

Pythonanywhere and HTML

Table of Contents

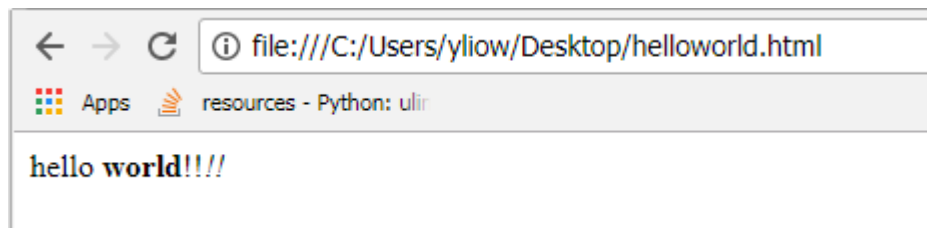
Pythonanywhere and HTML.....	1
Web pages.....	2
Web server.....	3
Web application.....	5
Header.....	12
Images.....	13
Put your own image files on the web.....	15
Lists.....	21
Tables.....	22
Forms.....	24
Forms – upload file.....	26

Web pages

Web page are just files written using HTML syntax. For instance save this on your Desktop with filename helloworld.html (use any text editor you like):

```
<html>
<body>
hello <b>world</b>!!<em>!!</em>
</body>
</html>
```

Open your browser and drag helloworld.html into the browser and you'll see this:



You can also open helloworld.html by double-clicking on it. The ... are HTML tags.

...	bold tag
...	emphasis tag

There are other tags. See <https://www.w3schools.com/html/>.

Web server

The helloworld.html is only on your machine. You can put this webpage on the internet. Create an account at (for instance) <http://www.pythonanywhere.com>. Click on “Pricing and signup”. Click on “Create a Beginner account”. This is free. Remember – be safe and do not use any user name that can be traced to you easily. For instance before you create an account at pythonanywhere.com, create a fake user account at gmail which you can use to confirm your registration at pythonanywhere.com.

Create your account

Username:

Email:

Password:

Password (again):


☒ I agree to the [Terms and Conditions](#)

We promise not to spam or pass your details on to anyone else.

(Replace alfkjghlafg with whatever string you like for your website and use your email address.)

Check your email account for the confirmation email and click on the confirmation link. Now you have a pythonanywhere account that allows you to put your webpage on the internet (and also write programs to generated webpages).

Click around and look for the Consoles tab:

 **python**anywhere [Dashboard](#) **Consoles** [Files](#) [Web](#) [Tasks](#) [Databases](#)

CPU Usage: 0% used – 0.00s of 100s. Resets in 16 hours, 28 minutes [More Info](#)

Start a new console:

Python: [3.5](#) / [3.4](#) / [3.3](#) / [2.7](#) / [2.6](#) IPython: [3.5](#) / [3.4](#) / [3.3](#) / [2.7](#) / [2.6](#) PyPy: [2.7](#)
Other: [Bash](#) | [MySQL](#)
Custom: [+](#)

If you click on “Bash”, you will get a linux shell for linux commands. Here’s the “ls” command to list files:



Bash console 5854464

```
01:27 ~ $  
01:27 ~ $  
01:27 ~ $ ls  
README.txt  
01:27 ~ $ █
```

So in my user “alfkjghlafg” account, there’s a README.txt file. This linux machine is a virtual machine and is connected to the internet. You can run many linux commands using this virtual machine.

Click on the pythonanywhere icon at the top and look for “Consoles” again and click on it. You will see the bash shell for linux command (id 5854464) is still running – you can click on the X to close the shell. Anyway back to webpages ...

Web application

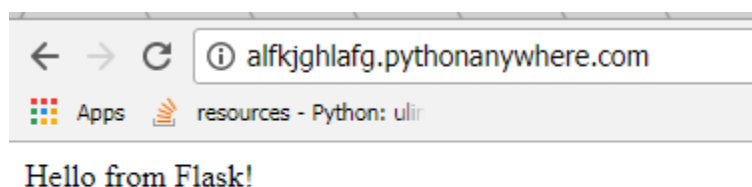
Now to create a web application (basically a program that creates webpages). Click on the “Web” tab:



Click on “Add a new web app”, then “Next”, then “Flask”, then “Python 2.7”, then “Next”. You’ll see this:



Look at the above ... your website is <http://alfkjghlafg.pythonanywhere.com>. In your browser, go to <http://alfkjghlafg.pythonanywhere.com> (or click on the above link in the red box) and you’ll see this:



When you created the web application, pythonanywhere has created a dummy web application for you that prints “Hello from Flask!”. A web application in this case is just a python program that creates web pages as a string. That’s all.

Now to find this python program and modify it.

Click on the pythonanywhere icon and then click on the Files tab:



Click on mysite/ (the directory containing the python program)

Directories

[New directory](#)[.emacs.d/](#)[.local/](#)[.virtualenvs/](#)[mysite/](#)

to get this:

Files

[New file](#)[flask_app.py](#)

2018-11-30 19:38 180 bytes

[flask_app.pyc](#)

2019-01-27 21:56 474 bytes

[Upload a file](#)

100MiB maximum size

The program is flask_app.py. Click on flask_app.py and you'll see the program:



/ home / alfkjghlafg / mysite / flask_app.py

```
1 |
2 # A very simple Flask Hello World app for you to get started with...
3
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10     return 'Hello from Flask!'
11
12
```

When you open a browser to <http://alfkjghlafg.pythonanywhere.com/>, pythonanywhere.com will run the above hello_world function inside this flask_app.py program.

Change the hello_world function to do this:



/ home / alfkjghlafg / mysite / flask_app.py

```
1 |
2 # A very simple Flask Hello World app for you to get started with...
3
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10     return '''
11     <html>
12     <body>
13     hello <b>world</b>!!<em>!!</em>
14     </body>
15     </html>
16     '''
17
18
```

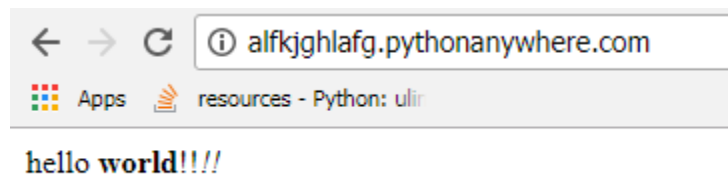
Remember to do Ctrl-S to save. Next, you need to reload your web application: Click on the pythonanywhere icon and click on the Web tab:

and then click on “Reload alfkjghlafg.pythonanywhere.com”.

