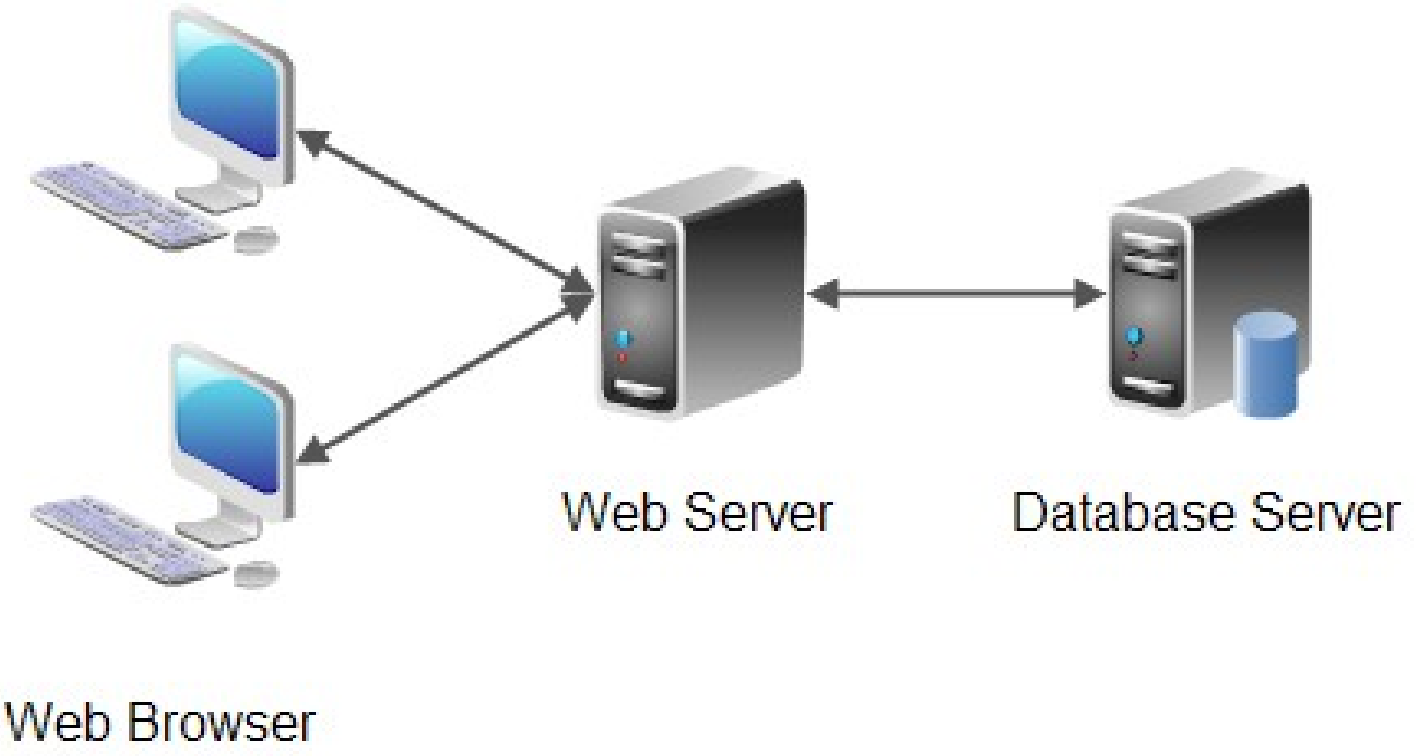
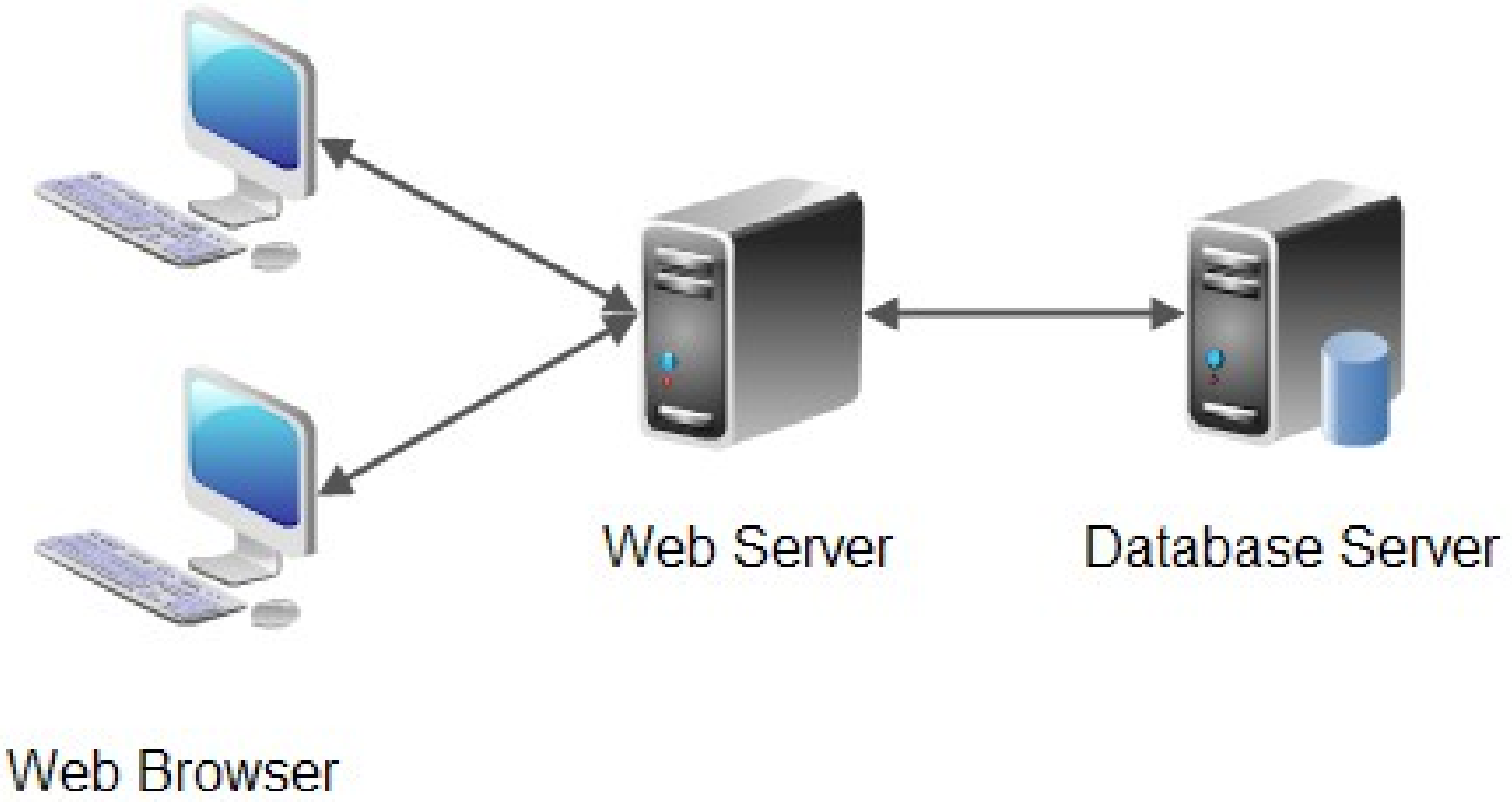
Databases and Webservers



