу и целиком, создавая при этом отдельную программу.

Как правило, скомпилированные программы выполняются быстрее и не требуют для выполнения дополнительных программ, так как уже переведены на машинный язык. Вместе с тем при каждом изменении текста программы требуется ее перекомпиляция, что создает трудности при разработке. Кроме того, скомпилированная программа может выполняться только на том же типе компьютеров и, как правило, под той же операционной системой, на которую был рассчитан компилятор. Чтобы создать исполняемый файл для машины другого типа, требуется новая компиляция.

Компилируемые языки обычно позволяют получить более быструю и, возможно, более компактную программу, и поэтому применяются для создания часто используемых программ.

Примерами компилируемых языков являются Pascal, C, C++, Erlang, Haskell, Rust, Go, Ada.

Интерпретируемые языки

Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет (интерпретирует) ее текст без предварительного перевода. При этом программа остается на исходном языке и не может быть запущена без интерпретатора. Можно сказать, что процессор компьютера — это интерпретатор машинного кода. Кратко говоря, интерпретатор переводит на машинный язык прямо во время исполнения программы.

Программы на интерпретируемых языках можно запускать сразу же после изменения, что облегчает разработку. Программа на интерпретируемом языке может быть зачастую запущена на разных типах машин и операционных систем без дополнительных усилий. Однако интерпретуемые программы выполняются заметно медленнее, чем компилируемые, кроме того, они не могут выполняться без дополнительной программы-интерпретатора.

Примерами интерпретуемых языков являются PHP, Perl, Ruby, Python, JavaScript. К интерпретируемым языкам также можно отнести все скриптовые языки.

Многие языки в наши дни имеют как компилируемые, так и интерпретируемые реализации, сводя разницу между ними к минимуму. Некоторые языки, **например, Java и C#**, находятся между компилируемыми и интерпретируемыми. А именно, программа компилируется не в машинный язык, а в машинно-независимый код низкого уровня, байт-код. Далее байт-код выполняется виртуальной машиной. Для выполнения **байт-кода** обычно используется интерпретация, хотя отдельные его части для ускорения работы программы могут быть транслированы в машинный код непосредственно во время выполнения программы по технологии компиляции «на лету». Для Java байт-код исполняется виртуальной машиной Java (**Java Virtual Machine, JVM**), для C# — **Common Language Runtime**.

Универсальные и специализированные

Универсальные языки

Универсальные языки программирования (УЯП) представляют собой языки, которые могут быть использованы для разработки программного обеспечения в различных областях. Они обладают мощными возможностями и гибкостью, что позволяет программистам решать широкий спектр задач.

- **C** является одним из самых влиятельных и широко используемых языков программирования. Он был разработан в 1970-х годах и до сих пор остается одним из наиболее популярных языков. **C** можно использовать для написания операционных систем, компиляторов, встраиваемого программного обеспечения и многих других приложений.
- **Java** является объектно-ориентированным языком программирования, который работает на машине виртуальной **JVM**. Он широко используется для разработки кросс-платформенных приложений, веб-серверов, игр и мобильных приложений на операционной системе **Android**.
- **Python** - это язык программирования, который отличается простотой и читаемостью. Он широко применяется в науке о данных, искусственном интеллекте, веб-разработке и автоматизации. **Python** также является популярным языком для начинающих программистов.
- **C++** является расширением языка **C**, но с поддержкой объектно-ориентированного программирования. Он широко применяется в разработке игр, приложений реального времени, системного программирования и т.д. **C++** также является одним из самых производительных языков программирования.
- **JavaScript** - это язык программирования, который выполняется непосредственно в браузере и используется для создания динамических веб-страниц и интерактивных веб-приложений. Он также может быть использован для серверного программирования на платформе **Node.js**.

Специализированные языки

Специализация в языках программирования касается, как правило, либо предметной области, например, математические вычисления (**Fortran, F#**), искусственный интеллект (**LISP**), веб-разработка (**PERL, PHP**), компьютерные игры (**Unity, Lua**), бухгалтерия (**1С**) и т.д., либо какой-то технологии программирования, например, многопоточность как в языке **Си-Омега (Cw)** или способ записи операторов **как в F#**.

- **Математические вычисления:** Fortran, F#
- **Математическое моделирование:** MatLab, Wolfram (Mathematica)
- **Искусственный интеллект:** LISP
- **На основе передачи сообщений:** Small Talk
- **Многопоточные приложения Cw**
- **Веб-разработка:** Perl, PHP, JavaScript
- **Базы данных:** SQL
- **Компьютерные игры:** Lua, Unity, Godot, Twine
- **Компьютерная графика:** MEL (Maya), MAX Script (3ds Max)
- **Бухгалтерия:** 1С

Алгоритмические и Языки описания данных

Алгоритмические языки

Алгоритмический язык программирования – формальный язык, используемый для записи, реализации и изучения алгоритмов. Всякий язык программирования является алгоритмическим языком, но не всякий алгоритмический язык пригоден для использования в качестве языка программирования.

Алгоритмический язык образуют три его составляющие: алфавит, синтаксис и семантика.

Алфавит – фиксированный для данного языка набор символов (букв, цифр, специальных знаков и т.д.), которые могут быть использованы при написании программы.

Синтаксис – правила построения из символов алфавита специальных конструкций, с помощью которых составляется алгоритм.

Семантика – система правил толкования конструкций языка.

Языки описания данных

Языки же описания данных предназначены только для описания данных для разных типов приложений. Эти языки можно считать необходимой нагрузкой к обычным алгоритмическим языкам. Например, если вы учите JavaScript для разработки веб-приложений, то скорее всего вам придется также изучить и синтаксис каскадных таблиц стилей CSS и язык описания данных JSON, в формате которого удобно передавать данные между веб-сервером и клиентом.

Или, например, язык работы с **базами данных SQL**, по сути, является языком для обработки и получения данных, но также включает в себя раздел Data Definition Language или Язык Описания Данных.

Структурные и
Объектно-Ориентированные языки программирования

Структурные языки

В основе этих языков лежит представление программы в виде иерархической структуры блоков. Любая программа состоит из трёх базовых управляющих структур: последовательность, ветвление, цикл. Рассмотрим основные структурные языки программирования: Basic, Pascal.

Basic

Этот язык программирования был придуман в **1963 году** преподавателями **Дартмутского Колледжа Джоном Кемени и Томасом Куртцом**. Основной задачей этого языка была возможность без опыта программирования создавать программы для своих задач. Сейчас же он превратился в обычный язык программирования с широким набором возможностей.

Преимущества языка Basic:

- Простота в использовании для начинающих.
- Ясные сообщения об ошибках.
- Не требует понимания работы аппаратного обеспечения.

Pascal

Pascal – один из самых известных языков программирования. Он был создан Никлаусом Виртом в 1968—1969 годах и используется для обучения программированию в старших классах школы и в вузах. Это один из первых языков, для которых характерна строгая типизация, потому что, по мнению Вирта, язык должен способствовать дисциплинированному программированию. Поэтому Pascal так активно используют в обучении. К 1980-м годам Паскаль стал основой для многочисленных учебных программ.

Преимущества языка Pascal:

- Лёгкий синтаксис.
- Невысокие аппаратные и системные требования.
- Универсальность.

Объектно-Ориентированные языки

Объектно-ориентированный язык программирования (ОО-язык) — язык, построенный на принципах объектно-ориентированного программирования.

В основе концепции объектно-ориентированного программирования лежит понятие объекта — некой сущности, которая объединяет в себе поля (данные) и методы (выполняемые объектом действия).

Например, объект **человек** может иметь поля **имя**, **фамилия** и методы **есть** и **спать**. Соответственно, в программе можем использовать операторы **Человек.Имя:="Иван"** и **Человек.Есть(пища)**.

Особенности

В современных ОО языках используются механизмы:

- **Наследование.** Создание нового класса объектов путём добавления новых элементов (методов). Некоторые ОО языки позволяют выполнять множественное наследование, то есть объединять в одном классе возможности нескольких других классов.
- **Инкапсуляция.** Скрытие деталей реализации, которое позволяет вносить изменения в части программы безболезненно для других её частей, что существенно упрощает сопровождение и модификацию ПО.
- **Полиморфизм.** При полиморфизме некоторые части (методы) родительского класса заменяются новыми, реализующими специфические для данного потомка действия. Таким образом, интерфейс классов остаётся прежним, а реализация методов с одинаковым названием и набором параметров различается. В ООП обычно применяется полиморфизм подтипов (называемый при этом просто «полиморфизмом»), нередко в форме позднего связывания.

Примеры: C++, Object Pascal, Java, C#

Таким образом, в языках структурного программирования алгоритмы на основе функций стоят как бы на первом месте, а данные для них можно брать откуда угодно. Не последнюю роль в этом сыграла идея автора кибернетики Норберта Винера о функции как о черном ящике, на вход которому можно подавать любые данные и наблюдать получаемый выход.

Для небольших задач типа сортировки данных или нахождения кратчайшего пути структурное программирование подходило идеально. Были найдены решения для большинства сложных алгоритмических задач. Появились фундаментальные труды, такие как многотомник **“Искусство программирования”** Дональда Кнута, который до сих пор считается настольной книгой для программистов.

Однако, увеличение сложности программ в результате привело к появлению и больших шансов на внесение ошибок в программы, так как возможность подставлять любые данные на вход процедурам и функциям влекло за собой побочные эффекты.

После появления объектно-ориентированных языков программирования, таких как **C++, Object Pascal, Java, C#**, а также новых аппаратных возможностей компьютеров, объемы программ и данных для них увеличились многократно, если не на порядки, что легко оценить хотя бы по объемам дистрибутивов программ, которые перестали помещаться сначала на дискеты, а потом и на компакт диски. А программирование снова как бы встало с головы на ноги.