Reload:

 Reload alfkjghlafg.pythonanywhere.com

After a second or two, open your browser to <http://alfkjghlafg.pythonanywhere.com/> and you’ll see this:

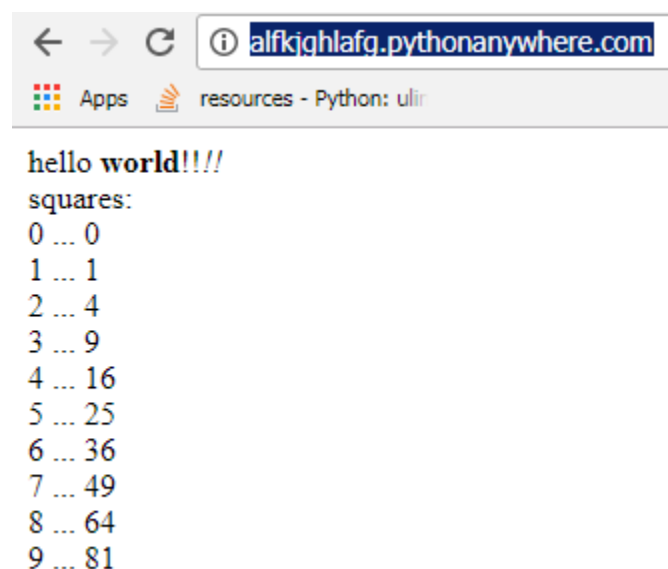


Now change the python program to this:

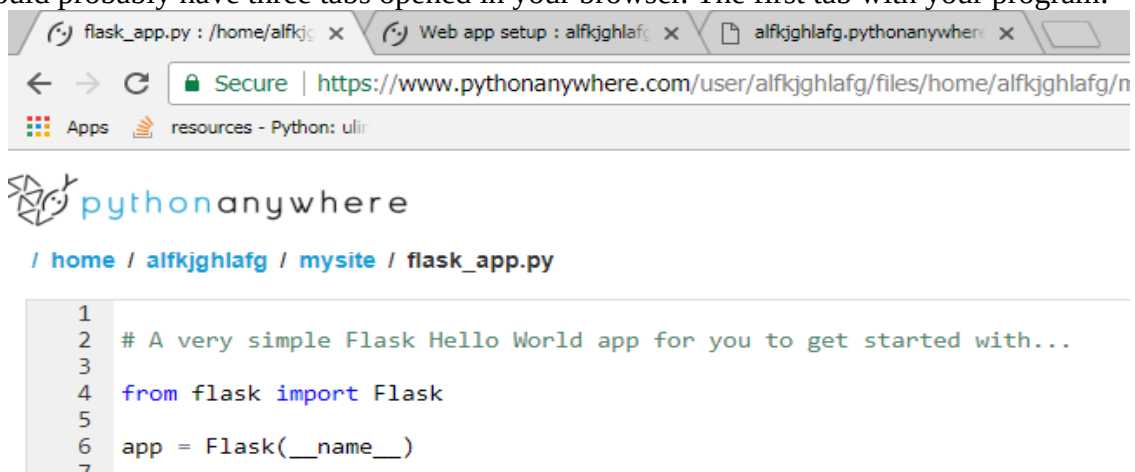

```
1
2 # A very simple Flask Hello World app for you to get started with...
3
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10
11     squares = ""
12     for i in range(10):
13         squares += "%s ... %s<br>" % (i, i * i)
14
15     return ''
16 <html>
17 <body>
18 hello <b>world</b>!!<em>!!</em>
19
20 <br>
21
22 squares:<br>
23 %s
24 </body>
25 </html>
26 ''' % squares
27
```

(The
 is the line break tag to go to the next line on the web page.)

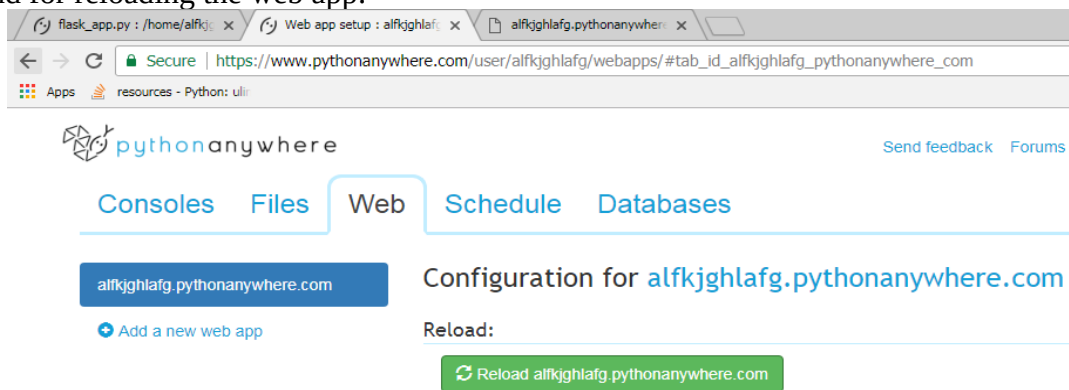
After reloading the web app and refreshing the browser at <http://alfkjghlafg.pythonanywhere.com/>, I get



