61A Extra Lecture 7

Announcements

Prefix Trees

All Words That Share a Prefix

Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

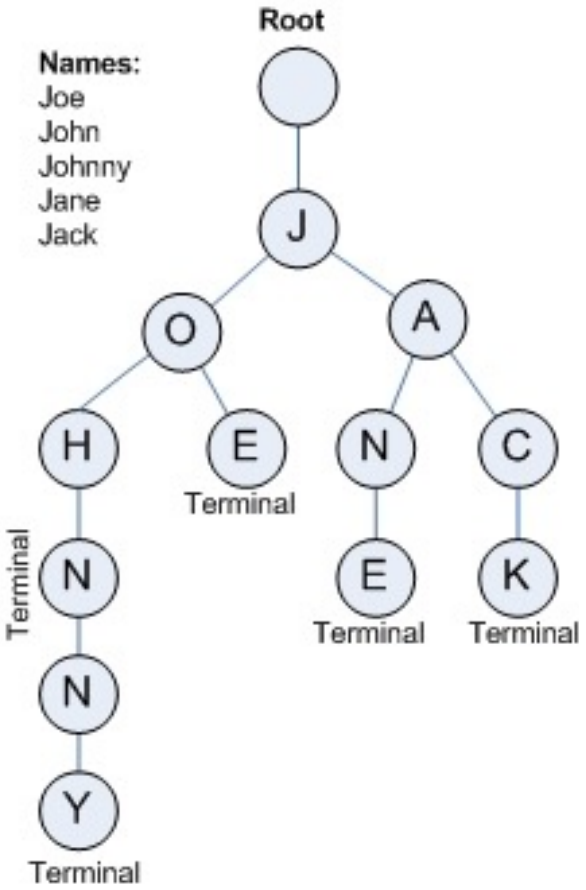


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

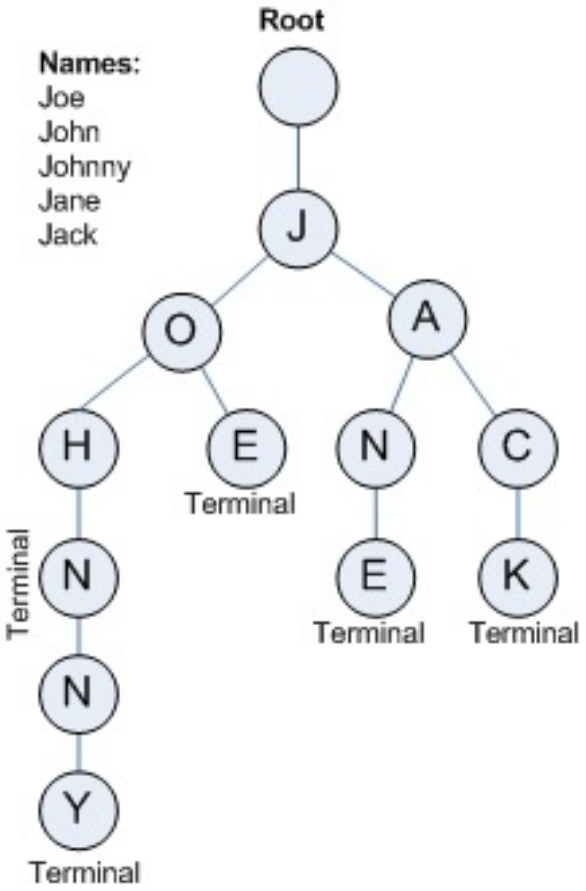


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

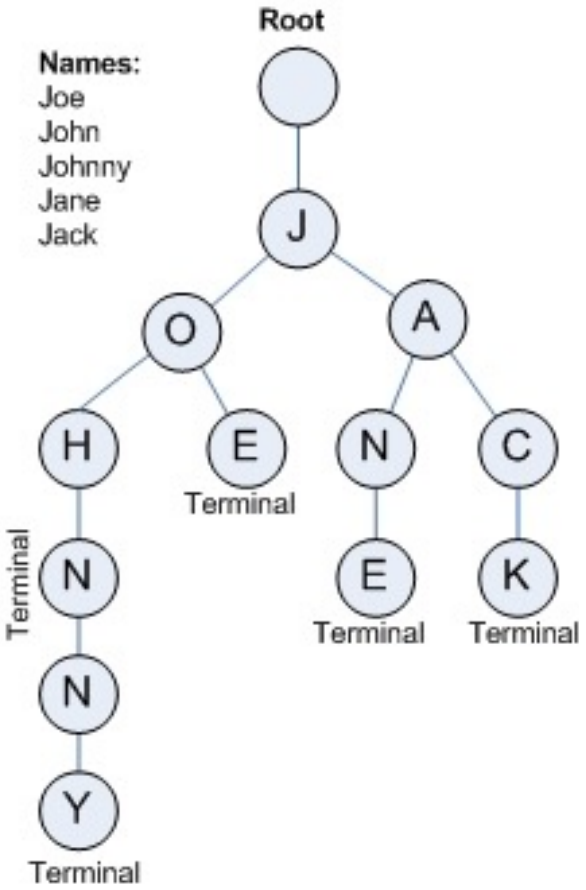


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

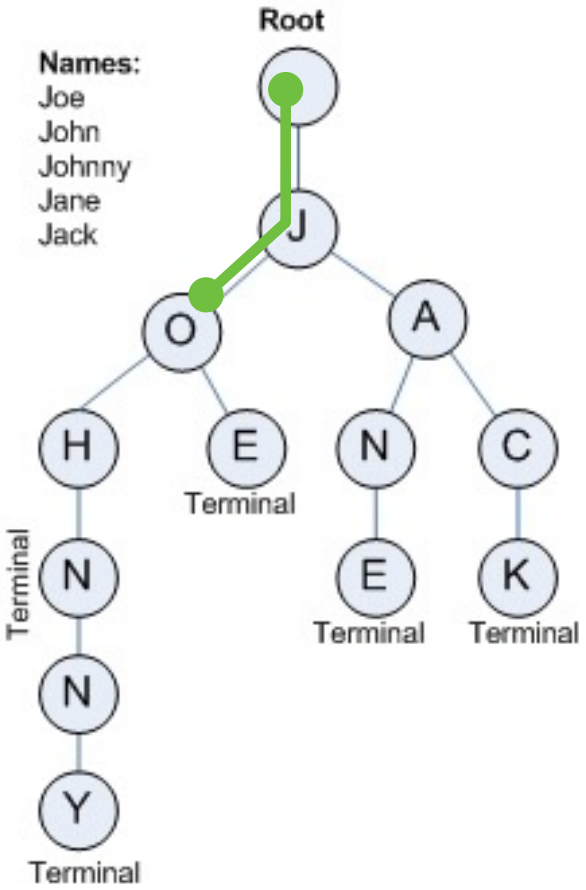


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

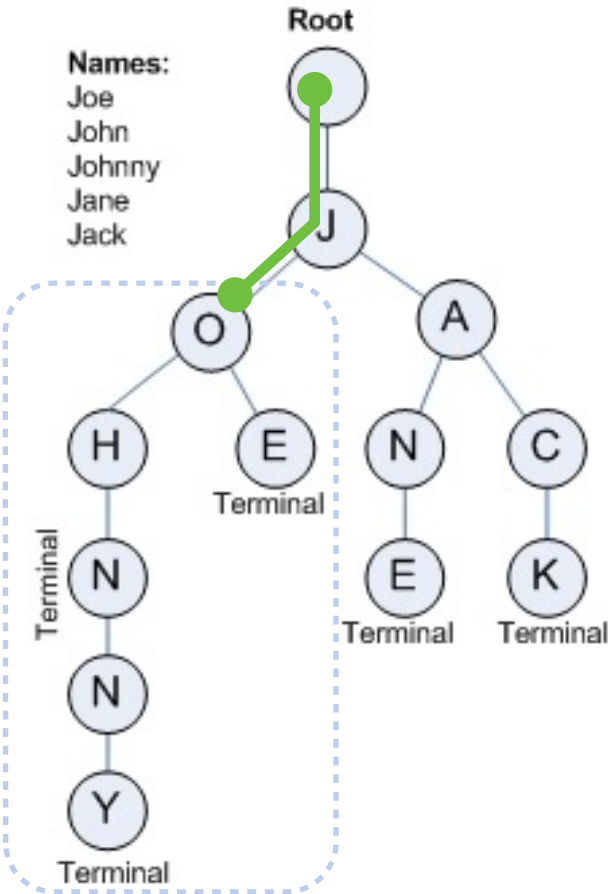


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

add: Follow a path from the root using a word, adding branches for each new letter until the end is reached

