

# Информатика

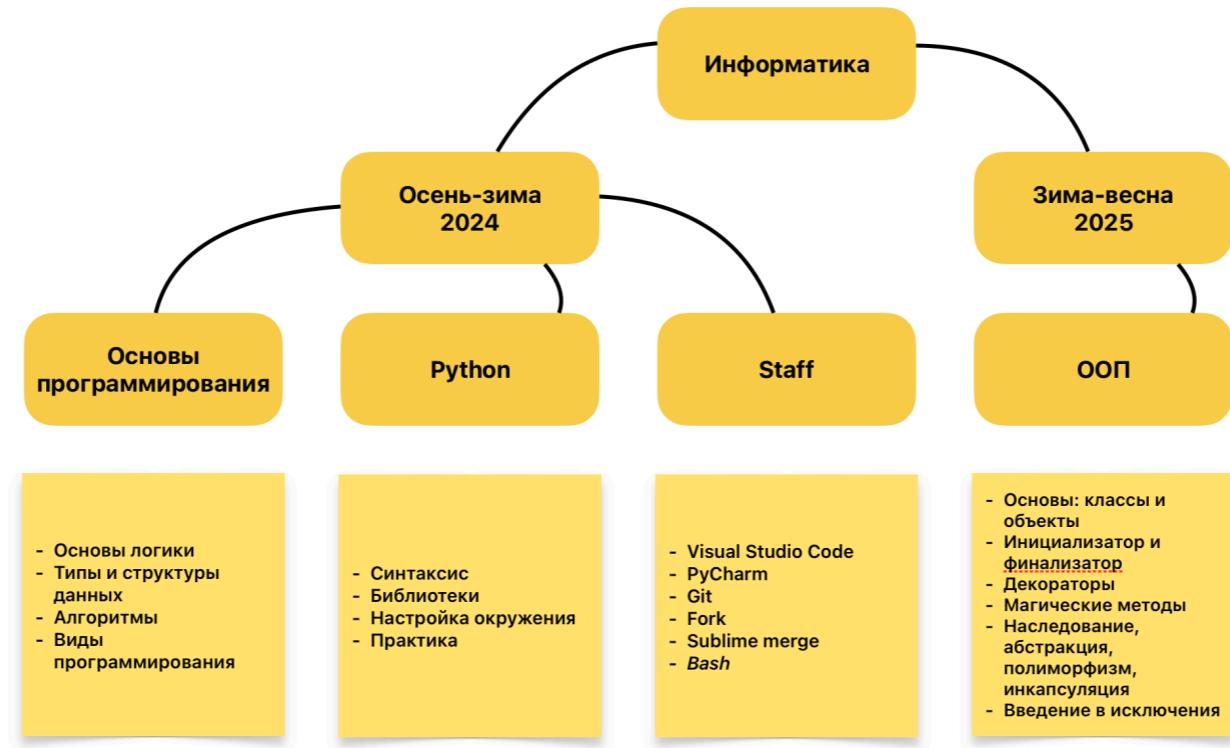
Баранов Максим Александрович

Осень 2024

# **О чём поговорим**

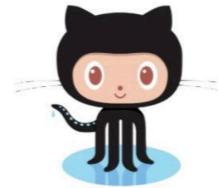
- Как устроен курс
- Что скачать и настроить
- Немного об истории программирования и языках
- Как получить зачет

