Server-Side Web Programming: Python (Part 1)

Robert M. Dondero, Ph.D
Princeton University

Objectives

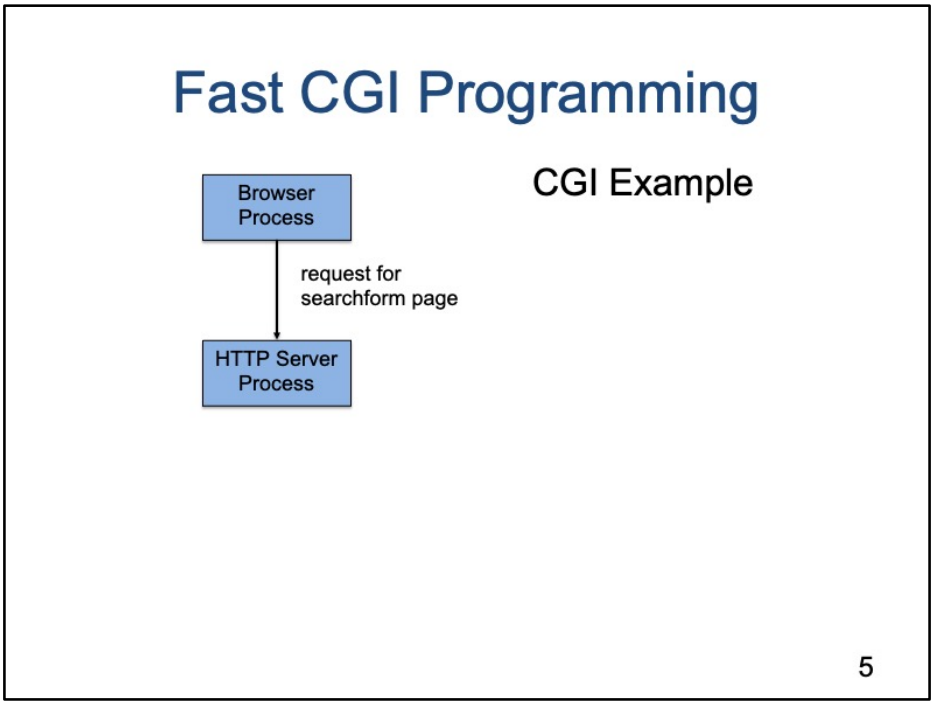
- We will cover:
  - Fast CGI programming
  - Python WSGI programming

Agenda

- **Fast CGI programming**
- Python WSGI programming

Fast CGI Programming

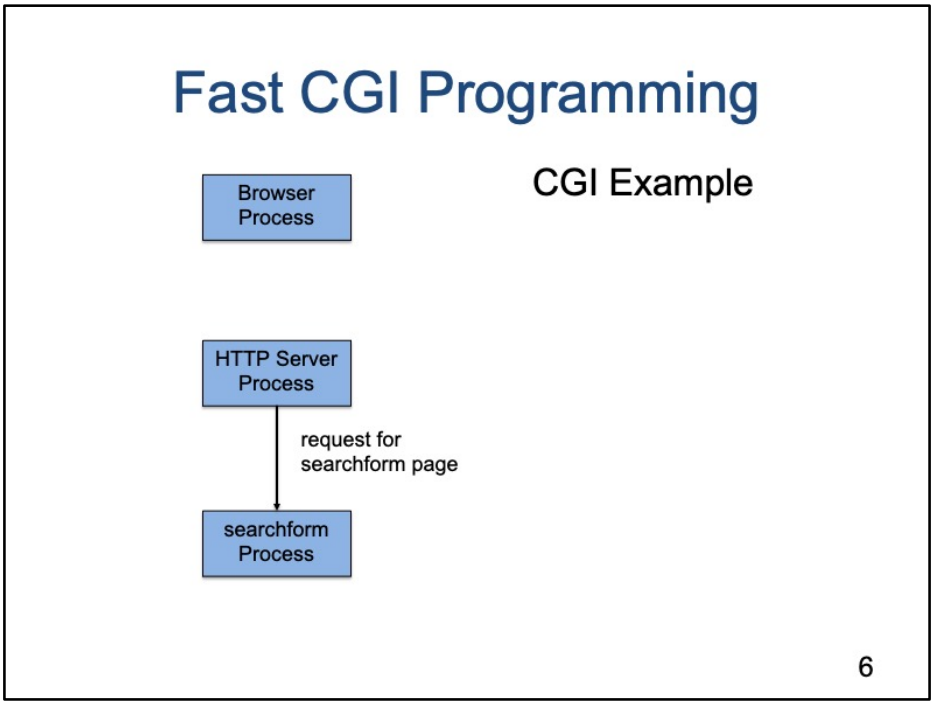
- **Problem**: CGI apps are slow
  - For each request, HTTP server forks/execs a new process
  - OK for low-volume website
  - Maybe not OK for high-volume website



Fast CGI Programming

[see slide]

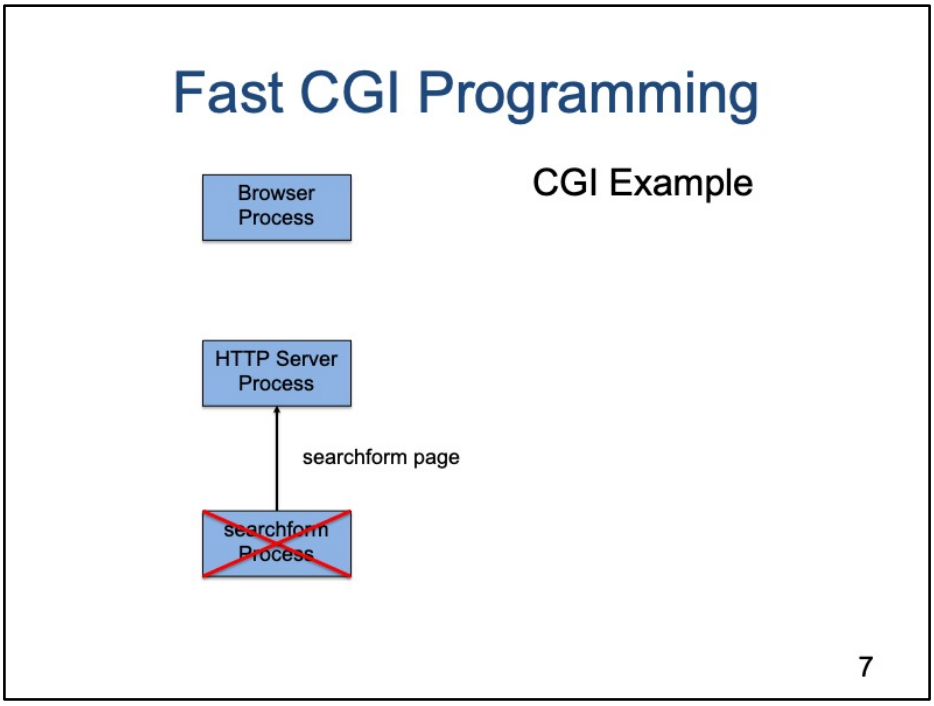
Browser process asks HTTP server process for searchform page



Fast CGI Programming

[see slide]

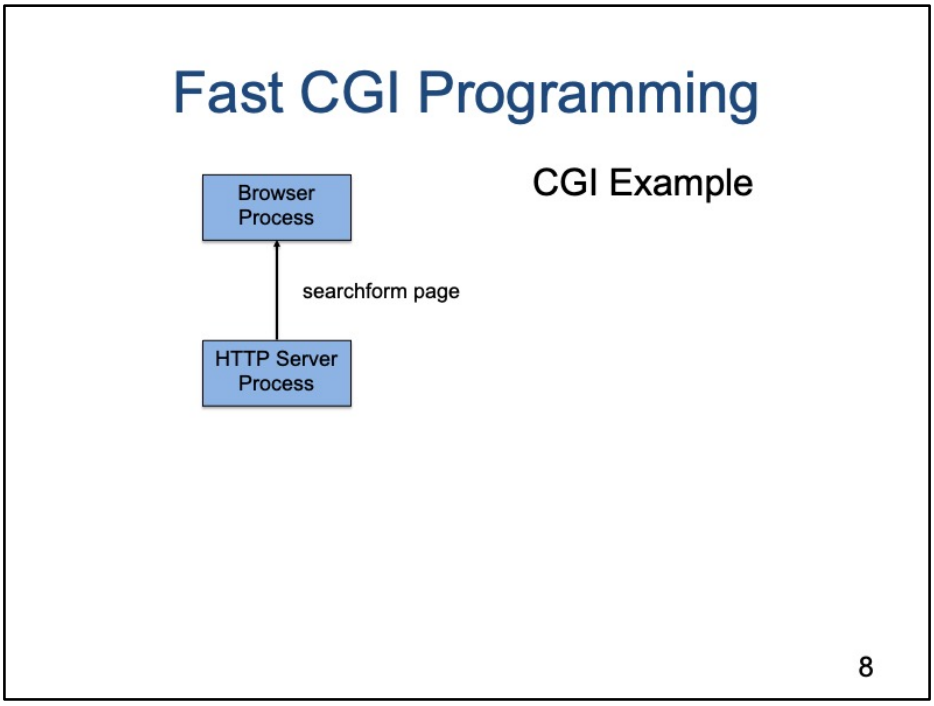
HTTP server process forks child process, execs searchform program