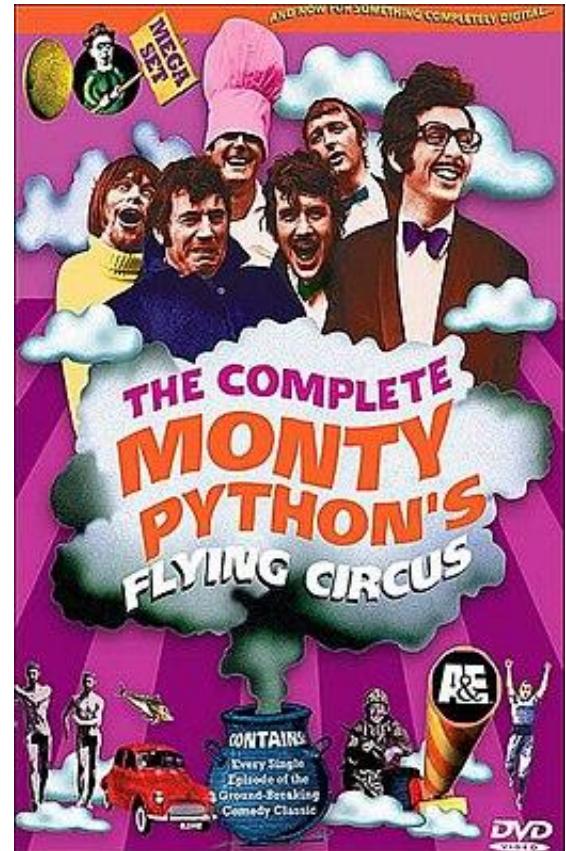
Краткая история Python

Язык **Python** разработал голландский программист **Гвидо Ван Россум** (Guido van Rossum) в 1991 году. Гвидо был фанатом британского комедийного сериала «Monty Python's Flying Circus», откуда и пришло название языка.



Гвидо Ван Россум

Изначально язык был полностью любительским проектом: Ван Россум просто хотел чем-то занять себя на рождественских каникулах.



Monty Python's Flying Circus

1. Версия 1.x



первая версия языка Python была выпущена в январе 1994 года. На тот момент Python уже предлагал простой и читаемый синтаксис, а также имел функции, поддерживающие процедурное программирование.

2. Версия 2.x



выпущена в октябре 2000 года, включала большое количество фундаментальных улучшений, таких как поддержка Unicode, список генераторов и списковых включений.

Python 2.7 - последнее обновление в ветке 2.x, выпущено в июле 2010 года. Python 2.7 стал одной из наиболее популярных версий Python и оставался активно использованным в течение многих лет.

3. Версия 3.x



обновление, которое внесло множество несовместимых изменений, выпущено в декабре 2008 года. Целью Python 3.0 было исправление некоторых недостатков и устранение неоднозначностей.

Python 3.x стал основной версией языка, а Python 2.x получил статус устаревшей ветки.

Python 2 VS Python 3

- Основные версии языка Python – **Python 2** и **Python 3**
- Версия **Python 2** считается устаревающей, версия 3 — более новой и современной.
- Почему не откажутся от второй версии? **Python 3** не имеет полной обратной совместимости с предыдущей версией: на **Python 2** написано очень много программ, и у разработчиков нет возможности переписать всё на новую версию.
- Мы будем пользоваться только **Python 3** и не будем говорить о **Python 2**.

Реализации интерпретатора Python

Питон интерпретируемый или компилируемый?

Первое, что необходимо понять: “Питон” – это интерфейс. Существует [спецификация](#), описывающая, что должен делать Питон, и как он должен себя вести (что справедливо для любого интерфейса). И существует несколько реализаций (что также справедливо для любого интерфейса).

Второе: “интерпретируемый” и “компилируемый” это свойства реализации, но не интерфейса.

Так что сам вопрос не совсем корректен.

В случае с самой распространенной реализацией (CPython: написанный на C, часто называемый просто “Python”, и, конечно, именно тот, который вы используете, если понятия не имеете о чем я толкую) ответ: интерпретируемый, с некоторой компиляцией. CPython компилирует* исходный код на Питоне в байткод, а затем интерпретирует этот байткод, запуская его в процессе.

*Замечание: это не совсем “компиляция” в традиционном смысле. Обычно, мы считаем, что “компиляция” это конвертация из высокоуровневого языка в машинный код. Тем не менее – в некотором роде это “компиляция”.

Байткод или машинный код

Очень важно понять разницу между байткодом и машинным (или нативным) кодом. Пожалуй, легче всего ее понять на примере:

- **Си компилируется в машинный код**, который впоследствии запускается напрямую процессором. Каждая инструкция заставляет процессор производить разные действия.
- **Java компилируется в байткод**, который впоследствии запускается на Виртуальной машине Java (Java Virtual Machine, JVM), абстрактном компьютере, который запускает программы. Каждая инструкция обрабатывается JVM, который взаимодействует с компьютером.

Сильно упрощая: **машинный код намного быстрее, но байткод лучше переносим и защищен**.

Машинный код может отличаться в зависимости от машины, тогда как байткод одинаковый на всех машинах. Можно сказать, что машинный код оптимизирован под вашу конфигурацию.

Возвращаясь к CPython, цепочка операций выглядит следующим образом:

- 1.CPython компилирует ваш исходный код на Питоне в байткод.
- 2.Этот байткод запускается на виртуальной машине CPython.

*Новички зачастую допускают, что Питон компилируемый из-за наличия .рус-файлов. Это отчасти верно: .рус-файлы – это скомпилированный байткод, который впоследствии интерпретируется. Так что если вы запускали ваш код на Питоне, и у вас есть .рус-файл, то во второй раз он будет работать быстрее, потому что ему не нужно будет заново компилироваться в байткод.

C_{Py}thon

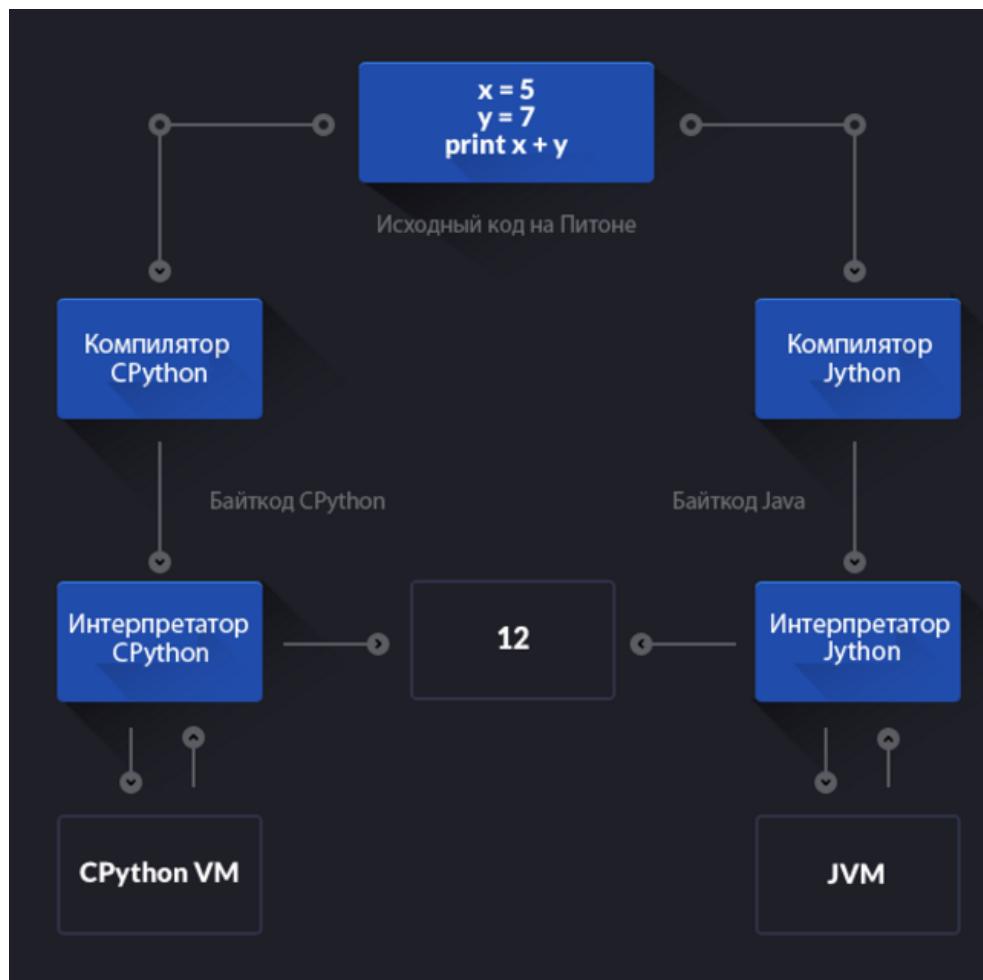
C_{Py}thon - это наиболее широко используемая реализация интерпретатора Python и является официальной реализацией языка. Изначально разработана Гвидо ван Россумом в 1989 году, C_{Py}thon написан на языке программирования С и предоставляет основные функциональные возможности интерпретатора Python, такие как компиляция, выполнение и интерпретация кода Python.

Ключевые особенности C_{Py}thon:

- **Интерпретация:** C_{Py}thon интерпретирует код Python, переводя его в промежуточный байт-код и исполняя его виртуальной машиной Python.
- **Сборка мусора:** C_{Py}thon использует сборку мусора, чтобы автоматически управлять памятью и освобождать объекты, на которые больше нет ссылок.
- **Динамическая типизация:** Python является динамически типизированным языком, и C_{Py}thon обеспечивает поддержку динамического связывания и определения типов переменных во время выполнения кода.
- **Расширяемость:** C_{Py}thon предоставляет возможность расширять его функциональность с помощью написания модулей на языке С или использования сторонних библиотек и модулей Python.

Альтернативная виртуальная машина: Jython

Jython - реализация Питона на **Java**, которая использует **JVM**. В то время как **CPython** генерирует **байткод** для запуска на **CPython VM**, **Jython** генерирует **байткод Java** для запуска на **JVM** (это то же самое, что генерируется при компиляции программы на **Java**).



Установка Python на Windows

Язык Python относится к свободному программному обеспечению, поэтому его можно скачать с официального сайта (python.org), свободно распространять и устанавливать на все современные операционные системы.

The screenshot shows the Python.org homepage. At the top, there is a navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation bar is the Python logo and the word "python" in lowercase. To the right of the logo are buttons for "Donate", a search bar with a magnifying glass icon, and a "GO" button. Further to the right are links for "Socialize" and other site sections. A red arrow points to the "Downloads" link in the main navigation menu, which is highlighted in blue. The main content area features a dark background with code snippets and text. One snippet shows Python 3 list comprehensions, and another shows the use of the enumerate function. To the right of the code snippets is a section titled "Compound Data Types" with a sub-section about lists. At the bottom right of the content area are numbered buttons from 1 to 5, likely for pagination.

