Python TUTORIAL

Python Concurrency

In this section, you'll learn about Python concurrency including multithreading, multiprocessing, and asynchronous programming from scratch.

What you'll learn:

- Build high-performance & responsive Python applications using concurrency techniques.
- Develop multithreaded applications using multithreading.
- Develop a program that processes tasks in parallel.
- Understand the single-threaded concurrency model.

Section 1. Multithreading

In this section, you'll have a good understanding of processes and threads and how to develop multithreaded programs.

- Understanding Processes and Threads (https://www.pythontutorial.net/advanced-python/differences-between-processes-and-threads/) – help you understand the processes and threads, and the main differences between them.

- Threading (https://www.pythontutorial.net/advanced-python/python-threading/) – show you how to use the threading module to develop a multi-threaded application.
- Multithreading Example (https://www.pythontutorial.net/advanced-python/python-multithreading-example/) – build a multithreaded program that scraps stock prices.
- Threading Event (https://www.pythontutorial.net/python-concurrency/python-threading-event/) – show you how to use the threading Event to communicate between threads.
- How to stop a thread (https://www.pythontutorial.net/python-concurrency/python-stop-thread/) – learn how to stop a child thread from the main thread.
- Daemon threads (https://www.pythontutorial.net/advanced-python/python-daemon-threads/) – learn about daemon threads.
- Thread-safe Queue (https://www.pythontutorial.net/advanced-python/python-thread-queue/) – show you how to use a thread-safe queue to exchange data safely between multiple threads.
- Thread Pools (https://www.pythontutorial.net/advanced-python/python-threadpoolexecutor/) – guide you on managing multiple threads efficiently using the thread pool.
- Threading Lock (https://www.pythontutorial.net/advanced-python/python-threading-lock/) – learn how to access a shared variable safely from multiple threads using a Lock object.

Section 2. Multiprocessing

In this section, you'll learn how to utilize the multiprocessing package to develop programs that run tasks in parallel.

- Multiprocessing (https://www.pythontutorial.net/advanced-python/python-multiprocessing/) – show you how to run code in parallel using the multiprocessing module.
- Process Pools (https://www.pythontutorial.net/advanced-python/python-processpoolexecutor/) – learn how to manage processes more efficiently by using a process pool.

Section 3. Async I/O

In this section, you'll how to utilize concurrency provided by the `asyncio` package to improve program performance, throughput, and responsiveness.

- Understanding Event loop (https://www.pythontutorial.net/python-concurrency/python-event-loop/) – explain how the event loop works and how `asyncio` package uses the event loop to achieve a single-threaded concurrency model.
- async/await (https://www.pythontutorial.net/python-concurrency/python-async-await/) – introduce to you coroutines and how to use the async and await keywords to define and pause coroutines.
- Creating tasks (https://www.pythontutorial.net/python-concurrency/python-asyncio-create_task/) – learn how to create tasks and schedule them for running on the event loop.
- Canceling tasks (https://www.pythontutorial.net/python-concurrency/python-cancel-tasks/) – show you how to cancel a task using the `cancel()` method of the `Task` object.
- Canceling a task with a timeout (https://www.pythontutorial.net/python-concurrency/python-asyncio-wait_for/) – show you how to use the `asyncio.wait_for()` function to cancel a task with a timeout.
- Future (https://www.pythontutorial.net/python-concurrency/python-asyncio-future/) – explain to you the Future object and awaitables.
- Running multiple tasks concurrently with gather() (https://www.pythontutorial.net/python-concurrency/python-asyncio-gather/) – run a list of tasks concurrently with the `asyncio.gather()` function.