Databases and Webservers



When you connect to a website, you will be connecting to a **webserver**. This server will access scripts, style sheets and other files and will probably also query a **database server**.

LAMP

- Linux — **Operating System**
- Apache — **Webserver**
- MySQL — **Database**
- PHP — **Script language**

The “LAMP stack” is the most common web solution stack in use today.

LAMP

- Linux — **Operating System**
- Apache — **Webserver**
- MySQL — **Database**
- PHP — **Script language**

The “LAMP stack” is the most common web solution stack in use today.

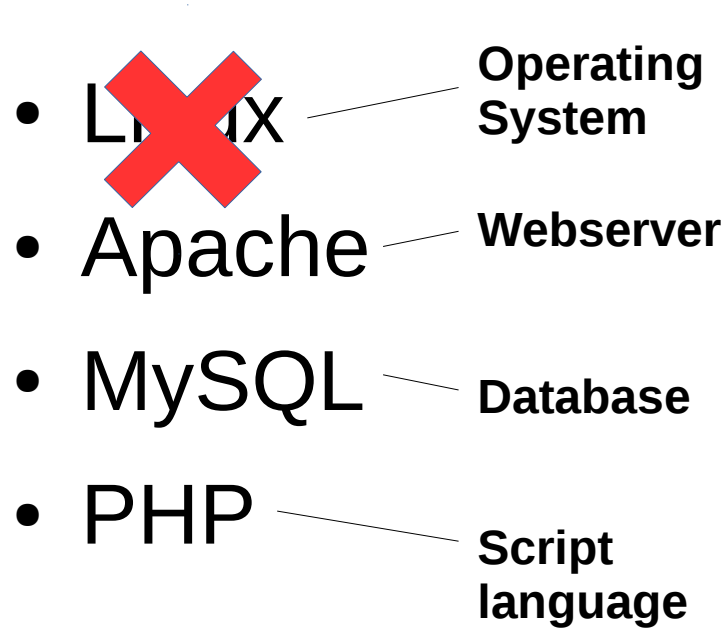
- Open Source
- Free
- Well Supported
- Massive Adoption

but for the Advanced Programming course...

