Concurrency

Announcements

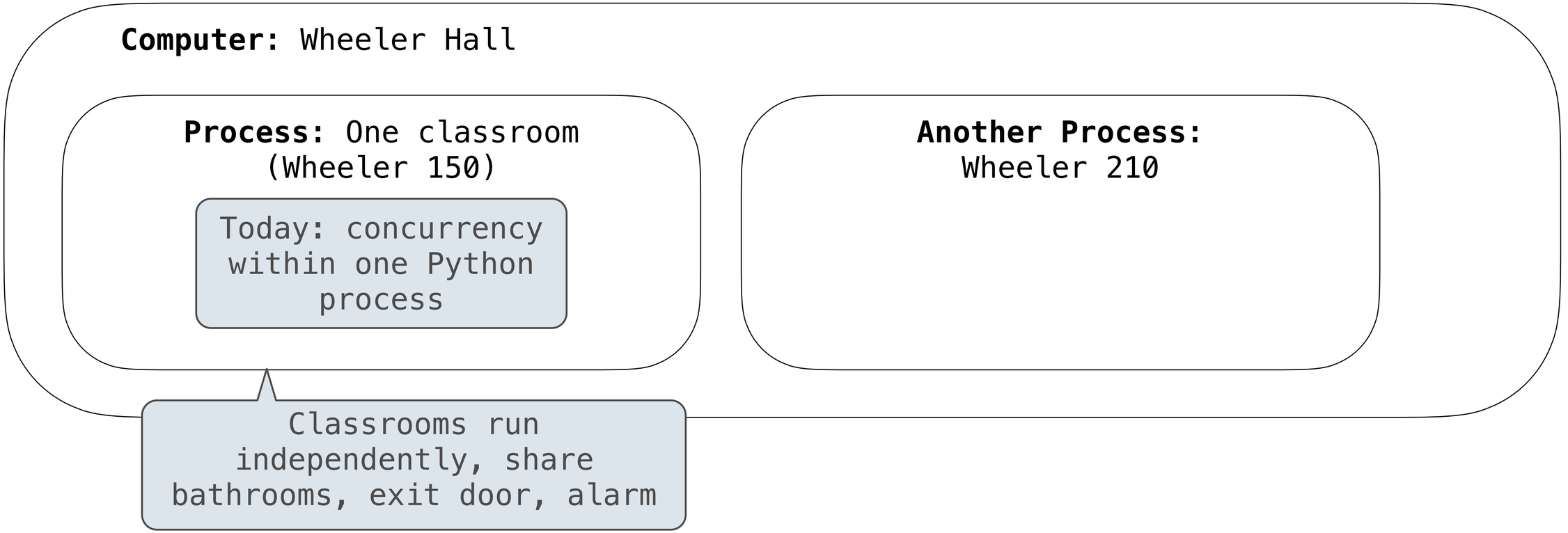
Concurrency On Your Laptop

(Demo)

Processes

Processes run mostly independently, share some resources (e.g., network connection)

python3 my_code.py starts one process



Concurrency:
task execution overlaps
(task B starts before task
A finishes)