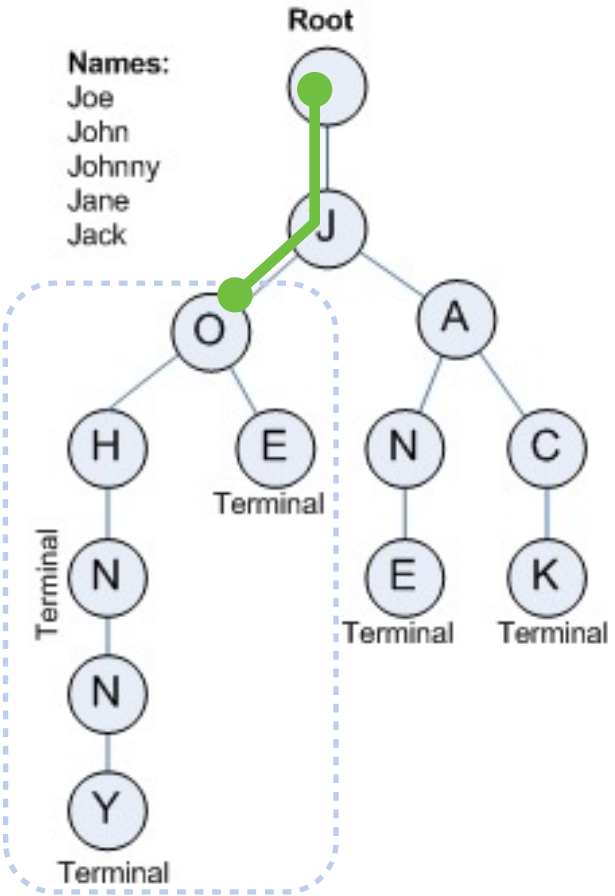


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

add: Follow a path from the root using a word, adding branches for each new letter until the end is reached

Example: "JANET"

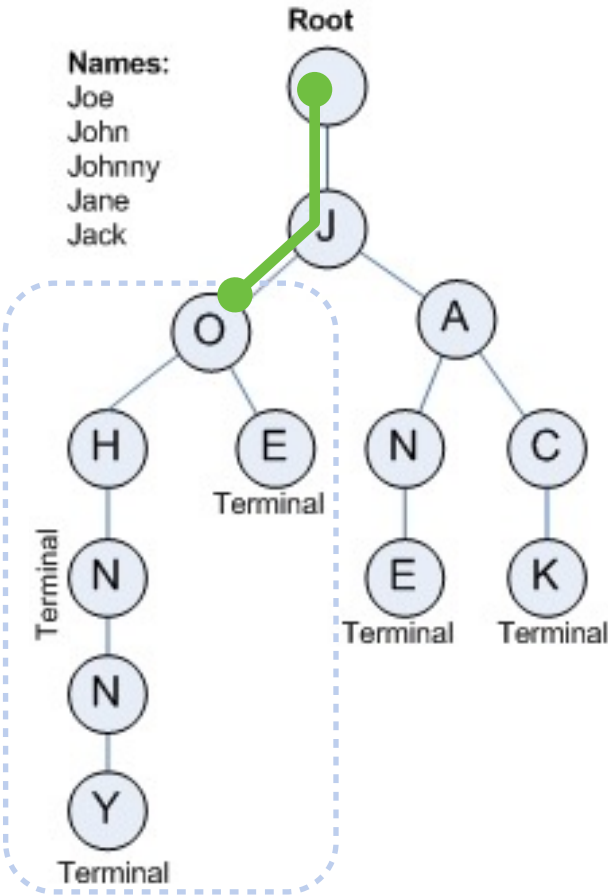


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

add: Follow a path from the root using a word, adding branches for each new letter until the end is reached

Example: "JANET"

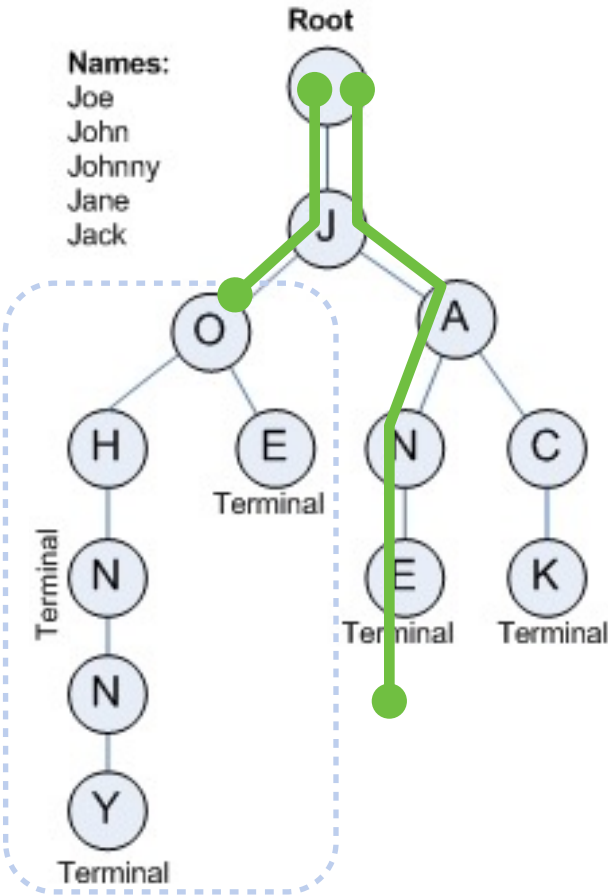


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

add: Follow a path from the root using a word, adding branches for each new letter until the end is reached

Example: "JANET"

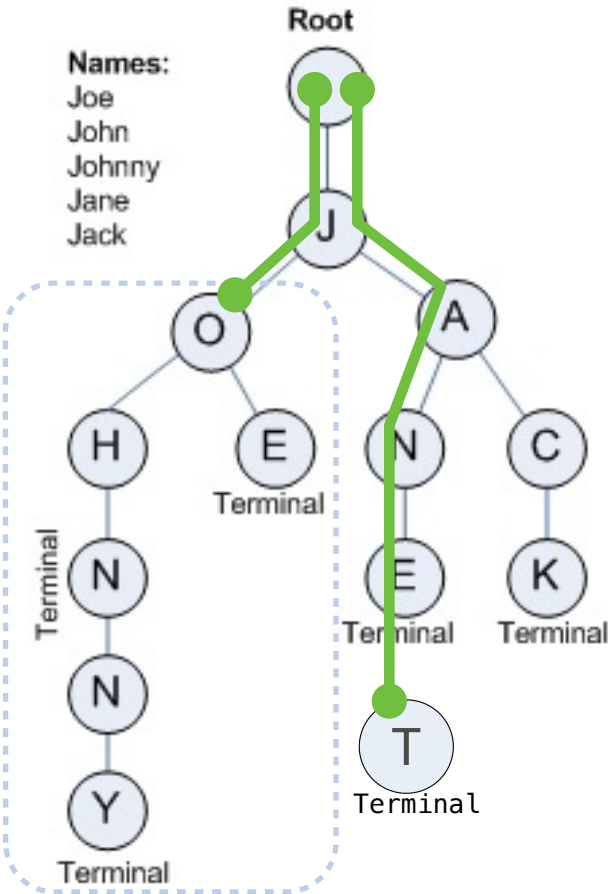


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

All Words That Share a Prefix

A prefix tree (or just "trie") indexes words by prefix

lookup: Follow a path from the root using a prefix, then enumerate everything below the resulting node

