

# An introduction to Flask: Create a web app and host it in the cloud

## Introduction

Web apps are everywhere nowadays. Did you know that building one doesn't have to take much time? In this workshop you will learn how to build a web app with Flask and host it in the cloud with PythonAnywhere. Flask is a lightweight web application framework which is designed to make getting started quick and easy with ability to scale up to complex applications. PythonAnywhere lets you host, run, and use Python in the cloud. It's a nice way to build fast and see the result immediately.

During this workshop you will build your own web app in just one hour! This web app allows you to keep a to-do list and will have the following main features:

- Create an item in the to-do list.
- Read and show the complete to-do list.
- Delete items from the list.

## Prerequisites

Create a beginners account on

<https://www.pythonanywhere.com/registration/register/beginner/>. A beginners account is a limited account with one web app at *your-username*.pythonanywhere.com, restricted outbound internet access from your apps and low CPU/bandwidth. But that's no problem for now.

## References

Do you want to know more about Flask, PythonAnywhere or MySQL? Check out the following links:

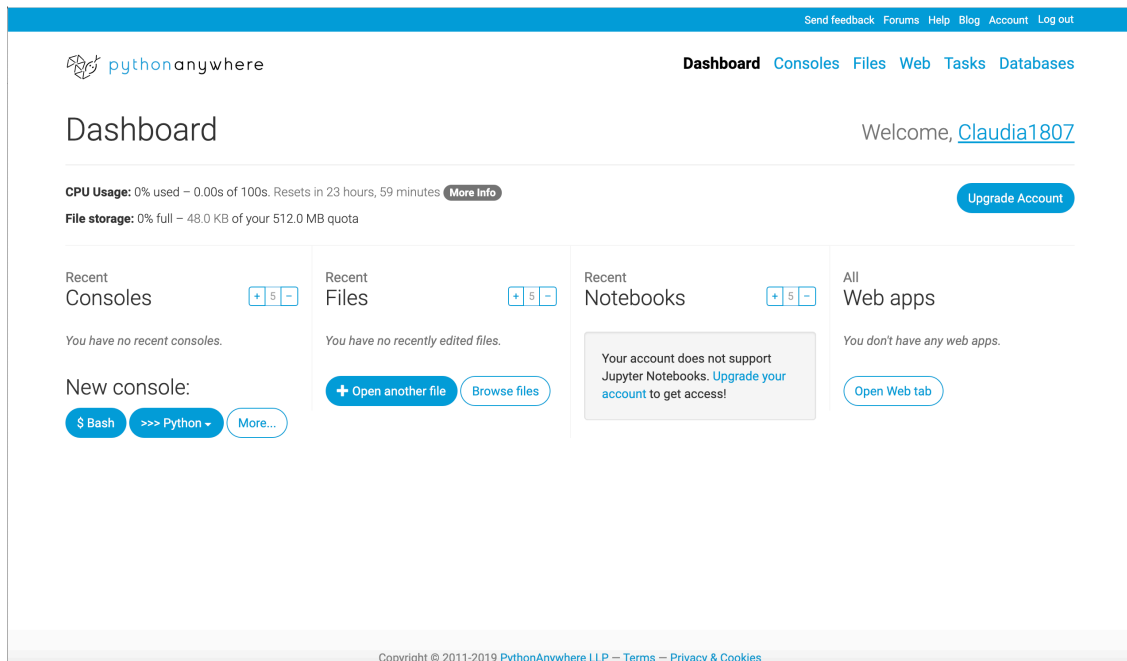
- Flask: <https://flask.palletsprojects.com/en/1.1.x/>
- PythonAnywhere help pages: <https://help.pythonanywhere.com/pages/>
- MySQL: [https://www.w3schools.com/php/php\\_mysql\\_intro.asp](https://www.w3schools.com/php/php_mysql_intro.asp)

This workshop is inspired by the blog '[A beginner's guide to building a simple database-backed Flask website on PythonAnywhere](#)'.

Now, let's get started and build the web app!  
Workshop starts at the next page

## Workshop Part I: Create Flask app and MySQL database.

Log in with your PythonAnywhere account and go to your [Dashboard](#).



