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.
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](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](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](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](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](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](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](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](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](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

Web browser

TCP Initialization Handshake

HTTP GET request of content

HTTP response with content

Follow-up requests for auxiliary content

Web server
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

Web browser

TCP Initialization Handshake

HTTP GET request of content

HTTP response with content

Follow-up requests for auxiliary content

(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

```python
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.

```python
x = 10

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

dosomething()
    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.

```python
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

x = 10
y = 10

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.

```python
don_something()
y = x
don_something()
x = y * 2

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

```
x: 10  x: 10  x: 20  x: 20
y: 10  y: 10  y: 10  z: 20
```

x = 10
y = x
x = y * 2
z = x
x = z + 1
Shared State and Race Conditions

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

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

```python
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.

```python
def do_something()
    y = x
    do_something()
    x = y * 2
    do_something()
    z = x
    do_something()
    x = z + 1
```

```
x = 10
x: 10
y: 10
x: 10
y: 10
x: 20
y: 10
z: 20
x: 20
y: 10
z: 20
x: 21
y: 10
z: 20
```
Shared State and Race Conditions

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

```python
x = 10

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

z = x
```

```plaintext
x: 10  x: 10  x: 20  x: 20  x: 21
y: 10  y: 10  y: 10  y: 10  
z: 20  z: 20  z: 20  z: 20
x: 10  x: 11
z: 10  z: 10
```
Shared State and Race Conditions

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

```python
def do_something():
    y = x
    do_something()
    x = y * 2

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

```
x = 10

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

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

```
x: 10
  y: 10
  x: 20
  y: 10
  z: 20
  x: 21

x: 10
  z: 10
  x: 11
  z: 10
  y: 11
  z: 10
```
Shared State and Race Conditions

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

```python
x = 10

do_something()
y = x

do_something()
x = y * 2

x = 10
y: 10

do_something()
y: 10
z: 20

x: 10
y: 10
z: 20

do_something()
z: 20

x: 10
y: 10
z: 20
x: 20
y: 10
z: 20
x: 21
y: 10
z: 20

x: 11
z: 10
x: 11
y: 11
z: 10
x: 22
y: 11
z: 10
```
Shared State and Race Conditions

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

```python
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.
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.

\[ x = 10 \]
\[ \text{do\_something()} \]
\[ y = x \]
\[ \text{do\_something()} \]
\[ x = y \times 2 \]
\[ \text{do\_something()} \]
\[ z = x \]
\[ \text{do\_something()} \]
\[ x = z + 1 \]
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
```

```
x: 10
y: 10
```

```
x: 20
ty: 10
z: 20
```

```
x: 21
ty: 10
z: 20
```

```
x: 10
y: 10
z: 10
```

```
x: 11
y: 10
z: 10
```

```
x: 11
y: 11
z: 10
```

```
x: 22
y: 11
z: 10
```
Shared State and Race Conditions

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

```python
x = 10

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

doz_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.

```python
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

\[
x = 10
s = \text{Lock()}
\]
A critical section is a sequence of statements that should be executed atomically.

```python
x = 10
s = Lock()
do_something()
s.acquire()
y = x
do_something()
x = y * 2
s.release()
```
A critical section is a sequence of statements that should be executed atomically.

```python
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.

```python
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()
```

- x: 10
- y: 10
- z: 10
- x: 20
- y: 10
- z: 20
- x: 21
- y: 10
- z: 20
- x: 11
- y: 11
- z: 10
- x: 22
- y: 11
- z: 10