

[Save Copy to Evernote](#)

Last updated: May 30, 2020

Что надо знать Python веб-разработчику для успешного поиска работы?

Это текстовая версия видео <https://www.youtube.com/watch?v=9kLI6R0heTQ>

Что надо знать Python веб-разработчику для успешного поиска работы?

Был вопрос в комментариях о том, что надо знать Python веб-разработчику для успешного приёма на работу — я попытаюсь ответить на этот вопрос. Это будет моё мнение, оно в каких-то моментах может отличаться от мнений других специалистов, но если вы будете знать и уметь то, о чём я расскажу в этом видео, вы с большой вероятностью сможете найти работу быстро.

При этом я осознанно не буду говорить здесь об уровне разработчика — разница между джуниором, middle и серворм разработчиком с точки зрения разных людей разная, и с точки зрения разных компаний тоже разная.

Чем больше из моего перечня вы знаете и умеете, тем лучше и тем выше ваши шансы быстро найти хорошую работу. В минимальном объёме вы должны знать и понимать все озвученные темы хотя бы по основам, но, понятно, что чем глубже вы будете их понимать, тем лучше.

И да, список будет длинным, я не хочу никого этим демотивировать — но люди, которые идут в отрасль тупо за зарплатой, считая, что здесь легко можно получить 150к за умение тыкать в кнопки, пусть немногого призадумаются. Чуть-чуть. Окей, поехали!

Итак, мы с вами говорим о веб разработке, а значит надо понимать, как собственно работает веб. Что входит в понимание того, как работает веб. Надо понимать, как в целом работает интернет, что такое IP адреса и как это все связывается в глобальную мировую сеть.

Надо знать, что такое DNS и как он работает, то есть что происходит, когда в браузере вы переходите на youtube.com. Надо понимать про hosts файл, которым вы можете изменить DNS настройки. Как физически запустить что-то на своём домене 2 или 3 уровня, то есть как привязать к домену свой сервер. Причем это надо уметь настроить физически на линукс сервере, например, в nginx.

Надо знать о разнице HTTP и HTTPS протоколов и опять же уметь ручками настроить на вашем домене работу по защищенному HTTPS протоколу.



[Terms of Service](#)[Privacy Policy](#)[Report Spam](#)

я и т.д.) запросов, чем они отличаются и
т. Какие есть основные HTTP статусы и
т за что они отвечают.

[Save Copy to Evernote](#)

Круто, если вы пришли с конспекта лекции №2. Всё работает на уровне HTTP.

Конечно, надо понимать, как работают и для чего нужны Cookies, какие они бывают, как с ними работать из JavaScript с клиентской части и как с ними работать с серверной части.

Как работают авторизация и аутентификация на веб-сайтах и в мобильных приложениях, которые хранят данные на сервере. Кстати, разницу между авторизацией и аутентификацией тоже надо понимать.

В идеале вы должны уметь настроить механизм сессий без веб-фреймворков, просто чтобы понимать, как это всё работает под капотом вашего любимого фреймворка, того же Django. Чтобы у вас было настоящее понимание о том, что там никакой магии не происходит.

Итак, это про базовое понимание того, как работает интернет и веб. Это азы и это база, которую надо знать веб-разработчикам.

Идём дальше.

Я глубоко убеждён в том, что все, все веб-разработчики должны знать хотя бы азы фронтенд технологий. Что сюда входит. Конечно же, HTML, язык гипертекстовой разметки. Конечно же, CSS, язык каскадных стилей, который позволяет оформлять веб-страницы. Используя HTML и CSS вы должны уметь минимально верстать страницы и какие-то базовые блоки. Я не говорю о том, что вы должны уметь это делать профессионально, но вы должны знать о семантической разметке, о настройке шрифтов, местоположений блоков на странице, margins и paddings, уметь использовать flex-боксы, флоаты, таблицы и тд. Вы должны вполне себе базово уметь верстать. Кстати, про адаптивную верстку тоже хорошо бы знать и уметь, все эти media queries и тд.