Can't use LAMP for this course:

- Linux — **Operating System**
- Apache — **Webserver**
- MySQL — **Database**
- PHP — **Script language**

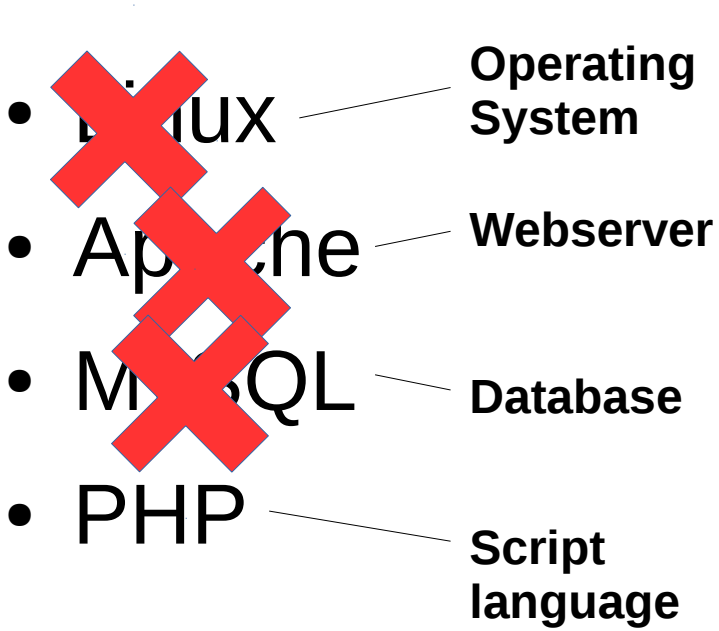
but for the Advanced Programming course...



Can't use LAMP for this course:

- People running windows not linux

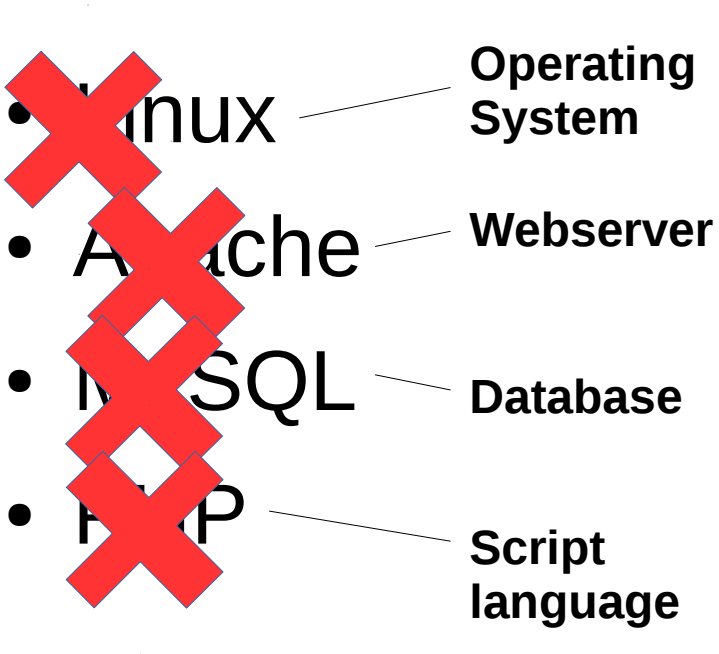
but for the Advanced Programming course...