Fast CGI Programming

[see slide]

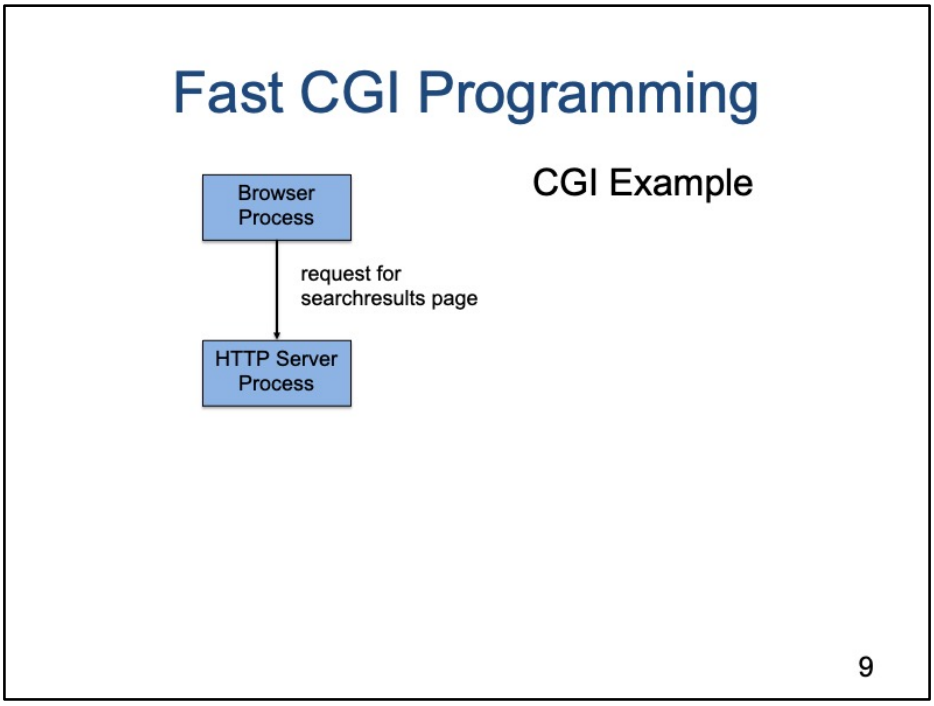
Searchform process passes searchform page to HTTP server process
Searchform process exits



Fast CGI Programming

[see slide]

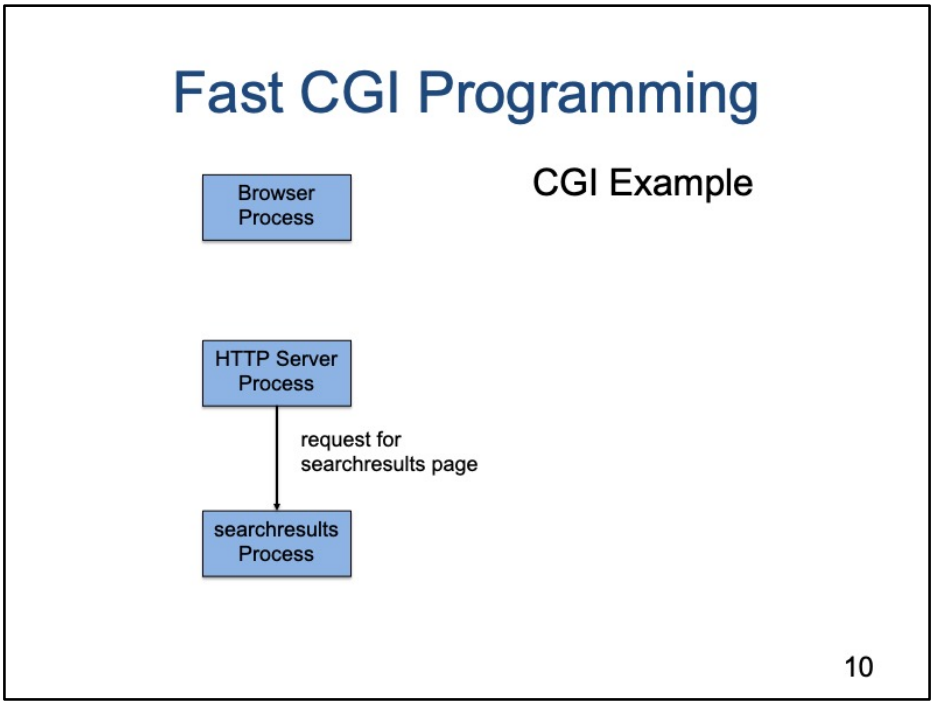
HTTP server process passes searchform page to browser



Fast CGI Programming

[see slide]

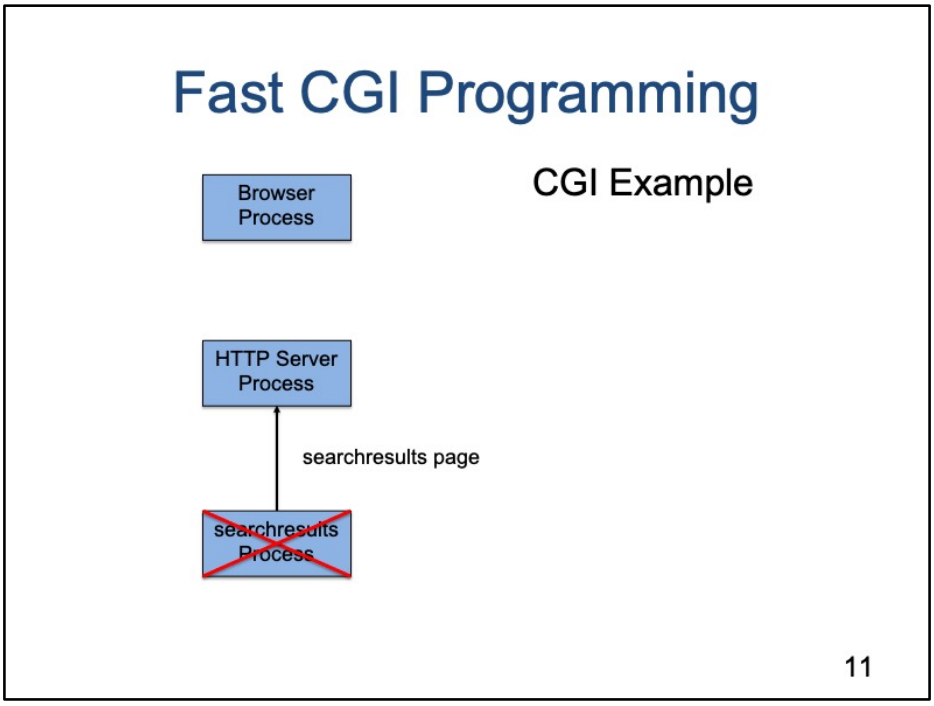
Browser process asks HTTP server process for searchresults page



Fast CGI Programming

[see slide]

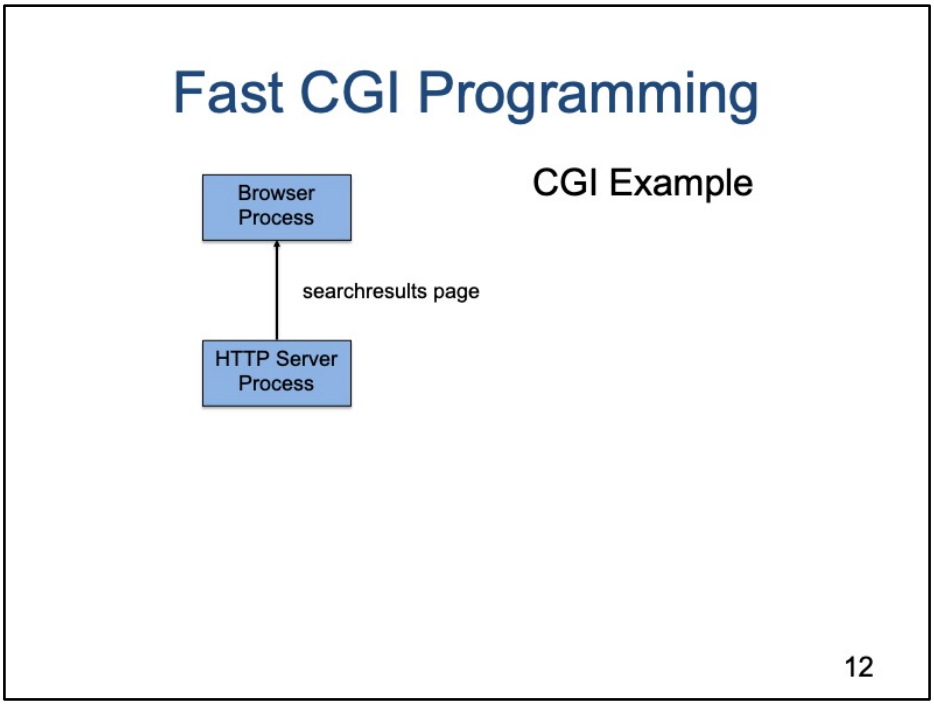
HTTP server process forks child process, execs searchresults program