The first step is to set-up all the components of your Flask web app:

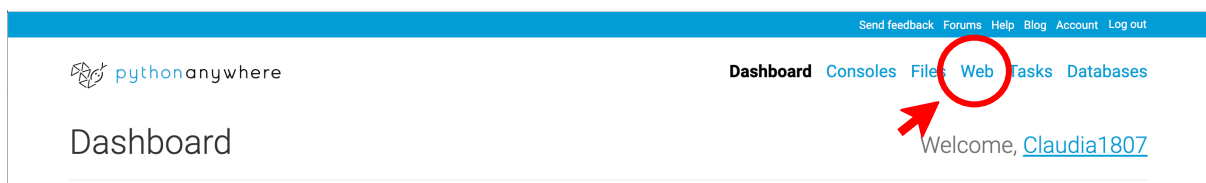
1. The web app itself.
2. A python script containing the code to run your Flask app.
3. A SQL database to store the data.

We will now guide you through PythonAnywhere to create all these components.

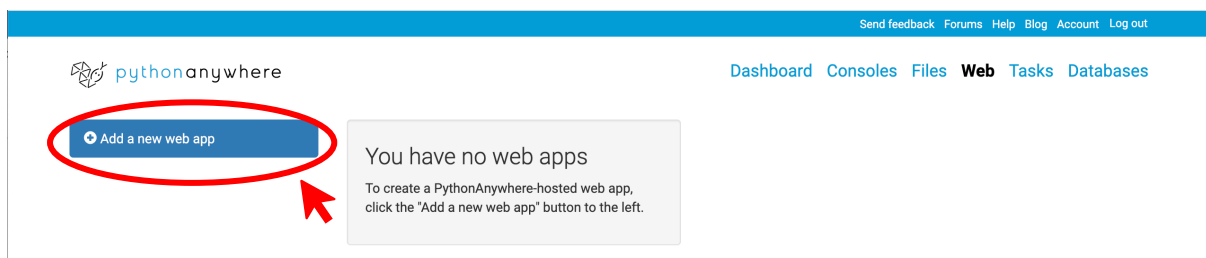
### Create Flask app

How can you create your first web app on PythonAnywhere?

1. Go to [Web](#).



2. Click [Add a new web app](#).



3. Click [Next](#).

Create new web app

### Your web app's domain name

Your account doesn't support custom domain names, so your PythonAnywhere web app will live at `Claudia1807.pythonanywhere.com`.

Want to change that? [Upgrade now!](#)

Otherwise, just click "Next" to continue.

Cancel

« Back

Next »

4. Select a Python Web framework. Choose [Flask](#).

Create new web app

### Select a Python Web framework

...or select "Manual configuration" if you want detailed control.

- » Django
- » web2py
- » **Flask**
- » Bottle
- » Manual configuration (including virtualenvs)

What other frameworks should we have here? Send us some feedback using the link at the top of the page!

Cancel

« Back

Next »

5. Select a Python version. Choose Python 3.7 (Flask 1.0.2).

Create new web app

### Select a Python version

- » Python 2.7 (Flask 1.0.2)
- » Python 3.4 (Flask 1.0.2)
- » Python 3.5 (Flask 1.0.2)
- » Python 3.6 (Flask 1.0.2)
- » Python 3.7 (Flask 1.0.2)

**Note:** If you'd like to use a different version of Flask to the default version, you can use a virtualenv for your web app. There are [instructions here](#).

Cancel

« Back

Next »

6. Enter a path for a Python file you wish to use to hold your Flask app. For example: `/home/<username>/mysite/flask_app.py`. Click [Next](#).


Create new web app

### Quickstart new Flask project

Enter a path for a Python file you wish to use to hold your Flask app. If this file already exists, its contents will be overwritten with the new app.

**Path**

/home/Claudia1807/mysite/flask\_app.py



Cancel

« Back

Next »

Well done! You have created your first Flask app! You should end up on the following webpage:

The screenshot shows the PythonAnywhere dashboard for a user named 'Claudia1807'. The top navigation bar includes links for 'Send feedback', 'Forums', 'Help', 'Blog', 'Account', and 'Log out'. The main header shows the PythonAnywhere logo and navigation links: 'Dashboard', 'Consoles', 'Files', 'Web' (selected), 'Tasks', and 'Databases'. A green notification bar at the top states: 'All done! Your web app is now set up. Details below.' Below this, the dashboard is titled 'Configuration for Claudia1807.pythonanywhere.com'. On the left, there's a button 'Add a new web app'. The main configuration area includes a 'Reload' section with a 'Reload Claudia1807.pythonanywhere.com' button. The 'Best before date' section explains that the free website will be disabled on Monday 13 January 2020 and provides a 'Run until 3 months from today' button. A note mentions that paying users' sites stay up forever. The 'Traffic' section shows a table of site activity:

How busy is your site?	
This month (previous month)	0 (0)
Today (yesterday)	0 (0)

This page contains the configuration of your web app consisting of:

- **Reload button**  
To reload your web app.
- **Best before date**  
You have to log in to PythonAnywhere once every three months and click [Run until 3 months from today](#) or your site will be disabled. You will receive an email one week before the site is disabled.
- **Traffic**  
An overview of the amount of traffic on your website.
- **Code**  
Contains links to the source code, working directory and WSGI configuration file. It also contains the Python version that your site is running (which you can adjust).
- **Virtualenv**  
Here you can enter the path to a virtualenv if you want to run different versions of Flask. We will not use a virtualenv in this workshop.
- **Log files**  
Contains links to different log files.
- **Static files**  
Contains URL and directory to static files like CSS, JavaScript or images.
- **Security**  
Contains options to force HTTPS and set password protection for your web app.
- **Delete**  
To delete your web app.

## Create Python script

PythonAnywhere has automatically created some code to run your Flask app. Let's take a look.

1. Go to [Code](#). Click on [Go to directory](#) after [Source code](#).

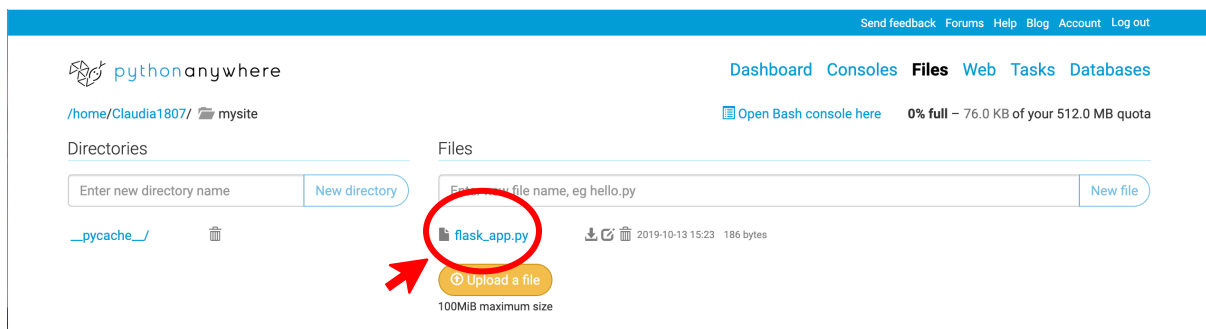
Code:

What your site is running.

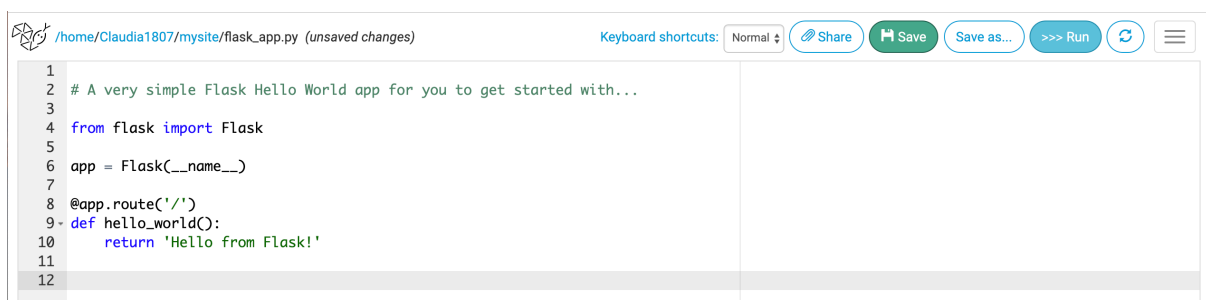
Source code: </home/Claudia1807/mysite>  
Working directory: </home/Claudia1807/>  
WSGI configuration file: [/var/www/claudia1807\\_pythonanywhere\\_com\\_wsgi.py](/var/www/claudia1807_pythonanywhere_com_wsgi.py)  
Python version: 3.7 



