-------------------------

Для проектов Flask определим какой-нибудь каталог на жестком диске.

Открываем PyCharm и создаем проект. Используем PIPENV

Указываем папку проекта **Wholesale** (создаем заранее ?)

## **Заходим в Терминал и сетапим Flask та flask-sqlalchemy:**

pipenv install flask

pipenv install flask-sqlalchemy

pipenv install Flask-Migrate

Формуємо основні файли FLASK…

Підключення БД.

Визначаємо налаштування БД –

SQLALCHEMY\_DATABASE\_URI = os.environ.get('DATABASE\_URL') or 'sqlite:///' + str(BASE\_DIR / "data" / "trade.db")

Визначаємо моделі таблиць:

! для ForeignKey назва таблиці з БД, а не назва Класу !

lass Unit(db.Model):  
 *""" --- Одиниці виміру --- """* unit\_code = db.Column(db.String(10), primary\_key=True)  
 unit\_name = db.Column(db.String(15), nullable=False)  
  
 def \_\_repr\_\_(self):  
 return f"Unit {self.unit\_name}"  
  
  
class Item(db.Model):  
 *""" --- Клієнти ---"""* id = db.Column(db.Integer, primary\_key=True)  
 item\_name = db.Column(db.String(50), nullable=False)  
 unit = db.Column(db.String(10), db.ForeignKey('unit.unit\_code'), nullable=False)  
 service = db.Column(db.Boolean, nullable=False, default=False)  
 item\_description = db.Column(db.String(100), nullable=False)  
  
 def \_\_repr\_\_(self):  
 return f"Item {self.id}, {self.item\_name}"

Далі у вікні терміналу:

**flask db init**

Оновлення таблиць при коригування моделей:

**flask db migrate -m "Initial migration."**

**flask db upgrade**

# Публикация в Интернете!

<https://tutorial.djangogirls.org/ru/deploy/>

**Примечание**: Эта глава может показаться сложной. Будь упорна, развертывание сайта на сервере является важной частью веб-разработки. Данная глава намеренно расположена в середине учебника для того, чтобы твой наставник смог помочь с таким мудреным процессом, как публикация сайта. Так ты сможешь самостоятельно закончить все главы, даже если время будет поджимать.

До настоящего момента твой сайт был доступен только для локального просмотра, теперь же ты узнаешь, как развернуть его на удалённом сервере! Развертывание (deploy) — это процесс публикации приложения в интернете, чтобы люди могли наконец увидеть твое творение :)

Как ты уже знаешь, веб-сайт должен располагаться на сервере. Есть много компаний, предоставляющих сервера в интернете. Мы воспользуемся услугами одной из них, с довольно простым процессом публикации: [PythonAnywhere](https://www.pythonanywhere.com/" \t "_blank). PythonAnywhere бесплатен для маленьких приложений с небольшим числом посетителей, и этого будет для нас более чем достаточно.

Другим внешним сервисом, которым мы воспользуемся, будет [GitHub](https://www.github.com/" \t "_blank) — сервис хостинга кода. Существуют и другие похожие сервисы, но практически у каждого программиста есть GitHub аккаунт, теперь будет и у тебя!

В итоге твой код будет в трёх местах. На локальном компьютере ты будешь заниматься разработкой и тестированием. Когда результат тебя устроит, ты загрузишь свою программу на GitHub. А твой сайт будет на PythonAnywhere, и ты сможешь обновлять его, просто загружая новую версию кода с GitHub.

# Git

**Примечание:** если ты уже выполнила установку, незачем повторять её вновь — можешь сразу переходить к следующему разделу и начать создание собственного Git-репозитория.

Git — это «система управления версиями», используемая множеством программистов. Эта программа отслеживает изменения, происходящие с файлами, чтобы впоследствии можно было восстановить состояние кода на нужный момент времени. Это немного похоже на функцию отслеживания изменений в Microsoft Word, но куда мощнее.