Fast CGI Programming

[see slide]

Searchresults process passes searchresults page to HTTP server process
Searchresults process exits



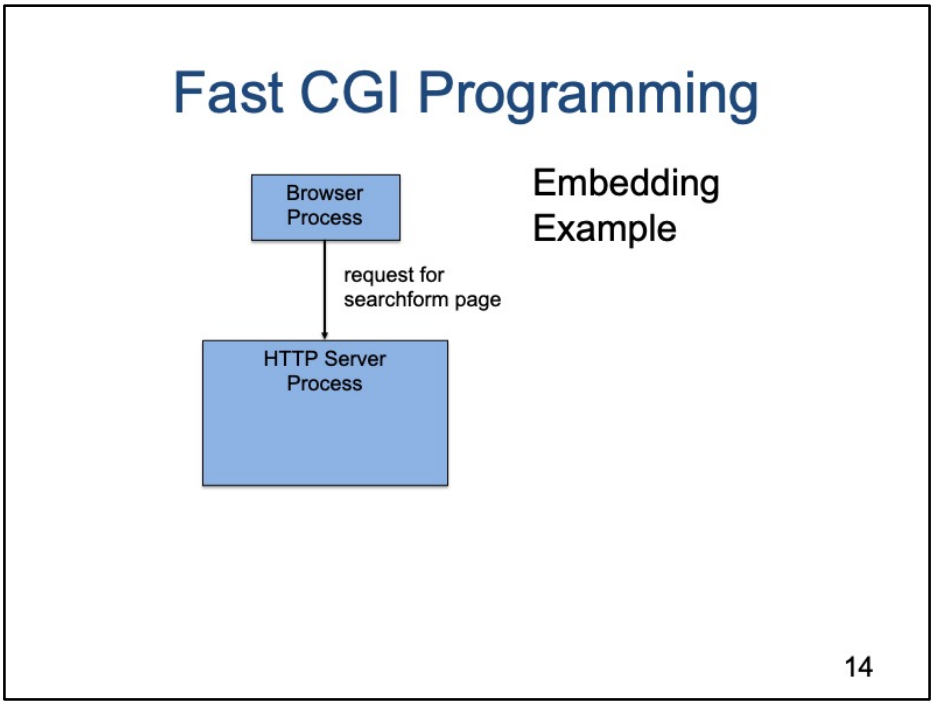
Fast CGI Programming

[see slide]

HTTP server process passes searchresults page to browser

Fast CGI Programming

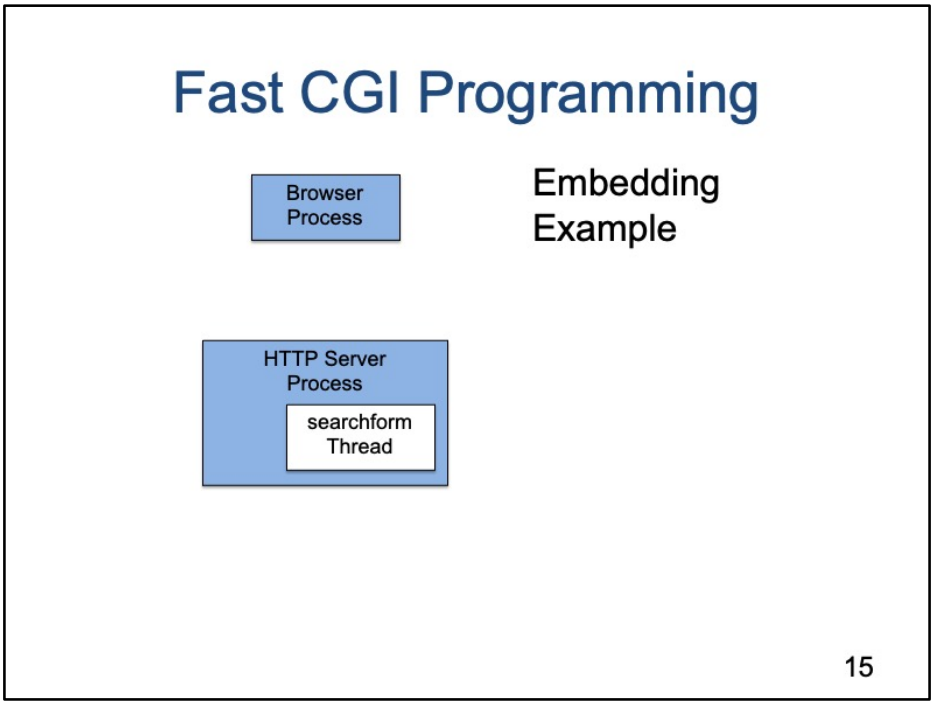
- **Solution 1**: Embed apps in HTTP server process
  - HTTP server does not **fork processes**
  - HTTP server **spawns threads** in its process
    - (See *Concurrency* lectures later in course)



Fast CGI Programming

[see slide]

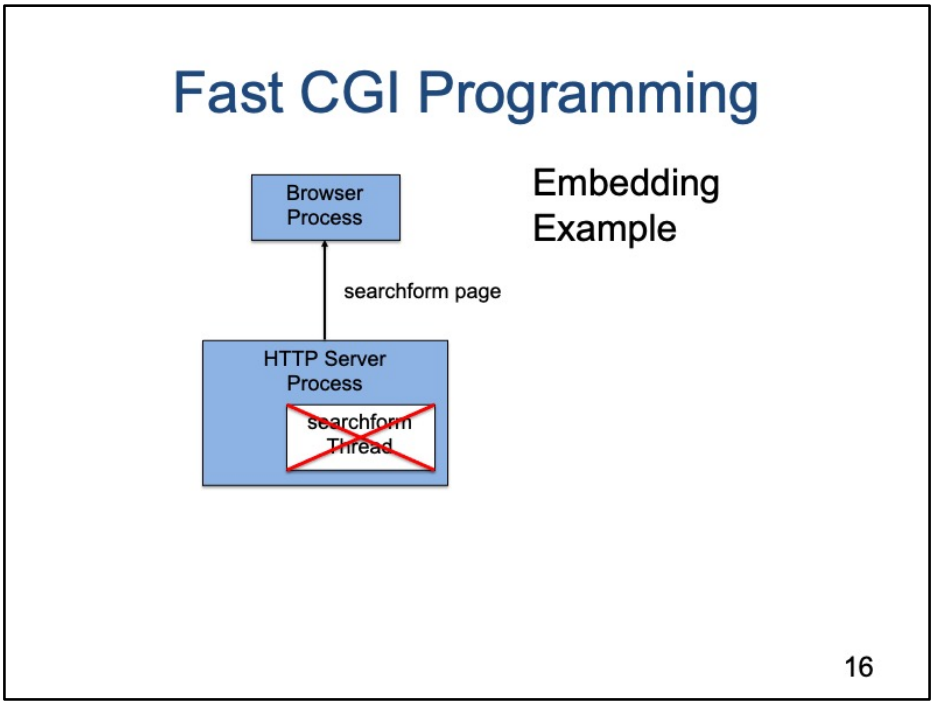
Browser process asks HTTP server process for searchform page



Fast CGI Programming

[see slide]