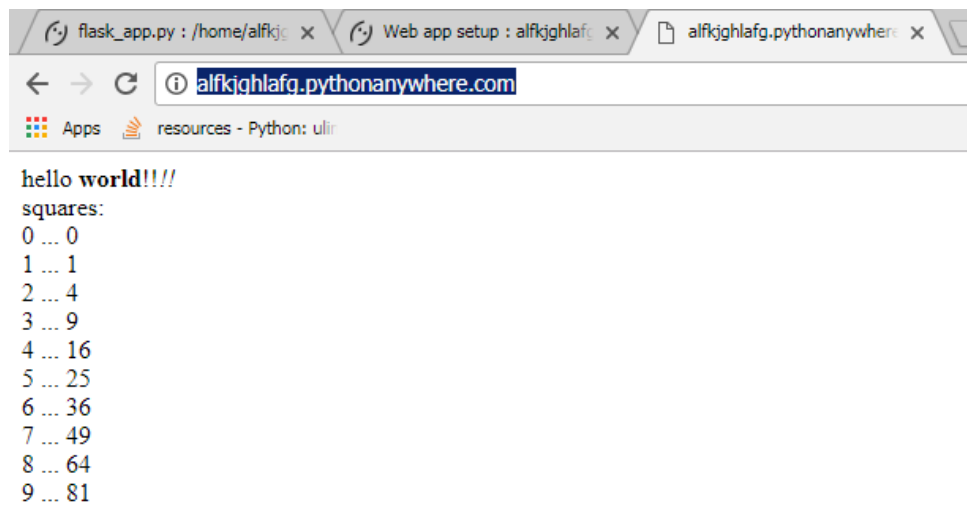
You should probably have three tabs opened in your browser. The first tab with your program:



The second for reloading the web app:



The third for viewing your webpage:



The screenshot shows a web browser with three tabs: 'flask_app.py : /home/alfkjhla...', 'Web app setup : alfkjhla...', and 'alfkjhla.pythonywhere...'. The address bar shows 'alfkjhla.pythonywhere.com'. The page content displays 'hello world!!!' followed by a list of squares from 0 to 9.

```
hello world!!!  
squares:  
0 ... 0  
1 ... 1  
2 ... 4  
3 ... 9  
4 ... 16  
5 ... 25  
6 ... 36  
7 ... 49  
8 ... 64  
9 ... 81
```

You can cycle through all the tabs using Ctrl-Tab.

Header

Now change your hello_world function to this:

`/ home / alfkjghlafg / mysite / flask_app.py`

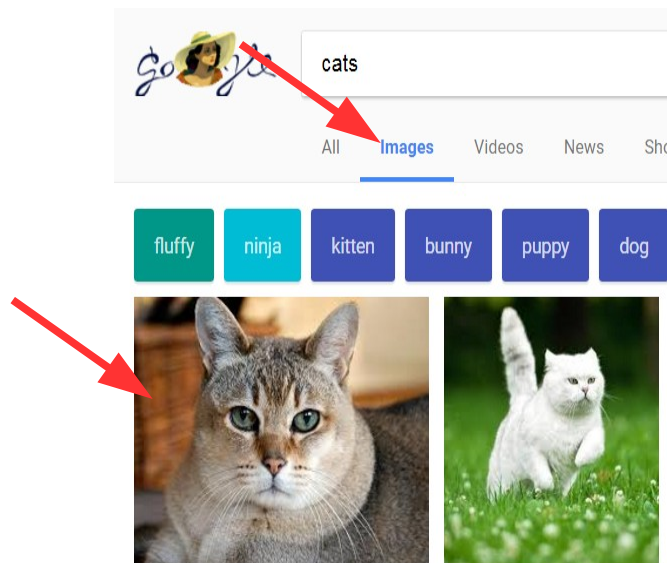
```
1 # A very simple flask hello world app for you to get started with!!!
2
3
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10
11     squares = ""
12     for i in range(10):
13         squares += "%s ... %s<br>" % (i, i * i)
14
15     return '''
16 <html>
17 <body>
18
19 <h1>Hello world and squares</h1>
20
21 hello <b>world</b>!!<em>!!</em>
22
23 <br>
24
25 squares:<br>
26 %s
27 </body>
28 </html>
29 ''' % squares
30
```

The `<h1>...</h1>` is the header tag. There's also `<h2>`, `<h3>`, ...

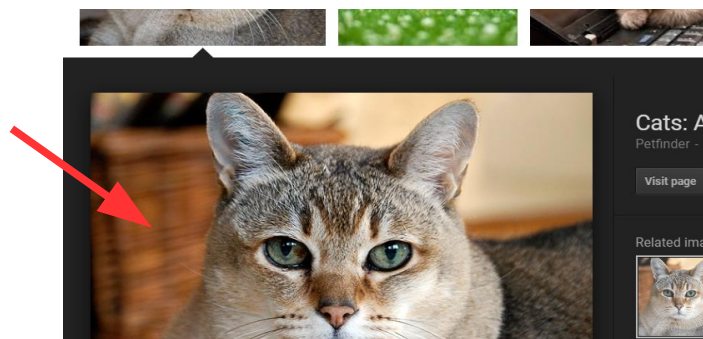
By the way, the `<body>` is the body tag and contains the main document body.

Images

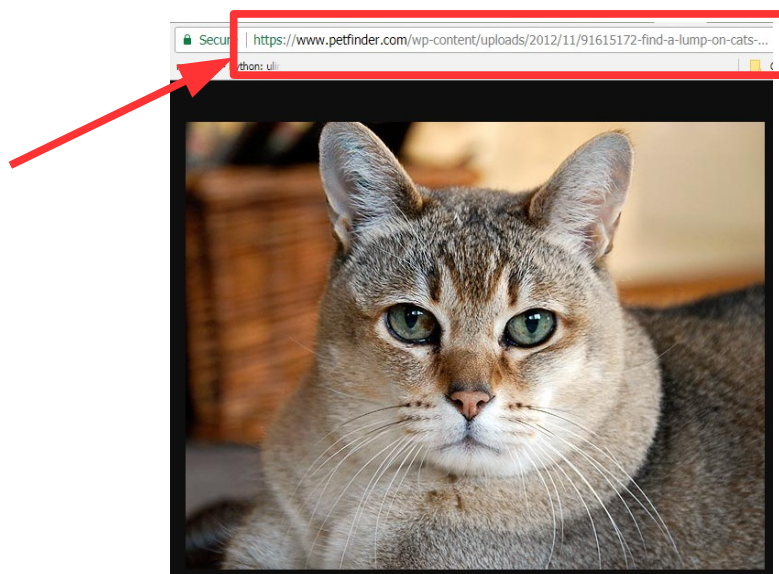
Now to put a cat picture in your web page. Google “cats”, click on “Images”, and click on the first cat



then right-click on this image



and select “Open image in new tab” to get this:



The URL of the image file is in that red box above:

<https://www.petfinder.com/wp-content/uploads/2012/11/91615172-find-a-lump-on-cats-skin-632x475.jpg>

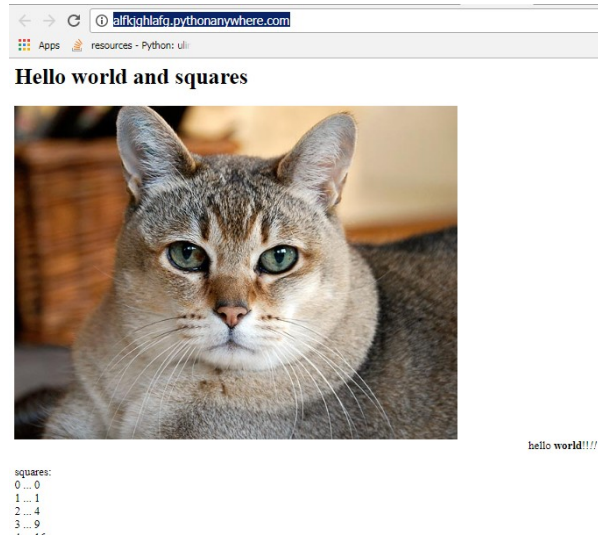
This image is (of course) on the web and has a URL address. (You cannot use an image on your laptop.) In the `hello_world` function of your web application create an image tag:

```
<img src=""></img>
```

In the image tag there's the `src` attribute – the value is the empty string now. Copy the cat's URL and put that into the `src` property. I'll put my image tag right after the `<h1>` tag:

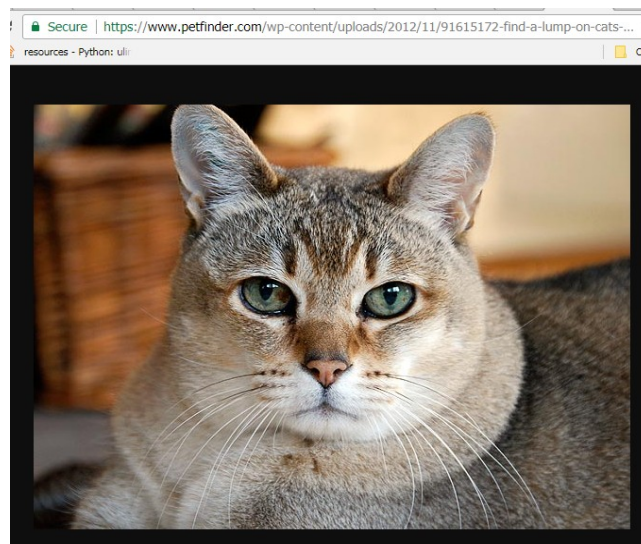
```
8 @app.route('/')
9 def hello_world():
10
11     squares = ""
12     for i in range(10):
13         squares += "%s ... %s<br>" % (i, i * i)
14
15     return ''
16 <html>
17 <body>
18
19 <h1>Hello world and squares</h1>
20
21 </img>
```

Don't forget to reload the application. When you view your web page, you'll get this:

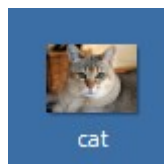


Put your own image files on the web

What if you want to use your own image files? Go back to this again:

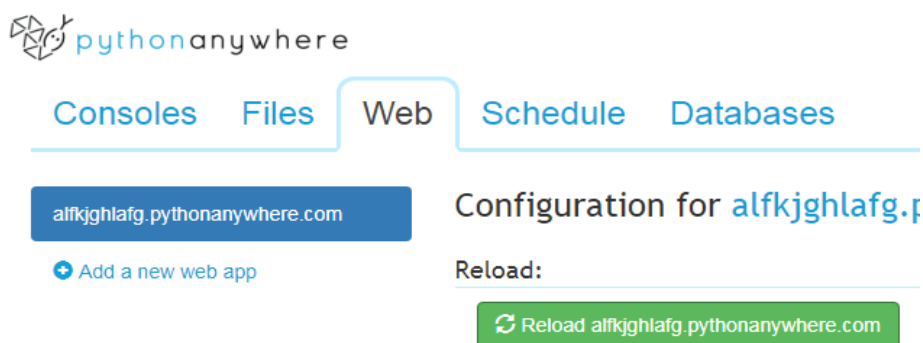


Right click on it and select “Save image as ...”. I saved it on my Desktop as “cat.jpg”:



I now need to make this image accessible through the web by putting this image file on my web server (which is connected to the internet) and also give this image file a web address (i.e. URL).

Click on pythonanywhere icon and click on the Web tab:



Scroll down to the “Static files” section:

Static files:

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
Enter URL	Enter path	

(Your image file is static and not generated by a program. Your web page on the other hand is generated by your python program and therefore is not static.)

Click on “Enter URL”, enter /static/ and click on the check mark to get this:

Static files:


Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
/static/	Enter path	
Enter URL	Enter path	

Click on first “Enter path” and enter the path below (replacing my user name with yours):

Static files:

Files that aren't dynamically generated by your code, like CSS, JavaScript or uploaded files, can be served much faster straight off the disk if you specify them here. You need to **Reload your web app** to activate any changes you make to the mappings below.

