Server-Side Web Programming: CGI (Part 1)

Copyright © 2026 by

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover…
  - Common Gateway Interface (CGI) programming
  - The HTTP GET method
  - The HTTP POST method

Agenda

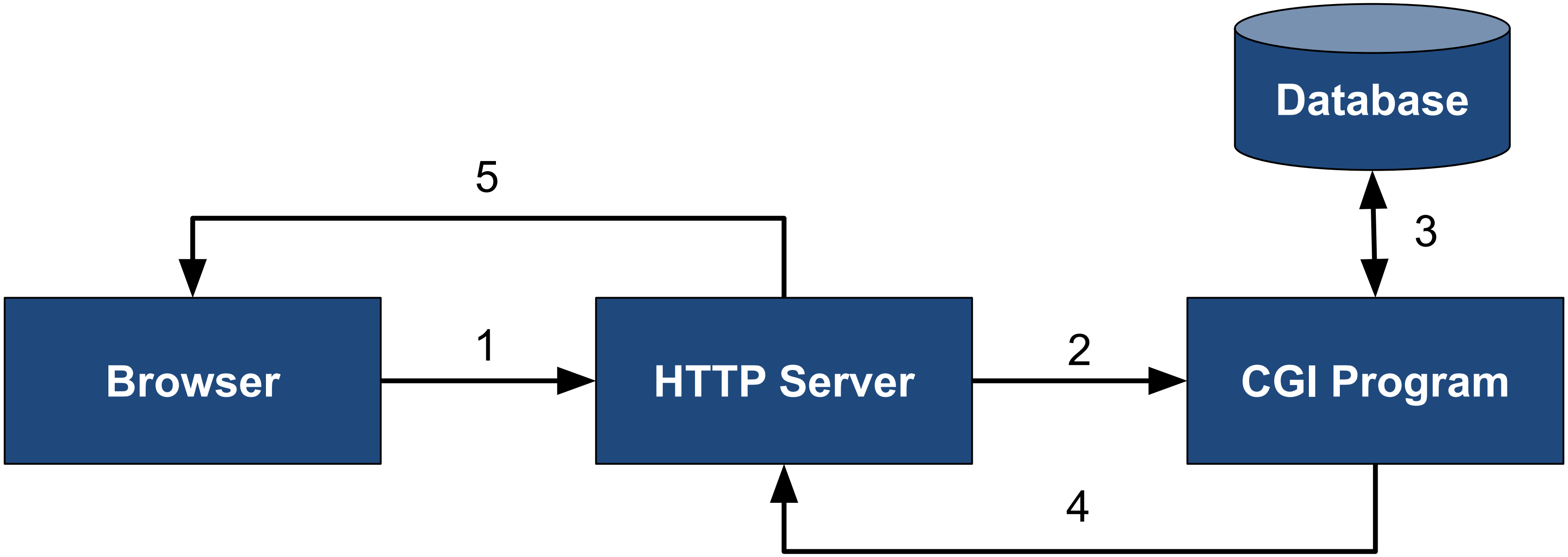
- **CGI programming**
- URL encoding
- CGI with arguments
- CGI using the HTTP POST method

CGI

- **Problem**:
  - Often HTTP server must generate HTML docs **dynamically**

CGI

- One **solution**:
- ***Common Gateway Interface (CGI)*** protocol



CGI

- **Question**: How does user command browser to command the HTTP server to execute a CGI program?
- **Answer 1**: User enters URL at top of browser

CGI

- Answer 1 elaboration

```
protocol://host:port/cgi-bin/somepgm.py
```

The default protocol is http
There is no default host
The default port is 80 for http, 443 for https

CGI

- **Question**: How does user command browser to command the HTTP server to execute a CGI program?
- **Answer 1**: User enters URL at top of browser
- **Answer 2**: User clicks on page link

CGI

- Answer 2 elaboration

```
<a href="protocol://host:port/cgi-bin/somepgm.py">
    ...
</a>
```

The default protocol is the protocol used to deliver this page

The default host is the host that delivered **this** page

The default port is the port from which **this** page was delivered

CGI

- **Question**: How does user command browser to command the HTTP server to execute a CGI program?
- **Answer 1**: User enters URL at top of browser
- **Answer 2**: User clicks on page link
- **Answer 3**: User submits a form

CGI

- Answer 3 elaboration

```
<form action="protocol://host:port/cgi-bin/somepgm.py" method="get">
   <input type="submit">
</form>
```

The default protocol is the protocol used to deliver this page
The default host is the host that delivered **this** page
The default port is the port from which **this** page was delivered