Идем дальше. JavaScript. Я глубоко убежден, что все веб-разработчики должны знать хотя бы азы JavaScript. В 2020м году надо знать уже не JS фреймворк JQuery, а нативный нормальный JavaScript и JavaScript API браузера. Document.querySelector, работа с CSS стилями из JavaScript, переменные, циклы, функции Javascript, работа с AJAX запросами (и на стороне фронтенда, то есть JS, и на стороне бэкенда, то есть Python"а), модель асинхронности JavaScript. Если знаете еще какой-то JS фреймворк (реакт или Vue), будете вообще красавчиком.

Итак, это с точки зрения фронтенда. Идём дальше, непосредственно к бэкенду.

Python. Конечно, нам надо знать Python. Что входит в серьезное знание любого языка программирования. Это а) знание его синтаксиса, б) знание его стандартной библиотеки, в) знание его основных third-party, то есть внешних библиотек.

Всё это вы должны знать максимально глубоко, если это ваша основная рабочая сфера деятельности. Обратите внимание, знание языка это не знание синтаксиса, этого пипец как мало. Понятно, надо знать синтаксис — определение переменных, функций, модулей и

пакетов, ветвления и циклы.

Save Copy to Evernote

Типы данных — числа, строки, последовательности и как с ними работать.

Последовательности все должны быть на кончиках пальцев — листы, дикты, таплы, множества. Как с ними работать, преобразовывать друг в друга, какие особенности. Слайсы и тд.

Форматирования строк, приведение типов и тд. Классы и объекты. Аргументы в функциях — позиционные и keyword, args и kwargs. Функции map, reduce, zip. Лямбда функции. Генераторы, декораторы. Исключения.

Всё это должно быть для вас просто как ваши пять пальцев. Вы не должны вообще задумываться об этих конструкциях языка, когда вы их используете. Для этого у вас должно быть достаточно количество практики.

Когда вы говорите на своём родном языке, например, русском, вы же не задумываетесь по ходу речи о существительных, прилагательных и глаголах, вы думаете о смысле, который вы при помощи своей речи передаёте. Точно так же у вас должно стать и с вашим основным языком, например, Python. Вы пишете код и думаете не о конструкциях языка, а о той логике, которую ваш код реализует, о той задаче, которую вы в данный момент программируете. Знание синтаксиса языка это абсолютно базовое знание, оно важно, но его мало.

Асинхронность в питоне важная часть языка. Какая она бывает, в чем разница, когда что имеет смысл применять. Надо нормально уметь пользоваться asyncio, конечно.

Знание станд библиотеки языка. Работа с файлами (обычными текстовыми и CSV) и аргументами командной строки. Пакет collections. Регулярные выражения. Логирование тоже очень важная тема. Работа с pickle дампами, с JSON. Работа с датами очень частая задача и вам нужно уметь свободно работать с датами и временем, в том числе с временными зонами (там pytz и тд).

Third-party библиотеки. Сюда входит например requests для работы с HTTP запросами. Aiohttp для отправки асинхронных запросов и в качестве асинхронного веб-сервера. Библиотеки для работы с Excel документами, PDF, XML, HTML (там beautiful soup тот же). Уметь работать с Celery и Redis.

Веб-фреймворки. Сюда относятся Django, Flask, Tornado и в принципе aiohttp. Всё знать не надо, но Django это мейнстрим, если знать только что-то одно, то его. Flask это микро-фреймворк, он маленький и простой, поэтому в целом его тоже можно изучить. Tornado сейчас изучать для написания на нем новых проектов наверное не стоит, лучше изучить aiohttp.

Итак, django. Что входит в минимальное знание Django. Создание проекта и приложений, модели и миграции, создание роутов и контроллеров, которые называются Views в Django, полезно знать тестирование в Django. Джанговый ORM, джанговый язык темплейтов. Надо знать правильную структуру Django проектов, понимание, где писать бизнес-логику. Как работать с джанговой админкой. Как работать с юзерами, авторизацией и аутентификацией в Django. Как работать со своими management командами в Django.