## Установка Git

**Windows**

Ты можешь загрузить Git с официального сайта [git-scm.com](https://git-scm.com/). Ты можешь нажимать "дальше, дальше, дальше" на всех этапах установки за исключением одного: на пятом шаге, который называется "Adjusting your PATH environment" (Настройка системной переменной Path), выбери "Use Git and optional Unix tools from the Windows Command Prompt" (Запуск Git и соответствующих Unix утилит через командную строку Windows, нижняя опция). Все остальные настройки можно оставить по умолчанию. Также неплохо будет выбрать опцию "Checkout Windows-style, commit Unix-style line endings".

После окончания установки не забудь перезапустить командную строку или PowerShell.

**OS X**

Загрузи Git с официального сайта [git-scm.com](https://git-scm.com/) и просто следуй инструкциям по установке.

**Примечание:** если ты используешь OS X 10.6, 10.7 или 10.8, тебе придётся установить git отсюда: [Установка Git для OS X Snow Leopard](https://sourceforge.net/projects/git-osx-installer/files/git-2.3.5-intel-universal-snow-leopard.dmg/download)

**Debian и Ubuntu**

command-line

$ sudo apt install git

**Fedora**

command-line

$ sudo dnf install git

**openSUSE**

command-line

$ sudo zypper install git

## Создаём Git-репозиторий

Git отслеживает изменения определенного набора файлов, который называется репозиторием (сокращенно "репо"). Давайте создадим такой для нашего проекта. Открой консоль и запусти эти команды в папке djangogirls:

**Примечание:** проверь текущий рабочий каталог с помощью команд pwd (OSX/Linux) или cd (Windows) перед инициализацией нового репозитория. Ты должна находиться в директории djangogirls.

command-line

$ git init

Initialized empty Git repository in ~/djangogirls/.git/

$ git config --global user.name "Your Name"

$ git config --global user.email you@example.com

Инициализировать git-репозиторий придется только один раз за проект (и тебе больше не придется вводить имя пользователя и адрес электронной почты).

Git будет отслеживать изменения всех файлов и каталогов в заданной директории, однако некоторые из них мы предпочли бы игнорировать. Для этого нам нужно создать файл .gitignore в корневом каталоге репозитория. Открой редактор и создай новый файл со следующим содержанием:

.gitignore

\*.pyc

\*~

\_\_pycache\_\_

myvenv

db.sqlite3

/static

.DS\_Store

И сохрани его как .gitignore в корневом каталоге "djangogirls".

**Примечание:** точка в начале имени файла имеет важное значение! Если у тебя есть проблемы с созданием таких файлов (Mac не позволит создать файл с названием, начинающимся с точки, через Finder, например), тогда используй кнопку «Сохранить как» в меню своего редактора кода, это точно поможет.

**Примечание:** среди файлов, которые мы перечислили в .gitignore, есть db.sqlite3. Этот файл содержит твою локальную базу данных, где будут храниться твои посты. Мы не добавляем его в репозиторий, поскольку твой сайт на PythonAnywhere будет использовать другую базу данных. Эта база данных тоже может быть SQLite, как на твоём рабочем компьютере, но обычно используется MySQL, которая может справиться с большим количеством посетителей, чем SQLite. В любом случае, поскольку мы не копируем базу данных SQLite на GitHub, все посты, которые ты уже создала, будут доступны только на твоём локальном компьютере, и тебе придётся заново создать их на опубликованном сайте. Рассматривай свою локальную базу данных как удобную игровую площадку, где ты можешь тестировать различные идеи и не бояться удалить настоящий пост из своего блога.

Используй команду git status перед git add или в любой другой момент, когда ты не уверена, что изменения — хорошая идея. Это убережёт тебя от таких неприятных сюрпризов, как добавление неправильных файлов. Команда git status возвращает информацию обо всех ранее неотслеживаемых/изменённых/добавленных в git файлах, а также статус ветки и многое другое. Результат должен быть похож на:

command-line

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

.gitignore

blog/

manage.py

mysite/

nothing added to commit but untracked files present (use "git add" to track)

И, наконец, мы сохраним наши изменения. Переключись на консоль и набери:

command-line

$ git add --all .