Can't use LAMP for this course:

- People running windows not linux
- Apache and MySQL too big to download

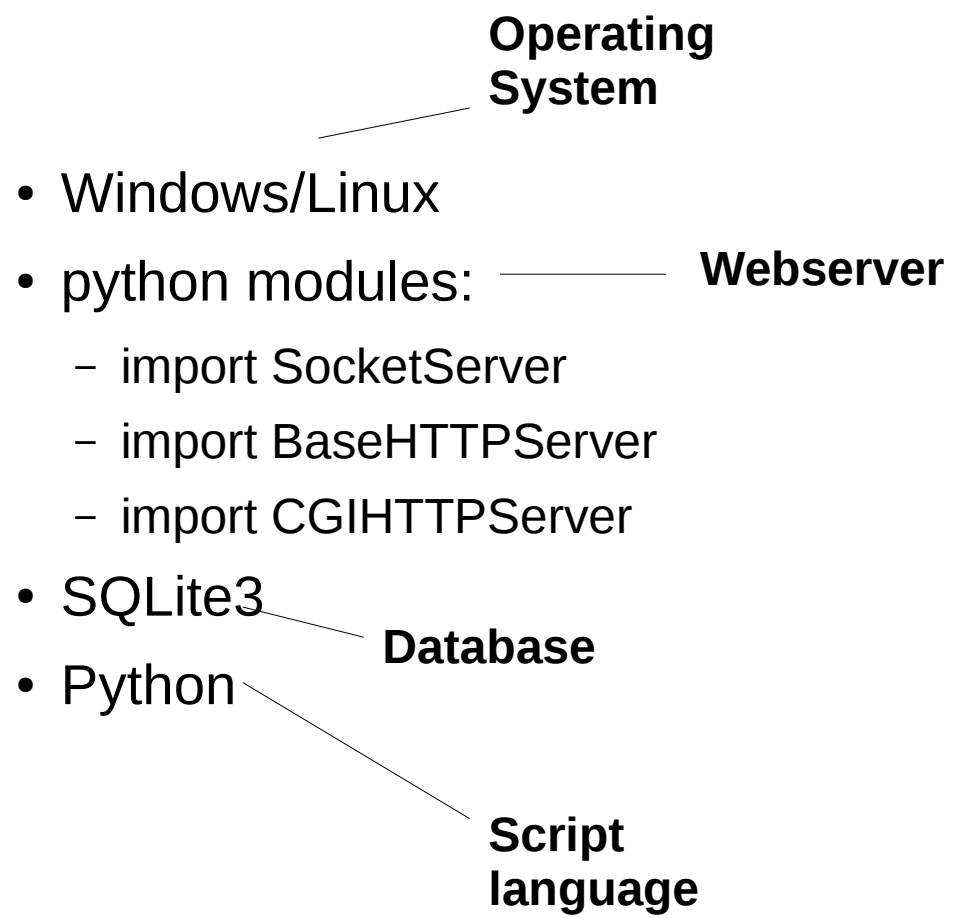
but for the Advanced Programming course...



Can't use LAMP for this course:

- People running windows not linux
- Apache and MySQL too big to download
- Not enough time to cover another language.

So we'll use smaller alternatives:



- Solution is cross platform
- modules already part of python
- SQLite lightweight
- Already know Python

To get started with our stack

- Download SQLite command line tool (this is useful for learning SQL).
- Download the webserver script

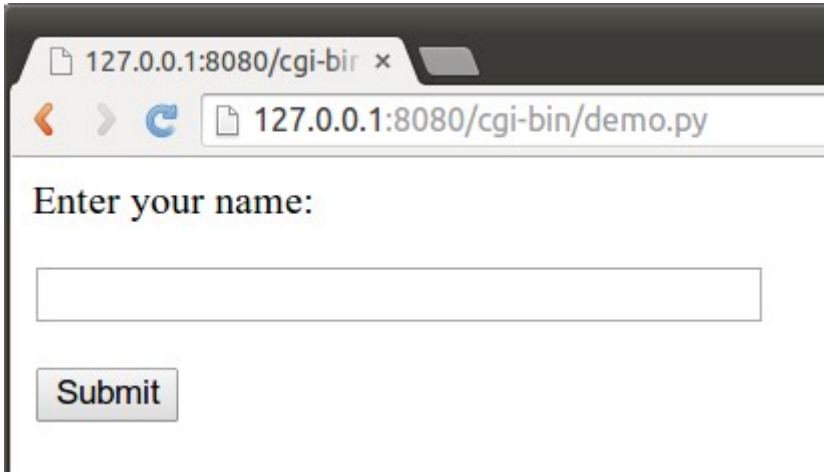
Checking things are working... 1. sqlite

- Open a terminal (in windows this is done by running **cmd**)
- **cd** to the right location
- run:
- **sqlite3 test.db**
- type:
- **.quit**
- check the `test.db` file now exists

It's important we always use sqlite3 and no other version as file formats are not compatible...