Python

PSF

Docs

PyPI

Jobs

Community

python™

Donate

Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [»» Learn More](#)

Python

PSF

Docs

PyPI

Jobs

Community



Donate



Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

Download the latest version for Windows

Download Python 3.8.1



Looking for Python with a different OS? Python for [Windows](#),
[Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#),
[Docker images](#)

Looking for Python 2.7? See below for specific releases



Install Python 3.8.1 (32-bit)

Select **Install Now** to install Python with default settings, or choose **Customize** to enable or disable features.



python
for
windows

Install Now

C:\Users\Timur\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation

Choose location and features

Install launcher for all users (recommended)

Add Python 3.8 to PATH

Cancel

Install Python 3.8.1 (32-bit)

Select [Install Now](#) to install Python with default settings, or choose [Customize](#) to enable or disable features.



 [Install Now](#)

C:\Users\Timur\AppData\Local\Programs\Python\Python38-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations



→ [Customize installation](#)

[Choose location and features](#)

[Install launcher for all users \(recommended\)](#)

[Add Python 3.8 to PATH](#)

[Cancel](#)

Optional Features

Documentation

Installs the Python documentation file.

pip

Installs pip, which can download and install other Python packages.

tcl/tk and IDLE

Installs tkinter and the IDLE development environment.

Python test suite

Installs the standard library test suite.

py launcher for all users (requires elevation)

Upgrades the global 'py' launcher from the previous version.



python
for
windows

Back

Next

Cancel

Advanced Options

- Install for all users
- Associate files with Python (requires the py launcher)
- Create shortcuts for installed applications
- Add Python to environment variables
- Precompile standard library
- Download debugging symbols
- Download debug binaries (requires VS 2017 or later)

Customize install location

Browse

python
for
windows

Back **Install**Cancel



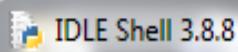
```
Python 3.8 (32-bit)
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:08:11) [MSC v.1928 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Привет")
Привет
>>>
```



IDLE Shell 3.8.8

```
File Edit Shell Debug Options Window Help
Python 3.8.8 (tags/v3.8.8:024d805, Feb 19 2021, 13:08:11) [MSC v.1928 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print ("Привет")
Привет
>>>
```

Ln: 5 Col: 4



File Edit Shell Debug Options Window Help

New File Ctrl+N :024d805, Feb 19 2021, 13:08:11) [MSC v.1928 32 bit (Intel)] on win32
Open... Ctrl+O "credits" or "license()" for more information.
Open Module... Alt+M
Recent Files ▾
Module Browser Alt+C
Path Browser

Save Ctrl+S
Save As... Ctrl+Shift+S
Save Copy As... Alt+Shift+S

Print Window Ctrl+P

Close Alt+F4
Exit Ctrl+Q

untitled

File Edit Format Run Options Window Help

```
print ("Привет")|
```

Ln: 1 Col: 16

IDLE Shell 3.8.8 - C:/Users/user/Desktop/111111111111111111111111.py (3.8.8)

untitled

File Edit Format Run Options Window Help

Type "print ("Привет")"
->>> print
Привет
->>>

Run Module F5
Run... Customized Shift+F5
Check Module Alt+X
Python Shell

The screenshot shows the IDLE Shell interface. On the left, there's a sidebar with 'File' and 'Edit' tabs. The main area has tabs for 'Python' and 'Type "...'. A code editor window titled '*untitled*' contains the following Python code:

```
print ("Привет")
```

Below the code, the Python prompt shows two lines: '>>> print' and 'Привет'. A context menu is open at the bottom of the code editor, with 'Run Module' highlighted. Other options in the menu are 'Run...', 'Customized', 'Shift+F5', 'Check Module', 'Alt+X', and 'Python Shell'. The menu has a light gray background and white text.

Google.Collab

CO Добро пожаловать в Colaboratory!

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Не удалось сохранить изменения

Поделиться

Содержание

+ Код + Текст Копировать на Диск

ОЗУ Диск Редактирова

Начало работы

Анализ и обработка данных

{x} Машинное обучение

Ресурсы по теме

Примеры

Раздел

Добро пожаловать в Colab!

Уже знакомы с Colab? В этом видео рассказывается о функциях, которые вы могли пропустить: интерактивных таблицах, истории выполненного кода и палитре команд.



Что такое Colab?

Colaboratory, или просто Colab, позволяет писать и выполнять код Python в браузере. При этом:

- не требуется никакой настройки;
- бесплатный доступ к графическим процессорам;
- предоставлять доступ к документам другим людям очень просто.

Это отличное решение для **студентов, специалистов по обработке данных и исследователей в области искусственного интеллекта**. Чтобы узнать больше, посмотрите [ознакомительное видео](#) или начните работу с инструментом ниже.

▼ Начало работы

Документ, который вы читаете, размещен не на статической веб-странице, а в интерактивной среде под названием **блокнот Colab**, позволяющей писать и выполнять код.

Например, вот **ячейка** с коротким скриптом Python, который позволяет рассчитать значение, выразить его в виде переменной и

<https://colab.research.google.com>



Добро пожаловать в Colaboratory!

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Не удалось сохранить изменения



Создать блокнот



Открыть блокнот



Загрузить блокнот



Переименовать



Сохранить копию на Диске



Сохранить копию как файл Gist для GitHub



Создать копию в GitHub



Сохранить

Ctrl+S Копировать на Диск



История версий



Скачать

Ctrl+P



Печать

Жаловать в Colab!

с Colab? В этом видео рассказывается о функциях, которые вы могли пропустить: интерактивных таблицах, истории кода и палитре команд.



ОЗУ
Диск

Что такое Colab?

Colaboratory, или просто Colab, позволяет писать и выполнять код Python в браузере. При этом:

- не требуется никакой настройки;
- бесплатный доступ к графическим процессорам;
- предоставлять доступ к документам другим людям очень просто.

Это отличное решение для студентов, специалистов по обработке данных и исследователей в области искусственного интеллекта. Чтобы узнать больше, посмотрите [ознакомительное видео](#) или начните работу с инструментом ниже.

▼ Начало работы

Документ, который вы читаете, размещен не на статической веб-странице, а в интерактивной среде под названием **блокнот Colab**, позволяющей писать и выполнять код.



Untitled0.ipynb



Комментировать

Файл Изменить Вид Вставка Среда выполнения Инструменты Справка Изменения сохранены

ОЗУ
Диск

+ Код + Текст

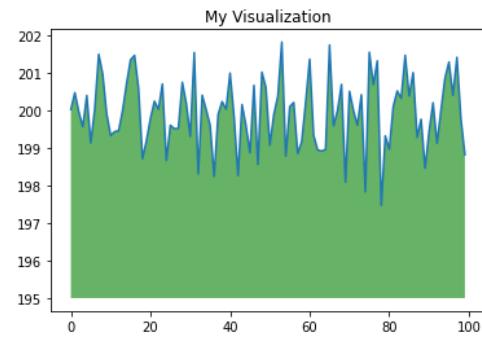
0
сек.

```
import numpy as np
from matplotlib import pyplot as plt

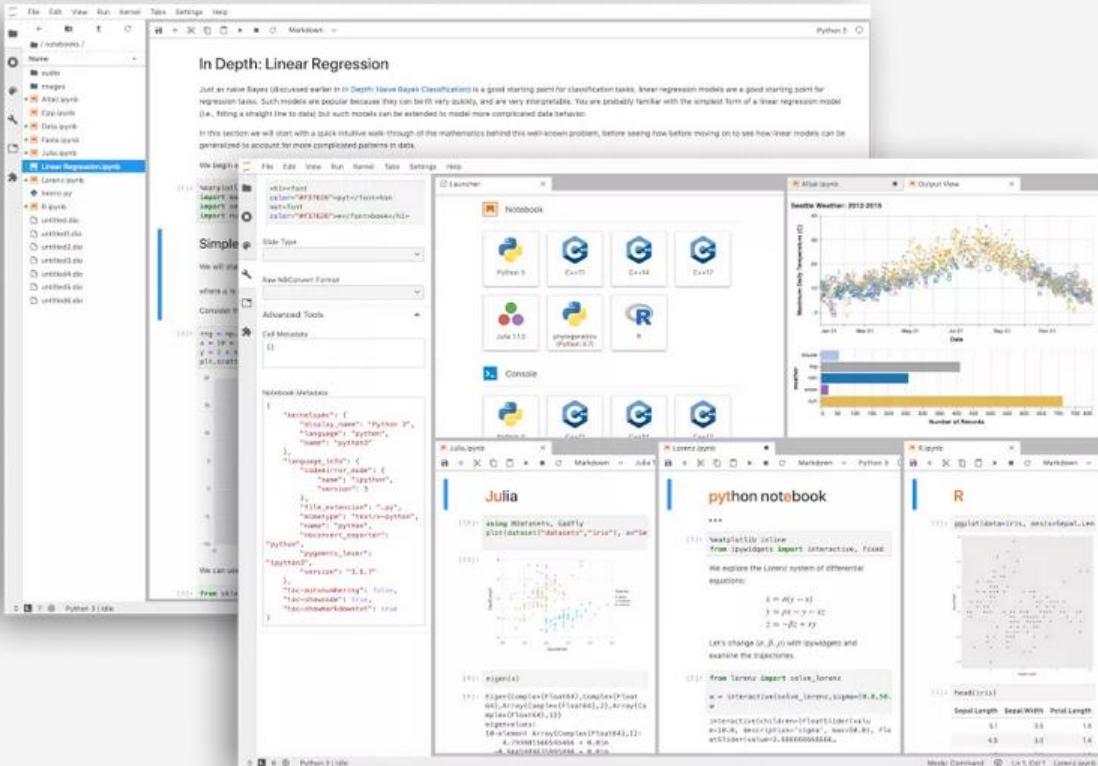
ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("My Visualization")
plt.show()
```



<https://jupyter.org/>



JupyterLab: A Next-Generation Notebook

JupyterLab is the latest web-based interactive development environment for data science. Its flexible interface allows users to configure and arrange multiple code editors, terminals, and data visualizations for computing, computational journalism, and machine learning. It also provides a platform to expand and enrich functionality.

[Try it in your browser](#)

[Install JupyterLab](#)

IDLE

IDLE Shell 3.9.6

File Edit Shell Debug Options Window Help

Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

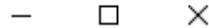
```
>>> import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("My Visualization")
plt.show()
SyntaxError: multiple statements found while compiling a single statement
>>> |
```

cmd Администратор: cmd.exe



Microsoft Windows [Version 10.0.19044.1949]

(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\System32>pip install numpy

Collecting numpy

 Downloading numpy-1.23.2-cp39-cp39-win_amd64.whl (14.7 MB)
 ██████████| 14.7 MB 1.8 kB/s

Installing collected packages: numpy

Successfully installed numpy-1.23.2

WARNING: You are using pip version 21.1.3; however, version 22.2.2 is available.