В идеале еще знать DRF, но в целом необязательно, при желании можно по ходу изучить, если со всем остальным у вас всё нормально

Что надо знать Python веб-разработчику для успешного поиска ра...

Так, это касательно питона, библиотек и фреймворков. Идём дальше к Save Copy to Evernote

Следование стандартам разработки — в случае питона PEP8.

Принципы написания качественного ПО. Про нейминг, про KISS, DRY, про ООП и про Solid, про основные шаблоны проектирования, про чистый код. Это огромная большая тема и да, это всё по большей части уже не junior тема, но вы с самого старта должны хотя бы базовые штуки из этого знать и применять.

Тестирование. Какие бывают тесты, чем они отличаются, когда что использовать. Какие есть методологии разработки, связанные с тестированием и почему они хороши (TDD и тд).

Инструментарий. ipython как удобный python shell, IDE (вероятнее всего Pycharm), pdb для отладки.

git — хотя бы базовые заведение репозиториев, ветки, коммиты, мержи.

chrome developer tools, работа с DOM деревом в ней, с сетью, с JS консолью.

Терминальные утилиты Linux, необходимые в разработке — cat, less, head/tail, grep и ripgrep, awk, xargs, htop и куча других. Про linux поговорим еще ниже.

SQL и реляционные СУБД. Если вы бэкендер, вам нужно знать базы данных и нужно нормально уверенно с ними уметь работать. То есть вы должны знать PostgreSQL, как ее поставить, минимально настроить, создавать там БД, таблицы, делать в них запросы. Причем запросы не просто уровня select 2+2, а join"ы с несколькими таблицами, со сложными группировками, having, оконными функциями и тд.

Надо знать про индексы и про хотя бы минимальную их настройку, про внешние ключи и нормализацию и денормализацию БД, конечно, надо понимать, плюсы и минусы нормализованной и денормализованной БД.

Linux

Если вы бэкенд разработчик, вы должны знать линукс и точка. Что значит знать линус. Это значит, что вы абсолютно комфортно можете решать все свои рабочие вопросы в терминале и браузере. Создание проекта, поиск в проекте, работа с исходным кодом проекта, настройка всех его компонентов, умение развернуть и настроить полноценный сервер для своего проекта. Кто-то скажет, что для этого есть админы и devops, но нет, друзья, вы тоже должны все это знать и уметь. Иначе — фронтенд я не хочу, линукс я не хочу, БД я не хочу. Это все равно что я могу только держать гвоздь, а забивать его молотком должен кто-то другой. Так не работает. Если вы бэкендер, то линукс это ваша среда, на которой будет работать ваш код, вы должны быть с ней на ты.

Save Copy to Evernote

Вы должны уметь полностью настроить для себя сервер. Nginx, какой-нибудь фиком или uwsgi, django, celery, postgresql, redis. Вам должно быть абсолютно комфортно работать в командной строке, в консоли.

Касательно докер. Да, не лишним будет знание докер, но вопреки расхожему мнению, знание докер не заменяет знание линукс, а добавляет к необходимости знания линукс еще и необходимость знания самого докера. Это важно и это нужно, но это дополнение к знаниям линукса и его инструментов.

Знание CI/CD инструментов кстати тоже будет очень неплохим, хотя бы Gitlab например.

Это касательно технологического стека. Вы должны их знать. Что я подразумеваю под знать? Вы знаете что-то, если вы с ним уже работали на практике. То есть практика это единственное мерило того, насколько вы действительно что-то знаете и умеете. Когда вы с чем-то начинаете работать, с любой технологией, сколько бы вы перед этим не прочли теории, вы в любом случае, всегда начнете собирать грабли, ошибки, сложности и тд. И реальный опыт, реальная ценность, она как раз в этих собранных вами граблях.

Есть еще пара важных моментов, это английский и умение гуглить. Английский — это умение читать английские технические тексты и умение задавать вопросы на английском, умение гуглить по-английски. Ну а умение гуглить позволяет вам самостоятельно и быстро находить ответы на любые свои вопросы в сети, это способность к быстрому и эффективному самообучению, что критично важно. Можно попытаться обойти какую-то проблему, э