# Структура курса

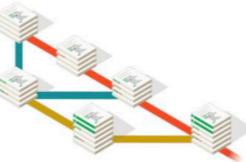


# GitHub

GitHub - интернет-сервис для хостинга git-репозиториев



Git – система контроля версий

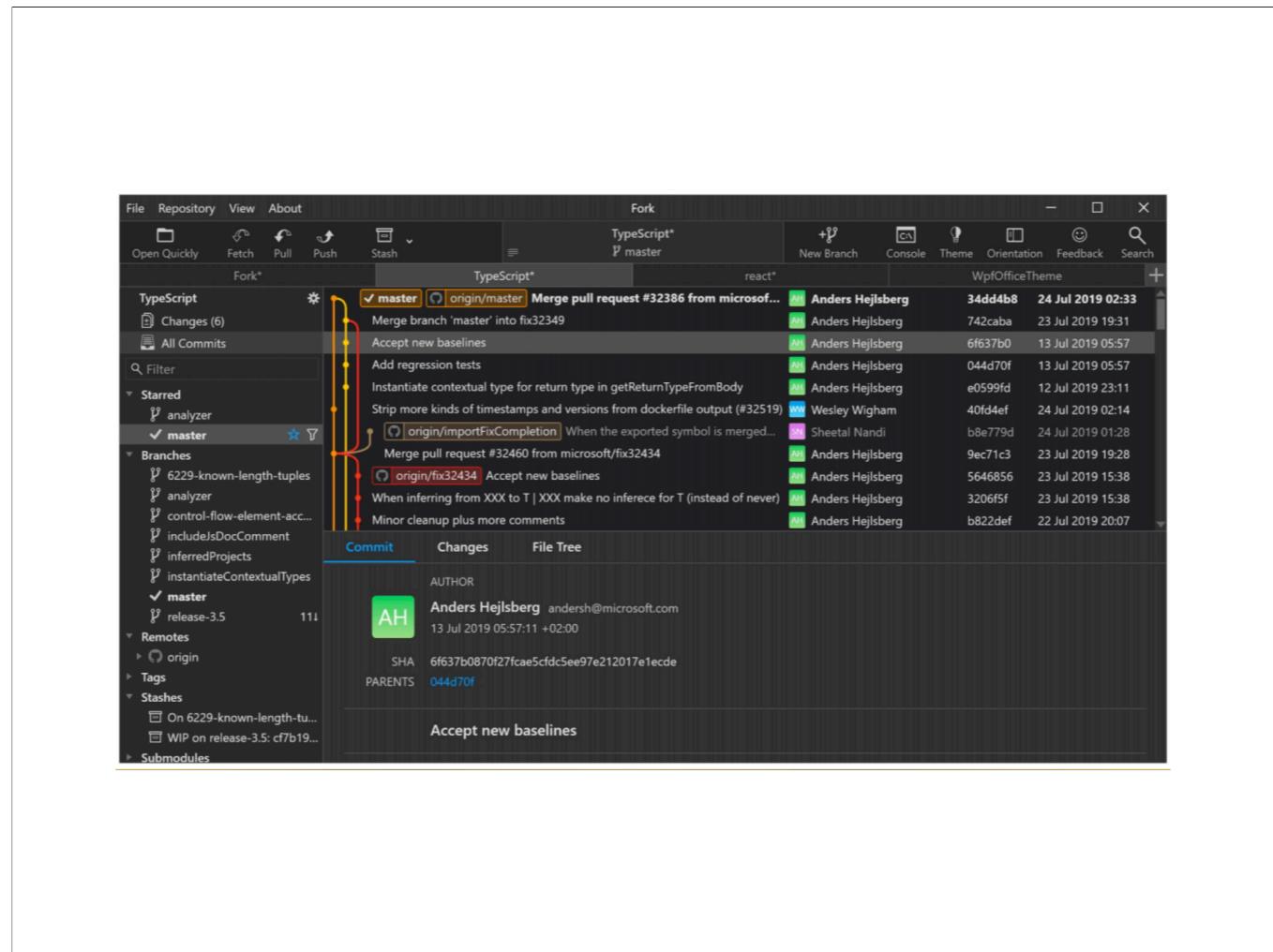


Git – это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но вы можете использовать его для любых файлов на ваш вкус). Изначально Git был создан Линусом Торвальдсом при разработке ядра Linux. Однако инструмент так понравился разработчикам, что в последствии, он получил широкое распространение и его стали использовать в других проектах. С его помощью вы можете сравнивать, анализировать, редактировать, сливать изменения и возвращаться назад к последнему сохранению. Этот процесс называется контролем версий.

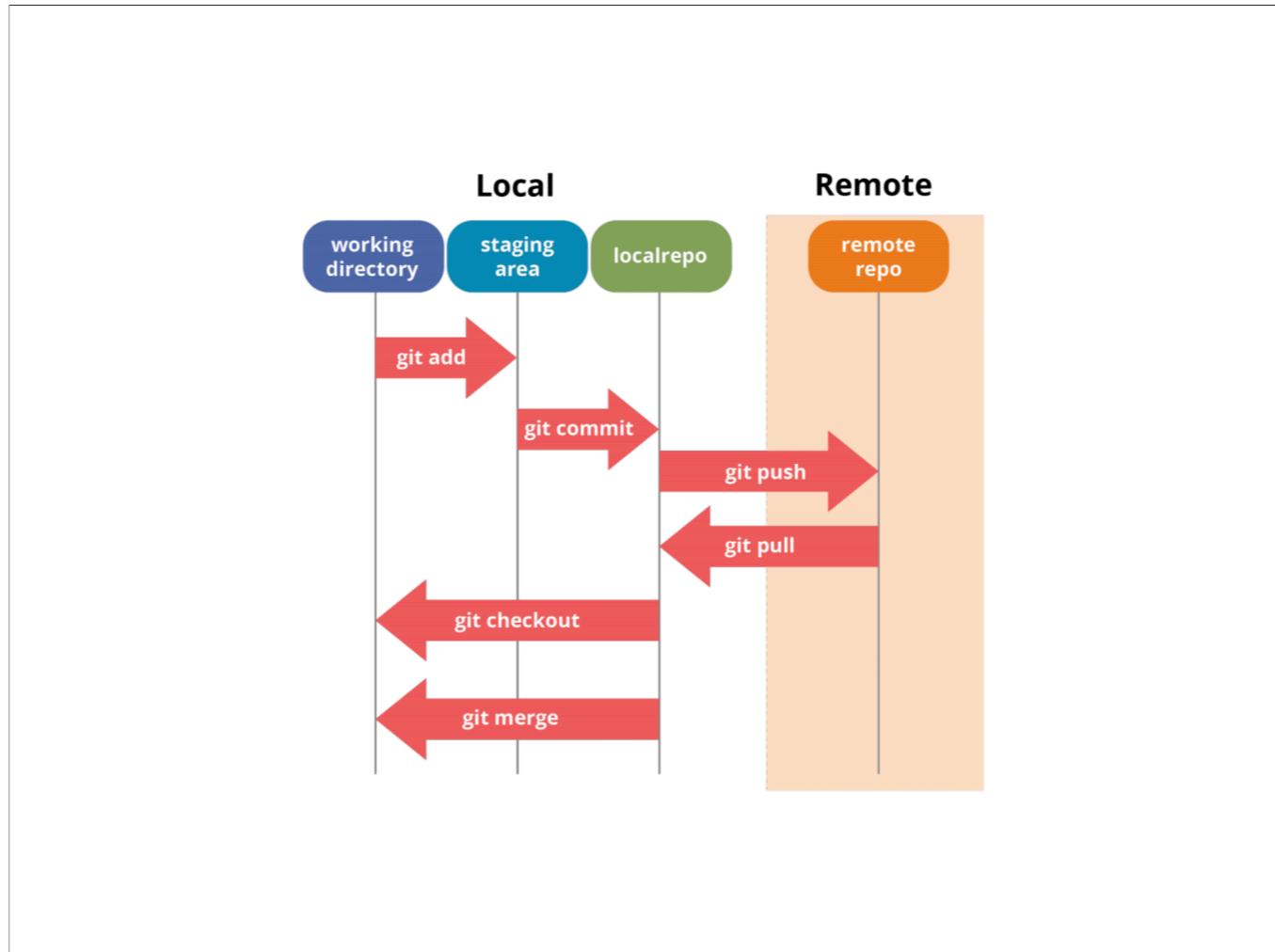
Для чего он нужен? Ну во-первых, чтобы отследить изменения, произошедшие с проектом, со временем. Проще говоря, мы можем посмотреть как менялись файлы программы, на всех этапах разработки и при необходимости вернуться назад и что-то отредактировать. Часто бывают ситуации, когда, во вполне себе работающий код, вам нужно внести определенные правки или улучшить какой-то функционал, по желанию заказчика. Однако после внедрения нововведений, вы с ужасом понимаете, что все сломалось. У вас начинается судорожно дергаться глаз, а в воздухе повисает немой вопрос: “Что делать?” Без системы контроля версий, вам надо было бы долго напряженно просматривать код, чтобы понять как было до того, как все перестало работать. С Гитом же, все что нужно сделать – это откатиться на коммит назад.

Во-вторых он чрезвычайно полезен при одновременной работе нескольких специалистов, над одним проектом. Без Гита случится коллапс, когда разработчики, скопировав весь код из главной папки и сделав с ним задуманное, попытаются одновременно вернуть весь код обратно. Git является распределенным, то есть не зависит от одного центрального сервера, на котором хранятся файлы. Вместо этого он работает полностью локально, сохраняя данные в директориях на жестком диске, которые называются репозиторием. Тем не менее, вы можете хранить копию репозитория онлайн, это сильно облегчает работу над одним проектом для нескольких людей. Для этого используются сайты вроде github и bitbucket.