Kay overlaps making lecture slides with writing discussion

Pratham starts Q1, then starts Q2, then finishes Q1

Parallelism:
Multiple tasks run
at the same time

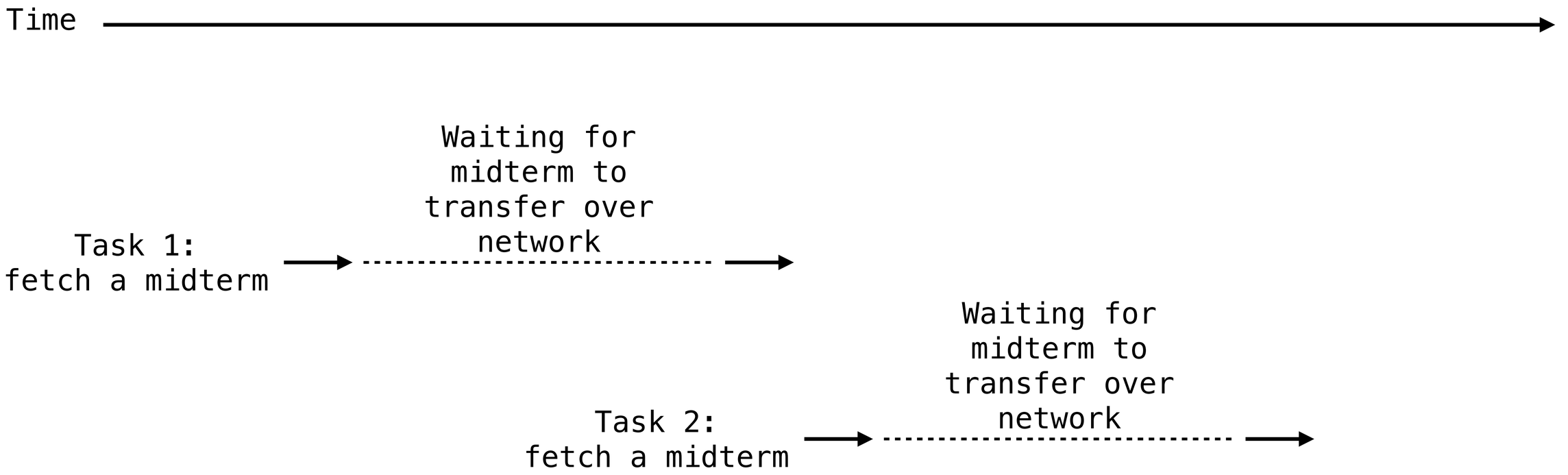
Kay works on lecture slides while John writes discussion

Skylar works on Q1 while Pratham works on Q2

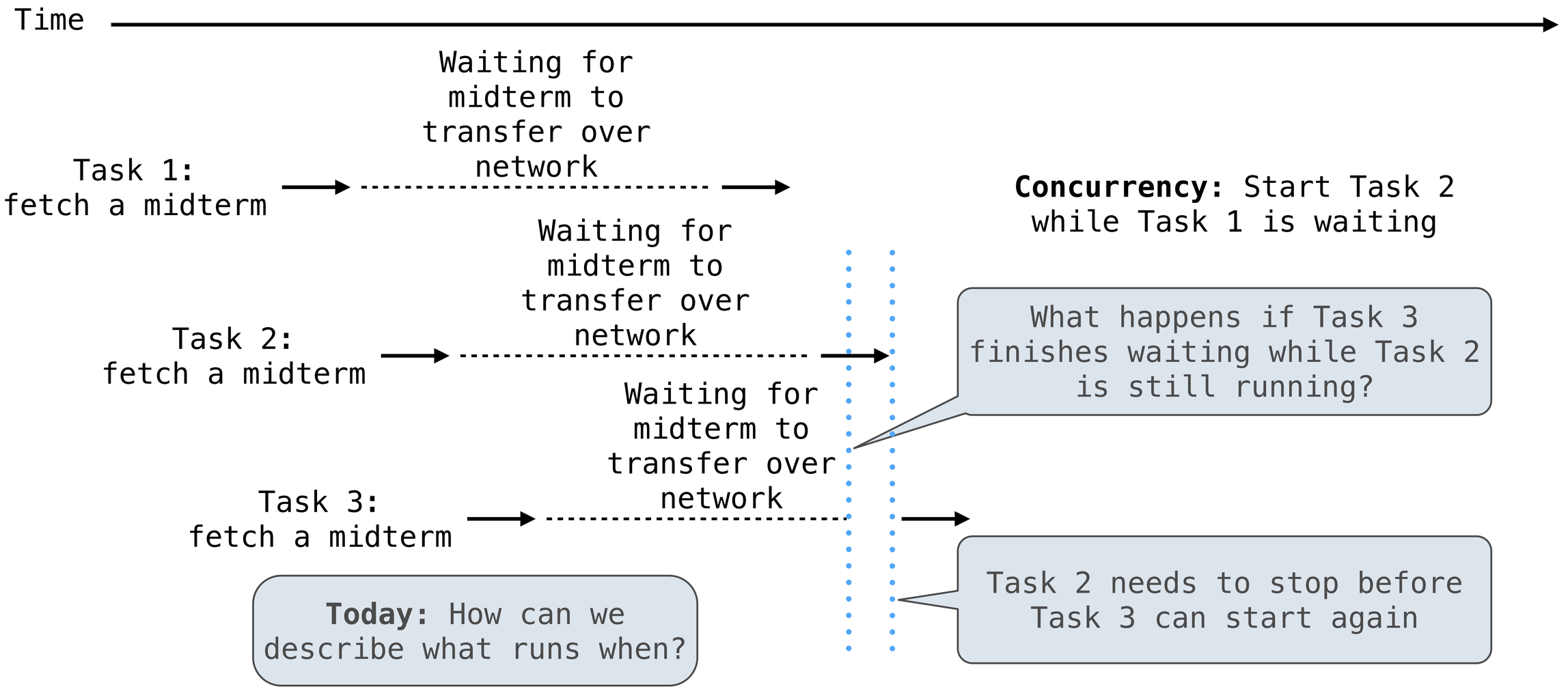
Single Python Process has a **Global Interpreter Lock.**
Only one line of Python code at a time