2. Click on the Python file (here: flask\_app.py).




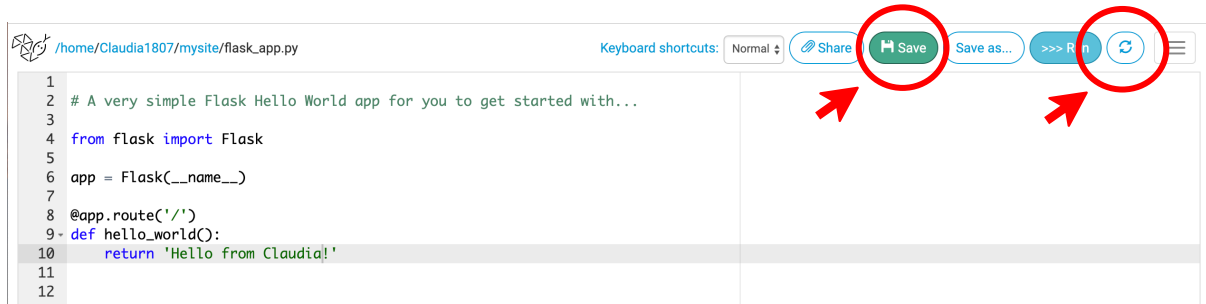
3. PythonAnywhere opens the editor where you can edit your code. We now have a simple Flask app that prints 'Hello World'.



What does this code do?

- Line 4 loads the Flask framework so that you can use it.
- Line 6 creates a Flask application to run your code.
- The decorator in line 8 specifies that the following function defines what happens when someone goes to the location '/' on your site. For example, if they go to <http://<username>.pythonanywhere.com/>. If you want to define what happens when they go to <http://<username>.pythonanywhere.com/foo> then you would use `@app.route('/foo')` instead.

- The function `hello_world()` on line 9 says that when someone goes to the location, they get back the (unformatted) text "Hello from Flask!".
- Let's see what the Flask app looks like in your browser. Go to `<username>.pythonanywhere.com`.
  - Now let's edit the Python code. Go to the editor and edit the text that is returned by the function `hello_world()` (so change 'Hello from Flask!' into some string of your choice). Then click [Save](#) and the reload button .



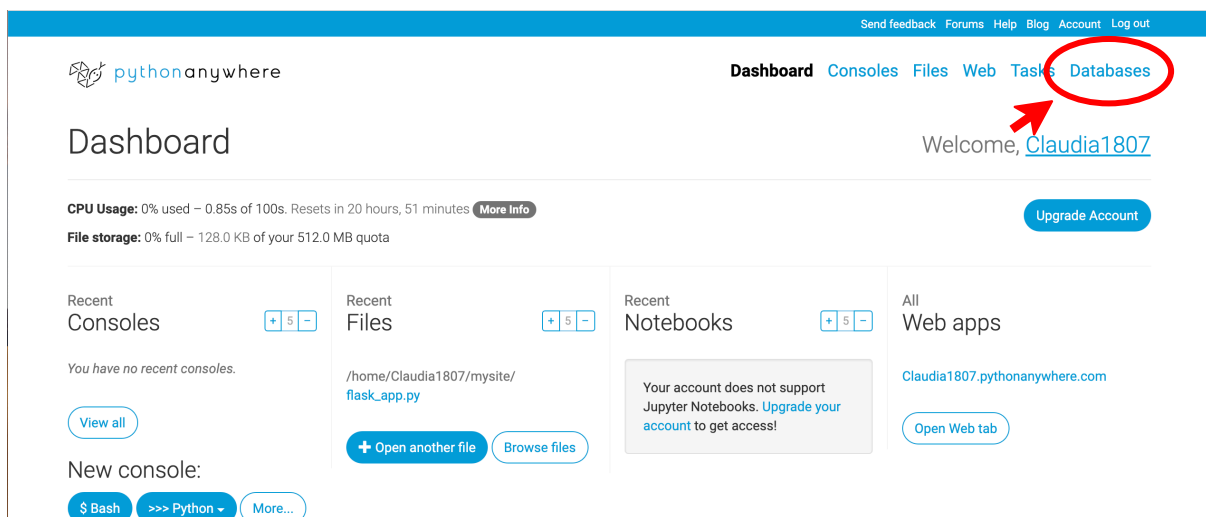
- Go again to `<username>.pythonanywhere.com`. Now the webpage should display your own text!