<https://git-fork.com/>



# КНИГА

git --fast-version-control

Search entire site...

**About**

**Documentation**

Reference  
[Book](#)  
Videos  
External Links

**Downloads**

[Community](#)

This book is available in [English](#).  
Full translation available in  
български език,  
Deutsch,  
Español,  
Français,  
Ελληνικά,  
日本語,  
한국어,  
Nederlands,  
Русский,  
Slovenčina,  
Tagalog,  
Українська

[View all translations](#)

## Book

The entire Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 3.0 license](#). Print versions of the book are available on [Amazon.com](#).

  
2nd Edition (2014)

### 1. Введение

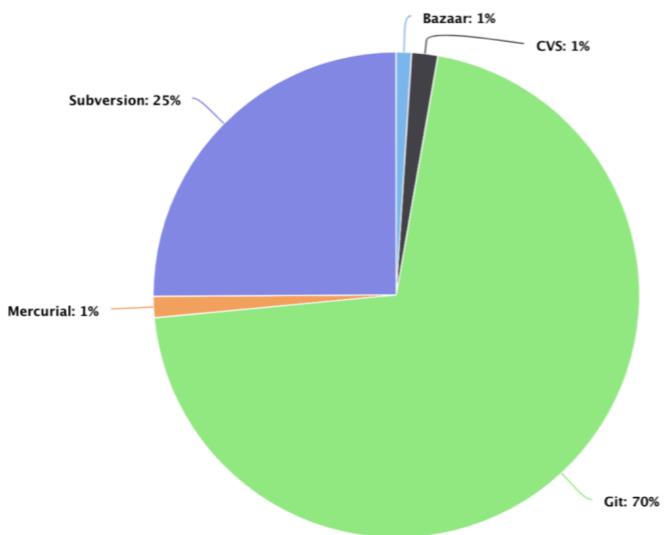
1.1 О системе контроля версий  
1.2 Краткая история Git  
1.3 Основы Git  
1.4 Командная строка  
1.5 Установка Git  
1.6 Первоначальная настройка Git  
1.7 Как получить помощь?  
1.8 Заключение

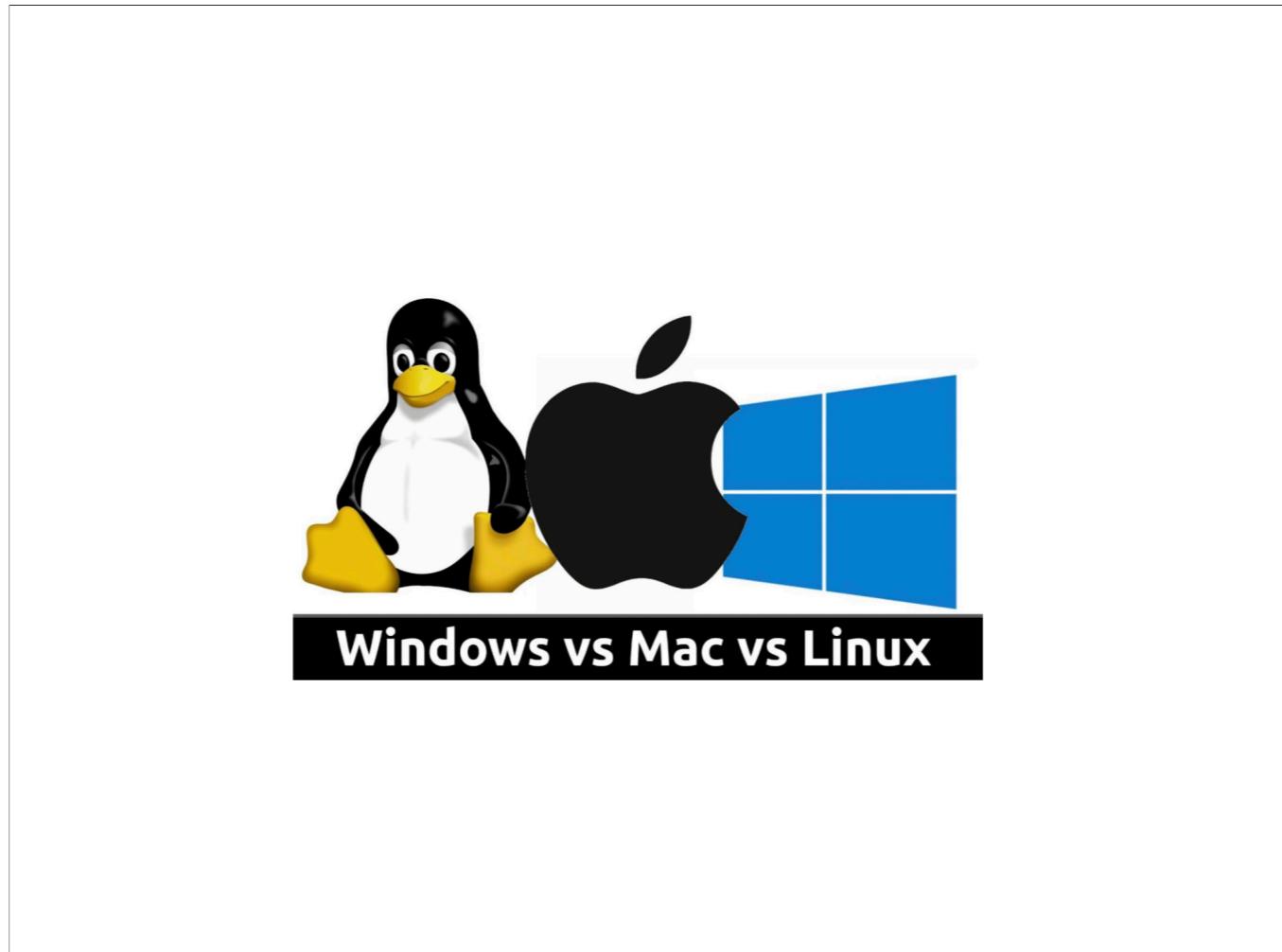
### 2. Основы Git

2.1 Создание Git-репозитория  
2.2 Запись изменений в репозиторий  
2.3 Просмотр истории коммитов  
2.4 Операции отмены  
2.5 Работа с удалёнными репозиториями  
2.6 Работа с метками  
2.7 Псевдонимы в Git

**Download Ebook**

 pdf    epub  
 mobi





## Операционная система Windows

Windows (Винда, Виндовс, «Окна» — от англ. «window») — это семейство операционных систем, разрабатываемых и выпускаемых корпорацией Microsoft с 1985 года. Сегодня линейка этих ОС включает разнообразные версии для домашних компьютеров (Windows 10/11 и ранее) и серверов (Windows Server 2019/2022 и ранее). Поддержка систем для мобильных устройств была полностью прекращена в 2019 году.