CGI

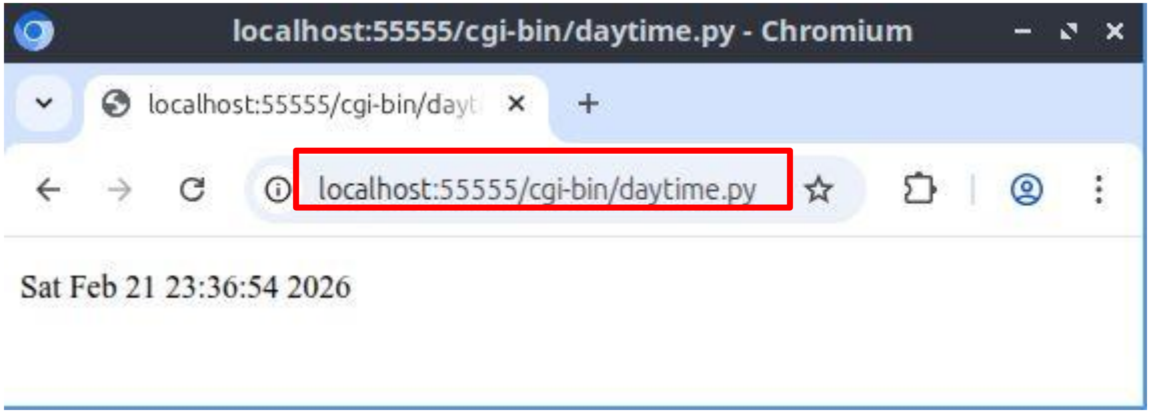
- See **DayTime** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

CGI

- See **DayTime** app (cont).

Approach 1:



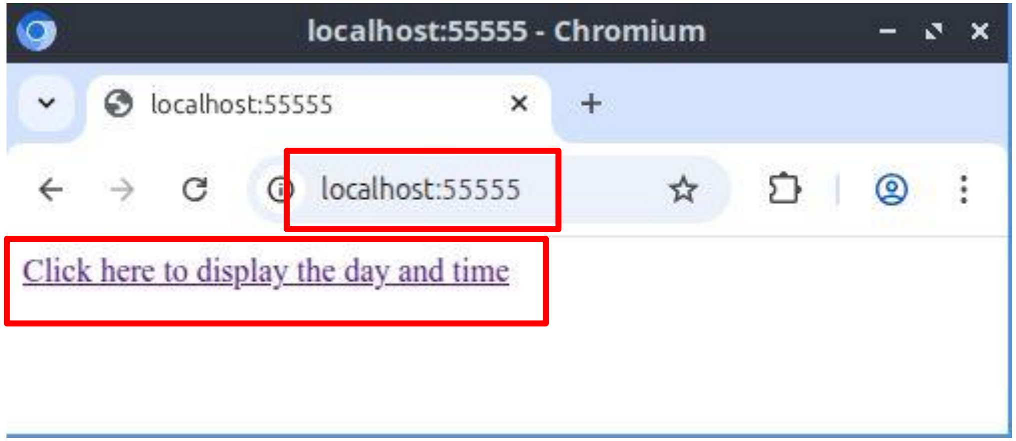
localhost:55555/cgi-bin/daytime.py - Chromium
localhost:55555/cgi-bin/dayt
localhost:55555/cgi-bin/daytime.py

```
Sat Feb 21 23:36:54 2026
```

CGI

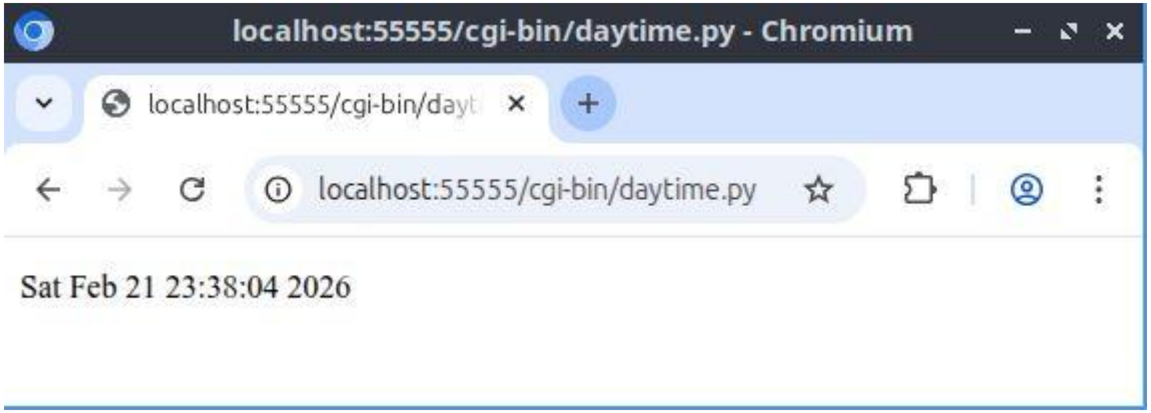
- See **DayTime** app (cont.)