In Part II of the workshop we will edit this code even more and create our own to-do app.

## Create MySQL database

We will now create a MySQL database to store the data that users send to the web app.

- Go back to the [Dashboard](#). Click [Databases](#).



- Initialize a new MySQL database. Enter a new password. **It is important to remember your password!** Click [Initialize MySQL](#).

3. You will end up on the [MySQL settings](#) page. Here you can create a new MySQL database. Enter a database name as part of the section [Create a database](#). Click [Create](#).

Congratulations, you've finished part I of this workshop! Now let's have a short break and continue with Part II on the next page.



## Workshop Part II: Code your app.

Well done! You have created your first Flask app and SQL database on PythonAnywhere. A web app in Flask consists of source code (here contained in the file `flask_app.py`), and templates, which are written in an extended version of HTML. The source code tells the web app what to do while the templates specify how it should be displayed.

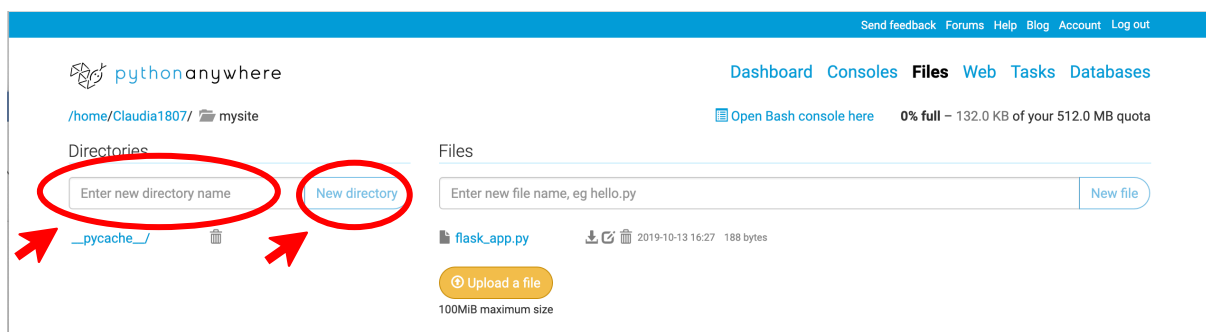
Now it's time to code. In part II of the workshop we will:

1. Add an HTML template to make your web app look fancy.
2. Edit the Python code that runs your Flask app.
3. Connect the SQL database to your Flask app to save all data.

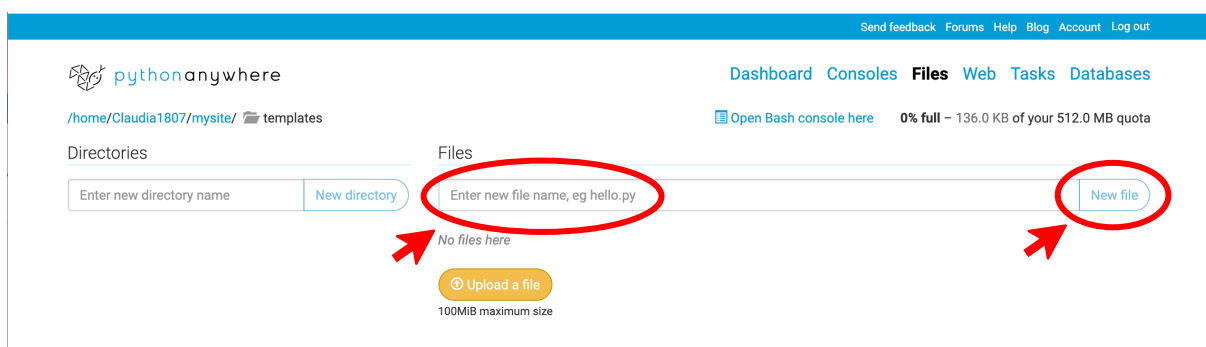
Let's get started!

### Add HTML template

1. Go to [Files](#) → folder *mysite*.
2. Create a new folder. In the field *Enter new directory name* write *templates*. Click on [New directory](#).



3. Create a new file. In the field *Enter new file name* write *main\_page.html*. Click on [New file](#).




4. We will give you an HTML template to work with. Open your `main_page.html` file. Paste the HTML code from the `main_page.html` file on GitHub into your own `main_page.html` file. Click [Save](#).
5. Go to your `flask_app.py` file. Remove all code.