Интерфейс пользовательской Винды — знакомая всем среда с рабочим столом, иконками программ/папок и панелью задач. Кнопка «Пуск» стала неотъемлемой частью работы с этой ОС, обеспечивая удобный доступ к основным функциям и приложениям. Система Windows славится не только простотой использования, но и широкой совместимостью с программным обеспечением и играми, что делает ее привлекательным выбором для многих пользователей.

## Операционная система Windows.

## Операционная система Linux

Linux (Линукс) — это целое древо операционных систем с открытым исходным кодом, «растущих» из одного общего ядра. Собственно, Линукс — это и есть ядро, а все основанные на нем ОС (их тысячи) являются дистрибутивами (Ubuntu, CentOS, AlmaLinux, Rocky, Kali Linux, Fedora и др.). Они могут отличаться лицензированием, специализацией, интерфейсами и даже командами. Благодаря открытости и свободе распространения многих Линукс-дистрибутивов, а также самого ядра, пользователи могут полностью контролировать и настраивать систему под свои нужды, что делает Linux по сути конструктором операционных систем. Существует несколько условно базовых дистрибутива (Slackware, Red Hat, Debian), которые стали фундаментом для многих последующих ОС семейства, но есть также и бесконечное количество отдельных сборок, позволяющих пользователям выбирать оптимальные

конфигурации.

Линукс был создан в 1991 году Линусом Торвальдсом, с тех пор множество разработчиков по всему миру внесли свой вклад в развитие и усовершенствование ОС.

## Сравнение Windows и Linux

Поскольку и Винда, и Линукс представляют группы операционных систем, сравнивать их между собой не совсем корректно, но в целом между этими семействами существует ощутимая разница по многим аспектам. Ниже мы разберем основные моменты, которые отличают Винду от Линукса.

### Первый пункт

Первый пункт, который мы рассмотрим, это ядро операционной системы. Ядро является наиболее основным и значимым компонентом любой ОС. Ядро Linux является монолитным, состоит из одного единственного файла, в случае необходимости расширения функционала дополнительно используют специальные модули.

Общение программ с ядром происходит с помощью системных вызовов. Они стандартизированы, а это значит, что одно и то же ПО (программное обеспечение) без переписывания может функционировать на разных платформах под управлением Linux.

Драйверы встроены в ядро. Большое количество программ размещается в пространстве пользователя, учитывая графическую оболочку. Такая структура ядра намного безопаснее, потому что если на этапе сборки ядра отключить поддержку модулей, запустить свой код на уровне ядра будет нереально.

Windows имеет кардинально другой вид ядра. Оно состоит из множества небольших частей библиотек dll, каждая из которых отвечает за свою функцию. Системные вызовы вообще не применяются. В замену этому пользовательские программы обращаются к библиотекам user32.dll, gdi32.dll, kernel32.dll, advapi32.dll, которые вызывают функции из ntdll.dll (напрямую имеет отношение к ядру).

Библиотека hal.dll управляет драйверами, которые подключаются к ядру отдельно. Пользовательский режим ядра дает возможность просто адаптировать систему к любому ПО. Но за это приходится жертвовать производительностью системы.

### Второй пункт

Второй пункт – отличия в файловой системе и дисках.

ОС Linux от ОС Windows отличается структурой файловой системы и это заметить совсем не трудно. Файловая система Linux начинается с корня, то есть с основного каталога системного раздела, а уже там состоится подключение всех других дисков по необходимых подкаталогах.

Сортировка файлов происходит по каталогам, которые зависят от типа: исполняемые – в /bin/, настройки – /etc/, а ресурсы – в /usr/.

Устройства хранения в Linux размещаются в алфавитном порядке, а разделы на них с помощью цифр. Операционная система Windows подает все в виде абстракции. Учитывая то, что диски и разделы имеют похожую классификацию как и в Linux, но все это скрыто самой ОС. Пользователь видит лишь диски C:, D:, E:, F: и т.п. Каждый из них представляет собой раздел на жестком диске, а детальная информация скрыта, что даже и лучше для неопытных пользователей. Если рассматривать распределения файлов, то отдельная программа находится в одной папке, со всеми и файлами, настройками и

ресурсами, которые исполняются.

## Хранение настроек и данных ОС

Как сохраняются настройки Linux? Это происходит в обычных файлах в файловой системе. Глобальные файлы расположены в папке /etc/. Они задействованы ко всем пользователям, которые используют этот ПК (персональный компьютер). Настройки для программ пользователя располагаются в скрытых подкаталогах домашнего каталога пользователя.

Это вполне удобно, особенно при переносе файлов на другой ПК. Программы создают свои конфигурационные файлы, которые имеют собственный синтаксис и редактируются вручную. Настройки можно выполнять с помощью графического интерфейса, правда, это не всегда понятно для пользователя, поэтому именно ручной вариант более приемлемый и простой.

В отличии от Linux, операционная система Windows сохраняет все необходимые настройки в реестре Windows. Они разделяются по специальным ветвям и ключам, а доступ к ним происходит быстро.

Данный способ безопасный и предоставляет функцию удаленного изменения настроек при помощи графических программ. Это имеет свои минусы: настройки не переносятся на другой ПК, в случае, если централизованная система настроек повреждена, это вредит всей системе. ПО довольно быстро заполняет реестр и занимает много места. Поэтому определиться, что более удобно Linux или Windows, это уже дело личных предпочтений пользователей.

## Пользователи и права

Какие особенности управления? Linux – многопользовательская система. Три уровня доступа к файлам: пользователь-владелец, группа пользователей и другие. Доступно три параметра доступа: чтение, запись и выполнение. Списки доступа ACL, SELinux и AppArmor разработаны для обеспечения безопасности, правда, они не очень популярны.