You should consider upgrading via the 'c:\program files\python39\python.exe -m pip install --upgrade pip' command.

C:\Windows\System32>

Администратор: cmd.exe



Microsoft Windows [Version 10.0.19044.1949]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\System32>pip install numpy

Collecting numpy

 Downloading numpy-1.23.2-cp39-cp39-win_amd64.whl (14.7 MB)
 |██████████| 14.7 MB 1.8 kB/s

Installing collected packages: numpy

Successfully installed numpy-1.23.2

WARNING: You are using pip version 21.1.3; however, version 22.2.2 is available.

You should consider upgrading via the 'c:\program files\python39\python.exe -m pip install --upgrade pip' command.

C:\Windows\System32>python -m pip install --upgrade pip

Requirement already satisfied: pip in c:\program files\python39\lib\site-packages (21.1.3)

Collecting pip

 Downloading pip-22.2.2-py3-none-any.whl (2.0 MB)
 |██████████| 2.0 MB 939 kB/s

Installing collected packages: pip

Attempting uninstall: pip

 Found existing installation: pip 21.1.3

 Uninstalling pip-21.1.3:

 Successfully uninstalled pip-21.1.3

Successfully installed pip-22.2.2

C:\Windows\System32>

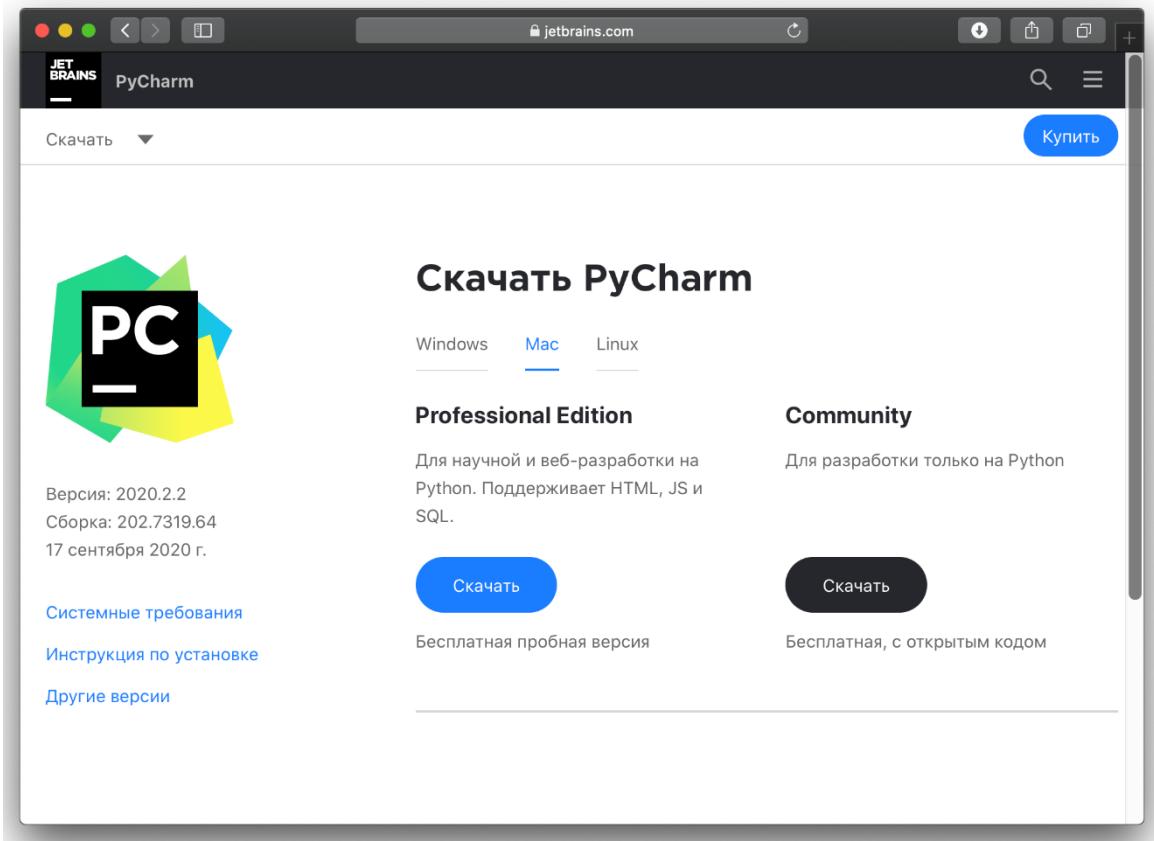
Администратор: cmd.exe

```
Microsoft Windows [Version 10.0.19044.1949]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Windows\System32>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.5.3-cp39-cp39-win_amd64.whl (7.2 MB)
    7.2/7.2 MB 4.5 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp39-cp39-win_amd64.whl (55 kB)
    55.4/55.4 kB 3.0 MB/s eta 0:00:00
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    247.7/247.7 kB 14.8 MB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.11.0-py3-none-any.whl (6.4 kB)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.37.1-py3-none-any.whl (957 kB)
    957.2/957.2 kB 8.7 MB/s eta 0:00:00
Collecting packaging>=20.0
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
    40.8/40.8 kB 1.9 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in c:\program files\python39\lib\site-packages (from matplotlib) (1.23.2)
Collecting pillow>=6.2.0
  Downloading Pillow-9.2.0-cp39-cp39-win_amd64.whl (3.3 MB)
    3.3/3.3 MB 9.1 MB/s eta 0:00:00
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
    98.3/98.3 kB 5.9 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, pyparsing, pillow, kiwisolver, fonttools, cycler, python-dateutil, packaging, matplotlib
Successfully installed cycler-0.11.0 fonttools-4.37.1 kiwisolver-1.4.4 matplotlib-3.5.3 packaging-21.3 pillow-9.2.0 pyparsing-3.0.9 py
C:\Windows\System32>
```



PyCharm



Сайт: <https://www.jetbrains.com/pycharm/>



Welcome to PyCharm



PyCharm
2022.3.1

Projects

Customize

Plugins

Learn

Welcome to PyCharm

Create a new project to start from scratch.

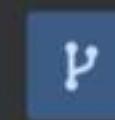
Open existing project from disk or version control.



New Project



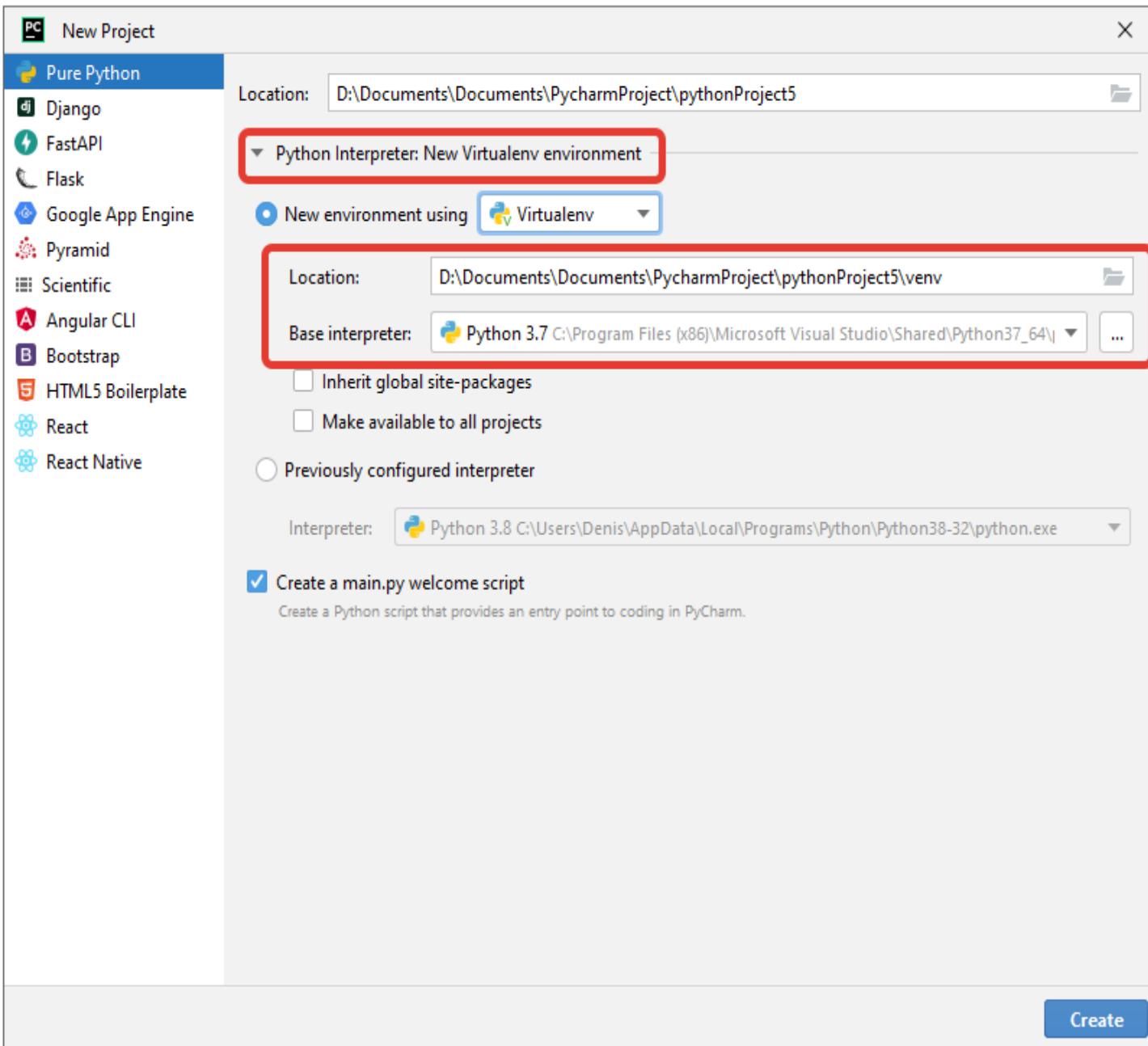
Open



Get from VCS



You can find the onboarding tour under Help > Learn IDE Features, where you can choose the **Get Acquainted with PyCharm** lesson.