Checking things are working... 2. python simple webserver

- To check it works run the `httpserver.py` python program
- Visit `http://127.0.0.1:8080`
- Edit the demo.py script in the cgi-bin folder
  - You may need to delete one or both of the #! lines at the top of the script depending on your operating system.
  - visit `http://127.0.0.1:8080/demo.html`



Learning a bit of SQL (using SQLite)

- We are going to make a simple database of people's favourite food and where they're from.
- Step 1: Create a database, run:

```
sqlite3 test.db
```

- Step 2: Create a table (see the sqlcheatsheet.jpg file for help on SQL queries):

```
CREATE TABLE people(name VARCHAR(200), food VARCHAR(200), homedistrict VARCHAR(200));
```

Learning a bit of SQL (using SQLite)

- 3: Try inserting a row into the table:
  `INSERT INTO people(name, food, homedistrict) VALUES ('bob','posho','kampala');`
- 4: Check the data is there:
  `SELECT * FROM people;`
- 5: Update the data:
  `UPDATE people SET homedistrict='gulu' WHERE name='bob';`

Hint: To get a more useful output turn headers on (this shows the names of the table columns):

`.headers ON`

Learning a bit of SQL (using SQLite)

- 6: We want to know where the districts are, add a new table:

```
  CREATE TABLE district(name
VARCHAR(200), lat FLOAT(10,6), long
FLOAT(10,6));
```

- 7: Add data:

```
  INSERT INTO district(name, lat, long)
VALUES ('kampala',0.317, 32.583);
  INSERT INTO district(name, lat, long)
VALUES ('tororo',0.75, 34.083);
```

Learning a bit of SQL (using SQLite)

- 8: Use a table JOIN to combine the two tables:

```
SELECT people.food, district.lat FROM people INNER JOIN district ON (people.homedistrict=district.name);
```

Combining the SQL server with the webserver

- Have a look at the demo.html file and the demo.py script, how does it work?
  - 1) When you visit demo.html the HTML you get describes a web form
  - 2) the ACTION parameter tells your browser where to submit the form. In this case to the cgi-bin/demo.py script.
  - 3) the demo.py script uses the cgi library to get out the 'name' that was passed and prints it.

Side Note: Security and Cross Site Scripting (XSS)

Cross-Site Scripting (XSS) attacks are a type of **injection**, in which malicious scripts are injected into otherwise trusted web sites.

XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user.

Flaws that allow these attacks to succeed are quite widespread and **occur anywhere a web application uses input from a user within the output it generates without validating or encoding it**.

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

XSS continued...

- E.g.:

- Or even a script:

  <body onload="alert('yeah');"></body>

https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

- We want to ask the user their name, favourite food and home district and put the results into the database.
- 1: Make a copy of the demo.py script and demo.html file, call them: food.py and food.html.

  Handy hint!!! Things won't work if you access the file directly with the browser. For instance if you just open the file. If you've done this you'll see **file://stuff** in the address bar. We want to access the file VIA THE WEBSERVER, so we want to see: **http://127.0.0.1:8080/stuff**
- 2: Change the HTML to ask for the person's:
  - name
  - favourite food
  - home district
- 3: Change the food.py script to process and display these new values.
- 4. Setup an SQLite3 database to store the files in (the lecture notes on how to do that are on this computer 10.10.10.94).
- 5. Modify food.py to insert these in the database.
- 6. Make a new script which lists the districts and favourite foods.

Database access from python using sqlite3!

- First try connecting to a database...

```
import sqlite3 as lite
try:
    con = lite.connect('stuff.db')
    cur = con.cursor()
except lite.Error, e:
    print "Error %s:" % e.args[0]
    sys.exit(1)
if con:
    con.close()
```

- Second, let's try selecting stuff from the database:

```
import sqlite3 as lite
try:
    con = lite.connect('stuff.db')
    cur = con.cursor()
    cur.execute('SELECT something FROM
atable WHERE stuff=?',(variable,))
    data = cur.fetchone()
    if (data==None):
        print “Not found anything”
    else:
        Print “Found: “+data
except lite.Error, e:
    print "Error %s:" % e.args[0]
    sys.exit(1)
if con:
    con.close()
```

- Third, we could insert a row into a table...

```
import sqlite3 as lite
try:
    A = 4;
    B = 2;
    con = lite.connect('stuff.db')
    cur = con.cursor()
    cur.execute('INSERT INTO atable
(something, blah) VALUES (?,?)',(A,B,))
        con.commit()
except lite.Error, e:
    print "Error %s:" % e.args[0]
    sys.exit(1)
if con:
    con.close()
```