Windows была разработана и рассчитана только на одного пользователя, и это влекло за собой проблемы в безопасности системы. Многопользовательская систем была добавлена немного позже. Она включает, кроме владельца, группы и других, подробные ACL списки доступа. Поэтому данное отличие этих двух ОС не слишком значительное.

## Особенности управления программами и обновлениями

В этом пункте ощущается большая разница между операционными системами Windows и Linux. Рассмотрим почему.

Linux имеет репозитории пакетов ПО. Нет особой необходимости скачивать программы с Сети. А это и безопасность, и надежность и возможность обновления. Процесс обновления происходит удобно с помощью одной команды сразу для всей системы у удобное для пользователя время.

Windows не имеет репозиториев. Необходимое ПО необходимо скачивать и самостоятельно устанавливать. Программы обновляются сами как и ОС, иногда в очень неудобное время для пользователя. А чтобы обновить, необходимо перезагрузить систему.

Как видим, отличия между этими двумя популярными операционными системами значительны. Но выбор ОС дело вкуса. Одним пользователям нравится Linux, другие не представляют свою жизнь без Windows. Использование той или иной ОС зависит, прежде всего, от целей и задач, которые преследует пользователь.

### Лицензирование

Windows распространяется по проприетарной лицензии, что означает, пользователи должны платить за использование продукта. Кроме того, у них нет доступа к исходному коду системы. Ядро закрыто от вмешательства кем-то помимо Microsoft.

В отличие от ОС Винды, большинство дистрибутивов Linux предоставляются с лицензией GPL (General Public License), которая дает пользователям свободу использовать, изменять и распространять систему в рамках определенных условий. Есть платные Линуксы, но все же обычно работать с дистрибутивами можно бесплатно.

### Простота использования

Мы уже отметили, что Windows имеет интерфейс, привычный для подавляющего большинства людей. Его очень просто использовать: все действия пользователя обычно сводятся к нажатию кнопок «Далее» — «Далее» — «Готово» и им подобным. В Винде есть удобные утилиты, которые решают любую проблему за пару кликов. Это хорошо для рядовых пользователей.

Линукс в большинстве своем является специфическим программным обеспечением, где очень многое нуждается в ручной настройке. Даже при наличии графического интерфейса человеку понадобятся более глубокие знания, чем элементарная компьютерная грамотность, чтобы установить нужные конфигурации.

### Работа службы поддержки

Винда — коммерческий продукт, поэтому пользователи этой системы могут обращаться за поддержкой в Microsoft на правах обладателей лицензий. Компания предоставляет обширную службу поддержки своим клиентам.

С Linux сложнее. В зависимости от лицензии, техподдержка может оказываться либо сообществом, либо компаниями-поставщиками дистрибутивов. Первый вариант реализуется куда чаще, но и сообщество разработчиков здесь хорошо развито, так что найти помочь не составит труда.

### Кастомизация

Открытый код Линукс обеспечивает всем системам этого семейства максимальную гибкость в кастомизации структуры и функциональности. Пользователь может изменять в Linux практически все, начиная от интерфейса и заканчивая ядром.

В Винде эта возможность крайне ограничена. У пользователя нет доступа к глубоким настройкам системы, он может контролировать лишь некоторые «поверхностные» функции.

### Скорость работы

Этот аспект будет зависеть от версии ОС, настроек и задач, которые выполняются каждой конкретной системой. Но при прочих равных Linux, как правило,

более производителен, чем Винда. Он эффективнее использует вычислительные ресурсы (особенно в легких дистрибутивах), его оболочки (такие как Xfce и LXQt) менее ресурсозатратны по сравнению с классическими оконными менеджерами Windows. Как итог, Linux-системы дают лучшую загрузку и быстрее выполняют задачи на слабом оборудовании. Винда, к сожалению, утяжелена большим количеством легаси-кода для совместимости со старыми программами. Линукс опережает Виндовс по скорости уже хотя бы потому, что дает пользователю возможность удалять любой ненужный код и неиспользуемые компоненты.

### Персонализация

Windows предоставляет ряд возможностей для изменения внешнего вида интерфейса: можно менять тему, обои и экран блокировки, иконки, цветовые и звуковые схемы, шрифты. Чтобы персонализировать что-то дополнительно, нужно обращаться к стороннему ПО.

Linux выходит далеко за пределы этого — опять же, благодаря открытости кода. ОС этого семейства предлагают бесконечные варианты персонализации, начиная от тем оформления и заканчивая компоновкой окон и даже полной сменой интерфейса.

### Конфиденциальность

Отличия в конфиденциальности между Windows и Linux частично обусловлены разными подходами к дизайну и философии этих операционных систем. Винда — проприетарная ОС, поэтому политика защиты личных данных пользователей устанавливается Microsoft. Она предоставляет инструменты для управления конфиденциальностью, но многие пользователи обеспокоены относительно сбора их данных корпорацией.

В Линукс весь код прозрачен, над ним постоянно трудится огромное мировое сообщество, что также обеспечивает общественный контроль за тем, как собираются и обрабатываются данные в дистрибутивах.

### Безопасность

В погоне за удобством пользователя, а также из-за повсеместного использования на домашних компьютерах Винда превратилась в главную цель для хакеров. В программировании удобство для пользователя означает удобство для вирусов, поэтому большая часть существующего вредоносного ПО создается именно под Windows. Майкрософт регулярно обновляет свои системы и устраняет уязвимости, но все равно без антивируса пользоваться Виндой довольно рискованно.

В свою очередь Линукс славится хорошим уровнем безопасности за счет разнообразия в структуре среди дистрибутивов и активной работы сообщества над устранением дыр. Кроме того, чтобы вирус мог навредить Linux-системе, должен использоваться root-доступ, дать который может только сам пользователь. Без него внести какие-то существенные изменения в систему не получится.

### Надежность

С самого первого выпуска Linux был ориентирован на обеспечение бесперебойной работы системы. По сей день безопасность и эффективное управление процессами являются главными преимуществами Линукс-дистрибутивов и придают им заслуженный статус стабильных и надежных операционных систем, особенно для использования на серверах.

И хотя Microsoft заметно повысила уровень надежности Винды, в этом вопросе она по-прежнему невыгодно отличается от Линукса. После установки обновлений ей чаще требуются перезагрузка, а из-за стремления к простому доступу и удобному использованию ОС юзером в ней упущены важные функции, отсутствие которых сказывается на стабильности и способствует появлению уязвимостей.

### Обновления

Windows имеет централизованную систему обновлений, что облегчает управление ими для пользователя. Microsoft регулярно выпускает обновления для

своей ОС, включая патчи безопасности, новые функции и улучшения производительности. При этом поддерживается Винда в течение всего срока ее использования.