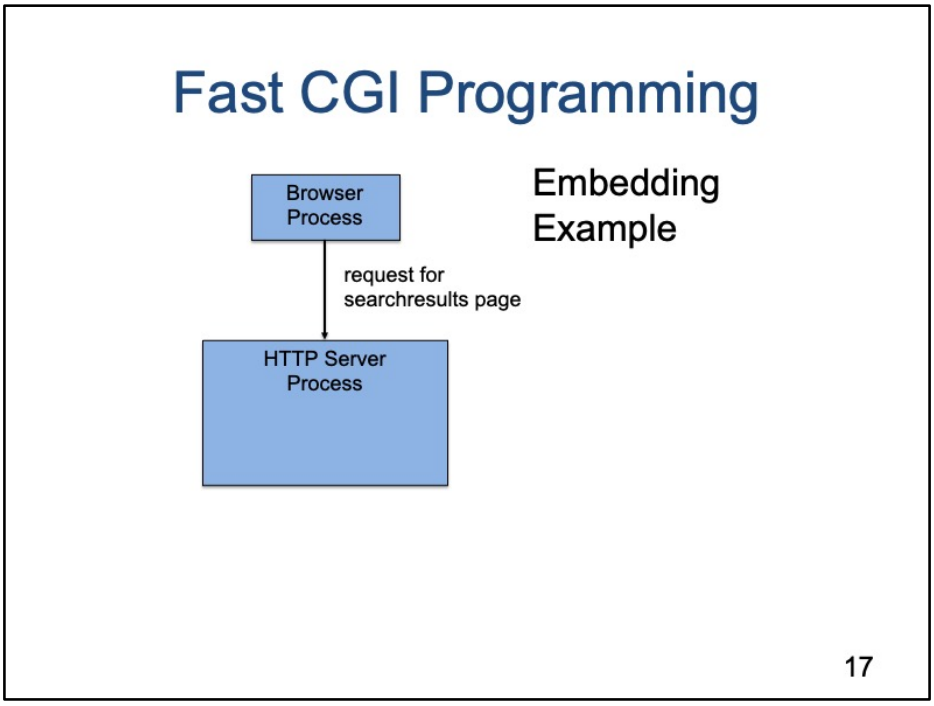
HTTP server process spawns new thread (See upcoming *Concurrency* lectures)
Thread generates searchform page



Fast CGI Programming

[see slide]

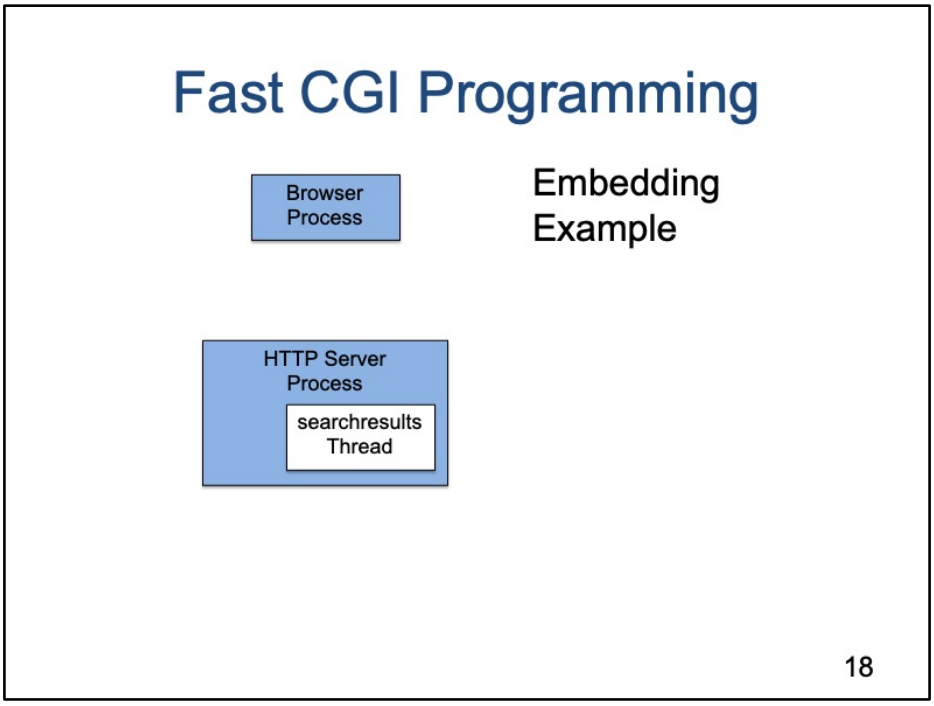
Searchform thread exits
HTTP server process sends searchform page to browser process



Fast CGI Programming

[see slide]

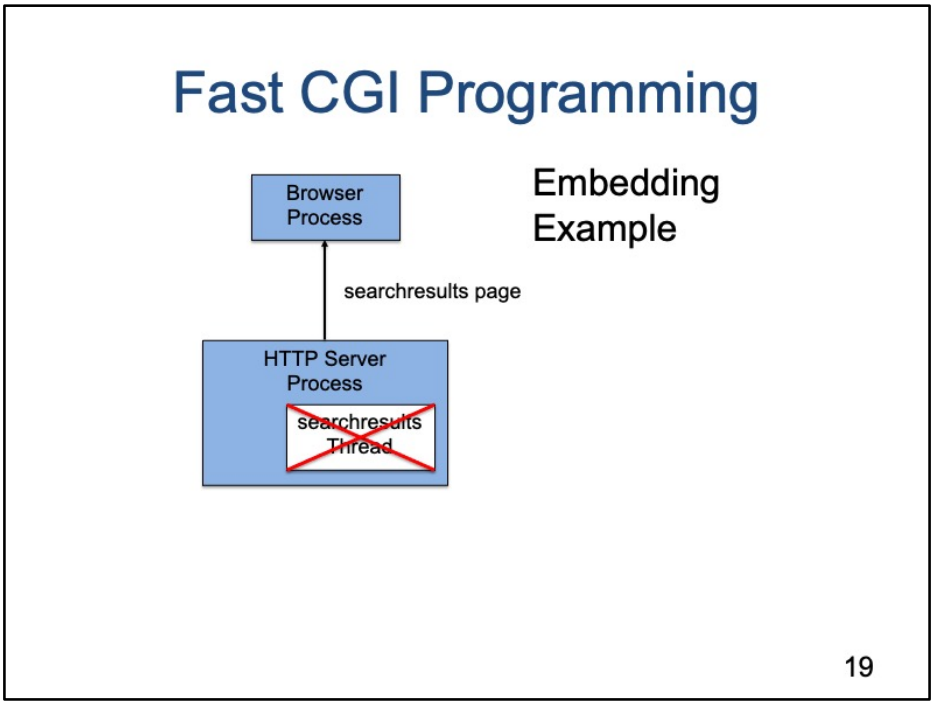
Browser process asks HTTP server process for searchresults page



Fast CGI Programming

[see slide]

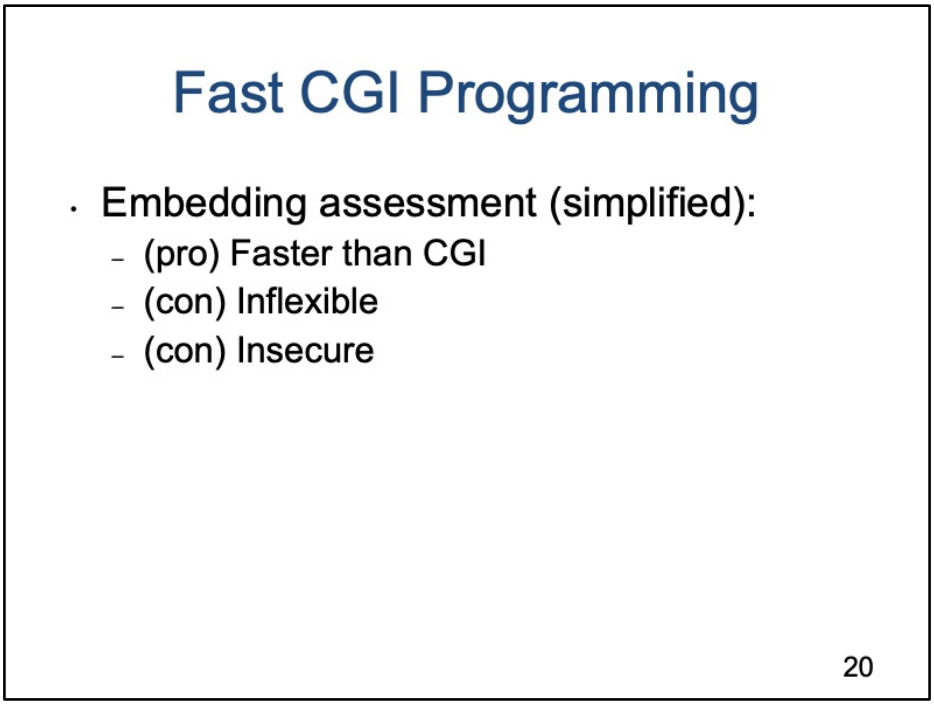
HTTP server process spawns new thread
Thread generates searchresults page



Fast CGI Programming

[see slide]

Searchresults thread exits
HTTP server process sends searchresults page to browser process