Approach 2:



localhost:55555

Click here to display the day and time


localhost:55555/cgi-bin/daytime.py - Chromium
localhost:55555/cgi-bin/dayt

localhost:55555/cgi-bin/daytime.py

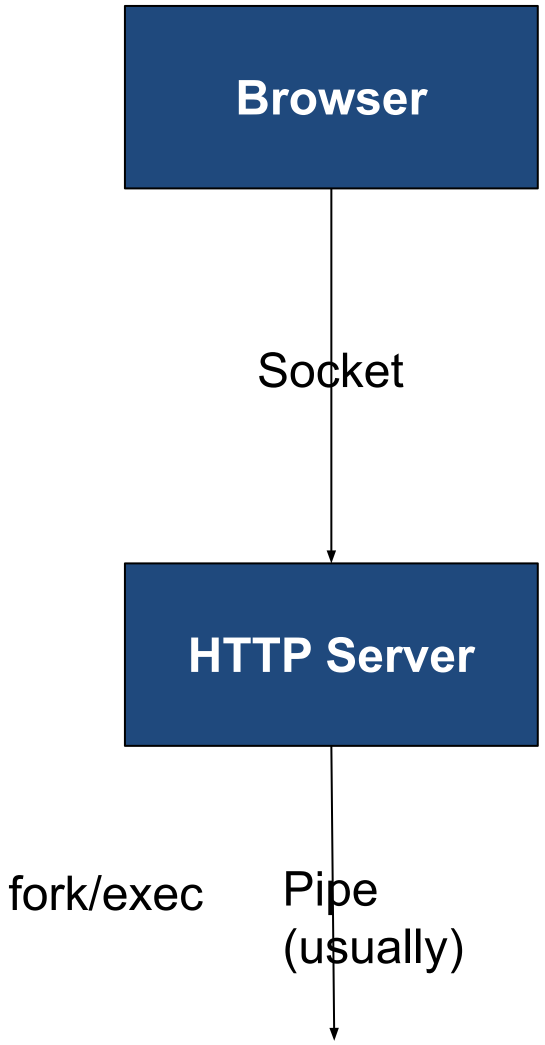
```
Sat Feb 21 23:38:04 2026
```

CGI

- See **DayTime** app (cont.)
  - **runserver.py**
  - **index.html**
  - **cgi-bin/daytime.py**

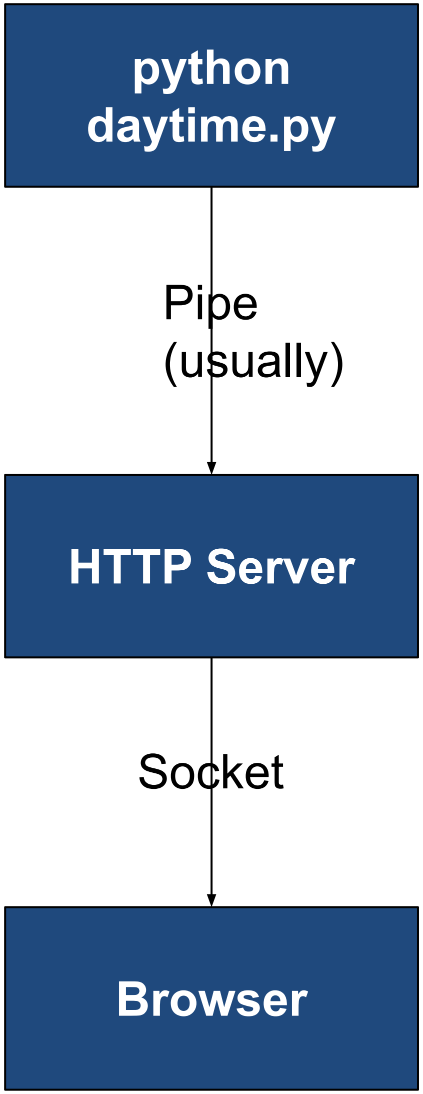
CGI with Arguments

User clicks page link



```
GET /cgi-bin/daytime.py HTTP/1.1
Host: localhost
<Blank line>
```

CGI with Arguments



```
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<body>
The current time
</body>
</html>
```

Writes to stdout

```
HTTP/1.1 200 OK
…
Content-Type: text/html;charset=utf-8

<!DOCTYPE html>
<html>
<body>
The current time
</body>
</html>
```

There are some additional headers

Agenda

- CGI programming
- **URL encoding**
- CGI with arguments
- CGI using the HTTP POST method

URL Encoding

- **Problem:**
  - URLs cannot contain spaces or special characters (", ', =, &, etc.)
- **Solution:**
  - ***URL encoding***

URL Encoding

- ***URL encoding***
  - A way of encoding a string such that it may appear within a URL
  - Each space is encoded as +
  - Each special character is encoded as:
    - `%nn` where `nn` are hex digits as per UTF-8, or
    - `%nn%nn`, or
    - `%nn%nn%nn`, or
    - `%nn%nn%nn%nn`

URL Encoding

- Python support for URL encoding
  - **`urllib.parse.quote_plus()`**
    - string → URL encoded string
  - **`urllib.parse.unquote_plus()`**
    - URL encoded string → string

URL Encoding

```
$ python
Python 3.12.3 (main, Jun 18 2025,
17:59:45) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> import urllib.parse
>>> urllib.parse.quote_plus('one')
'one'
>>> urllib.parse.quote_plus('t wo')
't+wo'
>>> urllib.parse.quote_plus('th"r ee')
'th%22r+ee'
>>>
urllib.parse.quote_plus('!@#$%^&*()')
'%21%40%23%24%25%5E%26%2A%28%29'
>>> quit()
$
```