В Линуксе обновления управляются менеджерами пакетов. Это требует больших усилий, но и дает больший контроль.

Linux vs Windows: основные моменты, которые помогут сделать выбор

Как и все остальное в жизни, выбирать операционную систему нужно с учетом ваших личных потребностей и предпочтений. Если вам нужна простота использования, совместимость с большим количеством программ и игр, то стоит остановиться на Винде. Если вам важна свобода, возможность кастомизации системы и высокая степень контроля над ней, Линукс-дистрибутивы — подходящий вариант.

## Заключение

Обе операционные системы имеют свои преимущества и недостатки, и решение должно быть принято на основе тщательного анализа ваших требований. Надеемся, наша статья помогла вам определиться с выбором ОС.

# Преимущества Linux

1 преимущество Linux для программистов — возможность научиться понимать связи кода, управлять ими и решать нестандартные задачи.

2 преимущество — автоматизация и логичность ОС, за счёт чего на выполнение задач уходит меньше времени.

3 преимущество — возможность читать открытый существующий код других разработчиков и таким способом развиваться.

4 преимущество — наборы слоёв абстракции в Linux, с их помощью в разы уменьшается объём работы.

5 преимущество — прямой доступ к периферийным устройствам и максимально простой интерфейс платформы.

# Пример

Приведём пример. Необходимо добавить изображение обложки к 40 файлам PDF

1. Запустим документ в PDF-редакторе.
2. Откроем главный файл с обложкой.
3. Добавим к нему документ PDF.
4. Сохраним этот файл как новый PDF-файл.
5. Проделаем всё это для всех старых документов PDF.

# Пример

Приведём пример. Необходимо добавить изображение обложки к 40 файлам PDF

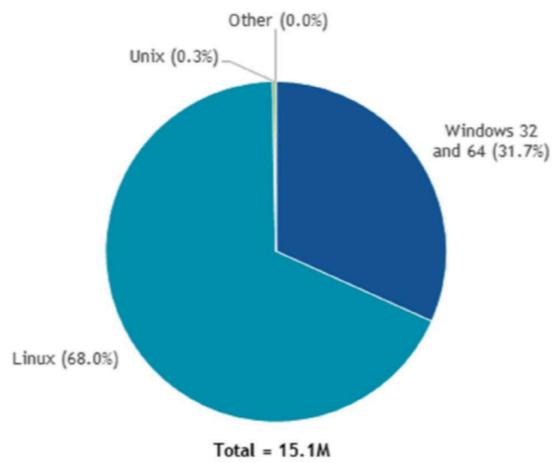
1

```
pdftk A=cover.pdf B=document_1.pdf \ cat A B \ output doc+cover_1.pdf
```

2

```
$ find ~/docs/ -name "*.pdf" | \ parallel pdftk A=cover.pdf B={} \ cat A B \ output  
{}.cover.pdf
```

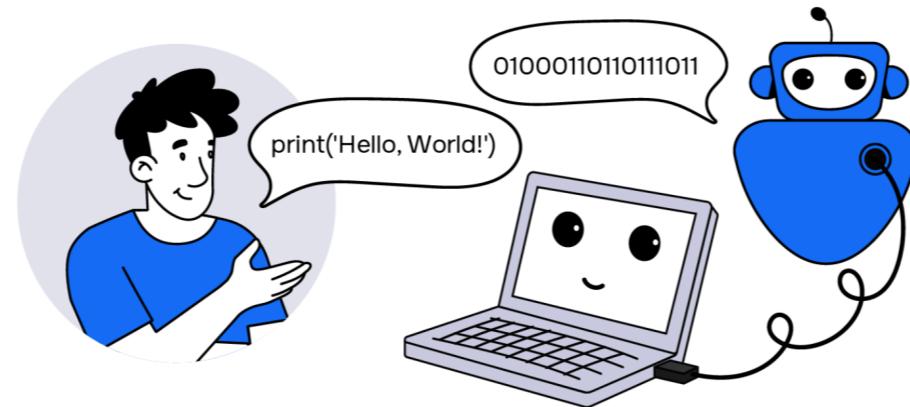
**Worldwide Server Operating Environment Shipments/Subscriptions and Nonpaid Deployment Share by Operating Environment, 2017**



Aug 2023	Aug 2022	Change	Programming Language	Ratings	Change
1	1		 Python	13.33%	-2.30%
2	2		 C	11.41%	-3.35%
3	4		 C++	10.63%	+0.49%
4	3		 Java	10.33%	-2.14%
5	5		 C#	7.04%	+1.64%
6	8		 JavaScript	3.29%	+0.89%
7	6		 Visual Basic	2.63%	-2.26%
8	9		 SQL	1.53%	-0.14%
9	7		 Assembly language	1.34%	-1.41%
10	10		 PHP	1.27%	-0.09%

# Виды языков программирования

- Компилируемые и интерпретируемые языки;
- Низкоуровневые и высокоуровневые языки;



На международные деловые встречи делегаты ездят с переводчиками, так как сами они могут не понимать иностранную речь.

Процессор, «мозг» компьютера, тоже не понимает код, который пишет разработчик на языке программирования. Процессор воспринимает лишь последовательность из нулей и единиц. Понять друг друга разработчику и компьютеру тоже помогают специальные переводчики: компиляторы и интерпретаторы.

Компилятор переводит весь код в машинный (состоящий из нулей и единиц) сразу же при запуске программы. Работу компилятора можно описать так:

Программист пишет программу на языке программирования → Запускает компилятор → Компилятор переводит всю программу в машинный код и кладет его в исполняемый файл (.exe в Windows)

К компилируемым языкам программирования относят C, C++, Java, Swift, Go.

Интерпретатор переводит код в машинный построчно каждый раз, когда запускается программа.

Программы, написанные на интерпретируемых языках, чаще всего запускаются медленнее компилируемых программ — как раз из-за построчного перевода кода. Однако процесс разработки на интерпретируемом языке быстрее, чем на компилируемом, так как программисту не нужно снова и снова компилировать программу в машинный язык.

К интерпретируемым языкам программирования относят Python, PHP, JavaScript.

## Низкоуровневые и высокоуровневые языки

До изобретения компилятора и интерпретатора программисты вручную писали код на машинном языке. Код, созданный на таком языке, разный для каждого компьютера, так как у каждого производителя процессоров свой собственный набор инструкций к написанию кода.