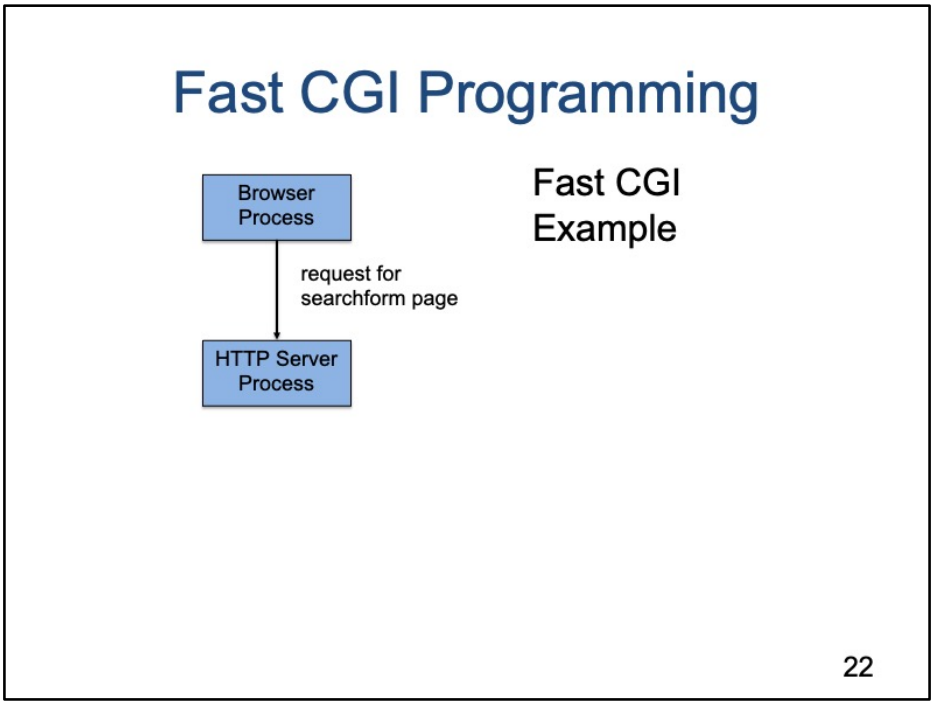
Fast CGI Programming

Embedding assessment (simplified):

(pro) Faster than CGI
Spawning a thread is faster than forking a process
(con) Inflexible
Poor load balancing; better for HTTP server to delegate work to other processes, maybe running on distinct computers
(con) Insecure
HTTP server typically runs as root
Applications run with same permissions as HTTP server

Fast CGI Programming

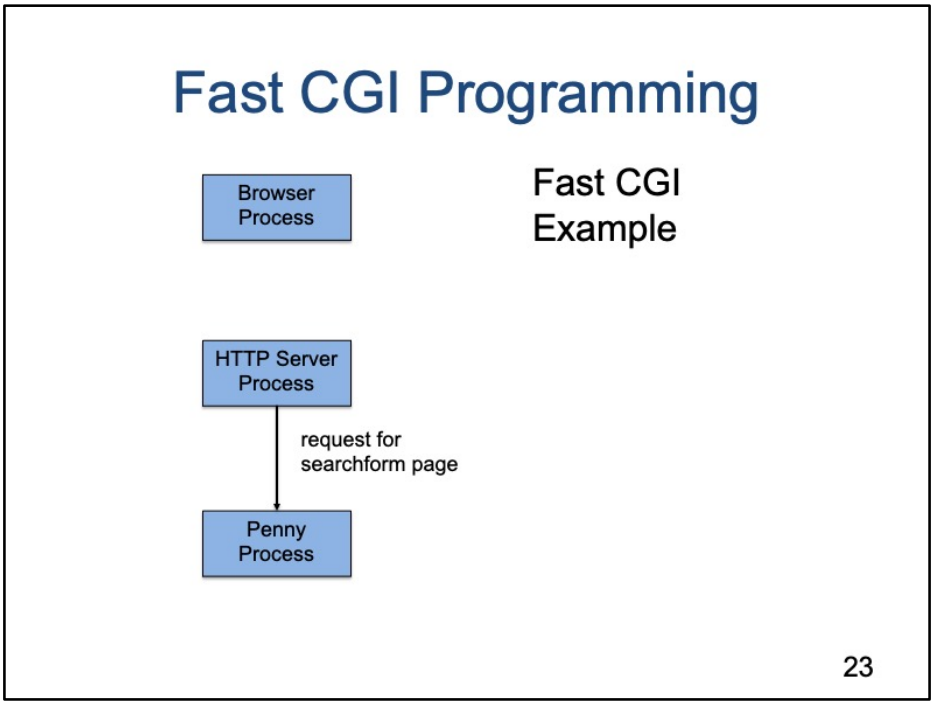
- **Solution 2**: ***Fast CGI***
  - HTTP server forks processes
    - CGI: one process for each **request**
    - Fast CGI: typically one process for each **application**
  - Processes persist
    - Unlike CGI



Fast CGI Programming

[see slide]

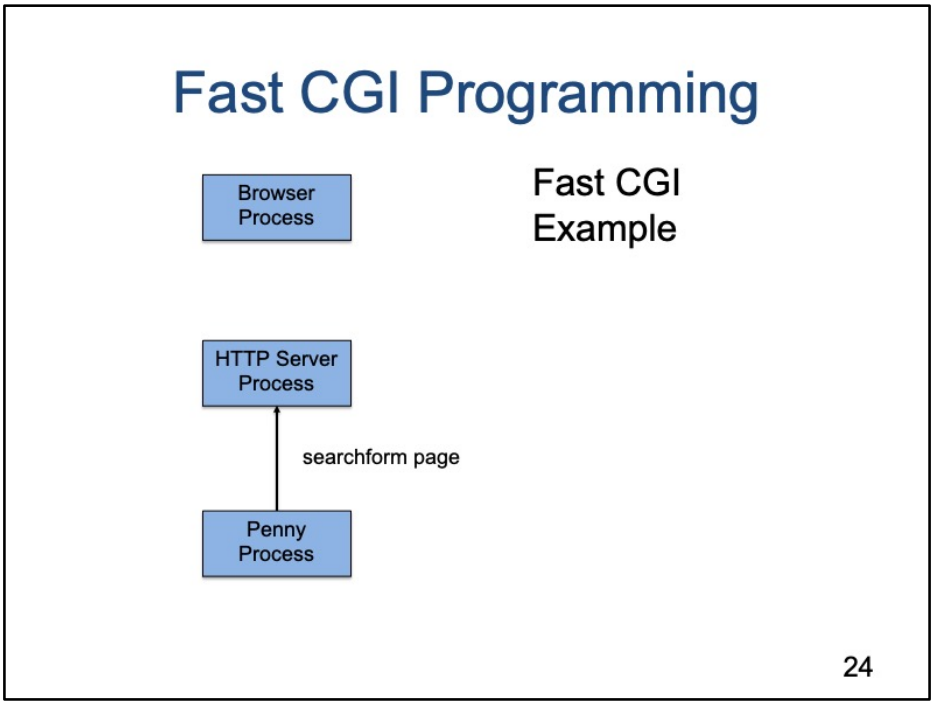
Browser process asks HTTP server process for searchform page



Fast CGI Programming

[see slide]

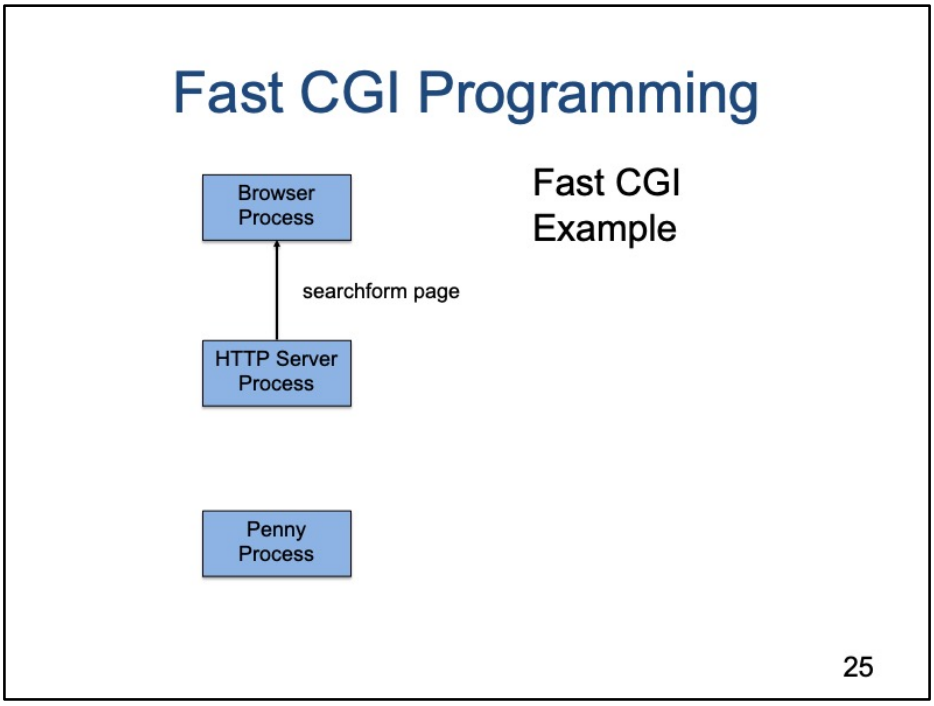
HTTP server process forks child process, execs Penny program