Agenda

- CGI programming
- URL encoding
- **CGI with arguments**
- CGI using the HTTP POST method

CGI with Arguments

- **Question**: How does browser pass arguments to a CGI program?
- **Answer 1**: User enters URL at top of browser with `name=value` pairs appended

CGI with Arguments

- Answer 1 elaboration

```
protocol://host:port/cgi-bin/somepgm.py?
    name1=value1&name2=value2
```

CGI with Arguments

- **Question**: How does browser pass arguments to a CGI program?
- **Answer 1**: User enters URL at top of browser
- **Answer 2**: User clicks on page link whose `href` attribute has `name=value` pairs appended

CGI with Arguments

- Answer 2 elaboration

```
<a href="protocol://host:port/cgi-bin/somepgm.py?
   name1=value1&name2=value2">
   ...
</a>
```

CGI with Arguments

- **Question**: How does browser pass arguments to a CGI program?
- **Answer 1**: User enters URL at top of browser
- **Answer 2**: User clicks on page link

  Answer 3: User submits form that has `input` elements of type `text`

CGI with Arguments

- Answer 3 elaboration

```
<form action="protocol://host:port/cgi-bin/somepgm.py" method="get">
   <input type="text" name="name1">
   <input type="text" name="name2">
   <input type="submit">
</form>
```

User enters *value1* and *value2* in `input` elements

Sends HTTP request as if generated by clicking on page link with href

`protocol://host:port/cgi-bin/somepgm.py?name1=value1&name2=value2`

CGI with Arguments

- Answer 3 elaboration (cont.)

```
<form action="protocol://host:port/cgi-bin/somepgm.py" method="get">
   <input type="text" name="name1" value="init1">
   <input type="text" name="name2" value="init2">
   <input type="submit" value="somelabel">
</form>
```

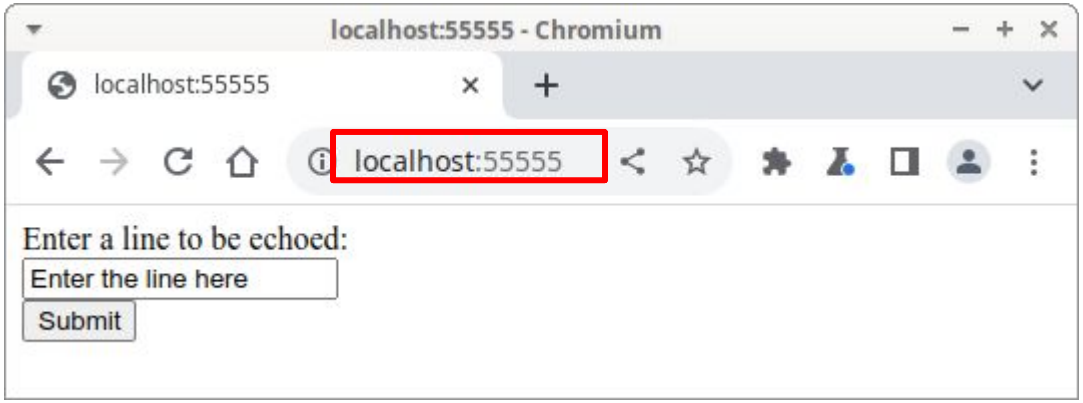
CGI with Arguments

- See **EchoGet** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

CGI with Arguments

- See **EchoGet** app (cont.)

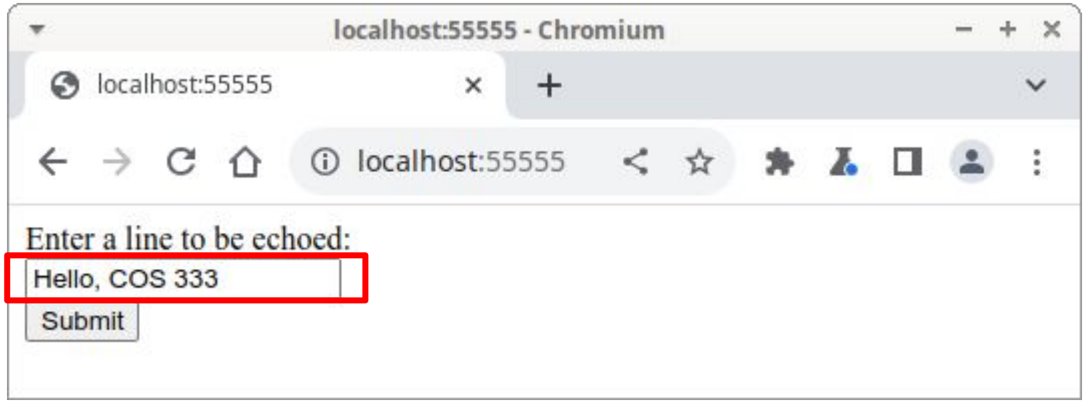


Enter a line to be echoed:

Enter the line here

CGI with Arguments

- See **EchoGet** app (cont.)