Разработчики тратили много времени и усилий, чтобы писать код на машинном языке. Со временем они научили компьютер понимать языки программирования, близкие по написанию к нашим естественным языкам. Так появилось условное деление языков программирования на высокоуровневые и низкоуровневые (куда и отнесли машинный язык).

К низкоуровневым языкам также отнесли язык ассемблера — надстройку над машинным языком. На нем программистам писать код чуть проще, чем на машинном. Однако все так же не слишком удобно — язык ассемблера, как и машинный язык, не содержит функций, структур данных, списков, объектов.

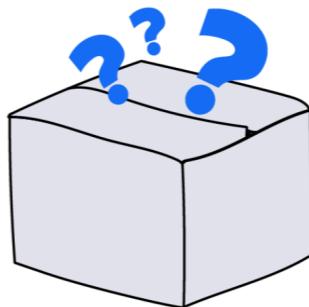
Код на высокоуровневых языках легко читаем и понятен программисту. Такие языки позволяют создавать программы, не переживая о совместимости кода с разными процессорами. Высокоуровневые языки требуют компиляции или интерпретации. К таким языкам относят Java, JavaScript, Python, Ruby, PHP и другие.

# Виды языков программирования

- Языки со статической типизацией и динамической типизацией;
- Универсальные и специальные языки;
- Эзотерические языки;
- Визуальные языки.



Статическая типизация



Динамическая типизация

## Языки со статической типизацией и динамической типизацией

Представьте, что вы переезжаете и собираете коробки с вещами. Чтобы понять, в какой из коробок лежат вещи для кухни, гостиной и спальни, вы их подписываете. Это и есть типизация.

Типизация — это набор правил, по которым язык программирования классифицирует информацию. Благодаря типизации компьютер понимает, какие данные ему нужно обработать, что с ними делать и какой объем памяти займет предстоящая операция.

Если у языка нет типизации, программист может присваивать переменной любой тип данных (строку, число), а потом отнести к этой же переменной другой тип данных. Это позволяет быстрее писать код, но в таком коде проще запутаться. Так же, как можно запутаться в неподписанных коробках.

Коробки можно подписывать по-разному — и у языков программирования типизация бывает разной. Статическая типизация определяет типы данных в программе до ее запуска (во время компиляции).

Языки со статической типизацией — Java, C++, Swift.

В динамически-типованных языках тип переменной определяется во время запуска программы. Объявлять тип переменной в явном виде не нужно: интерпретатор определяет его в процессе работы программы. Программу, созданную на языке с динамической типизацией, сложнее отлаживать и расширять ее функциональность.

## Универсальные и специальные языки

Существуют языки, которые могут использоваться для различных целей и областей применения. Их называют универсальными. Например, C++ подходит как для разработки несложных вычислительных программ, так и для создания высоконагруженных приложений, игр.

Специальные языки программирования (или доменно-специфические) подходят для решения определенного круга задач. Это, например, SQL, на котором можно писать запросы к базам данных, HTML или CSS, предназначенные для проектирования и верстки веб-страниц.

## Эзотерические языки

Существует группа языков, которая не подходит для решения практических задач, а применяется для проведения экспериментов или развлечения.

Эзотерические языки дают возможность разработчикам воплощать необычные идеи, исследовать возможности языков, а также помогают развивать гибкость ума и воображение.

Так программа на языке программирования Piet выглядит как абстрактная картина, Chef записывает программы в качестве кулинарных рецептов, а команды ArnoldC представляют собой цитаты из фильмов с Арнольдом Шварценеггером.

## Визуальные языки

Иногда большие объемы данных сложно описать текстовым кодом. Зато их удобно разбить по блокам и представить в виде схемы. В таких ситуациях удобно использовать визуальные языки программирования. Так код будет написан при помощи графических элементов (блоков и изображений).

Работа с визуальными языками подходит для начинающих специалистов, так как написание кода выполняется более просто и интуитивно. К визуальным языкам относят Scratch, Blockly, Substance Designer.

## Язык программирования PYTHON



- высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение кроссплатформенности написанных на нём программ.

**Разработчик языка Python – голландский программист Гвидо ван Россум**



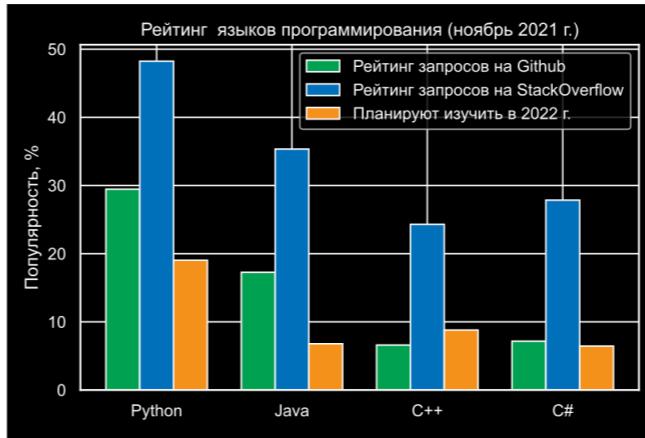
## Почему PYTHON?

### Кто использует Python?

- Google
- YouTube
- Dropbox
- BitTorrent
- iRobot
- Netflix и Yelp
- Intel, Cisco
- NASA

### Сильные стороны

- Качество программного кода
- Продуктивность труда
- Кроссплатформенность
- Библиотеки



## 1. Синтаксис

Язык имеет простой синтаксис и отлично подходит начинающим программистам. Код Python чист и прост для понимания. Вот несколько принципов языка:

- Красивое лучше, чем уродливое.
- Простое лучше, чем сложное.
- Читаемость имеет значение.

Python также располагает хорошей системой обработки и отчётов об ошибках, что тоже упрощает работу.

## 2. Отлично подходит для старта в программировании.

Python доступен для новичков. Если вы только делаете первые шаги в программировании Python – это отличный выбор. Изучение этого языка требует сравнительно немного времени. Помимо понятности и простоты, его преимуществом является доступность обучающих материалов. Это сильно упрощает освоение языка. Существует множество доступных курсов, которые позволяют вам освоить основы Python. Например, наш Python. Base pack: </courses/python-base-pack/>

## 3. Множество доступных сред разработки

Большая часть того, вам захочется делать на Python, уже написана за вас в стандартной библиотеке. Можно легко найти бесплатные продукты, которые позволяют значительно сэкономить время. Главное преимущество того, что код был написан до вас — он точно не содержит ошибок.