Fast CGI Programming

[see slide]

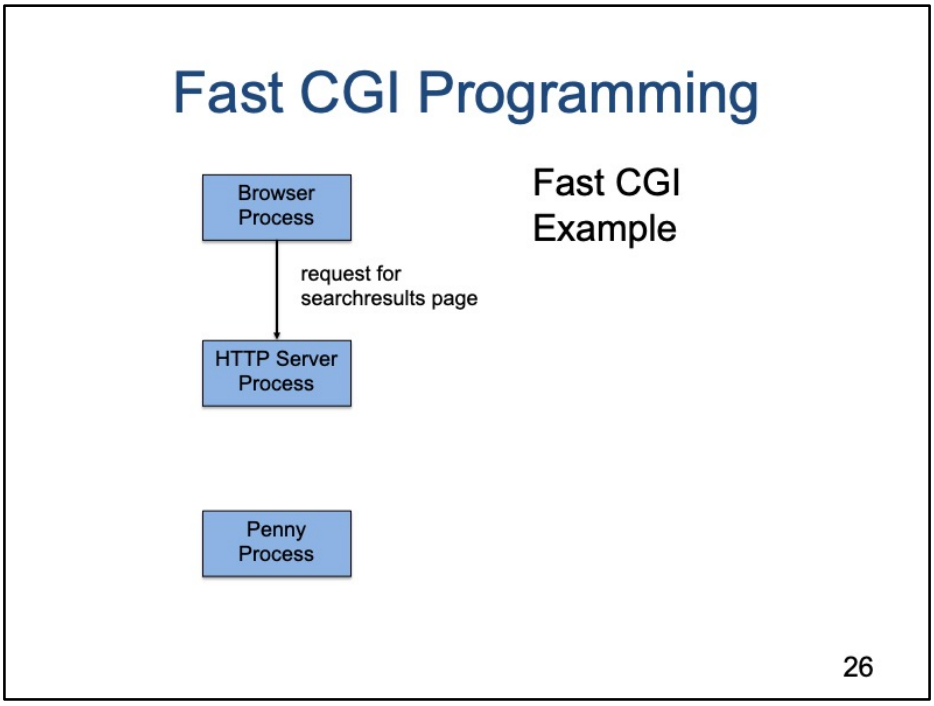
Penny process passes searchform page to HTTP server process
Penny process persists



Fast CGI Programming

[see slide]

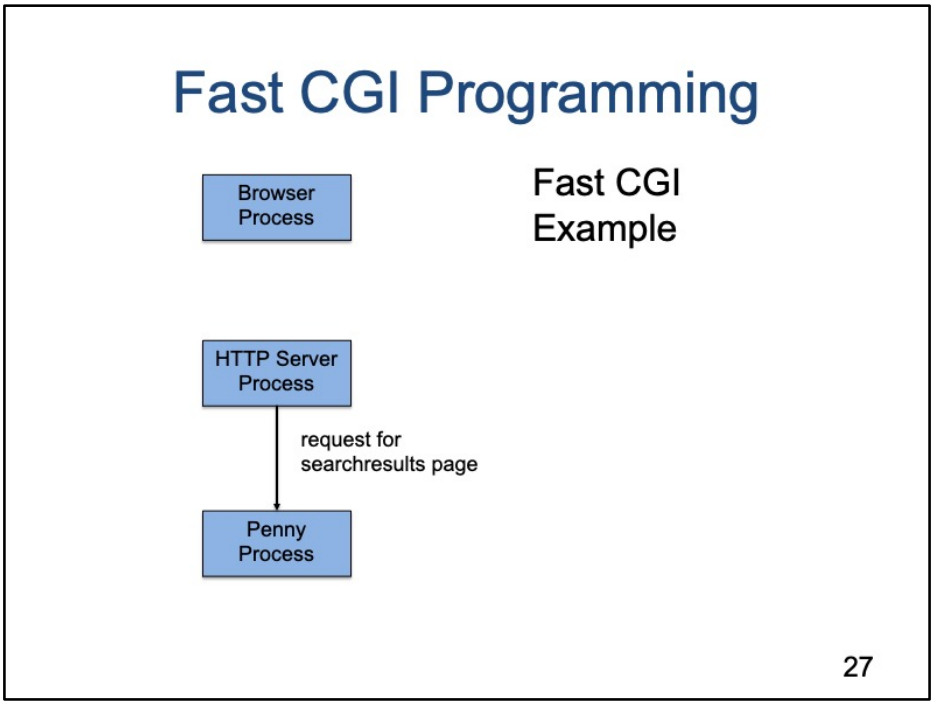
HTTP server process passes searchform page to browser



Fast CGI Programming

[see slide]

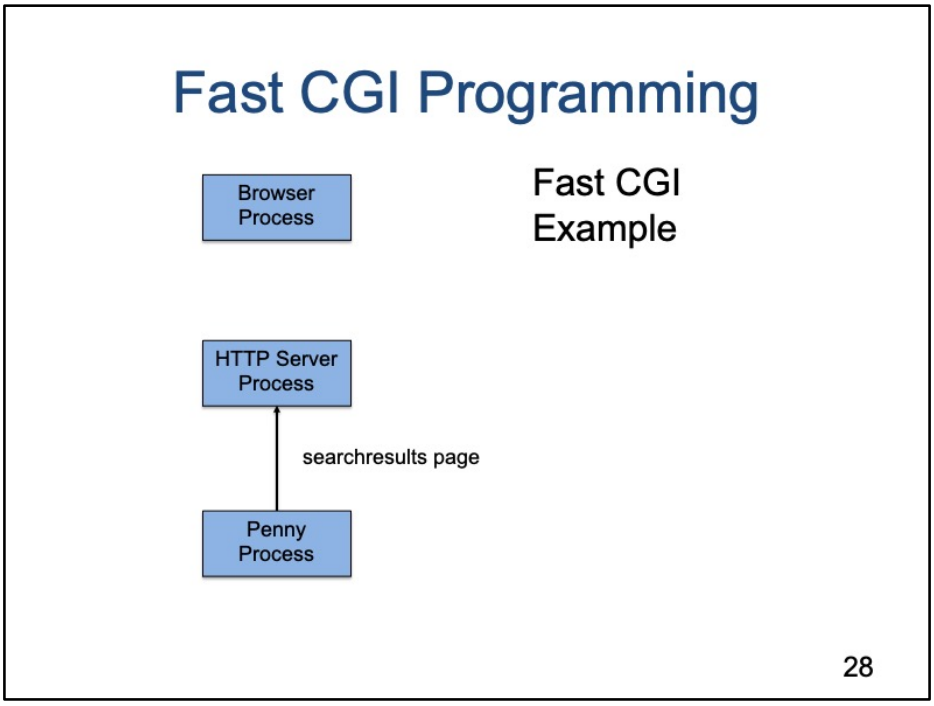
Browser process asks HTTP server process for searchresults page



Fast CGI Programming

[see slide]

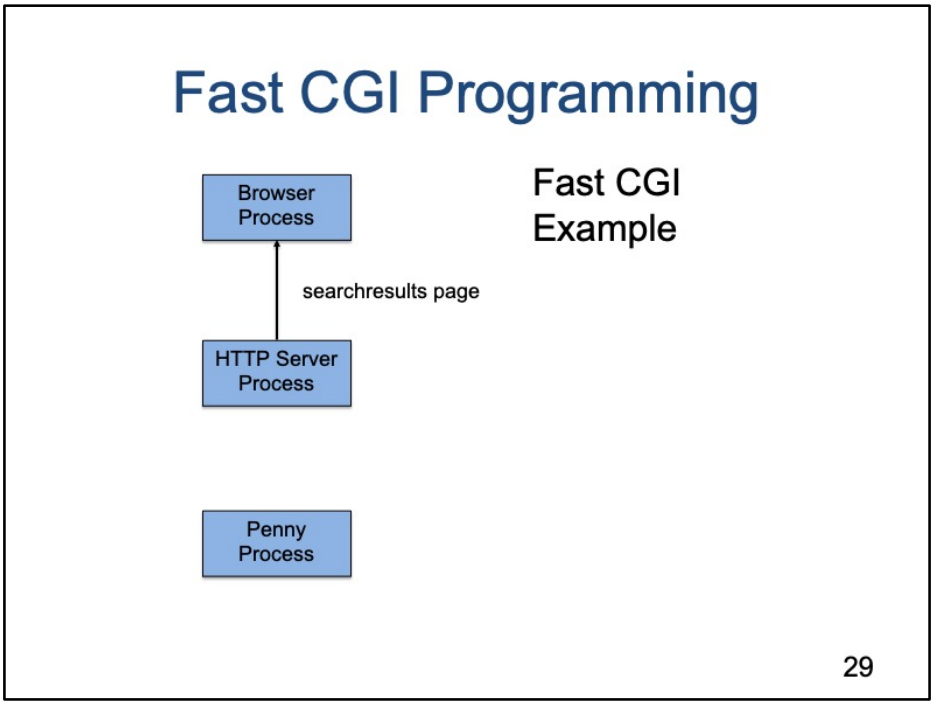
HTTP server process sends request to existing Penny process