URL	Directory	Delete
/static/	/home/alfkjghlafg/mysite/static/	
Enter URL	Enter path	

What this means is that if someone opens a browser to the address (or if a web page links to it):

<http://alfkjghlafg.pythonanywhere.com/static/cat.jpg>

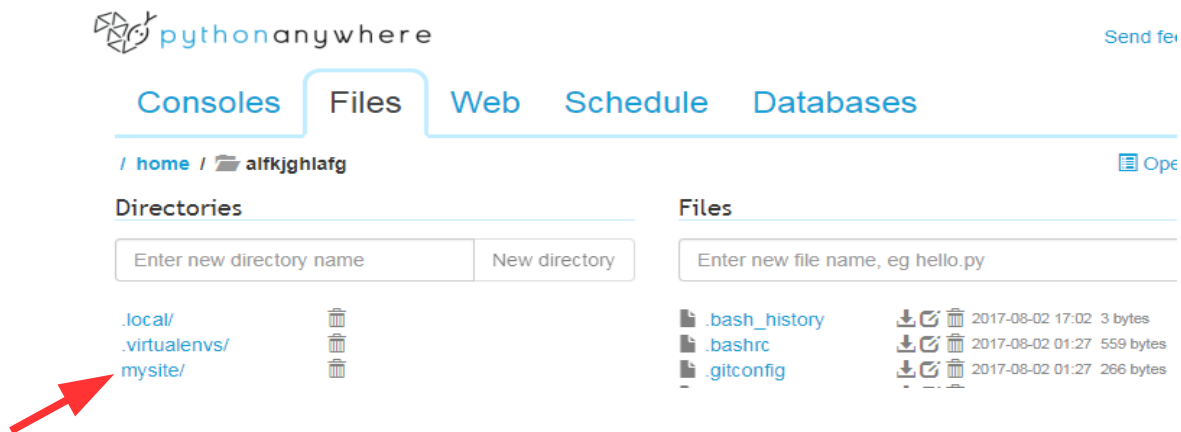
then the internet and pythonanywhere will translate the above URL to this path in your linux web server:

`/home/alfkjghlafg/mysite/static/cat.jpg`

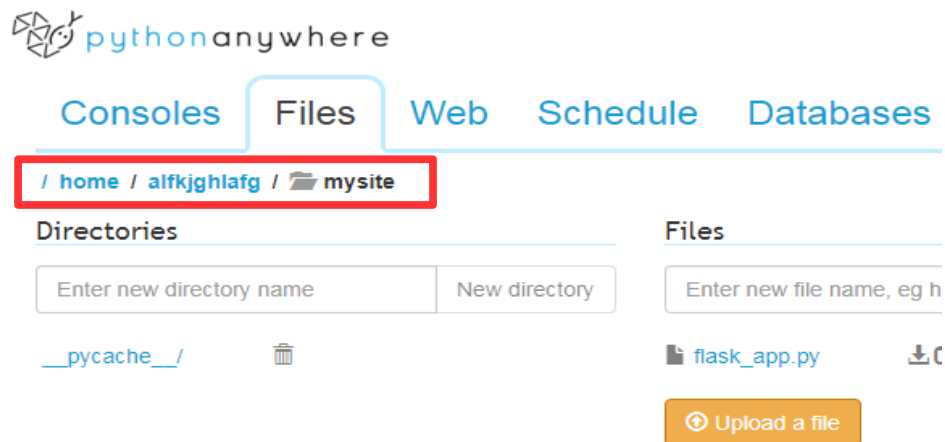
Remember to reload your web application.

Now we need to create the directory `/home/alfkjghlafg/mysite/static/` and upload `cat.jpg` (in your laptop) to this directory in your linux web server.

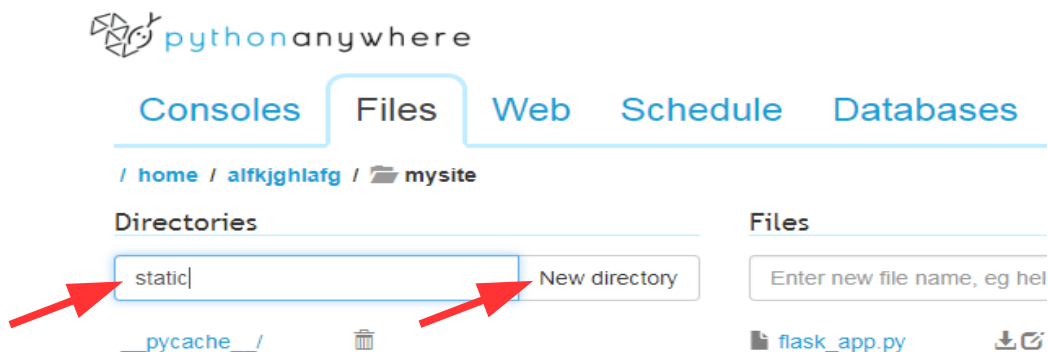
Click on the pythonanywhere icon and then click on the Files tab and then click on `mysite/` (see red arrow):



You see this:



You are now looking at the files of /home/alfkjghlafg/mysite/ directory. For instance in this directory is your flask_app.py. Create the static directory here by entering static and clicking on “New directory”:



You now see this:

[Consoles](#)[Files](#)[Web](#)[Schedule](#)[Databases](#)[/ home](#) / [alfkjghlafg](#) / [mysite](#) / [static](#)

Directories

Files

No files here

You are now in /home/alfkjghlafg/mysite/static/ directory. Click on Upload a file and browse for your cat.jpg on your laptop:

[Consoles](#)[Files](#)[Web](#)[Schedule](#)[Databases](#)[/ home](#) / [alfkjghlafg](#) / [mysite](#) / [static](#)

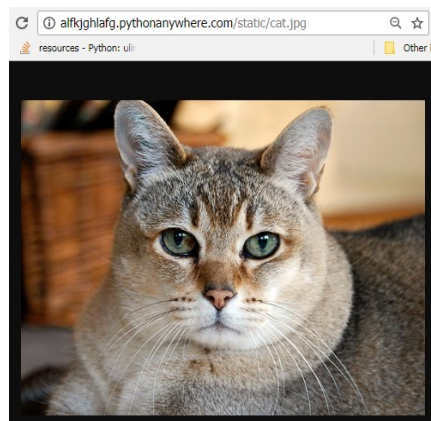
Directories

Files

You can now test if cat.jpg is accessible through the web. Open a browser and enter the