The screenshot shows the PyCharm IDE interface with a Python project open. The project structure on the left includes a folder named '1' containing a 'venv' library root and a file 'main.py'. The code editor on the right displays the following Python script:

```
# This is a sample Python script.  
# Press Shift+F10 to execute it or replace it with your code.  
# Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.  
  
def print_hi(name):  
    # Use a breakpoint in the code line below to debug your script.  
    print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.  
  
# Press the green button in the gutter to run the script.  
if __name__ == '__main__':  
    print_hi('PyCharm')  
  
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

A red circular marker is placed on the line 'print(f'Hi, {name}')' at line 9, indicating a breakpoint. The PyCharm interface features a dark theme with light-colored code blocks. At the bottom, there are tabs for Version Control, TODO, Problems, Terminal, Python Packages, Python Console, and Services. A status bar at the very bottom provides information about indexing and file paths.

The screenshot shows the PyCharm IDE interface with a dark theme. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. A tab at the top right indicates "1 - main.py - Administrator".

The left sidebar features a "Project" view showing a directory structure with a file named "main.py" selected. Other items in the project include "venv library root", "External Libraries", and "Scratches and Consoles".

The main code editor window displays the following Python script:

```
# Press the green button in the gutter to run the script.
if __name__ == '__main__':
    print_hi('PyCharm')

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("My Visualization")
plt.show()
```

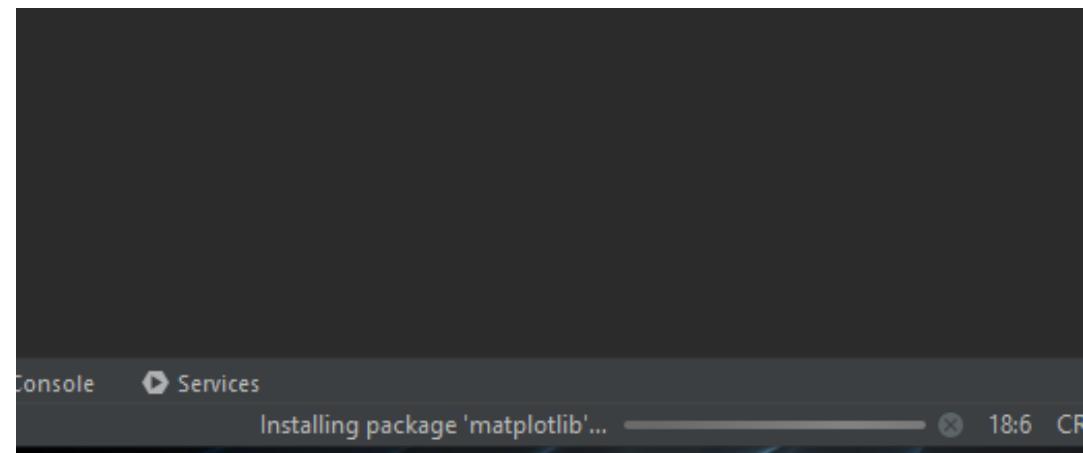
The code editor highlights several lines with red squiggly underlines, indicating syntax errors. Lines 17 and 18 specifically highlight the imports of `numpy` and `matplotlib.pyplot`.

The bottom section shows the "Run" tool window. It lists a single run configuration named "main" with the command "C:\Users\rilan\Desktop\2\1\venv\Scripts\python.exe C:/Users/rilan/Desktop/2/1/main.py". Below this, a "Traceback (most recent call last)" section shows the error message:

```
Hi, PyCharm
Traceback (most recent call last):
  File "C:\Users\rilan\Desktop\2\1\main.py", line 17, in <module>
    import numpy as np
ModuleNotFoundError: No module named 'numpy'
```

At the very bottom, the status bar displays "PEP 8: W292 no newline at end of file" and "27:11 CRLF UTF-8 4 spaces Python 3.9 (1)".

```
# See PyCharm help at https://www.jetbrains.com/help/pycharm/
import numpy as np
from matp No module named 'numpy' ...
ys = 200 Install package numpy Alt+Shift+Enter More actions... Alt+Enter
x = [x for x in range(len(ys))]
```



File Edit View Navigate Code Refactor Run Tools VCS Window Help 1 - main.py - Administrator

1 main.py

Figure 1

My Visualization

```
in the gutter to run the script.  
:  
https://www.jetbrains.com/help/pycharm/  
plot as plt  
n(100)  
(ys))]  
  