Fast CGI Programming

[see slide]

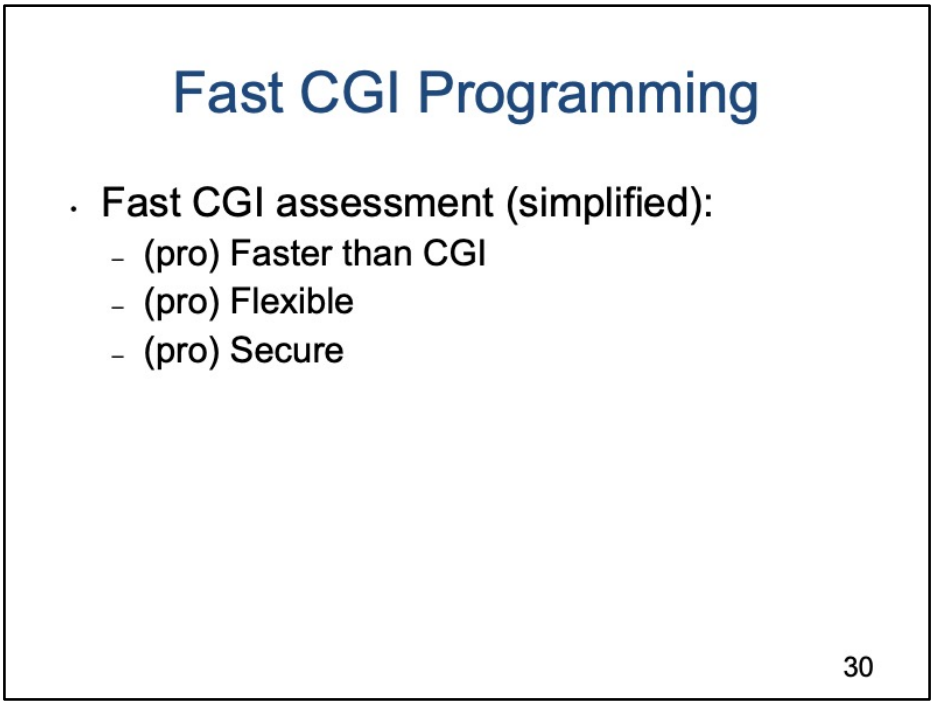
Penny process passes searchresults page to HTTP server process
Penny process persists



Fast CGI Programming

[see slide]

HTTP server process passes searchresults page to browser



Fast CGI Programming

Fast CGI assessment (simplified):

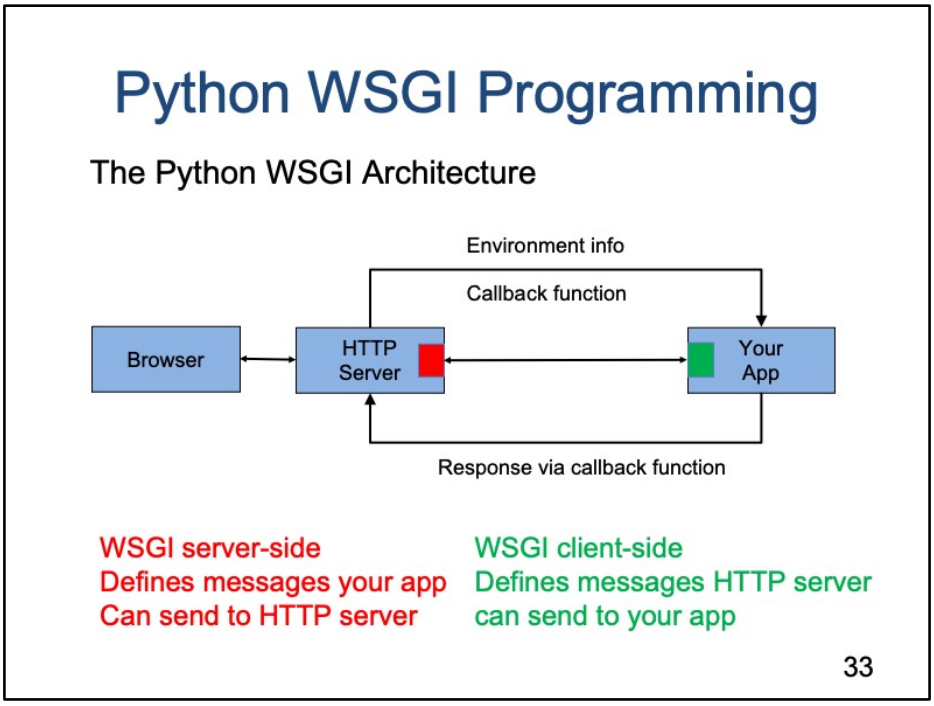
(pro) Faster than CGI
(pro) Flexible
    Good load balancing; HTTP server can delegate work to pool of app servers, maybe running on different computers
    Etc.
(pro) Secure
    Apps can run with more limited permissions than HTTP server has

Agenda

- Fast CGI programming
- **Python WSGI programming**

Python WSGI Programming

- **Python solution**:
- ***Web Server Gateway Interface (WSGI)***
  - A specification of two interfaces…



Python WSGI Programming

Informally, we can call the interfaces the “green” interface and the “red” interface

The **green** interface defines messages that the HTTP server can send to your app

The **red** interface defines messages that your app can send to the HTTP server

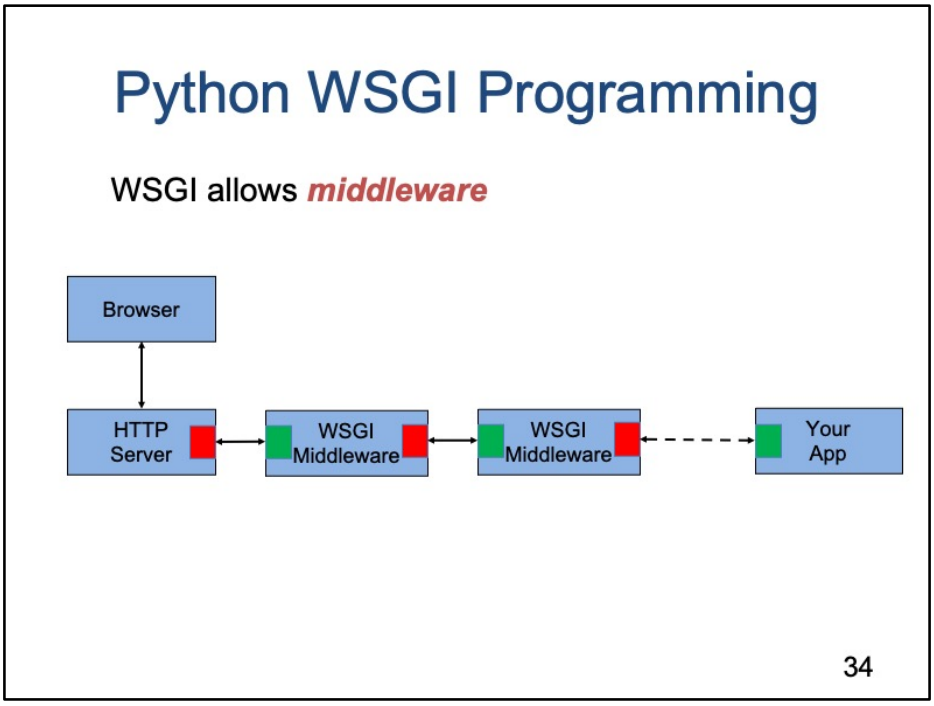
More precisely:

The HTTP server sends to your app:

Environment info (which includes the HTTP request), and

A callback function

Your app then calls the callback function to send its HTTP response to the HTTP server



Python WSGI Programming

[see slide]

Informally, red must be connected to green

Informally, a module that implements both the red interface and the green interface can serve as **middleware**

Such a module can be spliced between the HTTP server and your app
Such a module "decorates" your app with additional functionality

Middleware modules exist that provide:
- App persistence (as with fast CGI)
- Load balancing among multiple app processes
  - Maybe on multiple computers
- Authentication…

Provides lots of flexibility
The way to structure Python Web apps

Python WSGI Programming

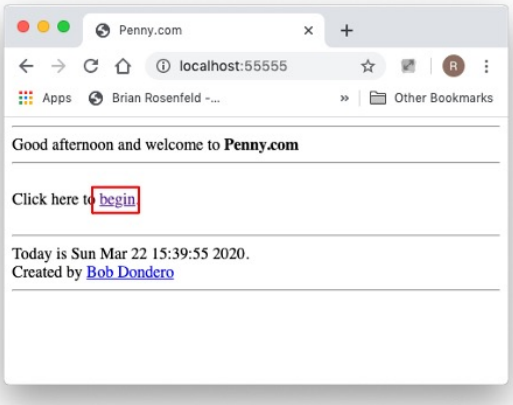
- See **PennyWsgi** app

```
$ python runserver.py 55555
Listening on port 55555
```