(Demo)

Waiting for Input / Output



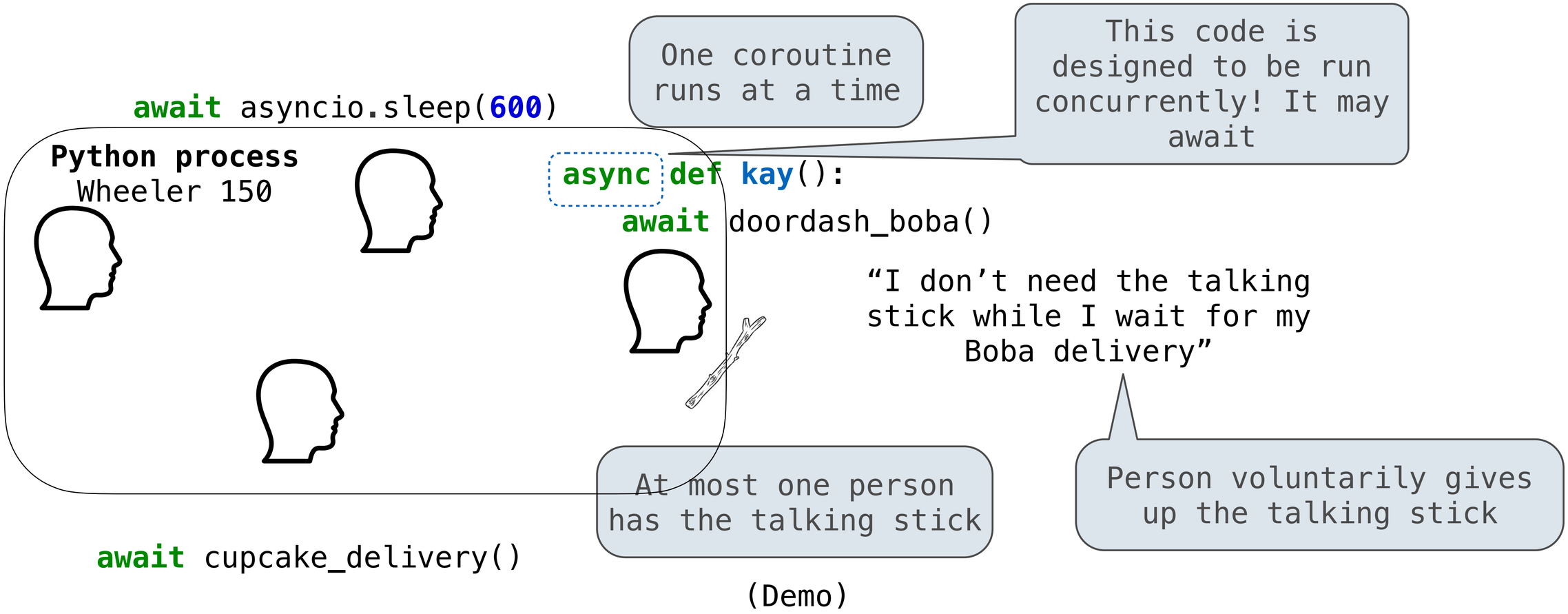
Do Work While Waiting on IO



Python: Coroutines

Coroutines: cooperative multitasking

Code runs until it voluntarily uses **await**



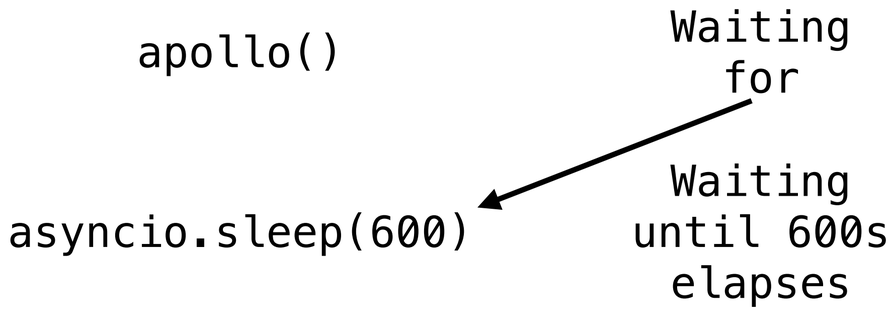
(Demo)

Python: Coroutines

```
async def apollo(seconds):
    await asyncio.sleep(seconds)

asyncio.run(apollo(600))
```

