Overview

Concurrency & Async Programming

Introduction

This lesson introduces the concepts of concurrent and asynchronous programming and the python tools available for their implementation. Concurrency is doing multiple things at the same time – in Python, this can be accomplished through several approaches: multi-threading, multi-processing and asynchronous programming.

Threading and multi-processing support true parallelism, while asynchronous techniques allow tasks to be completed in unknown order. Python recently added features to support async, this lesson will explain how to use it.

These techniques can be applied to allow interfaces to be responsive while work is being done, and/or to boost performance with parallelizable tasks.

Learning Objectives

Upon successful completion of this lesson, you will be able to:

- Identify when to apply each of core techniques:
  - multi threading
  - multi processing
  - async
- Create a simple multi-threaded program with a message queue
- Create a simple multi-processing program with a message queue
- Create a web-api client with the asyncio package

New Words, Concepts, and Tools

- Concurrency
- Threading
- Multiprocessing
- Message Queues
- Coroutines
- Async

Prerequisites

Beyond the content covered on this and the prior class, there are no specific prerequisites for this lesson.

Optional Reading

Fluent Python:

- Ch. 16: Coroutines
- Ch. 17: Concurrency with Futures
- Ch. 18: Concurrency with asyncio

Threading:

Grok the GIL

Async:

- Unyielding: a take on threads vs async
- The Asyncio Cheat Sheet: This is a pretty helpful, how to do it guide.
- Nathaniel Smith's Some thoughts on asynchronous API design
- The built in asyncio suffers from a lot of legacy -- for a new take without any pre- 3.5 legacy: The Trio async package

Suggested Workflow

- Explore the "Before you Start" readings and video
- Work through the lesson content pages
- Watch the required videos
- Do the practice activity