Runs test WSGI server that's bundled with Python

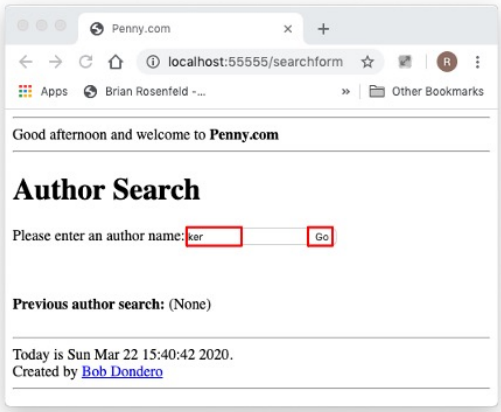
Python WSGI Programming

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



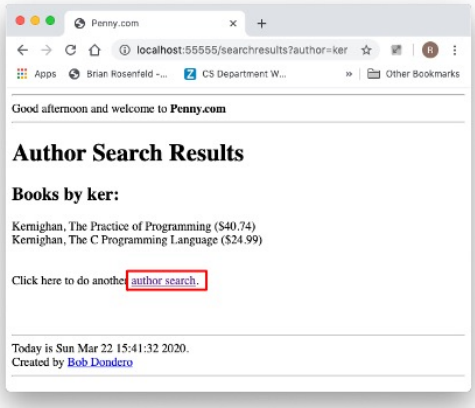
Python WSGI Programming

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



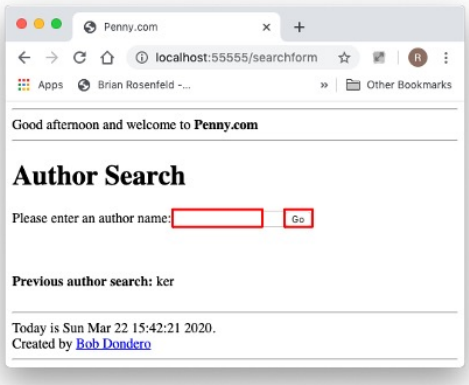
Python WSGI Programming

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



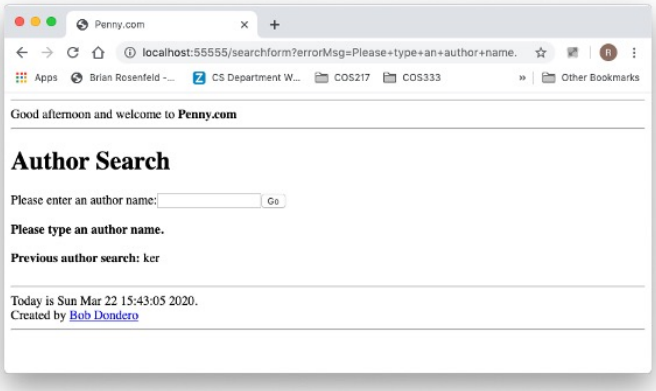
Python WSGI Programming

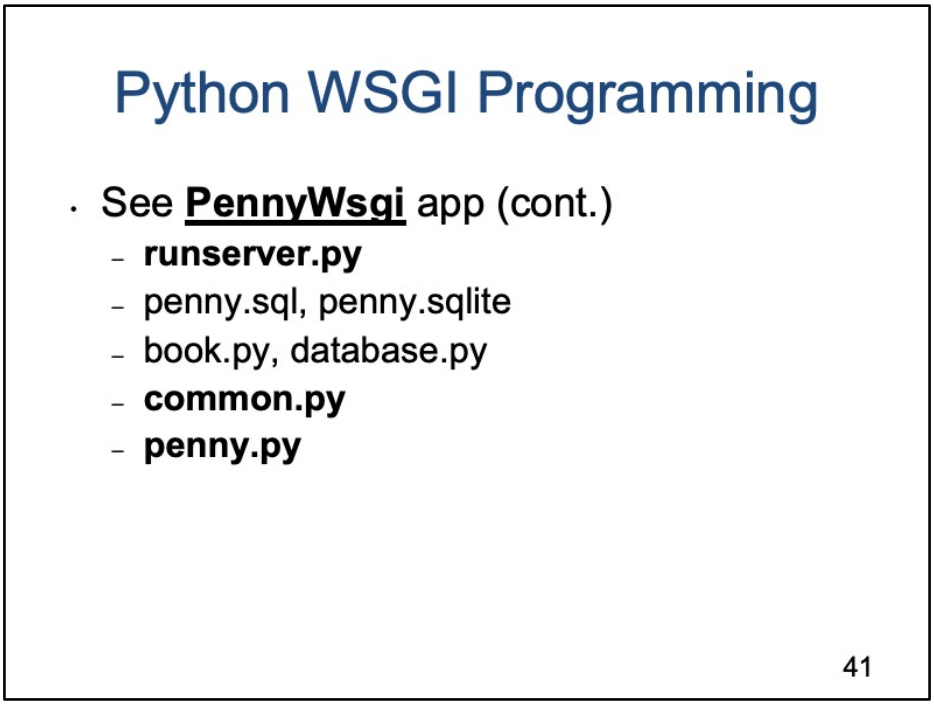
- See **PennyWsgi** app (cont.)



Python WSGI Programming

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





Python WSGI Programming

[see slide]

Code notes: runserver.py

Runs the Python standard HTTP/WSGI test server

The IP address 0.0.0.0 means “all IP addresses on the local computer”

Code notes: common.py

Defines utility functions, each of which returns (does not print) a string

Code notes: penny.py

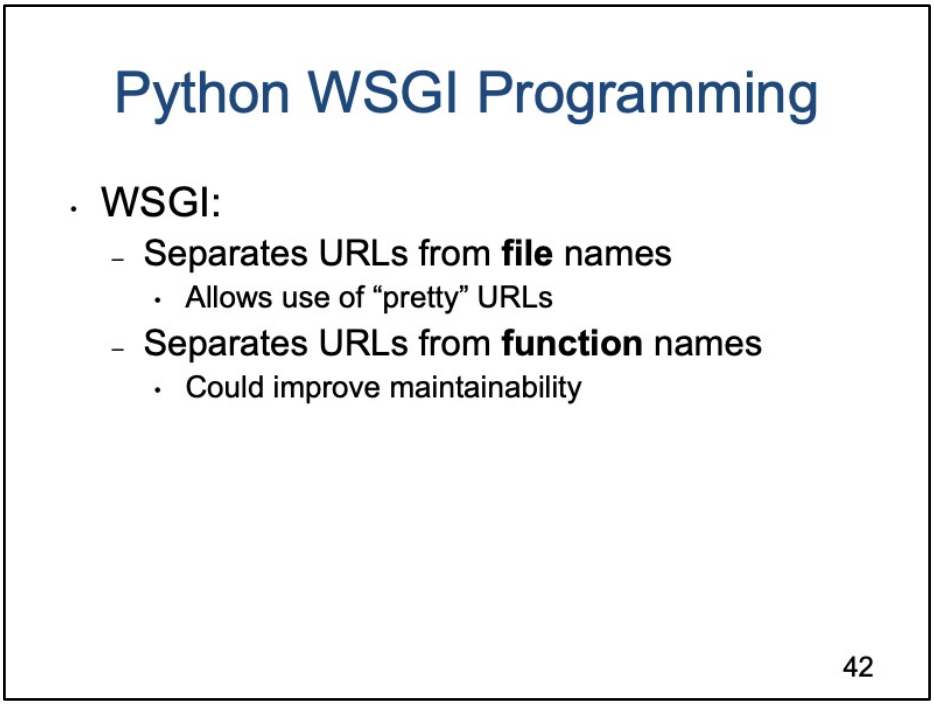
The environ parameter to app() contains environment info

The HTTP request, name=value pairs, cookies, etc.

The start_response parameter to app() is a function pointer

Page-generating functions call it to send HTTP response headers to the HTTP server

Each page-generating function returns a list containing an HTML document expressed as a bytes object



Python WSGI Programming

WSGI separates URLs from **file** names

CGI: File names (searchform.py, searchresults.py) are in URLs
WSGI: URLs contain arbitrary strings (searchform, searchresults), which the program maps to functions
Allows you to use "pretty" URLs

WSGI separates URLs from **function** names

Can map multiple URLs to same function
Example: This application maps / and /index to the same index() function
Can change function name without changing URL, or vice versa
Could improve maintainability

Summary

- We have covered:
  - Fast CGI programming
  - Python WSGI programming