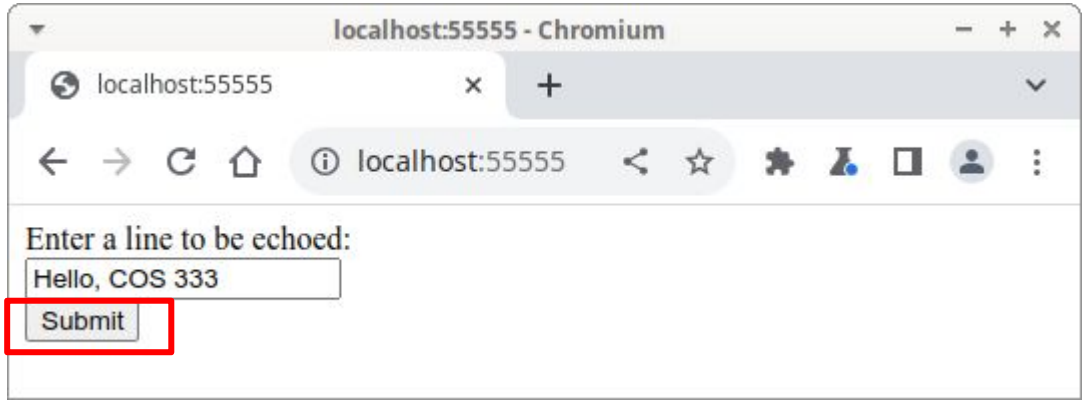


Enter a line to be echoed:

Hello, COS 333

CGI with Arguments

- See **EchoGet** app (cont.)

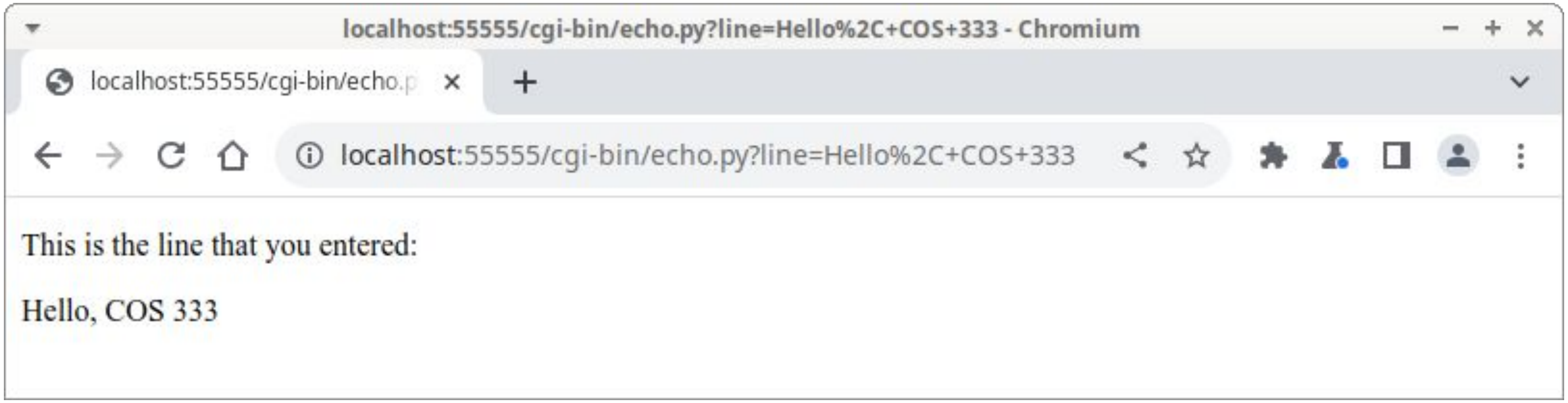


Enter a line to be echoed:

Hello, COS 333

CGI with Arguments

- See **EchoGet** app (cont.)



localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333 - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py?line=Hello%2C+COS+333

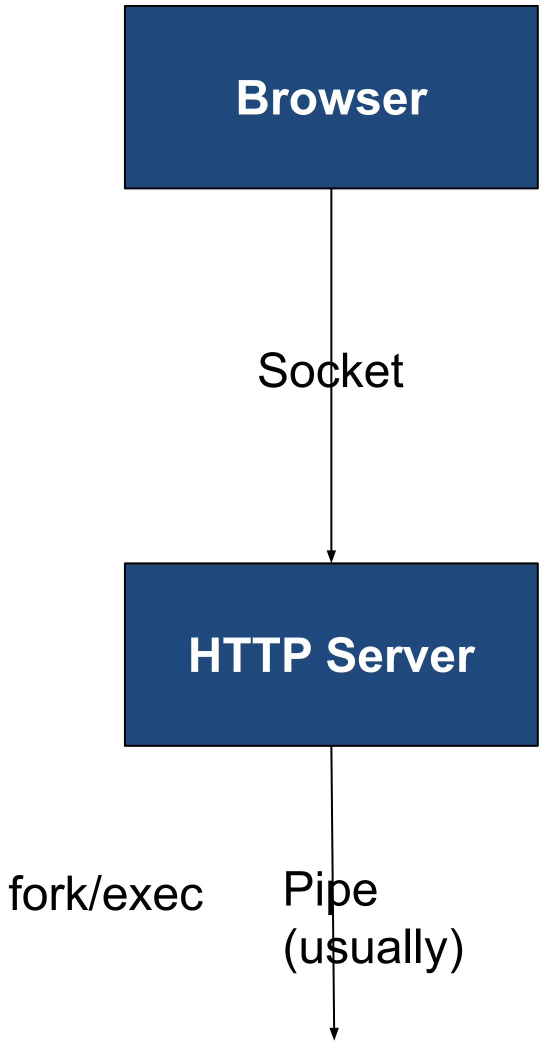
This is the line that you entered:

Hello, COS 333

CGI with Arguments

- See **EchoGet** app (cont.)
  - runserver.py
  - **index.html**
  - **cgi-bin/parseargs.py**
  - **cgi-bin/echo.py**

CGI with Arguments



Browser

Socket

HTTP Server

fork/exec

Pipe (usually)

User enters “Hello, COS 333” and clicks form “submit” button

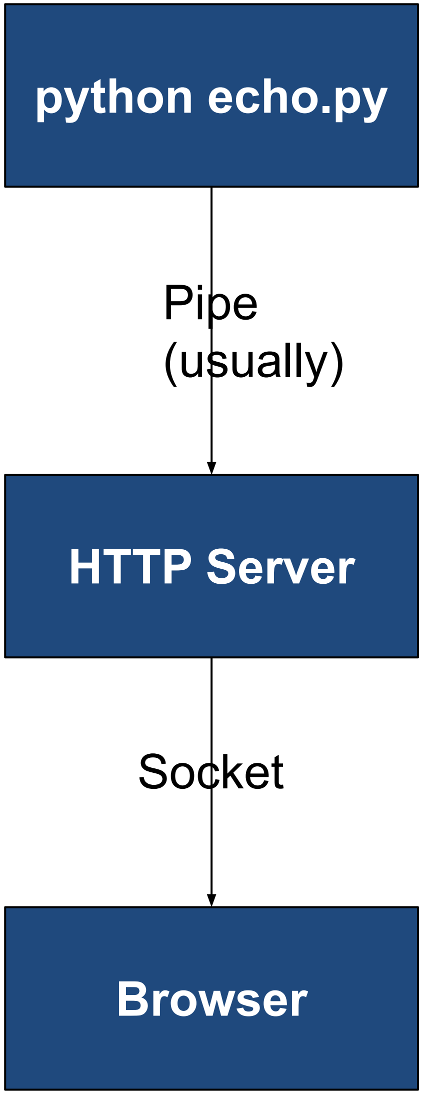
Names and values are URL encoded

```
GET /cgi-bin/echo.py?line=Hello%2C+COS+333 HTTP/1.1
Host: localhost
<Blank line>
```

Environment variable

```
QUERY_STRING: line=Hello%2C+COS+333
```

CGI with Arguments



```
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<body>
<p>Hello, COS 333</p>
</body>
</html>
```

Uses
QUERY_STRING
Writes to stdout

```
HTTP/1.1 200 OK
…
Content-Type: text/html;charset=utf-8

<!DOCTYPE html>
<html>
<body>
<p>This is the line that you entered:</p>
<p>Hello, COS 333</p>
</body>
</html>
```

There are
some additional
headers

Agenda

- CGI programming
- URL encoding
- CGI with arguments
- **CGI using the HTTP POST method**

CGI Using POST

- **So far:** CGI with the GET method

- **Question**: How does user command browser to communicate with HTTP server using CGI with the **POST** method?
- **Answer**: User submits form

CGI Using POST

- Answer elaboration

```
<form action="protocol://host:port/cgi-bin/somepgm.py" method="post">
   <input type="text" name="name1" value="init1">
   <input type="text" name="name2" value="init2">
   <input type="submit" value="somelabel">
</form>
```

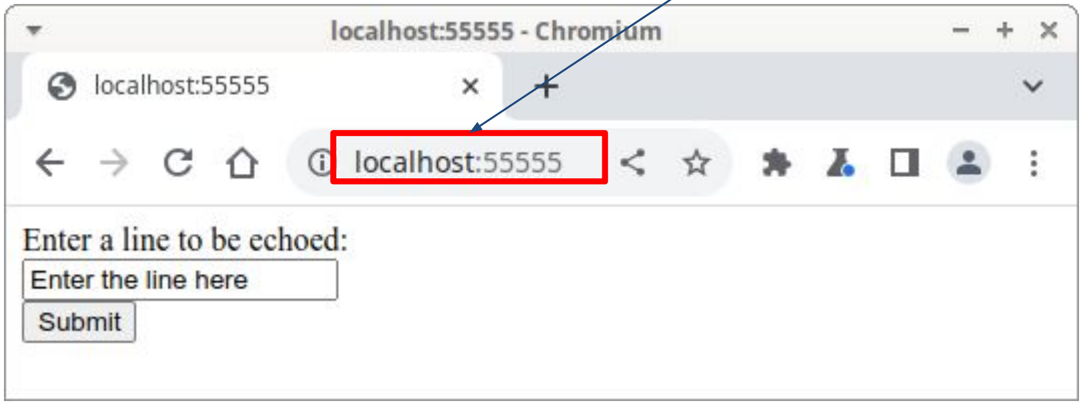
CGI Using POST

- See **EchoPost** app

```
$ python runserver.py 55555
Serving HTTP on 0.0.0.0 port 55555 (http://0.0.0.0:55555/) ...
```