95, where=(ys > 195), facecolor='g', alpha=0.6)  
on")
```

Home Back Forward Plus Search Magnifying Glass Copy

C:\Users\plan\Desktop\2\1\venv\Scripts\python.exe C:/Users/plan/Desktop/2/1/main.py
Hi, PyCharm

Structure Bookmarks Version Control Run TODO Problems Terminal Python Packages Python Console Services

PEP 8: W292 no newline at end of file 27:11 CRLF UTF-8 4 spaces Python 3.9 (1)

После этого его можно запустить
несколькими способами:

- использовать горячие клавиши: **Ctrl + Shift + F10** для Windows и Linux;
- правой кнопкой мыши щёлкнуть по вкладке `hello.ru` и во всплывающем окне выбрать пункт **Run 'hello'**;
- нажать на значок возле номера строки и во всплывающем окне выбрать пункт **Run 'hello'**.



Visual Studio Code

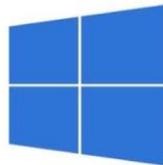
Установка VS Code

ooo code.visualstudio.com

Visual Studio Code Docs Updates Blog API Extensions FAQ Learn Download

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.


[Windows](#)
Windows 10, 11


[.deb](#)
Debian, Ubuntu


[.rpm](#)
Red Hat, Fedora, SUSE


[Mac](#)
macOS 10.11+

User Installer [x64](#) [x86](#) [Arm64](#)
System Installer [x64](#) [x86](#) [Arm64](#)
.zip [x64](#) [x86](#) [Arm64](#)
CLI [x64](#) [x86](#) [Arm64](#)

.deb [x64](#) [Arm32](#) [Arm64](#)
.rpm [x64](#) [Arm32](#) [Arm64](#)
.tar.gz [x64](#) [Arm32](#) [Arm64](#)
Snap [Snap Store](#)

CLI [x64](#) [Arm32](#) [Arm64](#)

.zip [Intel chip](#) [Apple silicon](#) [Universal](#)
CLI [Intel chip](#) [Apple silicon](#)



← →

Search



✖ ⌘ 0 △ 0 ⌘ 0



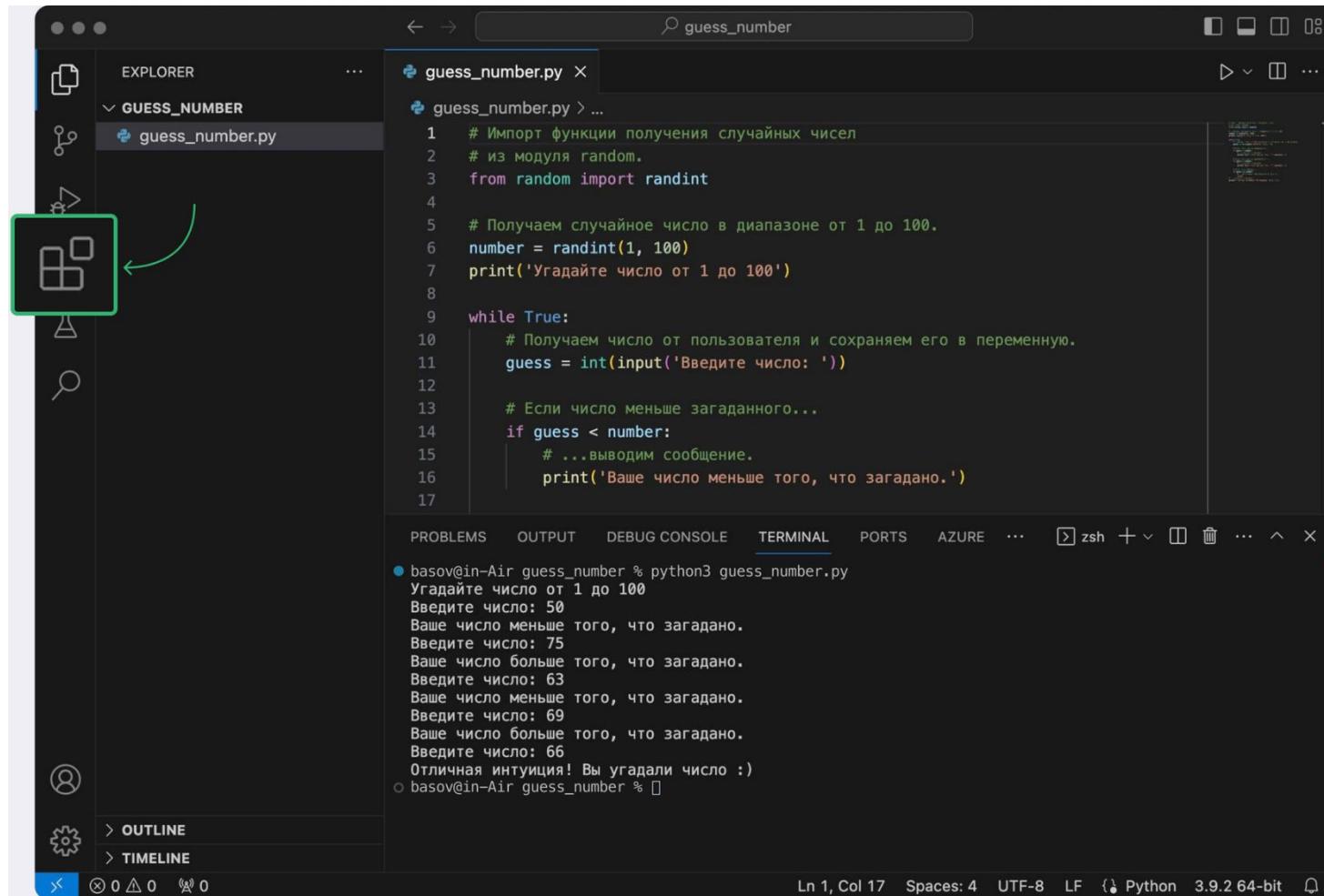
Show All Commands ⌘ P

Open File or Folder ⌘ O

Open Recent ⌘ R

New Untitled Text File ⌘ N

Настраиваем VS Code под Python



Найдите и установите следующие расширения:

- **Python** от Microsoft. Оно добавляет в редактор подсветку синтаксиса и ошибок, описание команд при наведении на них указателем мыши, расширенную поддержку языка Python для всех версий ≥ 3.7 и другие инструменты работы с кодом.
- **Pylance** — ещё одно дополнение от Microsoft, которое отвечает за автодополнение кода и другие подсказки.

Линтеры

Проконтролировать соответствие кода установленным стандартам можно с помощью **программ-линтеров**. Слово *linter* в переводе с английского обозначает ролик для чистки одежды от ворсинок и пыли.

Включение линтера в VS Code

Линтер можно установить как расширение для VS Code — скачать расширения можно в [Marketplace](#).

Например, там есть:

Линтер	Расширение
Pylint	https://marketplace.visualstudio.com/items?itemName=ms-python pylint
Flake8	https://marketplace.visualstudio.com/items?itemName=ms-python.flake8
Mypy	https://marketplace.visualstudio.com/items?itemName=ms-python.mypy-type-checker

Проверка ваших самостоятельных заданий будет проводиться с помощью линтера **Flake8**, и вы сможете избежать многих ошибок, если тоже будете проверять свой код с его помощью.

Этот линтер сочетает в себе функциональность нескольких анализаторов кода и весьма популярен среди разработчиков на Python.

Перейдите [по ссылке на страницу Flake8 в Visual Studio Marketplace](#) или найдите линтер **Flake8** через интерфейс VS Code:

Включение форматера в VS Code

Линтеры помогают избежать огромного количества неприятностей, анализируя код на предмет общих синтаксических, стилистических и функциональных ошибок. Они выступают в роли первой линии обороны, предохраняя от многих распространённых проблем.

Но линтеры обращают внимание в основном на техническую и структурную сторону кода, оставляя в стороне его визуальное представление и читаемость.

И здесь на сцену выходят форматеры кода. Если линтеры помогают разработчикам писать правильный код, то форматеры заботятся о том, чтобы этот код был написан красиво и читаемо. Форматеры автоматически структурируют код, следуя заданным стилевым правилам и конвенциям.