<http://alfkjghlafg.pythonanywhere.com/static/cat.jpg>

Sure enough you see this:

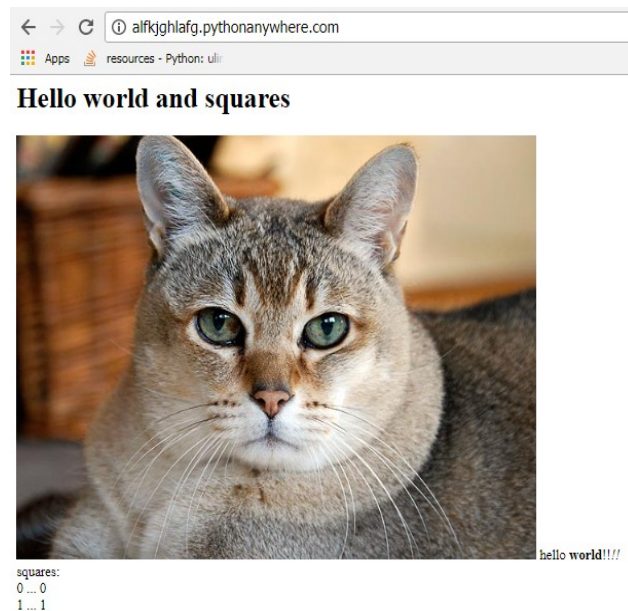


Note that the image URL is now pointing to your web server <http://alfkjghlafg.pythonanywhere.com/> and not to <http://www.petfinder.com>.

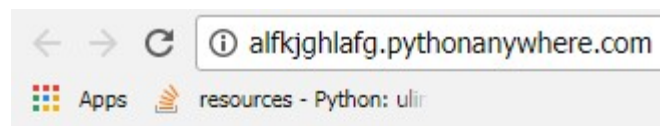
Finally change the img tag in your python code to this:

```
4 from flask import Flask
5
6 app = Flask(__name__)
7
8 @app.route('/')
9 def hello_world():
10
11     squares = ""
12     for i in range(10):
13         squares += "%s ... %s<br>" % (i, i * i)
14
15     return ''
16 <html>
17 <body>
18
19 <h1>Hello world and squares</h1>
20
21 </img>
22
23
24 hello <b>world</b>!!<em>!!</em>
25
```

reload the web application and you'll see this:



Now, the image used is hosted in your web server and not at petfinder.com web server. Read the following pages https://www.w3schools.com/tags/tag_img.asp and see if you can make the cat smaller:



Hello world and squares



hello world!!!!

squares:

0 ... 0

1 ... 1

2 ... 4

3 ... 9

4 ... 16

5 ... 25

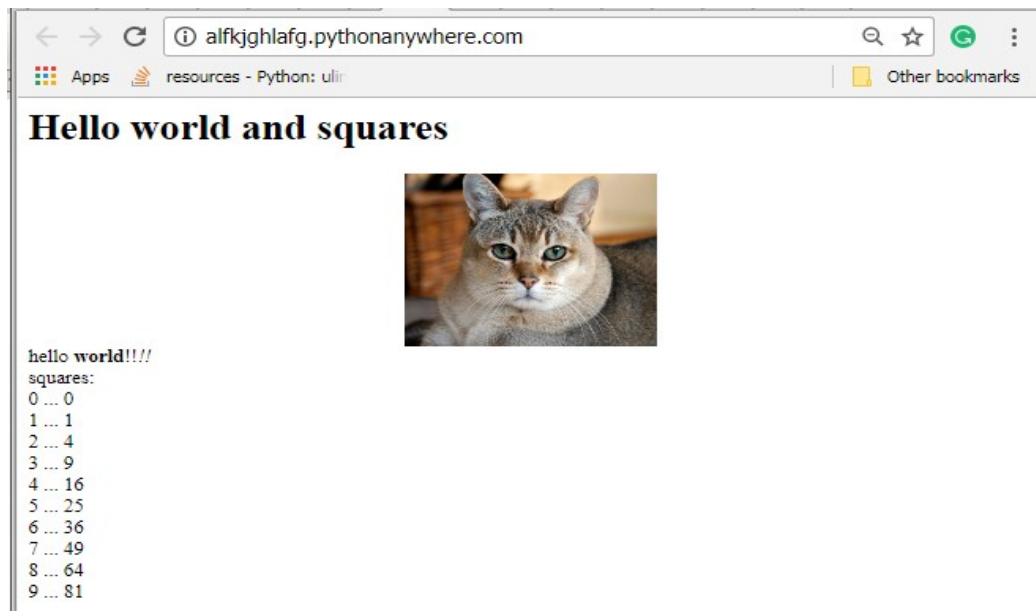
6 ... 36

7 ... 49

8 ... 64

9 ... 81

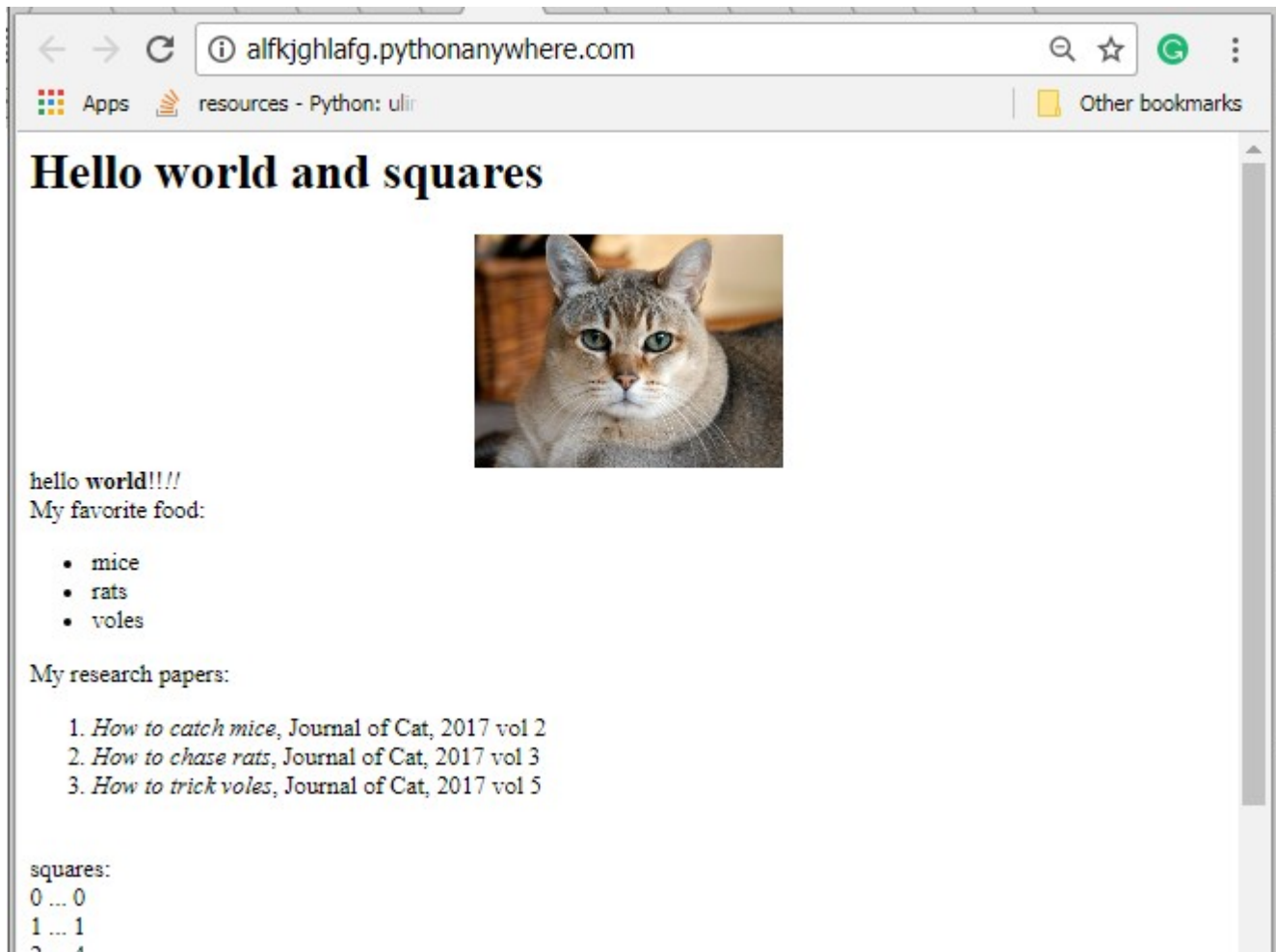
Google around and see if you can do this:



(the cat is centered).

Lists

Google “html unordered list” and “html ordered list” and see if you can do this:



Tables

The “older” way of doing tables is by using the `<table>` tag.


Put this in your python program:

```
39
40 - Here's my office hours if you have questions about my papers:
41 <table border='1'>
42     <tr>
43         <td>Mon</td>
44         <td>12-1</td>
45     </tr>
46     <tr>
47         <td>Fri</td>
48         <td>12-12:01</td>
49     </tr>
50 </table>
51
```

Note that the table is made up of rows created using the table row `<tr>` tag. Each row is made up of table data tags `<td>`.

Don't forget to reload the app. You'll get this:

Hello world and squares



hello world!!!!

My favorite food:

- mice
- rats
- voles

My research papers:

1. *How to catch mice*, Journal of Cat, 2017 vol 2
2. *How to chase rats*, Journal of Cat, 2017 vol 3
3. *How to trick voles*, Journal of Cat, 2017 vol 5

Here are my office hours if you have questions about my papers:

Mon	12-1
Fri	12-12:01

Go ahead and rewrite the squares data into a table:

hello **world!!!!**

My favorite food:

- mice
- rats
- voles

My research papers:

1. *How to catch mice*, Journal of Cat, 2017 vol 2
2. *How to chase rats*, Journal of Cat, 2017 vol 3
3. *How to trick voles*, Journal of Cat, 2017 vol 5

Here are my office hours if you have questions about my papers:

Mon	12-1
Fri	12-12:01

squares:

0	0
1	1
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81

Forms

Now add this function somewhere in your python file:

```
@app.route("/zzz")
def zzz():
    return """<html>
<body>
<h1>Guest book</h1>
<form>
Name: <input type='text' name='fullname' /> <br>
Favorite integer: <input type='text' name='fav_int' /> <br>
<input type='submit' value='Go' />
</form>
</body>
</html>"""
```

Reload your app, and point your browser to <http://....pythonanywhere.com/zzz> with your user name at the beginning of the URL instead of

You should see a form. You can enter info into the fields. But you cannot submit the form yet.

Exercise. In your main page (the hello world page), add a link so that when you click on the link, you arrive at the guest book page.

Now add this to you zzz function:

```
@add.route("/zzz")
def zzz():
    return """<html>
<body>
<h1>Guest book</h1>
<form action='get' method='xxx'>
Name: <input type='text' name='fullname' /> <br>
Favorite integer: <input type='text' name='fav_int' /> <br>
<input type='submit' value='Go' />
</form>
</body>
</html>"""
```

When you click on the submit button, your browser will request the execution of function xxx. Of course you need to create such a function. So add this:

```
@app.route("/xxx")
def xxx():
    return """<html>
<body>
<h1>Thanks</h1>
```

```
</body>
</html>"""
```

Of course you probably want to process the user's input into the guest book fields. So add this to your xxx function:

```
@app.route("/xxx", methods=['GET', 'POST'])
def xxx():
    fullname = request.args.get("fullname", "[NO NAME]")
    fav_int = int(request.args.get("fav_int", -1))
    if fav_int == 42:
        comment = "Meow!!! My fav int is also 42!!!"
    else:
        comment = "Why is 42 not your fav int???"
    plus_one = fav_int + 1

    return """<html>
<body>
<h1>Thanks</h1>

<p>Hi %s. Thanks for visiting my website.</p>

<p>I'm the only cat in the world that can do math. The number after
your fav int is %s. Purr.</p>

<p>I'm going to learn how to add your to a database next.</p>

<p>%s</p>

</body>
</html>""" % (fullname, plus_one, comment)
```

