Question #57074

- How I can create a web page in python?

Look at (and become bewildered by) the large number of Python Web Frameworks available.

1.Decide you're going to make your decision based on popularity, and pick Django [1].
2.Become frustrated when Django (even with the available add-on libraries) doesn't fit your intended type of site exactly (you aren't building a CMS, after all), and decide to try some other frameworks.
3.Decide that none of the other frameworks (even the ones that fit your intended use better) are well enough supported / a good enough fit / small enough to understand, so you decide to create your own framework that does exactly what you need.
4.Realize, after maintaining your own framework for a while, that hey, maybe you could release it as open source to spread the maintenance burden around a bit[2].
5.You know, maintaining an open source project is kinda hard work, and you're not really getting a lot of help, just complaints about missing documentation.
6.Realize that for the same effort, you could have been adding documentation or 3rd-party libraries for an existing framework and you could have had the best of both worlds. Pick the Pyramid [3] framework as the project to merge with/switch to/join because it leaves you with the greatest degree of freedom.
7.Occasionally still build a site with Django, because after all, the CMS-like use cases it is designed for are actually pretty common, and it does have a large and active community.

[1] At one point, the answer here would have been Zope or Plone
[2] Notice how #5 contributes to the problem in #1.
[3] Depending on your actual use-case, your answer here might be very different such as Flask (Python framework), Twisted, Tornado , or any number of other Python Web Frameworks each with it's own sweet spot.

The short answer is illustrated by the following code snippet.
For more details, please read the entire answer.
To build a website with Python, you simply create an empty file, put the following code inside it, name it with a

```
.py
```

extension and run it. If you visit

```
localhost:5000
```

you will see the website. Yes, it's just a page with plain text, but it is still a website.

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def home():

    return "Website content goes here."

if __name__ == '__main__':

    app.run(debug=True)
```

The above code uses the flask framework which provides a web app prototype written in Python. If you don't have flask, you can install by typing

```
pip install flask
```

in the terminal/command line.
Then you use Python functions to return output to the visited URL. In this case we returned a Python string.
In real life though, you would want to return HTML pages instead of plain Python strings. For that you would need to use the

```
render_template
```

method. Here is the updated code:

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route('/')

def home():

    return render_template("home.html")

if __name__ == '__main__':

    app.run(debug=True)
```

Of course you need to create a

```
home.html
```

file in this case and that should be located inside a folder called

```
templates
```

which you should create at the same directory level with your Python file.
That would be a good start!
Some more tips:
- It's a good practice to run your app inside a virtual environment. You can create a virtual environment using the virtualenv library:

```
pip install virtualenv

python -m venv foldername
```

- If you want to apply CSS styling to your HTML files, create a folder named

```
static
```

in the same directory level with

```
templates
```

. Then you could create subfolders (e.g. css, javascript, images, etc.) inside that folder and put the respective files inside them.
Then you link to them from your HTML pages more or less like this:

```
<link rel="stylesheet" href="{{ url_for('static',
filename='css/main.css') }}">
```

- Lastly, you need to deploy your website online so that others can visit it through a public URL. I would suggest Heroku cloud. It's relatively easy to deploy there and they have a free hosting plan along paid plans. You will need a few tools to deploy your Python website and those are the Heroku Toolbelt and Git, and you will also need to create a Heroku account. And don't forget to set debut to False before you deploy. That will keep your app secure.

- How to make a database also in python?

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

- •GadFly
- •mSQL
- •MySQL

- •PostgreSQL
- •Microsoft SQL Server 2000
- •Informix
- •Interbase
- •Oracle
- •Sybase

Here is the list of available Python database interfaces: Python Database Interfaces and APIs .You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

- •Importing the API module.
- •Acquiring a connection with the database.
- •Issuing SQL statements and stored procedures.
- •Closing the connection

We would learn all the concepts using MySQL, so let us talk about MySQLdb module.

What is MySQLdb?

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

How do I Install MySQLdb?

Before proceeding, you make sure you have MySQLdb installed on your machine. Just type the following in your Python script and execute it:

```
#!/usr/bin/python


import MySQLdb
```

If it produces the following result, then it means MySQLdb module is not installed:

```
Traceback (most recent call last):
   File "test.py", line 3, in <module>
      import MySQLdb
ImportError: No module named MySQLdb
```

To install MySQLdb module, download it from MySQLdb Download page and proceed

as follows:

```
$ gunzip MySQL-python-1.2.2.tar.gz
$ tar -xvf MySQL-python-1.2.2.tar
$ cd MySQL-python-1.2.2
$ python setup.py build
$ python setup.py install
```

Note: Make sure you have root privilege to install above module.

Database Connection

Before connecting to a MySQL database, make sure of the followings −

- •You have created a database TESTDB.
- •You have created a table EMPLOYEE in TESTDB.
- •This table has fields FIRST_NAME, LAST_NAME, AGE, SEX and INCOME.
- •User ID "testuser" and password "test123" are set to access TESTDB.
- •Python module MySQLdb is installed properly on your machine.
- •You have gone through MySQL tutorial to understand MySQL Basics.

Example

Following is the example of connecting with MySQL database "TESTDB"

```
#!/usr/bin/python

import MySQLdb
# Open database connection
db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )
# prepare a cursor object using cursor() method
cursor = db.cursor()
# execute SQL query using execute() method.
cursor.execute("SELECT VERSION()")
# Fetch a single row using fetchone() method.
data = cursor.fetchone()
print "Database version : %s " % data

# disconnect from server
db.close()
```

While running this script, it is producing the following result in my Linux machine.

```
Database version : 5.0.45
```

If a connection is established with the datasource, then a Connection Object is returned and saved into **db** for further use, otherwise **db** is set to None. Next, **db** object is used to create a **cursor** object, which in turn is used to execute SQL queries. Finally, before coming out, it ensures that database connection is closed and resources are released.

Creating Database Table

Once a database connection is established, we are ready to create tables or records into the database tables using **execute** method of the created cursor.

Example

Let us create Database table EMPLOYEE:

```
#!/usr/bin/python

import MySQLdb

# Open database connection

db = MySQLdb.connect("localhost","testuser","test123","TESTDB" )

# prepare a cursor object using cursor() method

cursor = db.cursor()

# Drop table if it already exist using execute() method.

cursor.execute("DROP TABLE IF EXISTS EMPLOYEE")

# Create table as per requirement

sql = """CREATE TABLE EMPLOYEE (

         FIRST_NAME  CHAR(20) NOT NULL,

         LAST_NAME  CHAR(20),

         AGE INT,

         SEX CHAR(1),

         INCOME FLOAT )"""


cursor.execute(sql)


# disconnect from server

db.close()
```