Например, форматер может автоматически выставить межстрочные интервалы, отступы и пробелы вокруг операторов. В итоге код становится более организованным и понятным.

Форматер тоже можно (и нужно!) установить в качестве плагина к вашему редактору кода. Для VS Code есть, например, форматеры [autopep8](#) и [Black Formatter](#).

Какой форматер использовать и использовать ли его вообще — решение за вами. Но помните, что при проверке самостоятельных работ форматирование кода тоже будет учитываться.

Чтобы результат работы выбранного форматера соответствовал стилю Практикума, может потребоваться его дополнительная настройка.

После установки форматера его стоит установить по умолчанию для файлов Python в VS Code. Для этого выполните следующие действия:

1. Откройте в VS Code любой файл с расширением .py.
2. Щёлкните правой кнопкой мыши на редакторе, чтобы вызвать контекстное меню.
3. Выберите пункт *Format Document With...*
4. В раскрывающемся меню выберите *Configure Default Formatter...*
5. Выберите из списка нужный форматер.

После этого отформатировать код можно, щёлкнув правой кнопкой мыши на редакторе и выбрав пункт *Format Document*.

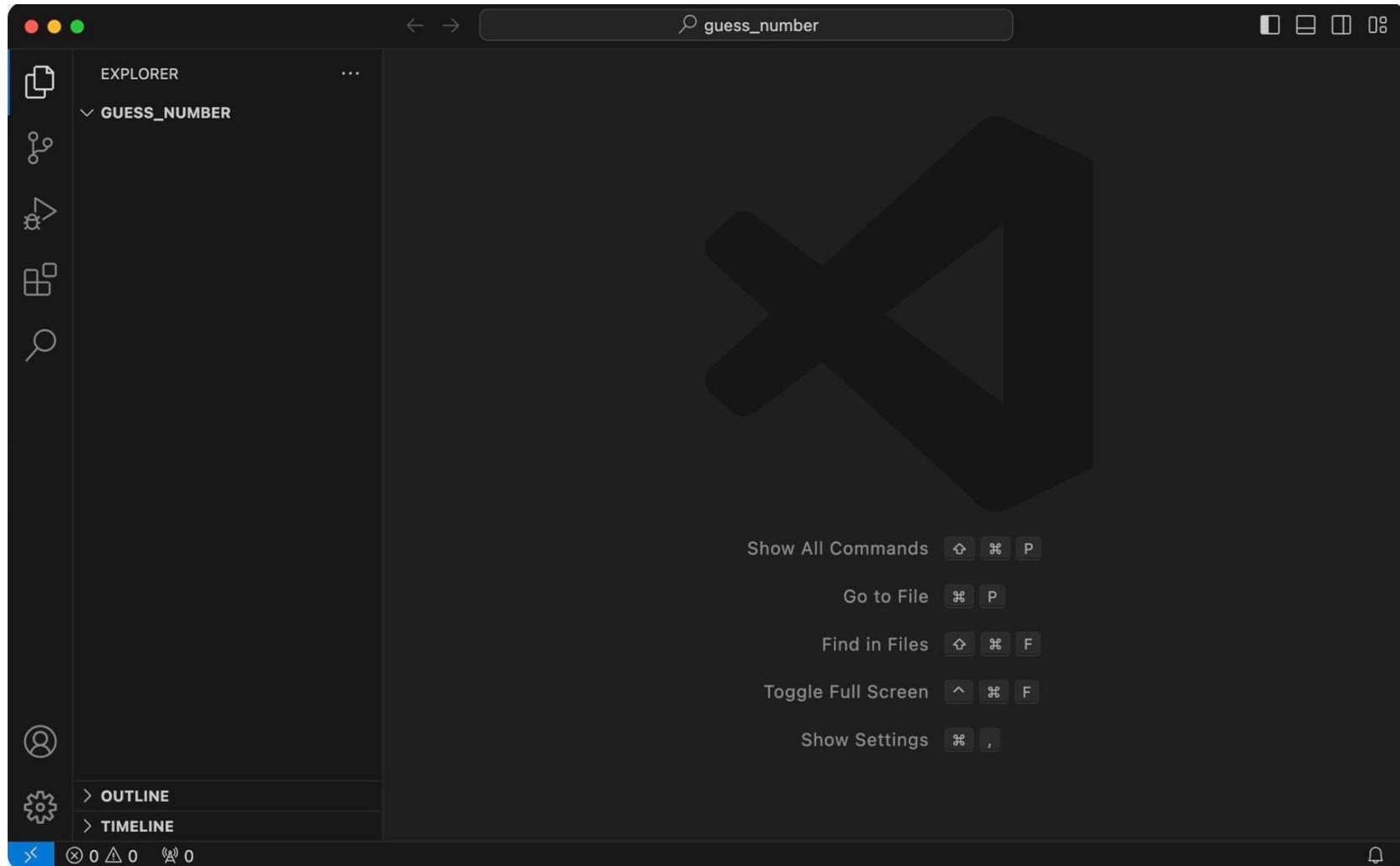
Russian Language Pack for Visual Studio Code

Языковой пакет для русского языка содержит локализацию интерфейса VS Code.

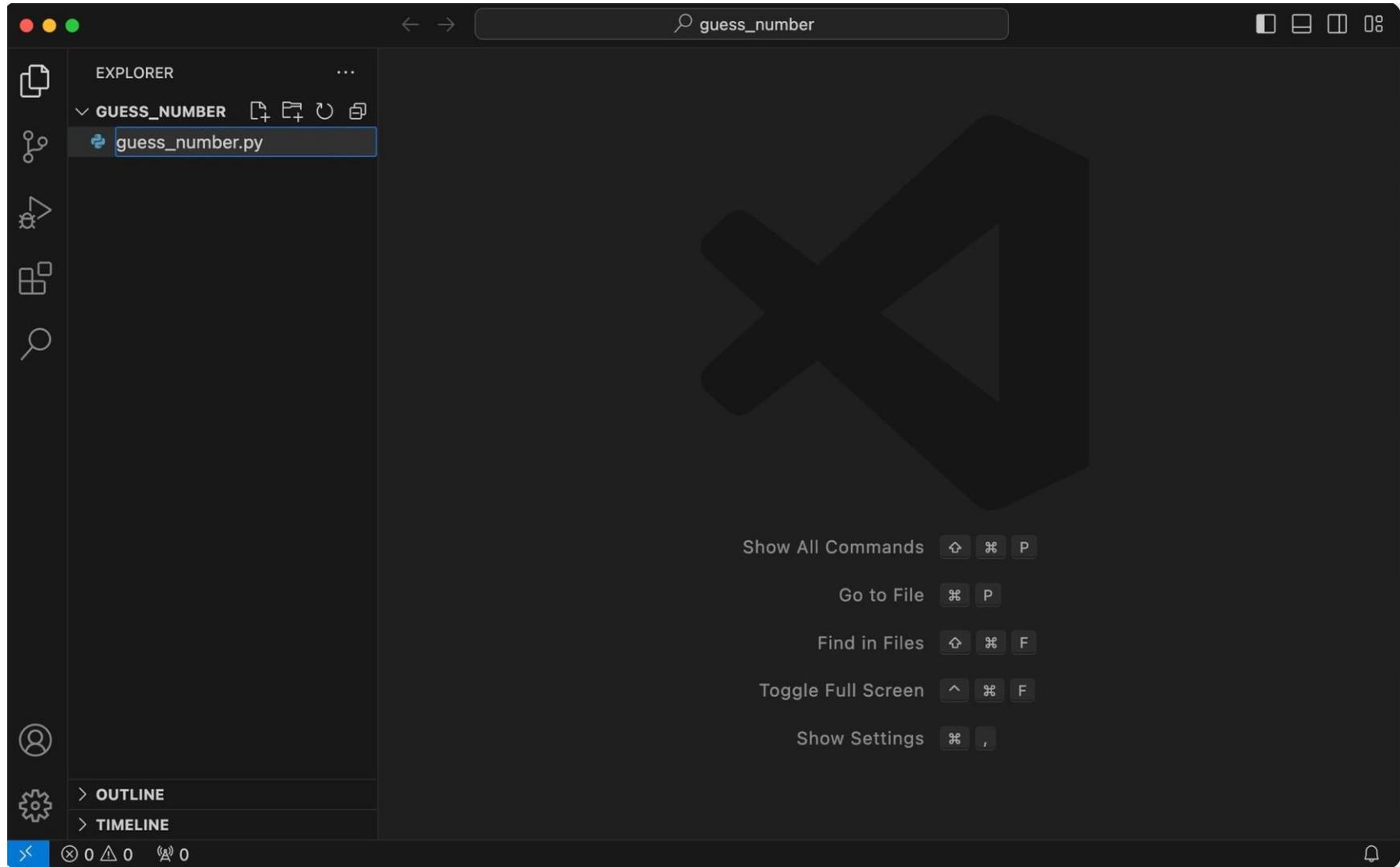
Нажмите Ctrl + Shift + P (Command ⌘ + Shift ↑ + P на Mac), чтобы открыть палитру команд, а затем начните вводить «отображен...» (“display”), чтобы отфильтровать команду «Настроить язык отображения» (“Configure Display Language”). Нажмите клавишу Ввод (Enter), и откроется список установленных языков по языковым стандартам, в котором будет выбран текущий языковой стандарт. Чтобы изменить язык пользовательского интерфейса, выберите другой языковой стандарт.

Как работать в VS Code

Чтобы начать работу с проектом в VS Code, нужно открыть корневую папку проекта. Выберите в главном меню пункты File → Open Folder.



Наведите курсор мыши на имя проекта, нажмите на иконку **New file** и введите название файла с расширением *.py*, например *guess_number.py*. В файлах с таким расширением хранится код, написанный на Python.



Сохраните изменения в файле:

- если работаете на Windows, нажмите сочетание клавиш **Ctrl+S**,
- если используете macOS — **Command+S**.

А ещё лучше — сразу настройте автосохранение. Для этого в меню **File** нужно поставить галочку напротив строки **Auto Save**.

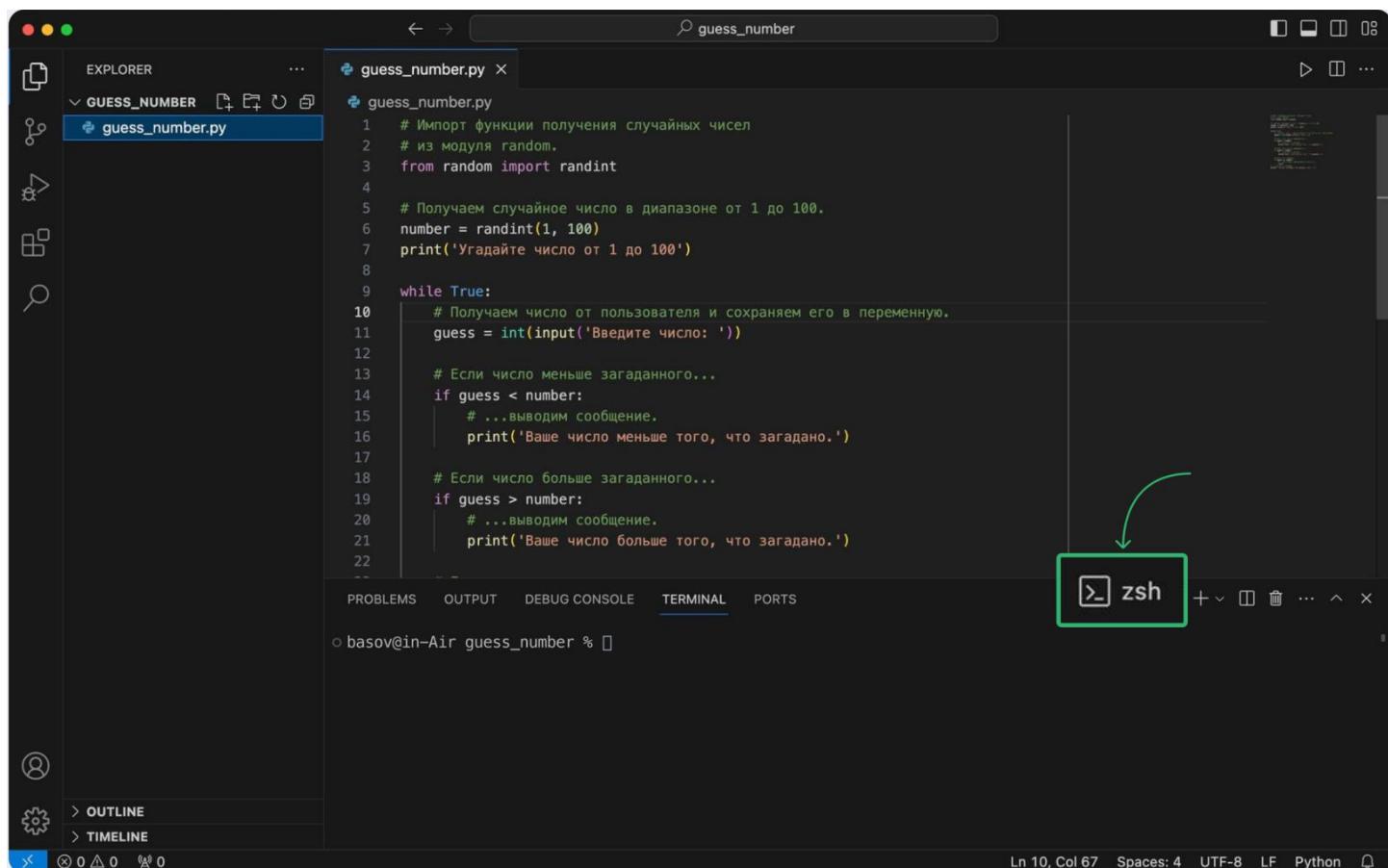
Терминал в редакторе кода

В главном меню выберите **Terminal → New Terminal**. Под кодом откроется окно с терминалом.

Терминал в редакторе кода

Теперь можно запустить программу и посмотреть, как она работает. Для этого вам понадобится терминал, но запускать его в отдельном окне не придётся. С терминалом можно работать прямо в среде VS Code.

В главном меню выберите Terminal → New Terminal. Под кодом откроется окно с терминалом.

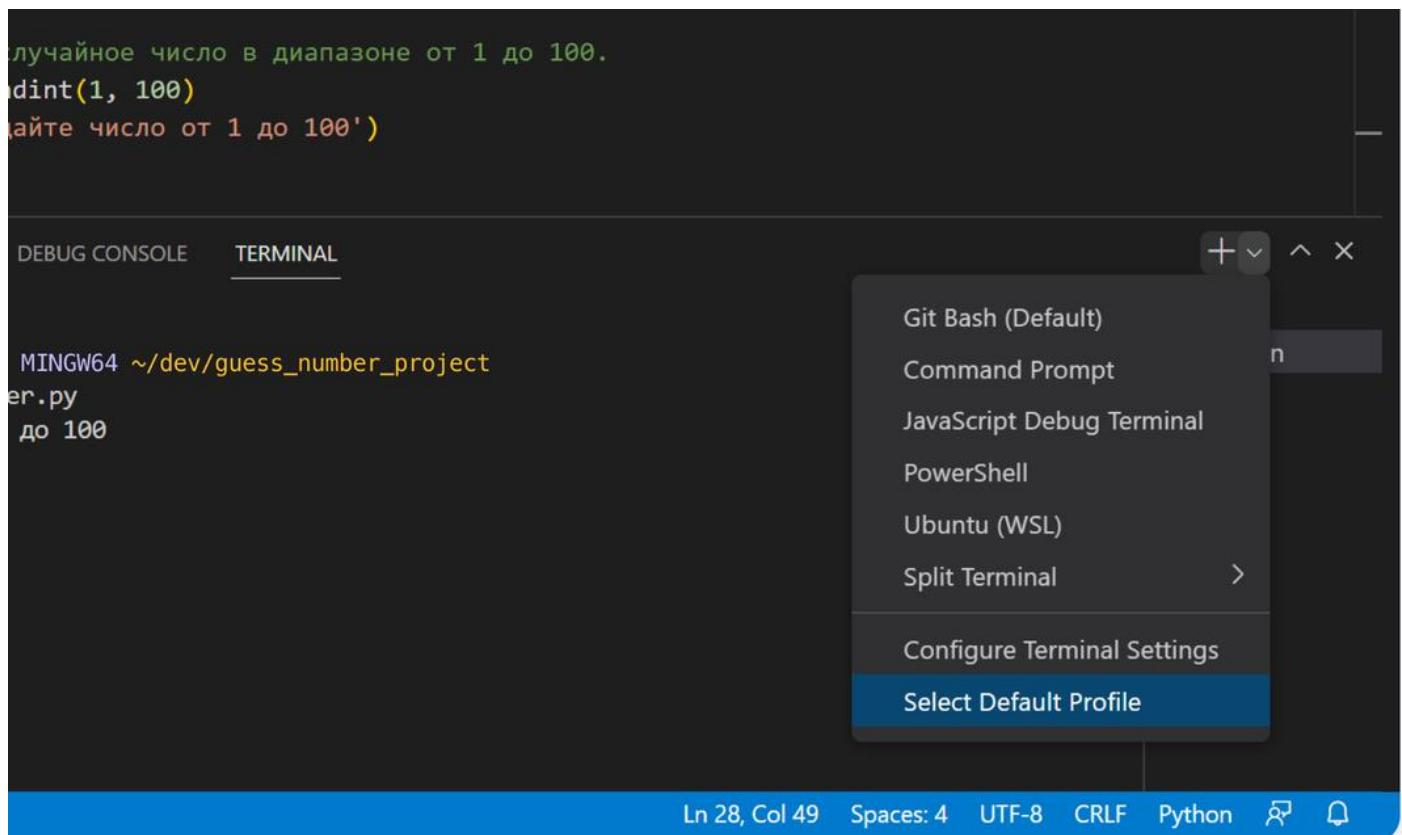


Если открылся не тот терминал

В ОС Windows может быть установлено несколько разных терминалов, и при первом включении терминала в VS Code, скорее всего, запустится PowerShell.

Измените настройки VS Code — пусть по умолчанию всегда запускается Git Bash.

В меню окна с терминалом нажмите кнопку «Вниз» рядом с плюсиком и выберите пункт Select Default Profile:



File Edit Selection View Go Run Terminal Help guess_number.py - guess_number_project - Visual Studio Code

EXPLORER GUESS_NUMBER_PROJECT guess_number.py Select your default terminal profile profiles contributed detected

Git Bash C:\Program Files\Git\bin\bash.exe --login
Command Prompt C:\WINDOWS\System32\cmd.exe
PowerShell C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe
Ubuntu (WSL) C:\WINDOWS\System32\wsl.exe -d Ubuntu
JavaScript Debug Terminal
Windows PowerShell C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe

```
7     print ('Угадайте число от 1 до 100')
8
9     while True:
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Stas@LAPYOP-0c243HR8 MINGW64 ~/dev/guess_number_project
\$ python guess_number.py
Угадайте число от 1 до 100
Введите число:

bash python

OUTLINE

0 △ 0 Ln 28, Col 49 Spaces: 4 UTF-8 CRLF Python 🔍 🔔

Чтобы запустить код, введите в терминале команду:

```
# Для Windows.python  
guess_number.py
```

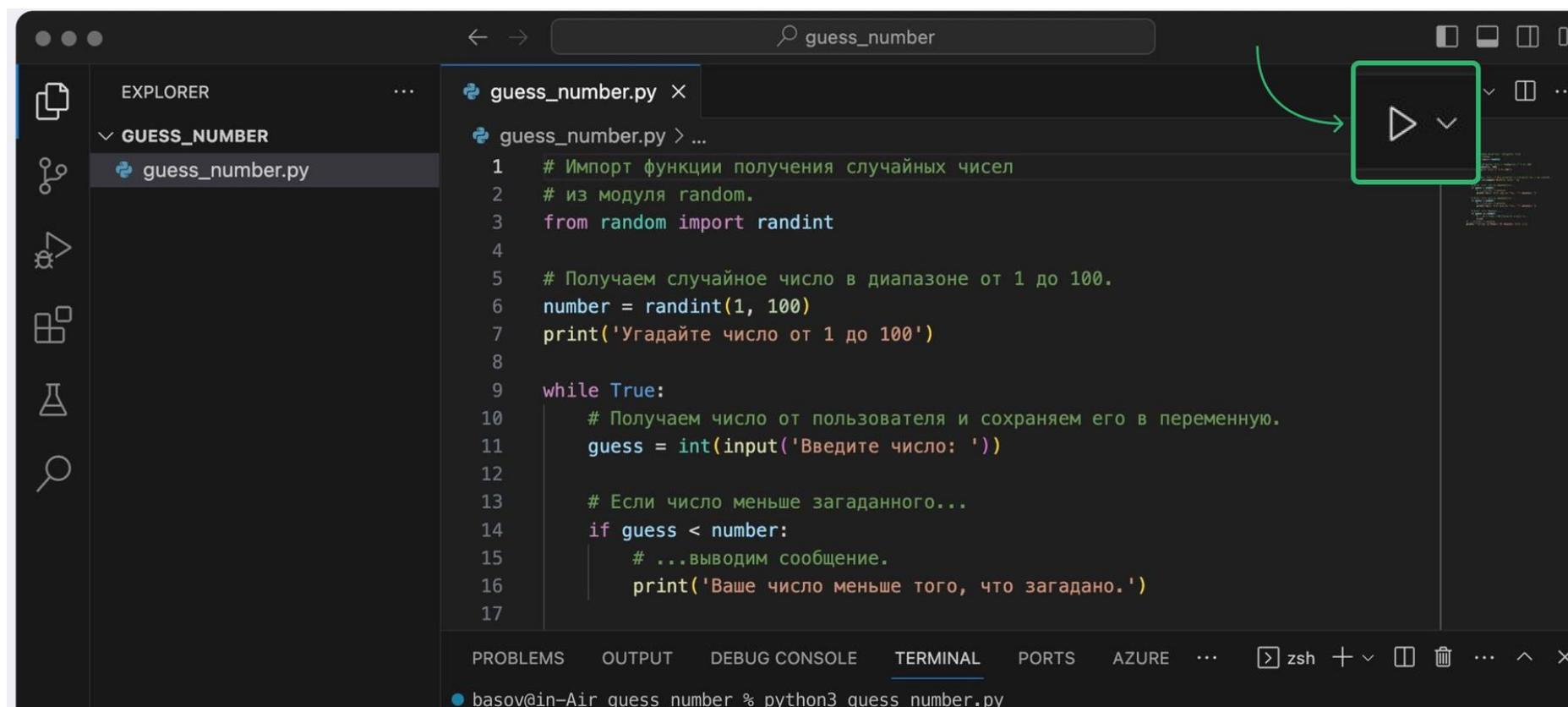
```
# Для Linux и macOS.  
python3 guess_number.py
```

```
# Интерпретатор Python, запусти на выполнение файл guess_number.py.
```

Кнопка Run

Ещё одна из возможностей, которую предоставляет редактор кода, — кнопка **Run Python File**. Эта кнопка запускает программу на выполнение.

Теперь скрипты можно не только запускать командой в терминале, но и просто нажать на кнопку, и программа запустится.



```
# Импорт функции получения случайных чисел
# из модуля random.
from random import randint

# Получаем случайное число в диапазоне от 1 до 100.
number = randint(1, 100)
print('Угадайте число от 1 до 100')

while True:
    # Получаем число от пользователя и сохраняем его в переменную.
    guess = int(input('Введите число: '))

    # Если число меньше загаданного...
    if guess < number:
        # ...выводим сообщение.
        print('Ваше число меньше того, что загадано.')
```

The screenshot shows the Visual Studio Code interface with a dark theme. On the left is the Explorer sidebar showing a folder named 'GUESS_NUMBER' containing a file 'guess_number.py'. The main editor area displays the code for a guessing game. In the top right corner, there is a toolbar with several icons, including a green highlighted 'Run' icon (a play button). Below the editor, the status bar shows the command line: 'basov@in-Air guess number % python3 guess_number.py'. The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), PORTS, AZURE, and other development tools.

Структура программы на Python

Неважно, на каком языке написана программа и что она делает, структура практически любой программы сводится к трём составляющим:

- ввод данных в программу;
- обработка введённых данных в программе;
- вывод результата обработки данных.

Импорт модулей: Обычно программа начинается с импорта необходимых модулей. Модуль - это файл с расширением .py, содержащий код и определения функций, классов и переменных. В Python существует множество стандартных модулей, а также сторонних библиотек, которые могут быть использованы для расширения функциональности программы.

```
# Пример импорта стандартного модуля  
import math
```

```
# Пример импорта конкретной функции из модуля  
from random import randint
```

Комментарии: Важно добавлять комментарии к коду, чтобы сделать его более понятным для других разработчиков. Комментарии помечаются символом # и не влияют на выполнение программы.

Точка входа в программу

```
if __name__ == "__main__"
```

Это стандартный код, который защищает пользователей от случайного вызова сценария, когда они не собирались этого делать, и его применение является хорошей практикой.

Это имеет значение для этих двух случаев использования:

- Мы запускаем его как основную программу с помощью `python filename.py`
- Импортируем файл в другой файл с помощью `import filename`

В последнем случае обычно мы хотим только импортировать модуль, а затем позже в коде выполнить некоторые функции или использовать класс из этого файла.

Когда интерпретатор Python читает исходный файл, он делает две вещи:

- Во-первых, он устанавливает несколько специальных переменных, таких как `__name__`.
- Затем он выполняет весь код, который находит в файле.

```
# This is testa.py

def func_one():
    print("Function One")

if __name__ == "__main__":
    print("Running testa")
    func_one()
```

Первый случай использования

Запустив его как основную программу с помощью `python testa.py`. Интерпретатор Python присвоит переменной `__name__` жестко закодированную строку «`main`», и таким образом код в операторе `if` будет выполнен:

```
# python testa.py

# Running testa
# Function One
```

Второй случай использования

Импортируем testa в качестве модуля.

Интерпретатор присвоит «testa» переменной `__name__` в модуле testa. Таким образом, код в операторе if не будет выполнен, и `func_one` не запустится.

```
# This is testb.py
import testa

if __name__ == "__main__":
    print("Running testb")
```

```
# python testb.py

# Running testb
```

PC File Edit View Navigate Code Refactor Run Tools VCS Window Help 1 - print_hi.py - Administrator

1) print_hi.py

Project 1 C:\Users\rilan\Desktop\2\1
 > venv library root
 print_C.py
 print_hi.py
 External Libraries
 > Python 3.9 (1) > C:\Users\rilan\Desktop\2\1\venv\Scripts
Scratches and Consoles

print_hi.py x print_C.py x

```
1 # This is a sample Python script.
2
3 # Press Shift+F10 to execute it or replace it with your code.
4 # Press Double Shift to search everywhere for classes, files, tool windows, actions, and settings.
5
6
7 def print_hi(name):
8     # Use a breakpoint in the code line below to debug your script.
9     print(f'Hi, {name}') # Press Ctrl+F8 to toggle the breakpoint.
10
11 print('__name__ = '__name__)
12 # Press the green button in the gutter to run the script.
13 if __name__ == '__main__':
14     print_hi('PyCharm')
15
16 # See PyCharm help at https://www.jetbrains.com/help/pycharm/
17
```

Run: print_hi x
C:\Users\rilan\Desktop\2\1\venv\Scripts\python.exe C:\Users\rilan\Desktop\2\1\print_hi.py
__name__ = __main__
Hi, PyCharm

Process finished with exit code 0

Version Control Run TODO Problems Terminal Python Packages Python Console Services
Packages installed successfully: Installed packages: 'matplotlib' (17 minutes ago)

17:1 CRLF UTF-8 4 spaces Python 3.9 (1)

File Edit View Navigate Code Refactor Run Tools VCS Window Help T - print_C.py - Administrator

1 > print_C.py

Project + - | Settings

1 C:\Users\rlan\Desktop\2\1
2 venv library root
3 print_C.py
4 print_hi.py

External Libraries
< Python 3.9 (1) > C:\Users\rlan\Desktop\2\1\venv\Sc
Scratches and Consoles

print_hi('C#')

Unresolved reference 'print_hi'

Import 'print_hi.print_hi()' Alt+Shift+Enter More actions... Alt+Enter

The screenshot shows the PyCharm IDE interface with a dark theme. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. A tab bar at the top right shows "1 - print_C.py - Administrator". The left sidebar has a "Project" view showing a folder structure with "print_hi.py" and "print_C.py" selected. Below it are "External Libraries" and "Scratches and Consoles". The main editor window contains the following Python code:

```
from print_hi import print_hi
print_hi('C#')
```

The "Run" tool window at the bottom shows the output of running "print_C.py":

```
C:\Users\rilan\Desktop\2\1\venv\Scripts\python.exe C:\Users\rilan\Desktop\2\1\print_C.py
__name__ = print_hi
Hi, C#
Process finished with exit code 0
```

The bottom status bar displays "PEP 8: W292 no newline at end of file", "3:15 CRLF UTF-8 4 spaces", and "Python 3.9 (1)".

Д/з

```
import numpy as np
from matplotlib import pyplot as plt
ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization")
plt.show()
```

- Заменить “Sample Visualization” на свою фамилию
- запустить в разных режимах, сделать скрины:
 1. Интерактивный режим (командный интерпретатор python)
 2. Скриптом из командной строки
 3. IDLE,
 4. Google.Collab
 5. PyCharm
 6. Visual Studio code