Example: "JO"

add: Follow a path from the root using a word, adding branches for each new letter until the end is reached

Example: "JANET"

(Demo)

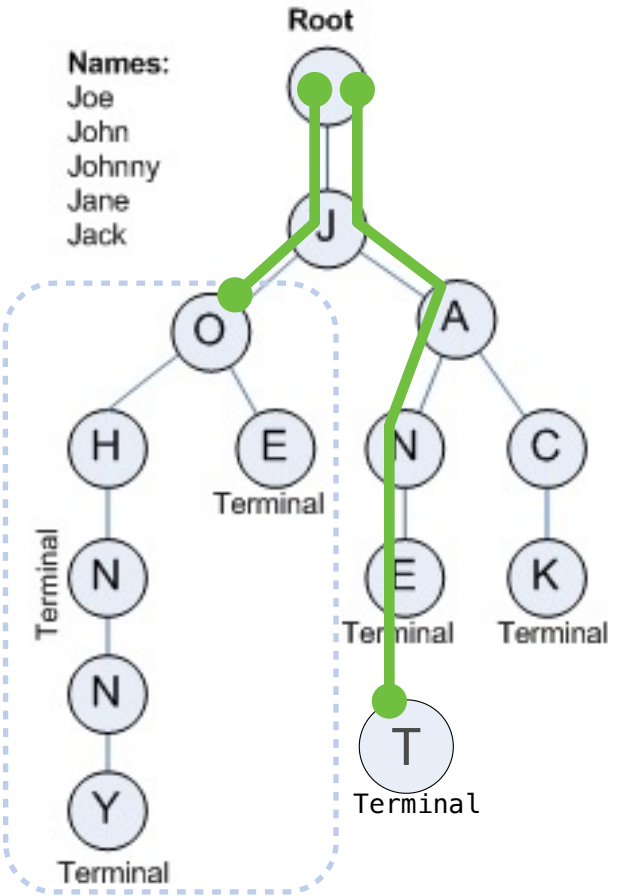


Image: http://www.codeproject.com/Articles/18033/Phone-Directory-Implementation-Using-TRIE

Flask

The Flask Web Framework

The Flask Web Framework

Translates HTTP requests (described in a future lecture) to Python function calls

The Flask Web Framework

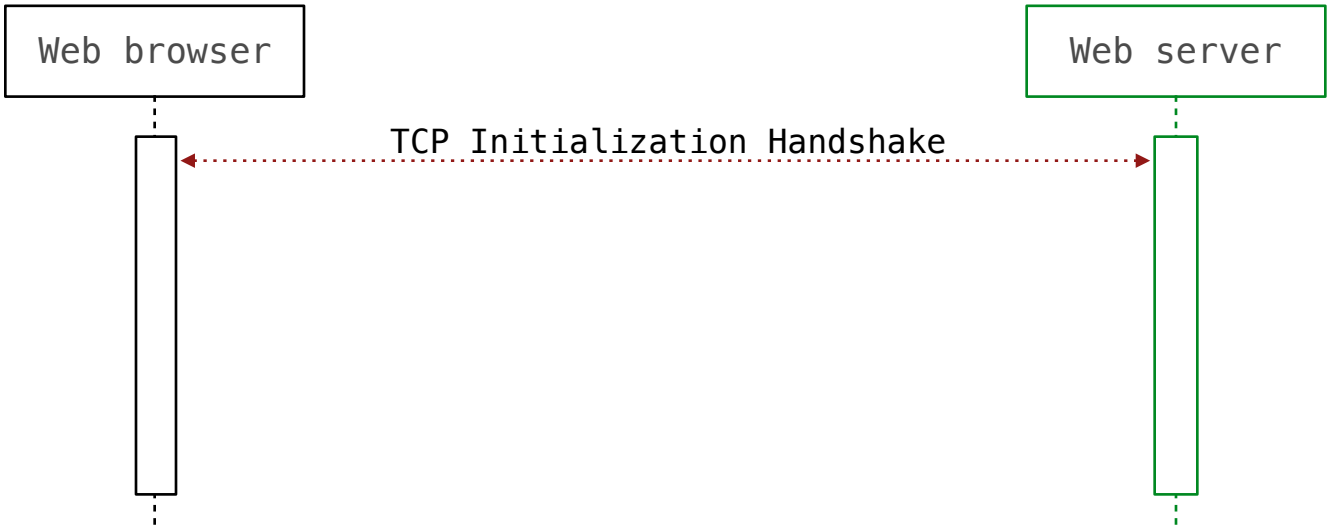
Translates HTTP requests (described in a future lecture) to Python function calls

Manages data exchange between a browser and a Python program

The Flask Web Framework

Translates HTTP requests (described in a future lecture) to Python function calls

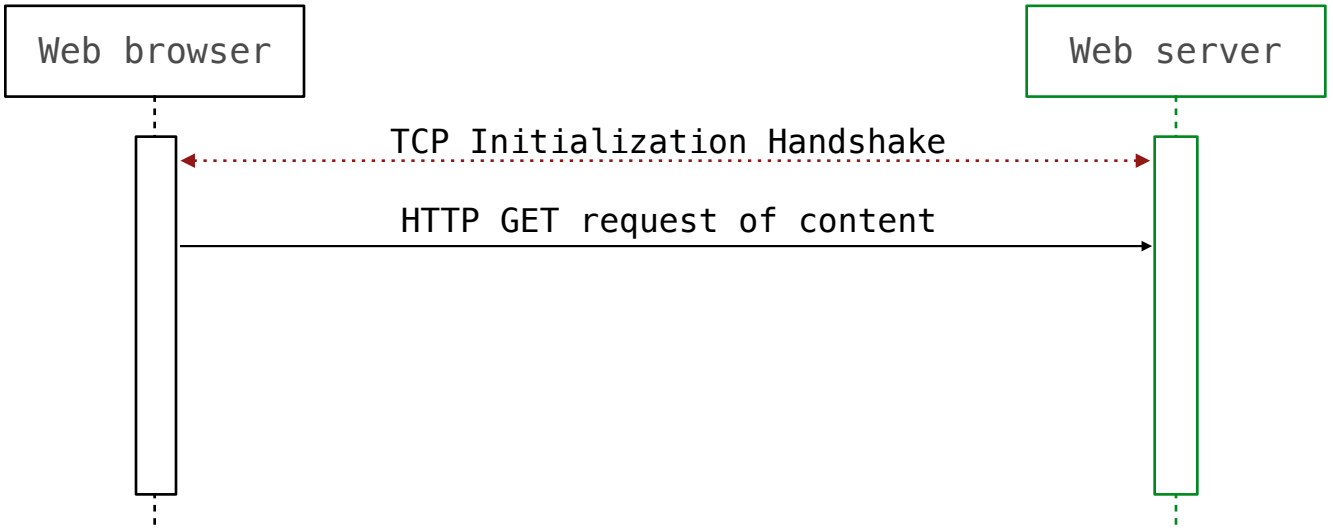
Manages data exchange between a browser and a Python program



The Flask Web Framework

Translates HTTP requests (described in a future lecture) to Python function calls