$ git commit -m "My Django Girls app, first commit"

[...]

13 files changed, 200 insertions(+)

create mode 100644 .gitignore

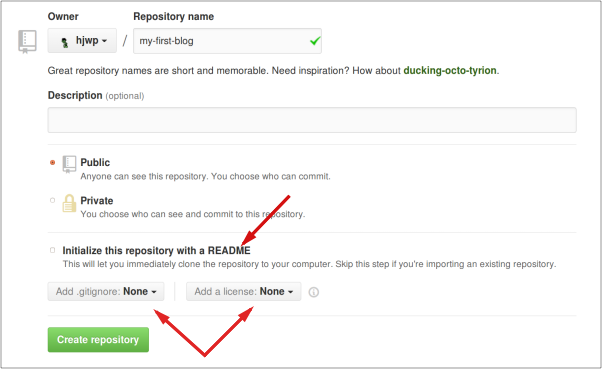
[...]

create mode 100644 mysite/wsgi.py

## Загружаем код в репозиторий GitHub

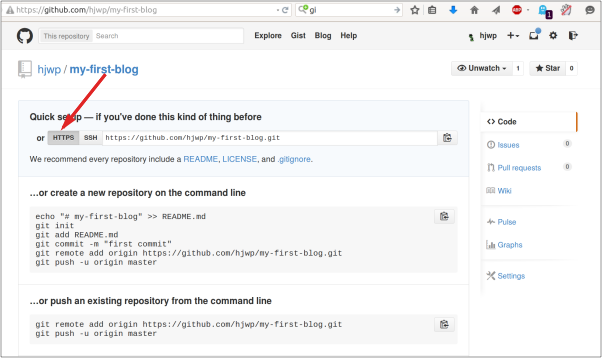
Зайди на [GitHub.com](https://www.github.com/) и создай новую бесплатную учётную запись (если это уже сделано во время подготовки к воркшопу — здорово!). Убедись, что запомнила свой пароль (добавь его в свой менеджер паролей, если им пользуешься).

Затем создай новый репозиторий и назови его "my-first-blog". Не выбирай опцию "initialise with a README", не создавай файл .gitignore (мы уже сделали это сами) и оставь лицензию None.



**Примечание:** имя репозитория my-first-blog для нас очень важно — ты можешь, конечно, придумать другое название, но оно будет встречаться множество раз в руководстве, и тебе придется заменять его каждый раз на своё. Будет проще для начала остановиться на нашем варианте: my-first-blog.

На следующем экране ты найдёшь URL для клонирования репозитория. Выбери вариант "HTTPS" и скопируй ссылку:



Теперь нужно связать локальный репозиторий с репозиторием на GitHub.

Напечатай у себя в консоли следующую команду (замени <your-github-username> на имя, указанное при создании аккаунта на GitHub, но без угловых скобок):

command-line

$ git remote add origin https://github.com/<your-github-username>/my-first-blog.git

$ git push -u origin master

Введи свое имя пользователя и пароль от аккаунта GitHub; ты должна увидеть что-то такое:

command-line

Username for 'https://github.com': hjwp

Password for 'https://hjwp@github.com':

Counting objects: 6, done.

Writing objects: 100% (6/6), 200 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To https://github.com/hjwp/my-first-blog.git

\* [new branch] master -> master

Branch master set up to track remote branch master from origin.