## 4. Универсальность

Python – кроссплатформенный и мультинаправленный язык. Он подходит для огромного множества целей: веб-разработка, разработка игр и приложений, анализ и визуализация данных, машинное обучение, искусственный интеллект и т.д. Для приложений python практически нет ограничений.

## 5. Python – быстрорастущий язык

Python- это самый популярный высокоуровневый язык программирования. Число людей, использующих этот язык программирования, растет стремительными темпами ежегодно. Изучив этот язык, вы станете частью большого комьюнити единомышленников.

## 6. Востребованность на рынке

Спрос на специалистов, владеющих этим языком, только растет. Вы легко можете в этом убедиться, посмотрев количество вакансий на сайтах. Например, грамотных аналитиков данных на Python сейчас сильно не хватает. Серьезных специалистов не так много и компаниям сложно найти работников. Спрос превышает предложение, поэтому освоив эту профессию, вы не останетесь без работы.

Освоить высокооплачиваемую профессию аналитика данных можно на нашем курсе:

[/courses/data\\_scientist/](/courses/data_scientist/)

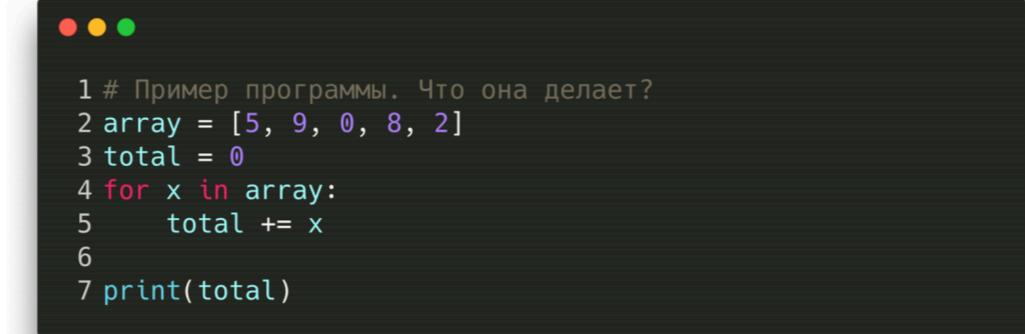
## Интерпретаторы python



Jython — реализация языка Python для выполнения программ на платформе Java. Первоначальное имя проекта — JPython, которое пришлось поменять из-за конфликта с одноимённым проектом (имя было занято на хостинге sourceforge.net). Является одновременно и компилятором, и интерпретатором.

Stackless Python, или просто Stackless — версия интерпретатора языка программирования Python, названная так из-за отказа от использования стандартного стека вызовов языка Си в пользу собственного стека. Наиболее впечатляющей особенностью Stackless являются микропотоки (англ. microthreads), которые позволяют избежать чрезмерного расхода системных ресурсов, присущего стандартным потокам операционной системы. В дополнение к стандартным возможностям Python, Stackless добавляет поддержку сопрограмм, коммуникационных каналов и сериализации задач.

PyPy — интерпретатор языка программирования Python. PyPy в начале своего существования был интерпретатором Python, написанным на Python. Текущие версии PyPy транслируются из RPython в Си и компилируются. В PyPy встроен трассирующий[англ.] JIT-компилятор, который может превращать код на Python в машинный код во время выполнения программы.



A screenshot of a macOS terminal window. The window has a dark background and three colored window control buttons (red, yellow, green) at the top left. The terminal text area contains the following Python code:

```
1 # Пример программы. Что она делает?
2 array = [5, 9, 0, 8, 2]
3 total = 0
4 for x in array:
5     total += x
6
7 print(total)
```

```
1 # Питонисткий путь
2 array = [5, 9, 0, 8, 2]
3 total = sum(array)
4
5 print(total)
```

## Шаг 1. Типы

```
1 # Простые типы
2 a = 1
3 b = 2.4
4 c = False
5 d = None
6 e = 42j # комплексные числа
7
8 # Страна
9 f = 'это строкка в юникоде \u041c'
10
11 # Последовательности
12 g = [1,2,3,4,5,6] # список
13 h = (1,'a',3.141) # кортеж (read only)
14 i = range(10) # диапазон [0,10)
15
16 # Словарь и множество
17 j = {'название': 'политех', 'дата': 1899} # словарь
18 k = {1,2,3,4,5,6} # множество
```

## Шаг 2. Операции

```
1 # Задаем имена
2 x = 1
3 y = 2
4 z = 3
5
6 # Можно задать в одну строчку
7 x, y, z = 1, 2, 3
8
9 # Переопределяем имя (не копируем!)
10 x = y
11
12 # Лайфхак: «обмен переменных»
13 x, y = y, x
14
15 # Арифметические операции (в порядке возрастания приоритета)
16 z = x + y
17 z = x * y
18 z = x / y
19 z = x // y # целочисленное деление (floor)
20 z = x % y # остаток от деления
21 z = x ** y # степень
```

### Шаг 3. Управление

```
1 # Управление
2 # >, >=, <, <=, ==, !=, is
3 # and, or, not
4 if a > b:
5     pass
6 elif a == b: # elif = else if
7     pass
8 else:
9     pass
10
11 # Циклы
12 for x in range(10): # диапазон [0,10)
13     pass
14 else: # дополнительно
15     pass
16
17 while a:
18     pass
19 else: # дополнительно
20     pass
```

## Шаг 4. Функции

```
1 # Функции
2 def func_name(x):
3     return x + 1
4
5 # Вызов функции (вернет 4)
6 func_name(3)
7 func_name(x=3) # можно явно указать название аргумента
```

```
345.py    X
old lectures > 345.py > ...
1  l = [5,6,7,8]
2  total = 0
3  for i in range(len(l)):
4      val = l[i]
5      total += val
6
7  print(total)
8
9  # Чуть проще
10 total = 0
11 for x in [5,6,7,8]:
12     total += x
13
14 print(total)
15
16 # Pythonic
17 total = sum([5,6,7,8])
18 print(total)

PROBLEMS    OUTPUT    TERMINAL    PORTS    DEBUG CONSOLE
/usr/local/bin/python3 /Users/max/Yandex.Disk.localized/Courses/Informatics/Inf
● max@LoAir Informatics % /usr/local/bin/python3 /Users/max/Yandex.Disk.localized
26
26
26
○ max@LoAir Informatics %
```

## ЧТО ЧИТАТЬ?