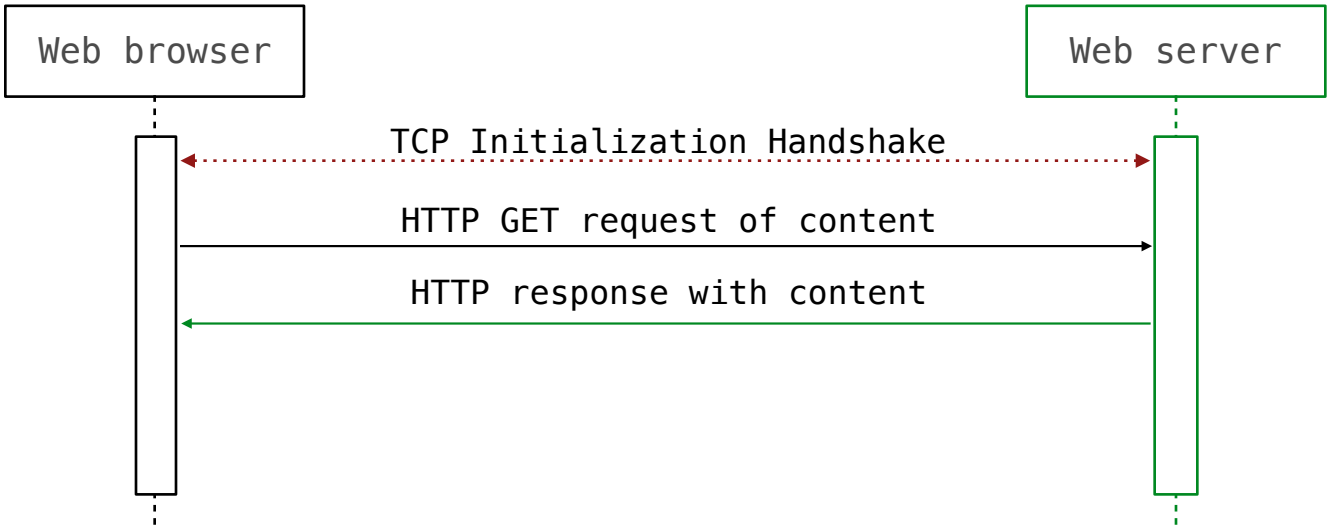
Manages data exchange between a browser and a Python program



The Flask Web Framework

Translates HTTP requests (described in a future lecture) to Python function calls

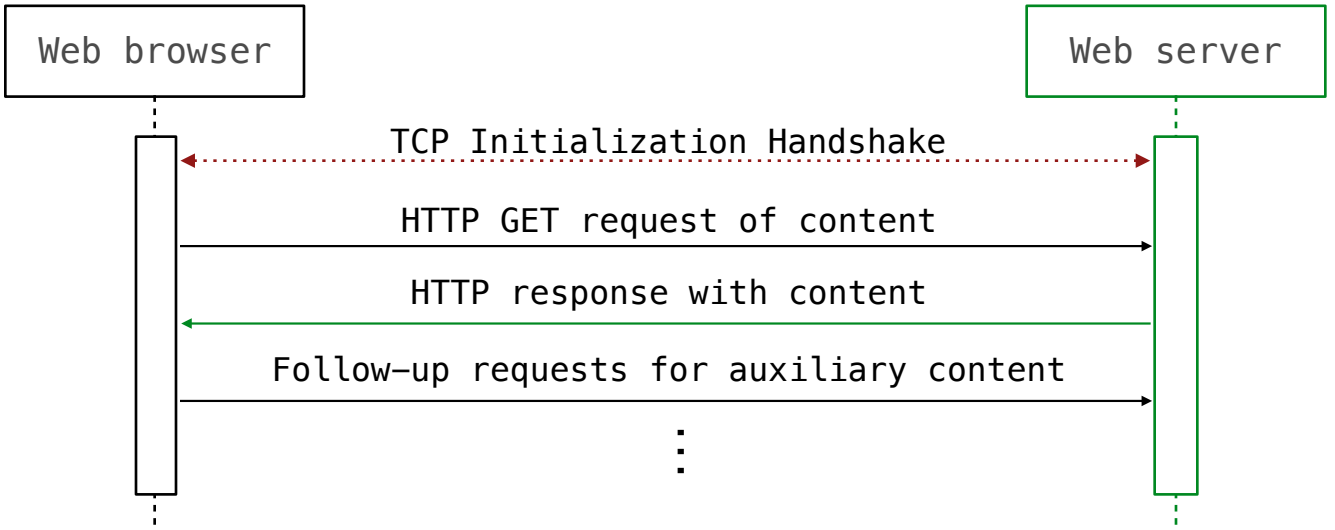
Manages data exchange between a browser and a Python program



The Flask Web Framework

Translates HTTP requests (described in a future lecture) to Python function calls

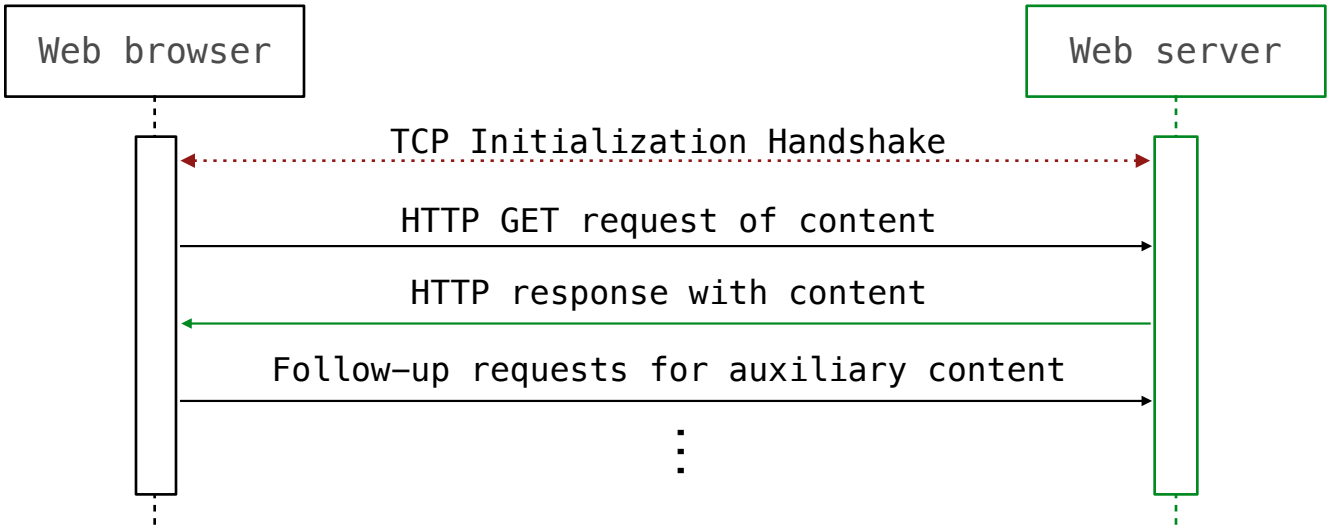
Manages data exchange between a browser and a Python program



The Flask Web Framework

Translates HTTP requests (described in a future lecture) to Python function calls

Manages data exchange between a browser and a Python program



(Demo)

Threads

Threads

Threads

A thread executes a function call

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

For high-latency operations such as web requests, threading can increase speed enormously

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

For high-latency operations such as web requests, threading can increase speed enormously

Thread(target=<function>, args=<args>): Create (but do not start) a thread of execution

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

For high-latency operations such as web requests, threading can increase speed enormously

Thread(target=<function>, args=<args>): Create (but do not start) a thread of execution

.start(): Start the function call, but do not wait for it to complete

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

For high-latency operations such as web requests, threading can increase speed enormously

Thread(target=<function>, args=<args>): Create (but do not start) a thread of execution

.start(): Start the function call, but do not wait for it to complete

.join(): Wait for the function call to complete (return value is ignored)

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

For high-latency operations such as web requests, threading can increase speed enormously

Thread(target=<function>, args=<args>): Create (but do not start) a thread of execution

.start(): Start the function call, but do not wait for it to complete

.join(): Wait for the function call to complete (return value is ignored)

.run(): Start the function call and wait for it to complete

Threads

A thread executes a function call

Multiple threads can execute different calls simultaneously

For high-latency operations such as web requests, threading can increase speed enormously

Thread(target=<function>, args=<args>): Create (but do not start) a thread of execution

.start(): Start the function call, but do not wait for it to complete

.join(): Wait for the function call to complete (return value is ignored)

.run(): Start the function call and wait for it to complete

(Demo)

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

x = 10

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```

```
do_something()
y = x
do_something()
x = y * 2
```

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```

```
do_something()
y = x
do_something()
x = y * 2
```

```
do_something()
z = x
do_something()
x = z + 1
```

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```

```
x: 10
```

```
do_something()
y = x
do_something()
x = y * 2
```

```
do_something()
z = x
do_something()
x = z + 1
```