CGI Using POST

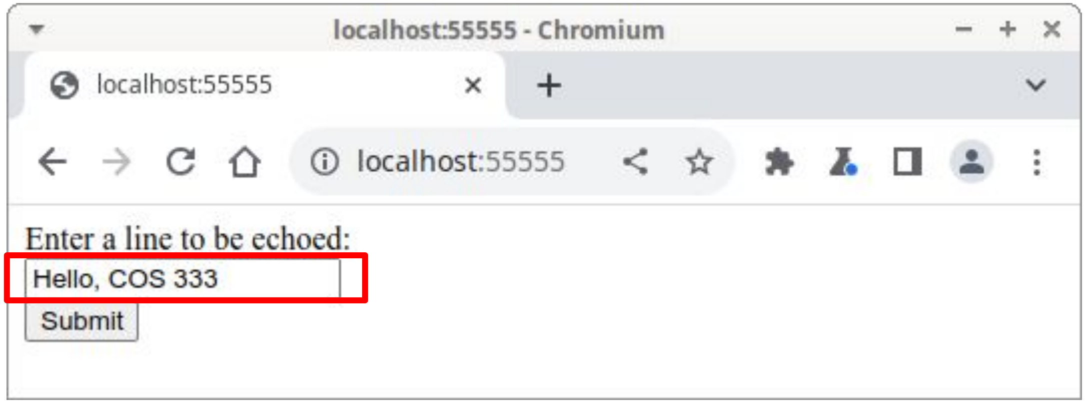
- See **EchoPost** app (cont.)



Default protocol is HTTP
Default page is index.html

CGI Using POST

- See **EchoPost** app (cont.)

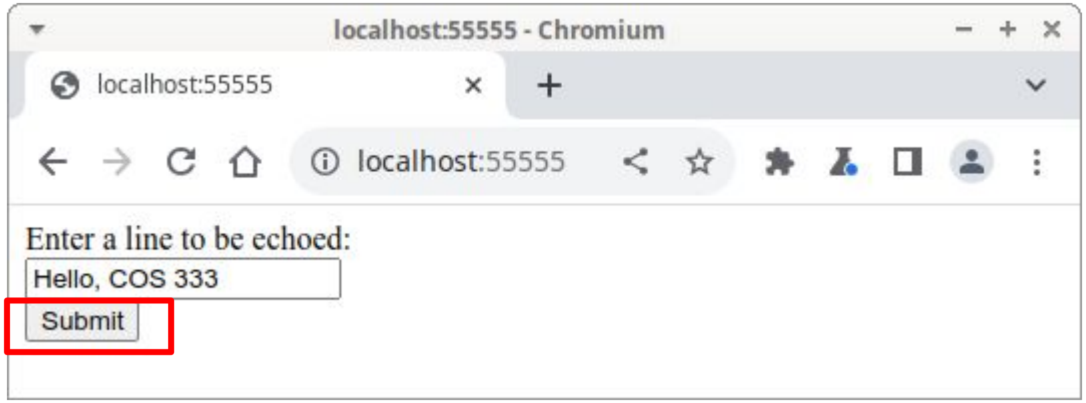


Enter a line to be echoed:

Hello, COS 333

CGI Using POST

- See **EchoPost** app (cont.)

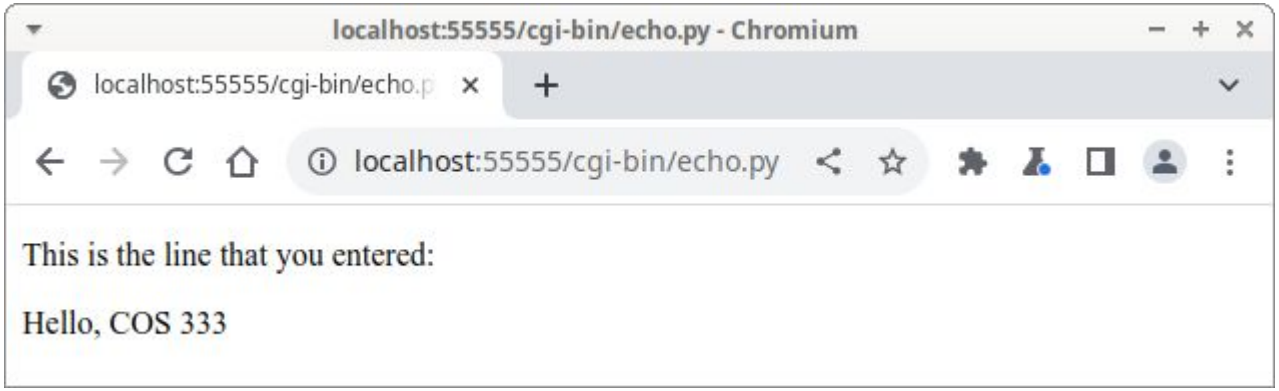


Enter a line to be echoed:

Hello, COS 333

CGI Using POST

- See **EchoPost** app (cont.)


localhost:55555/cgi-bin/echo.py - Chromium
localhost:55555/cgi-bin/echo.p
localhost:55555/cgi-bin/echo.py

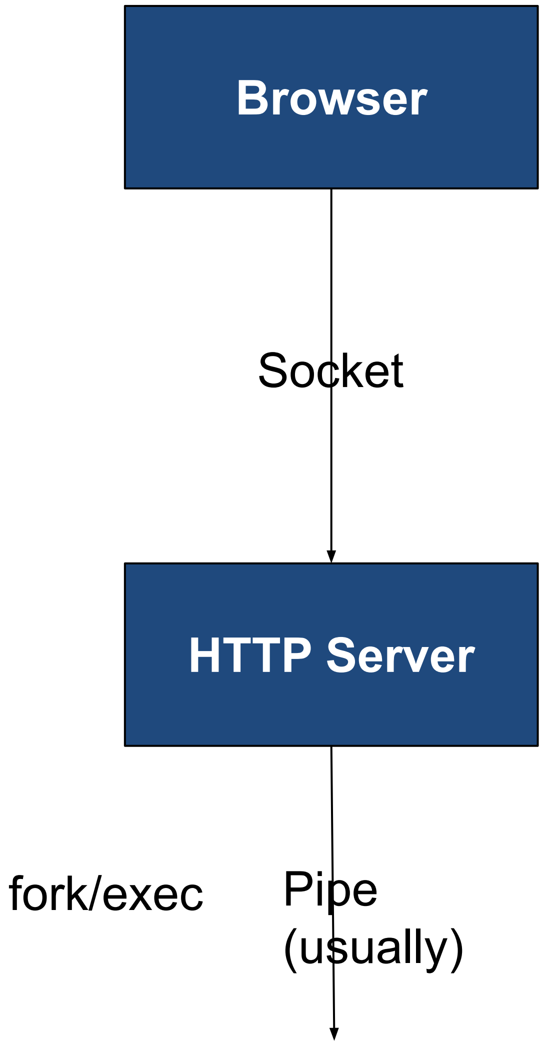
This is the line that you entered:

Hello, COS 333

CGI Using POST

- See **EchoPost** app (cont.)
  - runserver.py
  - **index.html**
  - cgi-bin/parseargs.py
  - **cgi-bin/echo.py**

CGI Using POST



Browser

Socket

HTTP Server

fork/exec

Pipe
(usually)

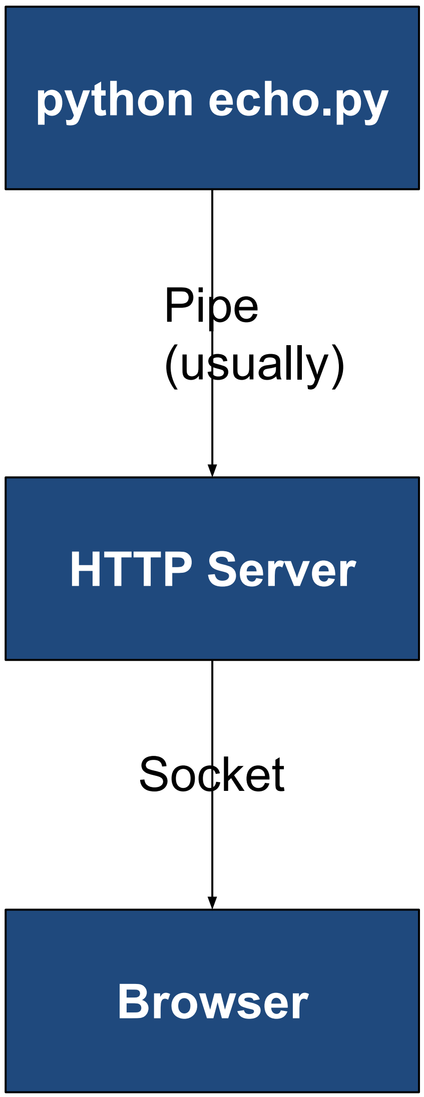
User enters “Hello, COS 333”
and clicks form “submit” button

```
POST /cgi-bin/echo.py HTTP/1.1
Host: localhost
Content-type: application/x-www-form-urlencoded
Content-length: 19

line=Hello%2C+COS+333
```

Binds stdin of child process to pipe
Writes to pipe: line=Hello%2C+COS+333

CGI Using POST



```
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<body>
<p>Hello, COS 333</p>
</body>
</html>
```

Reads from stdin
Writes to stdout

```
HTTP/1.1 200 OK
…
Content-Type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<body>\
<p>This is the line that you entered:</p>
<p>Hello, COS 333</p>
</body>
</html>
```

There are some additional headers

Lecture Summary

- In this lecture we covered:
  - Common Gateway Interface (CGI) programming
  - The HTTP GET method
  - The HTTP POST method