Твой код теперь на GitHub. Зайди на сайт и проверь! Ты найдешь его в хорошей компании: [фреймворк Django](https://github.com/django/django" \t "_blank), [этот учебник](https://github.com/DjangoGirls/tutorial), а также многие другие великолепные проекты с исходным кодом размещены на GitHub :)

# Настройка блога на PythonAnywhere

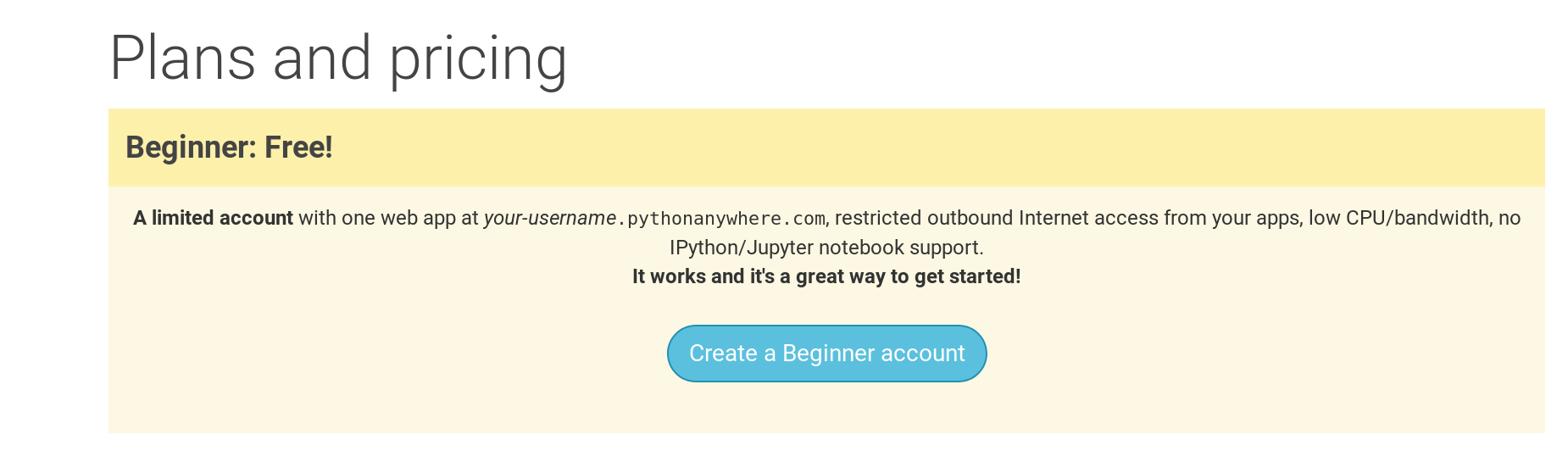
## Регистрация на PythonAnywhere

**Примечание:** возможно, ты уже завела учётную запись на PythonAnywhere ранее — если так, нет нужды повторять это вновь.

PythonAnywhere — это сервис по запуску кода на Python в облаке. Мы будем использовать его, чтобы разместить наш сайт «вживую» в интернете.

Создай аккаунт уровня "Beginner" на PythonAnywhere. Он бесплатный, так что кредитка не понадобится.

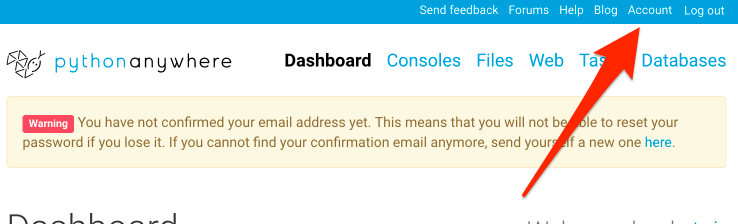
* [www.pythonanywhere.com](https://www.pythonanywhere.com/)



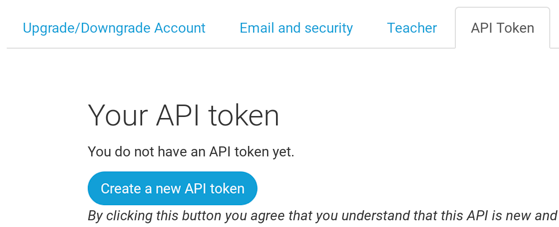
**Примечание:** при выборе имени пользователя помни, что URL блога примет вид yourusername.pythonanywhere.com, так что остановись либо на своём нике, либо на имени, связанном с тематикой блога. Кроме того, убедись в том, что запомнила пароль (сохрани его в своём менеджере паролей, если им пользуешься).