Give up control until asyncio.sleep(seconds) finishes

Coroutines:



Python: Coroutines

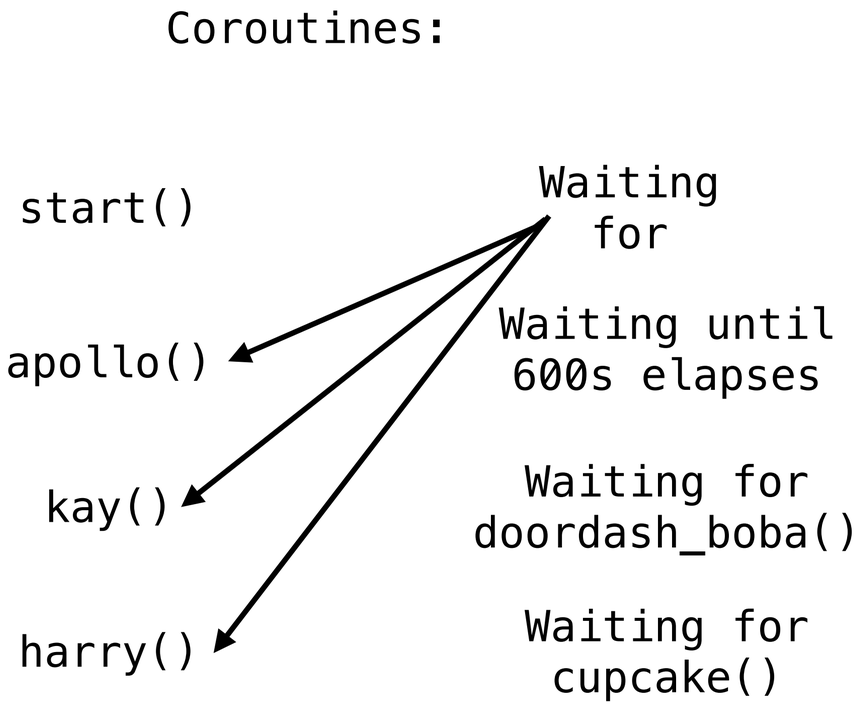
```
async def apollo():
    await asyncio.sleep(600)

async def kay():
    await doordash_boba()

async def harry():
    await cupcake()

async def start():
    await asyncio.gather(
        apollo(),
        kay(),
        harry(),
    )

asyncio.run(start())
```

A coroutine function (may await!)