6. Add the following code.

```
from flask import Flask, render_template

app = Flask(__name__)
app.config["DEBUG"] = True

@app.route("/")
def index():
    return render_template("main_page.html")
```

7. Click [Save](#) and reload button .
8. Go to `<username>.pythonanywhere.com`. Look at the design of your web app. That's what is described in your HTML template.

### Connect SQL database

We will now connect the Flask app to the MySQL database using the Python package *SQLAlchemy*.

1. Go to [Files](#). Open the file that contains the Python code (here: flask\_app.py).
2. Change the comment on line 2 that describes your Flask app. Remove the `hello_world()` function. **Before** the decorator `@app.route` add the following code:

```
SQLALCHEMY_DATABASE_URI =
"mysql+mysqlconnector://{username}:{password}@{hostname}/{databasename}".format(
    username="<the username from the 'Databases' tab>",
    password="<the password you set on the 'Databases' tab>",
    hostname="<the database host address from the 'Databases' tab>",
    databasename="<the database name you chose>"
)

app.config["SQLALCHEMY_DATABASE_URI"] = SQLALCHEMY_DATABASE_URI
app.config["SQLALCHEMY_POOL_RECYCLE"] = 299
app.config["SQLALCHEMY_TRACK_MODIFICATIONS"] = False
```

Fill in your own *username*, *password*, *hostname* and *databasename*. You can find this information on the [Databases](#) tab. The *databasename* is equal to `<username>$_<databasename>`. You don't have to edit the app configurations.

3. Add the following code after the code that you just added (the code that configures the database):

```
db = SQLAlchemy(app)
```

4. Add an extra import statement at the top of your code:

```
from flask_sqlalchemy import SQLAlchemy
```

- Next, we need to add a *model*. A model is a Python class that specifies the data that you want to store in the MySQL database. Here's the class definition for our model. Put this code just after the code you just added.

```
class Item(db.Model):
    __tablename__ = "items"
    id = db.Column(db.Integer, primary_key = True)
    name = db.Column(db.String(4096))
```

- Now that we've defined our model, we need to create the database tables. This is a one-time operation. Save your Python file, then go to [Files](#) → [/mysite](#) and click on [Open Bash console here](#). Note: it is important to open the Bash console in the folder `mysite`. Enter within your Bash console:

```
ipython3.7
```

This starts a new iPython 3.7 console.

- Once the console is running, you need to import the database manager. Type in your console:

```
from flask_app import db
```

```
db.create_all()
```

(Press [Enter](#) after each command)

```
19:11 ~/mysite $ ipython3.7
Python 3.7.0 (default, Aug 22 2018, 20:50:05)
Type "copyright", "credits" or "license" for more information.

IPython 4.1.2 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.

In [1]: from flask_app import db

In [2]: db.create_all()

In [3]: █
```

Your database is created!

## Code your app!

We will now add the following features to your web app by editing the Python code:

- Add items to your to-do list.
- Delete items from your to-do list.

### Add items to your to-do list


Now let's add some code to add items to your to do list.

1. Go back to your Python file (here: flask\_app.py).
2. Replace the decorator `@app.route` and the function `index()` with the following code:

```
@app.route("/", methods = ["GET", "POST"])
def index():
    if request.method == "GET":
        return render_template("main_page.html", items = Item.query.all())
    else:
        item = Item(name=request.form["item"])
        db.session.add(item)
        db.session.commit()
        return redirect(url_for('index'))
```

3. Replace the first import statement with:

```
from flask import Flask, redirect, render_template, request, url_for
```


4. Click [Save](#) and reload button .
5. Go to URL of your web app.
6. Can you add items to your to-do list?

### Remove items from your to-do list

You also want to remove items from your to-do list once you've done it.

1. Go back to your Python file (here: flask\_app.py).
2. Add the following decorator and function:

```
@app.route("/delete", methods=["POST"])
def delete():
    name = request.form.get("item")
    item = Item.query.filter_by(name=name).first()
    db.session.delete(item)
    db.session.commit()
    return redirect(url_for('index'))
```

3. Click [Save](#) and reload button .
4. Go to the URL of your web app.
5. Can you remove items from your to-do list?

Congratulations, you've built your first web app with Flask and PythonAnywhere!