## Создание API токена для PythonAnywhere

Это нужно будет сделать только один раз. Когда ты зарегистрируешься на PythonAnywhere, откроется панель управления (dashboard). На ней в правом верхнем углу будет ссылка на страницу «Account»:

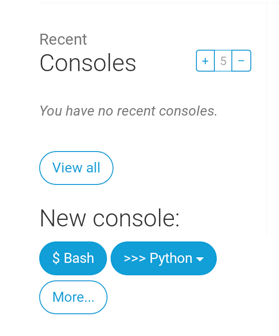


Там выбери вкладку «API token» и нажми кнопку, на которой написано «Create new API token» (создать новый API token).



## Настройка сайта на PythonAnywhere

Вернись на [главную страницу PythonAnywhere](https://www.pythonanywhere.com/), кликнув логотип. Затем запусти Bash-консоль. Нажав на кнопку bash ты запускаешь командную строку, которая находится на серверах PythonAnywhere. Эта командная строка аналогична той, что есть на твоём собственном компьютере.



**Примечание:** PythonAnywhere использует Linux, так что если ты используешь Windows, то терминал и команды могут немного отличаться от того, к чему ты привыкла на своём компьютере.

Чтобы опубликовать сайт на PythonAnywhere, нужно загрузить на PythonAnywhere твой код с Github и затем настроить PythonAnywhere так, чтобы он распознал твой код и запустил твоё веб-приложение. Существуют способы сделать это «вручную», но для PythonAnywhere есть программа-помощник, которая сделает это для тебя. Давай её установим.

PythonAnywhere command-line

$ pip3.6 install --user pythonanywhere

Когда ты это запустишь, в консоли будет печататься лог установки. Он начнётся с чего-то вроде Collecting pythonanywhere, а последней будет строчка Successfully installed (...) pythonanywhere- (...).

Теперь запустим эту вспомогательную утилиту, которую ты только что установила. Она настроит твоё приложение, скачав его код с GitHub. Напечатай следующее в консоли PythonAnywhere (не забудь использовать свой ник на GitHub вместо <your-github-username>, URL в консольной команде должен совпадать с URL, используемым в команде clone):

PythonAnywhere command-line

$ pa\_autoconfigure\_django.py https://github.com/<your-github-username>/my-first-blog.git

Утилита будет печатать в консоль, что она делает:

* Скачивает твой код с GitHub
* Создаёт виртуальное окружение на PythonAnywhere, такое же, как на твоём компьютере
* Обновляет твой файл настроек с настройками деплоя
* Создаёт базу данных на PythonAnywhere, используя команду manage.py migrate
* Разбирается с твоими статическими файлами (о них будет дальше)
* Настраивает PythonAnywhere так, чтобы твоё приложение было доступно в интернете

Все эти шаги автоматизированы на PythonAnywhere, но они совершенно такие же, какие надо было бы совершить с любым другим хостинговым сервисом.

Главное, на что нужно обратить внимание сейчас, — это то, что твоя база данных на PythonAnywhere никак не связана с базой данных на твоём компьютере. Поэтому там будут разные посты и разные аккаунты администраторов. Как следствие, для базы на PythonAnywhere необходимо создать аккаунт администратора так же, как ты это делала у себя локально с помощью команды createsuperuser. На PythonAnywhere заранее активировано виртуальное окружение, так что всё, что тебе нужно сделать — это запустить в консоли PythonAnywhere команду:

PythonAnywhere command-line

(ola.pythonanywhere.com) $ python manage.py createsuperuser

Введи параметры для своего пользователя-админа. Лучше всего использовать те же самые данные, что и у тебя на локальном компьютере, чтобы избежать путаницы, если ты конечно не хочешь сделать пароль на сервере PythonAnywhere более надёжным.