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```

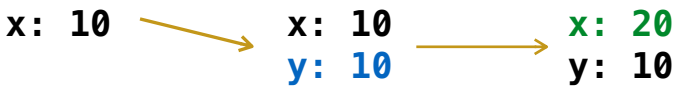

```
do_something()
y = x
do_something()
x = y * 2
```

```
do_something()
z = x
do_something()
x = z + 1
```

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```



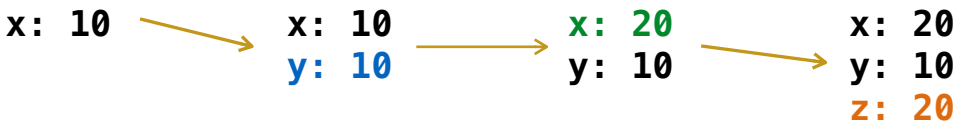

```
do_something()
y = x
do_something()
x = y * 2
```

```
do_something()
z = x
do_something()
x = z + 1
```

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```



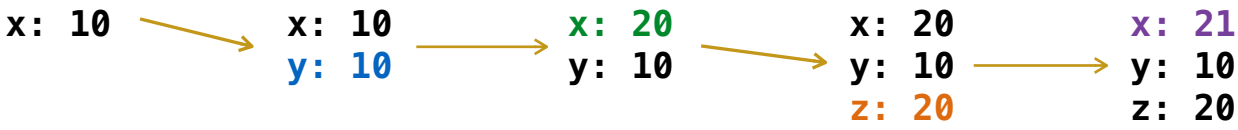

```
do_something()
y = x
do_something()
x = y * 2
```

```
do_something()
z = x
do_something()
x = z + 1
```

Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10
```




```
do_something()
y = x
do_something()
x = y * 2
```

```
do_something()
z = x
do_something()
x = z + 1
```

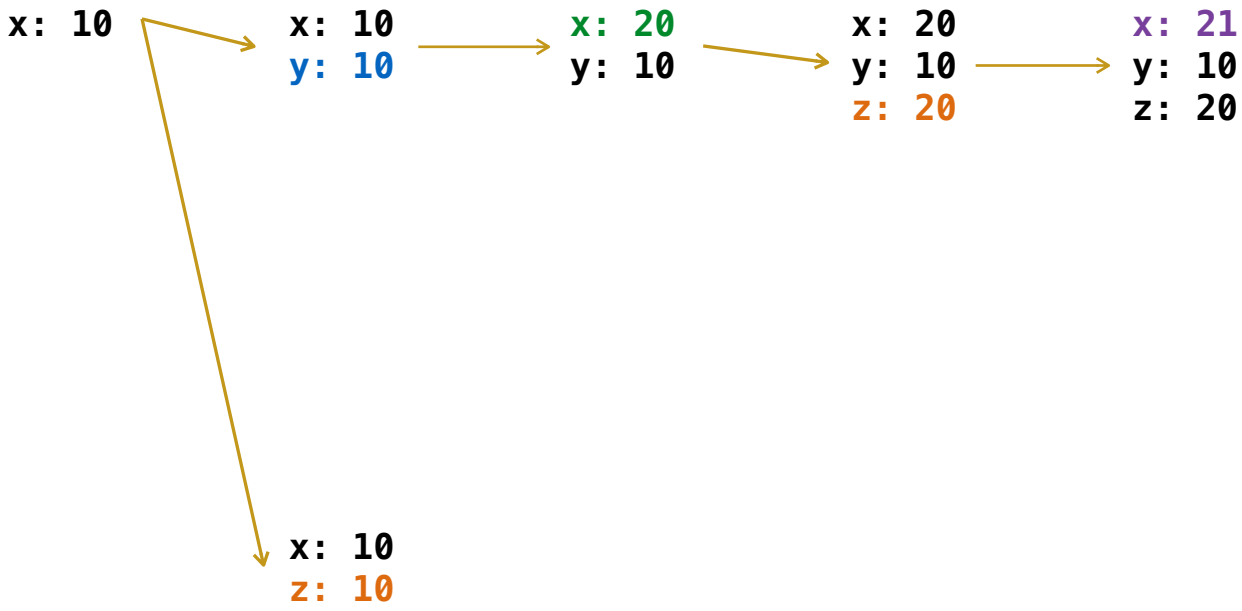
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



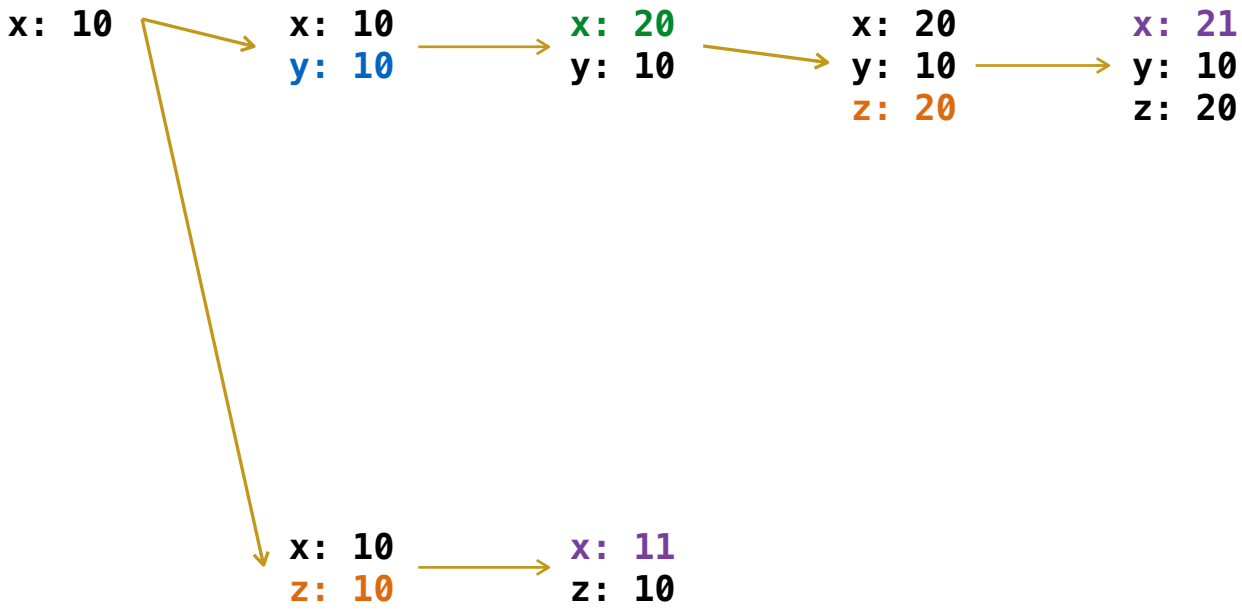
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



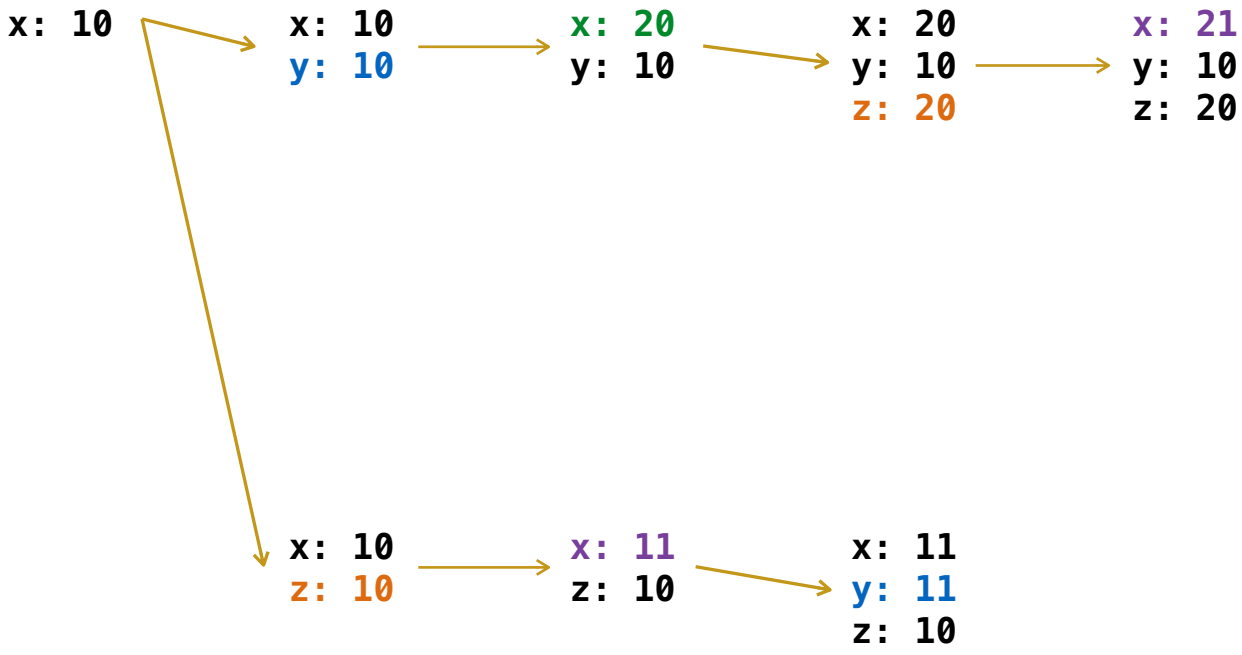
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



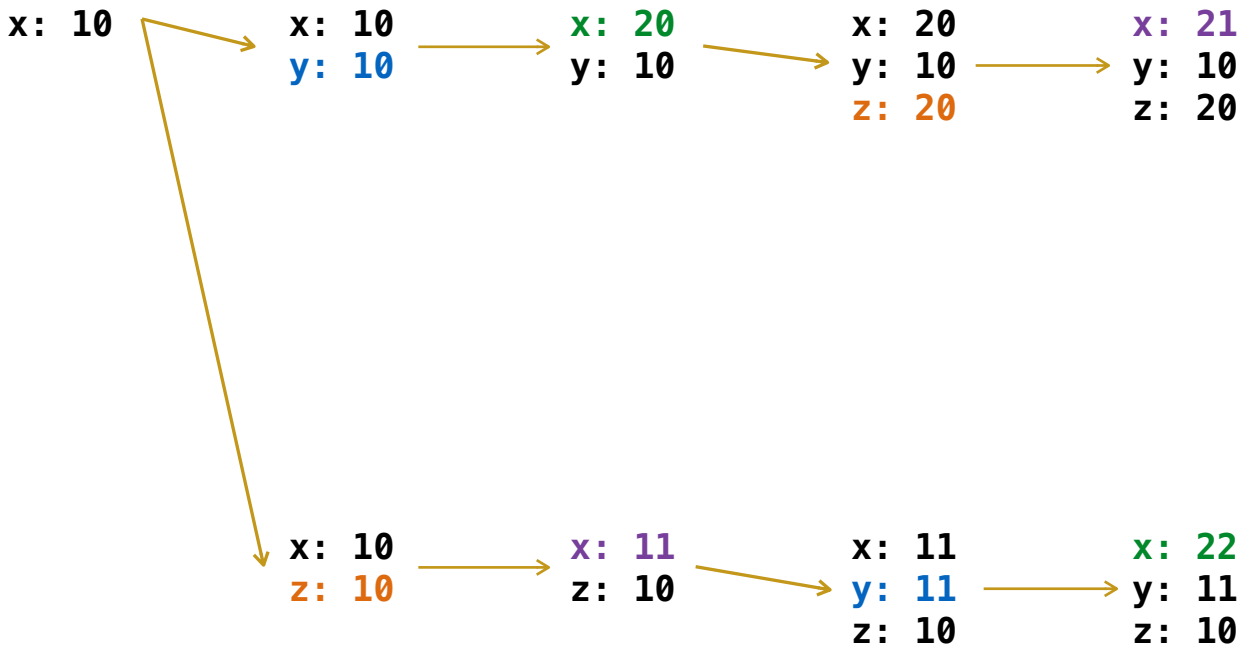
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



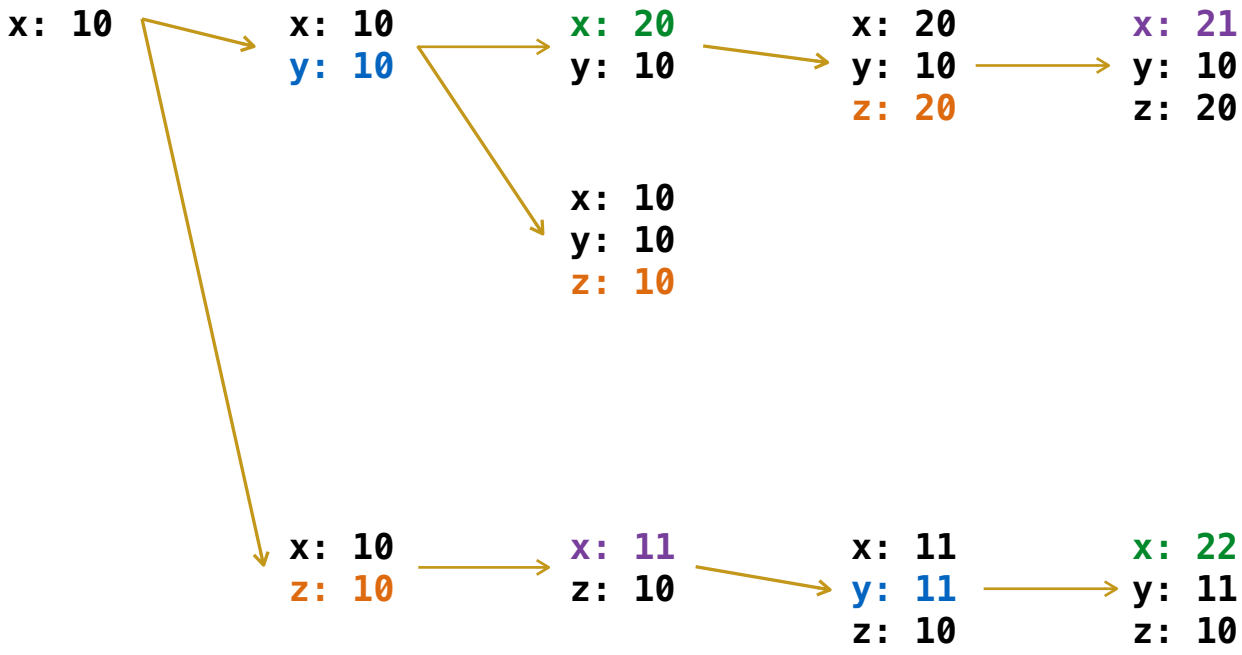
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



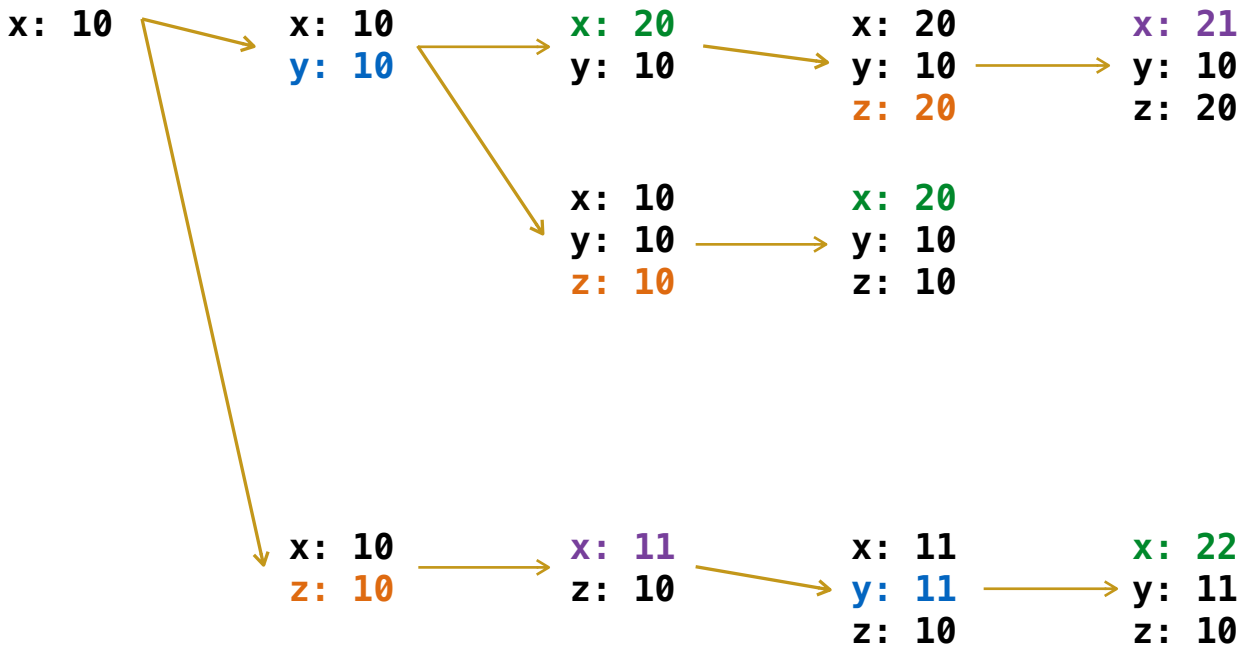
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



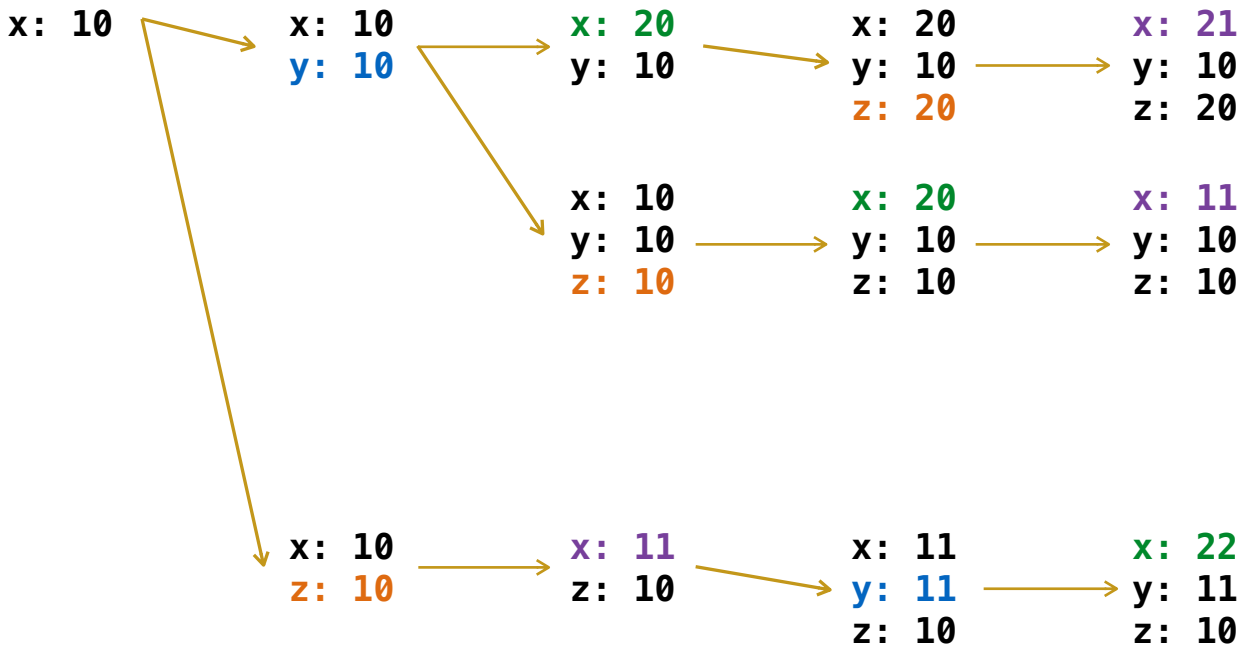