Give up control until asyncio.sleep(600) finishes

Run all of the awaitables (e.g., coroutines) concurrently

Start an event loop (environment that knows how to handle concurrent work)



Examples

```
async def sleep1():
    await asyncio.sleep(2)
    await asyncio.sleep(2)
    await asyncio.sleep(2)

asyncio.run(sleep1())
```

```
async def sleep2():
    await asyncio.gather(
        asyncio.sleep(2),
        asyncio.sleep(2),
        asyncio.sleep(2))

asyncio.run(sleep2())
```

```
async def blocking_sleep(seconds):
    time.sleep(seconds)

async def sleep3():
    await asyncio.gather(
        blocking_sleep(2),
        blocking_sleep(2),
        blocking_sleep(2))

asyncio.run(sleep3())
```

How long does each coroutine take to run?

pollev.com/cs61a

How can you make a non-async function run concurrently?

```
async def new_blocking_sleep(seconds):
    await asyncio.to_thread(lambda: time.sleep(seconds))

async def sleep4():
    await asyncio.gather(new_blocking_sleep(2),
                         new_blocking_sleep(2),
                         new_blocking_sleep(2))

asyncio.run(sleep4())
```

Run in a separate thread! Python decides when it starts and stops

Fetching Midterms Concurrently

(Demo)

What’s hard about concurrency?

How do you describe concurrent code?

    await, async, asyncio.gather(), asyncio.run(), asyncio.to_thread()

How does concurrent code share state?

Shared State: In General

Pratham

Skylar

Changing shared state concurrently is hard!

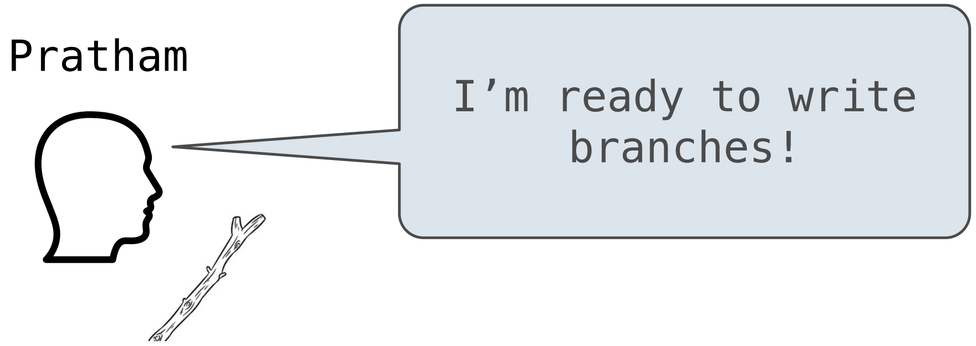
61A Discussion 5: Trees

```
    elif label(t) != p[0]:
        return False
    else:
        for b in
```

61A Discussion 5: Trees

```
    >>> has_path(t1, [3, 5])        # This path does
    True
    >>> has_path(t1, [3, 5, 6])     # This path goes
    True
    >>> has_path(t1, [3, 4, 5, 6])  # There is no pat
    False
    """
    if p == [label(t)]:  # when len(p) == 1 and p is
        return True
    elif label(t) != p[0]:
        return False
    else:
        while not t.is_leaf():
```

Shared State: Coroutines



Skylar

61A Discussion 5: Trees

61A Discussion 5: Trees

```
elif label(t) != p[0]:
    return False
else:
    while not t.is_leaf():
```

After you await, the state may have changed

61A Discussion 5: Trees

61A Discussion 5: Trees

```
elif label(t) != p[0]:
    return False
else:
    while not t.is_leaf():
```

Shared State: Coroutines

```
async def pratham(worksheet):
    if worksheet.stuck():
        result = await text_problem_to_friend()
    worksheet.write(result)
```

After await, worksheet may be different!