Сейчас, если хочешь, посмотри на файлы на PythonAnywhere с помощью команды ls:

PythonAnywhere command-line

(ola.pythonanywhere.com) $ ls

blog db.sqlite3 manage.py mysite requirements.txt static

(ola.pythonanywhere.com) $ ls blog/

\_\_init\_\_.py \_\_pycache\_\_ admin.py apps.py migrations models.py

tests.py views.py

Ты также можешь заглянуть на страницу «Files» и посмотреть, что лежит на сервере, используя встроенный в PythonAnywhere файловый менеджер. (Со страницы «Console» ты можешь попасть на другие страницы PythonAnywhere, используя кнопку меню в правом верхнем углу. Находясь на какой-либо странице, ты можешь найти ссылки на другие вверху.)

## Ты в сети!

Ура, твой сайт теперь доступен всем в интернете! Ты можешь найти ссылку на него на странице «Web» на PythonAnywhere. Этой ссылкой можно делиться с кем хочешь :)

**Примечание** Это туториал для начинающих, поэтому публикуя сайт, мы использовали несколько хаков, которые не очень хороши с точки зрения безопасности. Если ты захочешь сделать что-то на основе этого проекта или начать новый, прочитай [Django deployment checklist](https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/" \t "_blank) для советов по безопасности на твоём сайте.

## Советы по отладке

Если ты столкнулась с ошибкой, запуская скрипт pa\_autoconfigure\_django.py, вот несколько частых причин:

* Ты забыла создать PythonAnywhere API токен
* Ошибка в GitHub URL
* Если ты видишь сообщение "Could not find your settings.py" (невозможно найти settings.py), это может быть вызвано тем, что ты не добавила все файлы в Git или не загрузила их на GitHub. Перечитай Git-секцию выше
* Если ты до этого уже регистрировала аккаунт на PythonAnywhere и наткнулась на ошибку collectstatic, вполне вероятно, что у тебя устаревшая версия SQLite (например, 3.8.2). В этом случае зарегистрируй новый аккаунт и попробуй ещё раз ввести команды, упомянутые в разделе **Настройка блога на PythonAnywhere**.

Если ты столкнулась с ошибкой при попытке открыть свой сайт, первое место, где нужно искать отладочную информацию — твой **error log**. Ты найдёшь ссылку на него на странице ["Web"](https://www.pythonanywhere.com/web_app_setup/) на PythonAnywhere. Глянь, есть ли там какие-либо сообщения об ошибках; самые новые находятся внизу.

Ещё есть [общие советы по отладке на PythonAnywhere](http://help.pythonanywhere.com/pages/DebuggingImportError).

И помни, твой тренер здесь, чтобы помочь!

# Ты в сети!

Стандартная страница твоего сайта должна включать приветствие "It worked!", точно так же как было на локальном компьютере. Попробуй добавить /admin/ к концу адреса сайта, и перейдёшь к панели администратора сайта. Войди под своим именем и паролем и увидишь форму для добавления новых записей в блог.

После того, как создашь несколько записей, ты можешь вернуться к своей локальной версии приложения (а не на PythonAnywhere). С этого момента для внесения изменений тебе нужно работать в своей локальной версии. Это обычный подход в веб-программировании: изменять код локально, загружать изменения на GitHub, а затем подтягивать изменения на сервер с сайтом. Такой подход позволяет тебе работать и экспериментировать, не рискуя сломать свой сайт. Круто, правда?

Ты заслужила огромную похвалу! Развёртывание сервера — одна из самых каверзных частей веб-разработки, и нередко уходит несколько дней, прежде чем заставишь всё работать. А у нас уже есть работающий в сети веб-сайт, вот так вот!