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



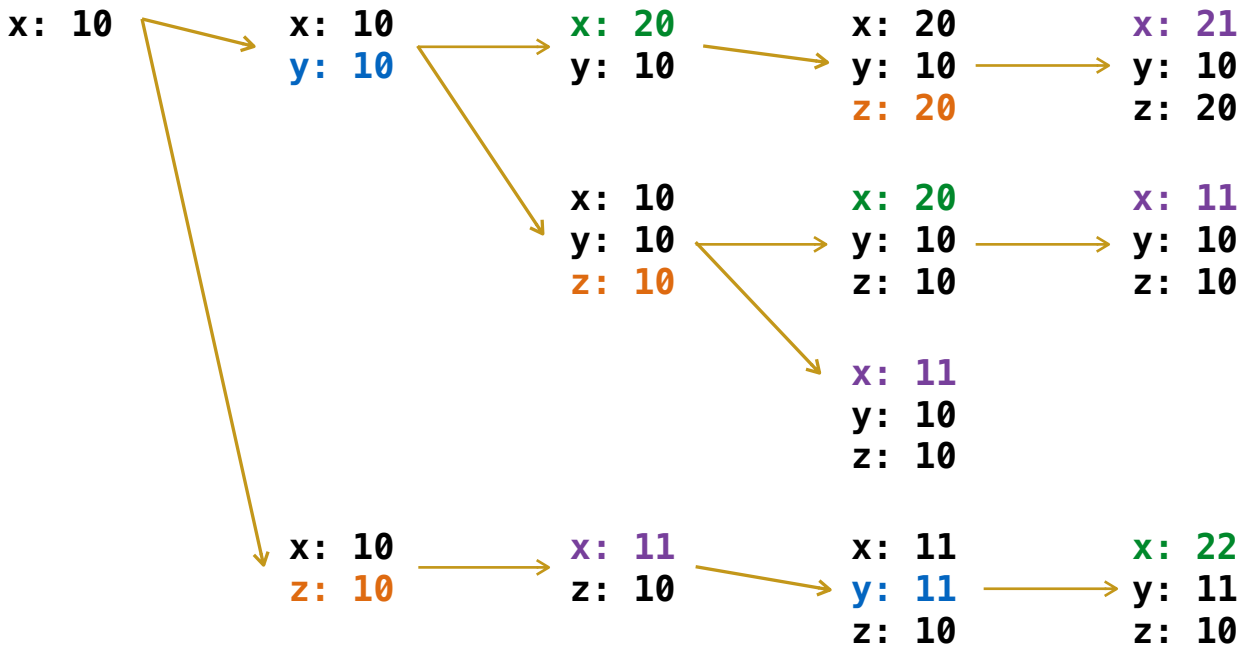
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



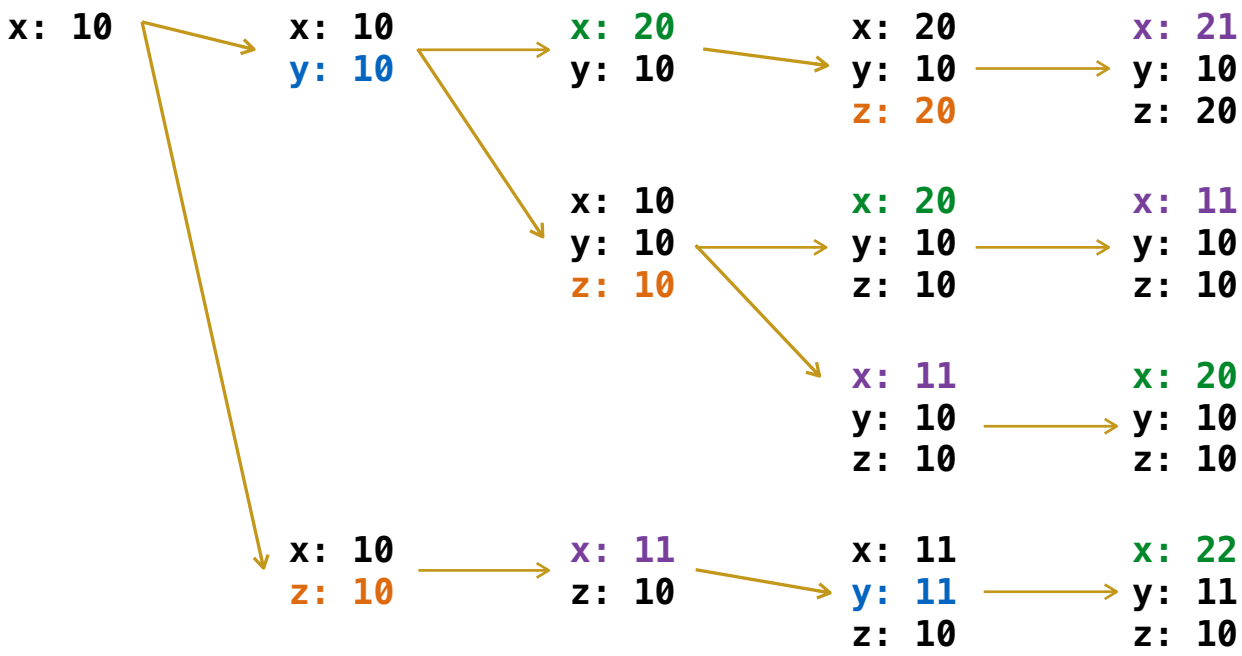
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



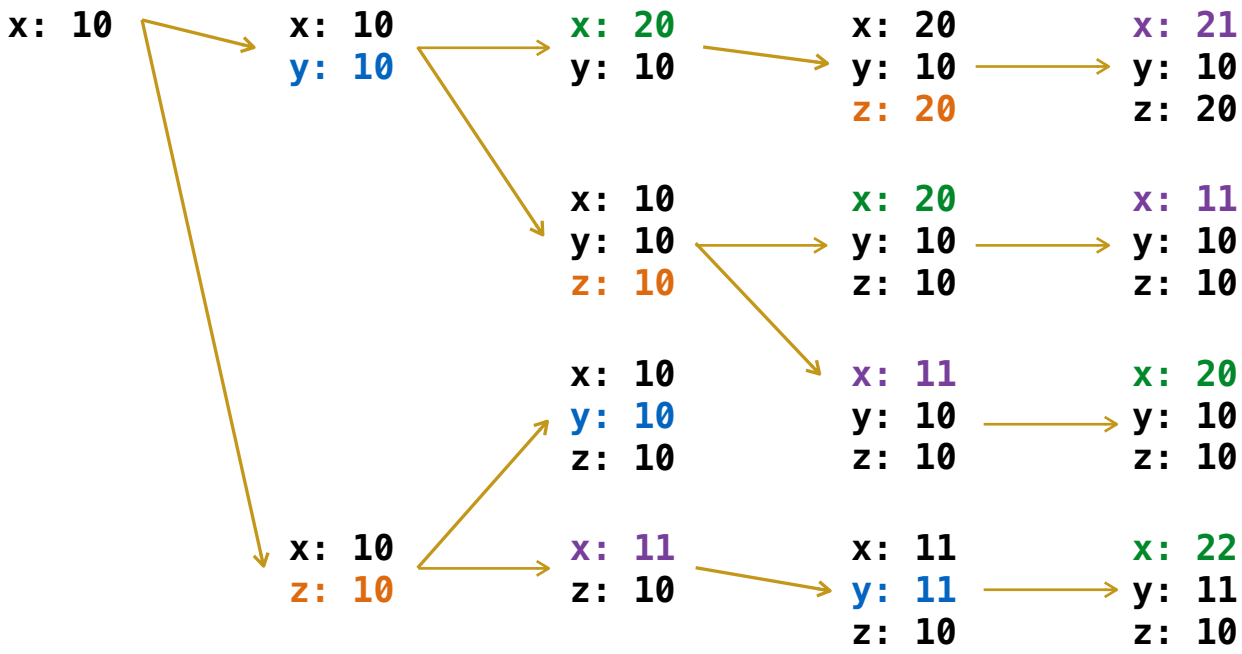
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



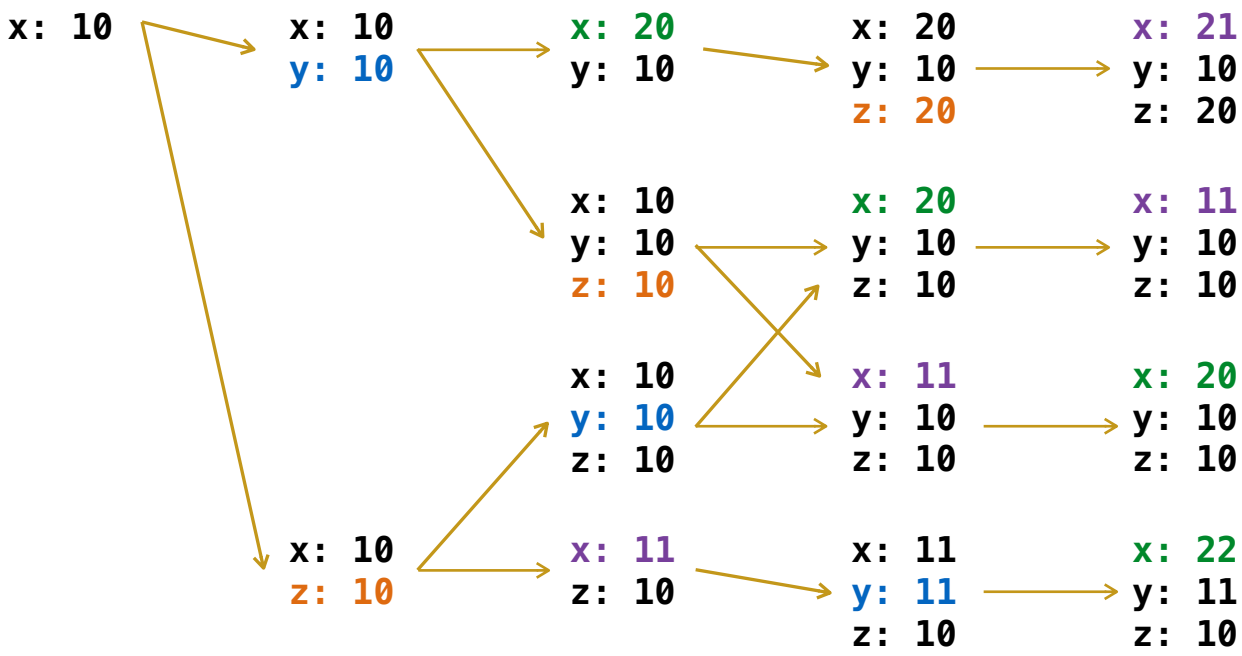
Shared State and Race Conditions

When multiple threads make changes to the same object, the result can be unpredictable

```
x = 10


do_something()
y = x
do_something()
x = y * 2


do_something()
z = x
do_something()
x = z + 1
```



Locks and Critical Sections

A critical section is a sequence of statements that should be executed atomically

Locks and Critical Sections

A critical section is a sequence of statements that should be executed atomically

```
x = 10
s = Lock()
```

Locks and Critical Sections

A critical section is a sequence of statements that should be executed atomically

```
x = 10
s = Lock()

do_something()
s.acquire()
y = x
do_something()
x = y * 2
s.release()
```

Locks and Critical Sections

A critical section is a sequence of statements that should be executed atomically

```
x = 10
s = Lock()

do_something()
s.acquire()
y = x
do_something()
x = y * 2
s.release()

do_something()
s.acquire()
z = x
do_something()
x = z + 1
s.release()
```

Locks and Critical Sections

A critical section is a sequence of statements that should be executed atomically

```
x = 10
s = Lock()

do_something()
s.acquire()
y = x
do_something()
x = y * 2
s.release()

do_something()
s.acquire()
z = x
do_something()
x = z + 1
s.release()
```