At the top of your python file, you import the request module by changing

```
from flask import Flask
```

to

```
from flask import Flask, request
```

Forms – upload file

Now to include a file upload. Make the following changes:

```
@app.route("/zzz")
def zzz():
    return """<html>
<body>
<h1>Guest book</h1>
<form action='xxx' method='post'
        enctype='multipart/form-data'>
Name: <input type='text' name='fullname' /> <br>
Favorite integer: <input type='text' name='fav_int' /> <br>
Your pic: <input type=file name='file0' /> <br>
<input type='submit' value='Go' />
</form>
</body>
</html>"""
```

The method='post' basically means when the HTML is submitted from the client (the browser), the communication with the webserver is through HTTP POST instead of HTTP GET. You don't have to know the details (you can always google). You just have to know that data is communicated between client and browser in a different way. The most important difference is that HTTP POST allows you to send more data to the server.

Now modify your xxx function. Your xxx function can handle HTTP GET and HTTP POST. Previously it was only through HTTP GET. We've changed zzz to perform HTTP POST. But it's also possible for function yyy to execute xxx through HTTP GET. To handle both cases, you can write your function like this:

```
@app.route("/xxx", methods=['GET', 'POST'])
def xxx():
    if request.method == 'POST':
        ...
    elif request.method == 'GET':
        # What we had before
        fullname = request.args.get("fullname", "[NO NAME]")
        fav_int = int(request.args.get("fav_int", -1))
        if fav_int == 42:
            comment = "Meow!!! My fav int is also 42!!!"
        else:
            comment = "Why is 42 not your fav int???"
        plus_one = fav_int + 1

    return ...
```

So here's the new xxx function:

```
@app.route("/xxx", methods=['GET', 'POST'])
def xxx():
    if request.method == 'POST':
        fullname = request.form.get("fullname", "[NO NAME]")
        fav_int = int(request.form.get("fav_int", -1))
        if fav_int == 42:
            comment = "Meow!!! My fav int is also 42!!!"
        else:
            comment = "Why is 42 not your fav int???"
        plus_one = fav_int + 1
        # save file
        if 'file0' in request.files:
            afile = request.files['file0'] # file object
            filename = afile.filename
            afile.save(filename)
            file_comment = "I've saved your %s!!!" % filename
        else:
            file_comment = "Hey ... you did not upload a file!!!"

    elif request.method == 'GET':
        # This should not happen
        fullname = request.args.get("fullname", "[NO NAME]")
        fav_int = int(request.args.get("fav_int", -1))
        if fav_int == 42:
            comment = "Meow!!! My fav int is also 42!!!"
        else:
            comment = "Why is 42 not your fav int???"
        plus_one = fav_int + 1

    return """<html>
<body>
<h1>Thanks</h1>

<p>Hi %s. Thanks for visiting my website.</p>

<p>I'm the only cat in the world that can do math. The number after
your fav int is %s. Purr.</p>

<p>I'm going to learn how to add your to a database next.</p>

<p>%s</p>

<p>%s</p>

</body>
</html>""" % (fullname, plus_one, comment, file_comment)
```

Notice that to get the name-value pairs from HTTP GET, you do

```
fullname = request.args.get("fullname", "[NO NAME]")
```

But for HTTP POST, you do

```
fullname = request.form.get("fullname", "[NO NAME]")
```

If you google online, you will find reasons why these are named as above. Basically HTTP GET sends the form input name-value pairs encoded in the URL while HTTP POST packs all the data from the form inputs (including the file) not into the URL but in the body of a message.

It's very important to (immediately) check where the file uploaded is saved. It depends on how the web server is configured. For the case of pythonanywhere.com with default flask setup, the default path is your user home directory. You probably want to keep it somewhere else. Also, it's NOT a good idea to use the filename of the file sent to you – that's dangerous. You should save the file using a filename you choose. You can of course save the filename (say in a database), but it should NOT be used as a real filename. I'll leave it to you to figure out why it's a security issue if you use the filename given to you. Google.

Another thing is you probably want to check the contents of the file to see if the file is malicious. Again google on how to check the contents of a file – if the user claims to upload a jpeg file, how would you verify that?

Yet another thing is to limit the file upload size. Again google on how to do that.

Exercise. Save the JPG file in your static directory. In your xxx function, for the return HTML page, include an IMG tag to show the image file was uploaded.

Exercise. Until you know how to use a database, save all visitors who signed your guest book using text file. Modify your guest book so that for a repeat visit, you flag the user and show the previously uploaded picture file and the favorite number that was saved.

MySQL

For storage, you can use MySQL. Click on the “Databases” tab and follow the instructions to set up your MySQL account and to create a database. Note that the database password is different from your pythonanywhere.com account password. Once your MySQL account is initialized you will see this:

MySQL settings

Connecting:

Use these settings in your web applications.

Database host address: `yliowtest.mysql.pythonanywhere-services.com`

Username: `yliowtest`

Refer to your MySQL notes. In an exercise, you were told to figure out how to connect to the MySQL server on a different machine, i.e., not localhost, your own machine. Now’s the time to figure out how to do it if you have not done so. HINT: Google knows everything. [ANOTHER HINT: There’s a parameter in the connection function for the host machine name]. You also want to include “pythonanywhere” in your google search because pythonanywhere changed the database name. [HINT: If you create your database with the name FASTFOOD, then the name is not exactly FASTFOOD. Spoiler on next page. But again Google knows everything.] It should not take you more than a hour to figure out how to do all the above.

Once you manage to make a connection, get a cursor, execute “use” to use your database, you can now do whatever SQL command you want to execute on your database through python in your web app.

*** SPOILER ***

The database name is [username]\$[FASTFOOD]. For instance if your name is SUPERMAN, then after making a connection to your MySQL server, you execute

“use SUPERMAN\$FASTFOOD”

in your cursor. You can of course specify the database name in your connection (see note). Of course you can also include your database name in your connection as a parameter.