Multithreading

Python II

What is threading

In computing, a process is an instance of a computer program that is being executed. Any process has 3 basic components:

1. An executable program.
2. The associated data needed by the program (variables, work space, buffers, etc.)
3. The execution context of the program (State of process)

A **thread** is an entity within a process that can be scheduled for execution. Also, it is the smallest unit of processing that can be performed in an OS (Operating System).

Threading in python

A **thread** is a separate flow of execution. This means that your program will have two things happening at once. But for most Python 3 implementations the different threads do not actually execute at the same time: they merely appear to.

Time.sleep() / enumerate()

time.sleep()

enumerate(iterable)

Thread start / daemon threads

To start a separate **thread**, you create a Thread instance and then tell it to .start():

In computer science, a daemon is a process that runs in the background

A **daemon** thread will shut down immediately when the program exits.

Main thread / thread.join

Each process has at least one thread. Main thread is the thread which starts first when the file is starting.

Making __main__ to wait for other threads to end.

ThreadPoolExecutor

There’s an easier way to start up a group of threads than the one you saw above. It’s called a **ThreadPoolExecutor**, and it’s part of the standard library in concurrent.futures