Lesson 10

Python Essentials

Overview

1. Asynchronous Operations
2. Calling Web APIs

Async Overview

- Asynchronous IO (async IO): pattern that has implementations across many programming languages.
- **async/await**: two new Python keywords that are used to define coroutines.
- **asyncio**: the Python package to implement async pattern.
- Consider using with operations take a long time
  - Web calls
  - Network IO
  - Complex data processing

Writing Async operations

- async: Flag to create a coroutine (function with an await call)
- await: "Pauses" code to wait for response
- create_task: Creates a handle (or coroutine) and schedules execution

```
async with aiohttp.ClientSession() as session:

    task_one = asyncio.create_task(load_data(session, 2))
    task_two = asyncio.create_task(load_data(session, 3))

    result_one = await task_one
    result_two = await task_two
```

Notes:

with will do automatically clean up, but we don't want it to clean up before we finish our async ops. The keyword asyc tells with to remember to wait.

Operations: - run: Runtime for asynchronous functions - create_task: Creates a handle (or coroutine) and schedules execution - gather: Create a collection of tasks to execute and wait for completion

Creating coroutines (functions using async/await): - async: Flag to create a coroutine (function with an await call) - await: "Pauses" code to wait for response

Calling Web APIs Overview

- You can call functions from programs hosted on web servers.
- Need: Address or Endpoint, method name, and parameters.
- Use request library

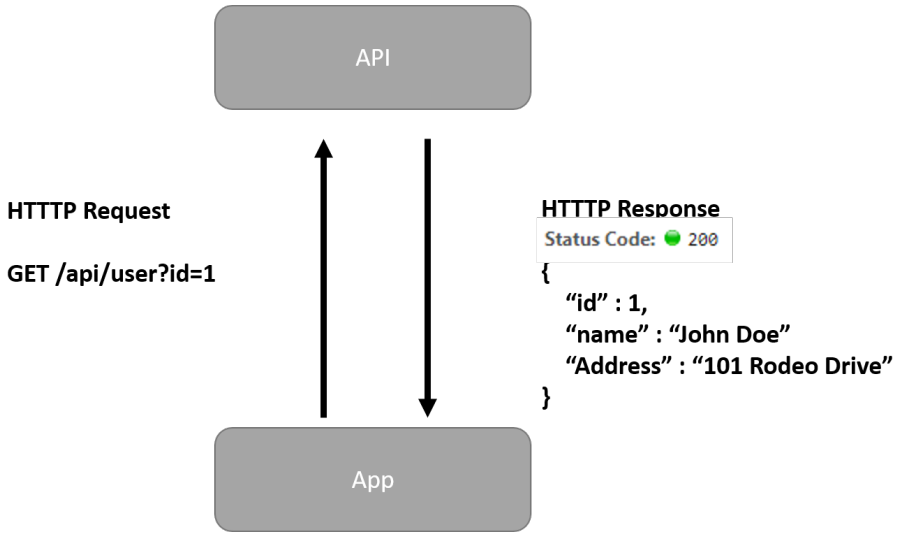

Figure 1: image

Notes: We don't want to stop everything just because one process is taking forever.

Async and Calling Web APIs

- By default, socket operations are blocking.

- requests.get(url) is not awaitable.
- But, almost everything in **aiohttp** is an awaitable coroutine
  - session.request()
  - response.text()