(Demo)

Concurrency in one slide

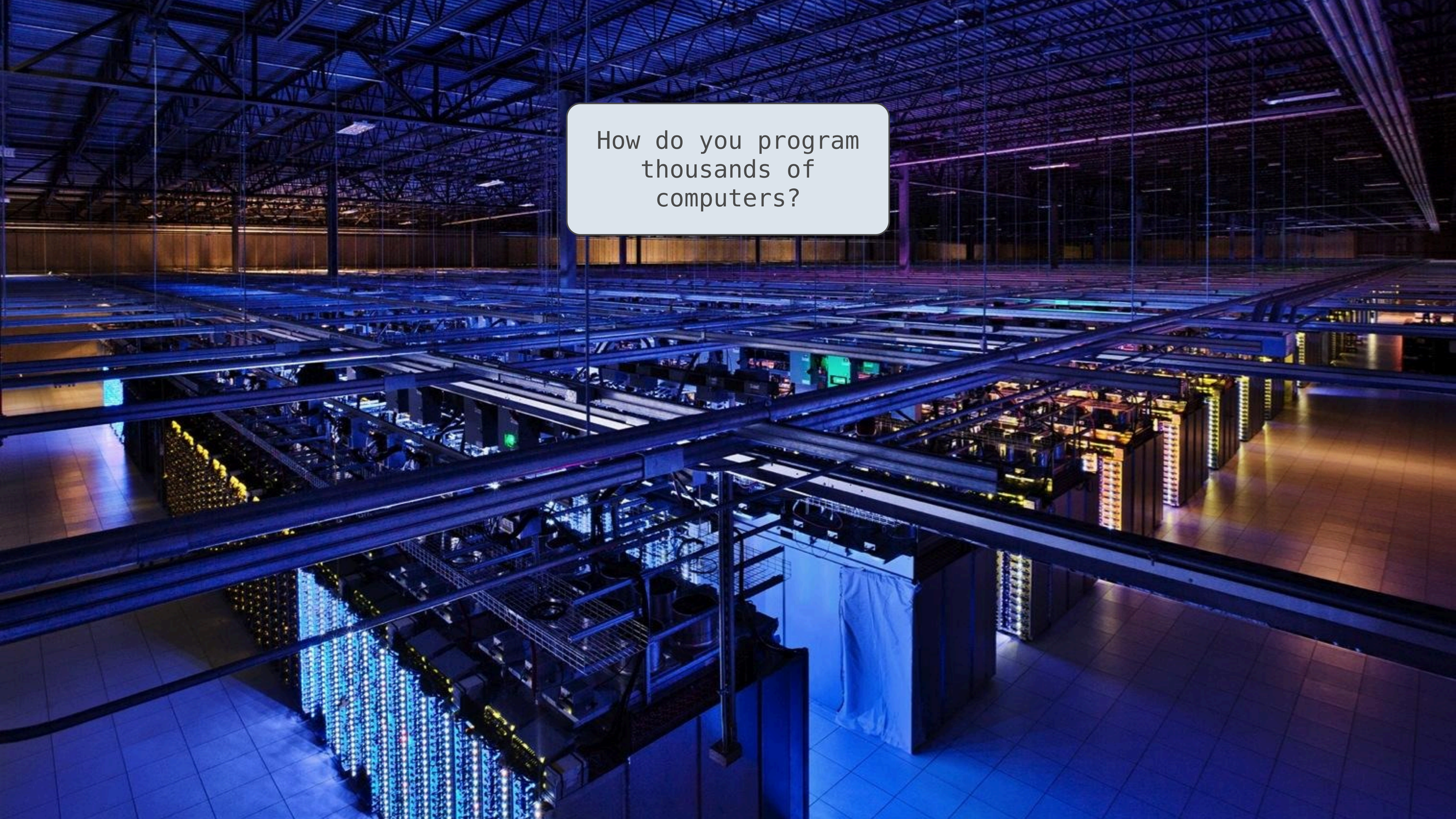
How do you describe concurrent code?

- `await`, `async`, `asyncio.run()`, `asyncio.to_thread()`

To run things concurrently: `asyncio.gather()`

How does concurrent code share state?

- Until you call `await`, nothing will change
- Mutable objects may be different after `await`


How do you program thousands of computers?