Daemon Thread

This lesson describes how daemon threads exit as soon as the main program thread exits.

Daemon Thread

In Greek mythology, a **daemon** was a supernatural being of a nature between gods and humans whereas in the computer science realm a daemon is a computer program that runs as a background process rather than being under the direct control of an interactive user. A daemon thread in Python runs in the background. The difference between a regular thread and a daemon thread is that **a Python program will not exit until all regular/user threads terminate.** However, a program may exit if the daemon thread is still not finished.

For folks coming from Java background, this is similar to how JVM treats daemon threads. Daemon threads don't hold up a Java program from terminating.

Marking a thread Daemon

We can mark a thread daemon by passing in true for the `daemon` field in the `Thread` class's constructor.

Creating a daemon thread

```
daemonThread = Thread(target=daemon_thread_task, daemon=True)
```

In the below snippet the main thread creates a **non-daemonic** thread and exits. However, the program continues to run. The code widget throws an error because the execution times out.

```
from threading import Thread
from threading import current_thread
import time


def regular_thread_task():
    while True:
        print("{0} executing".format(current_thread().getName()))
        time.sleep(1)

regularThread = Thread(target=regular_thread_task, name="regularThread", daemon=False)
regularThread.start()  # start the regular thread

time.sleep(3)

print("Main thread exiting but Python program doesn't")
```

Run Save Reset

Select lesson text to explore the Explain with AI Premium Offering

Output 30.27s

```
regularThread executing
regularThread executing
regularThread executing
regularThread executing
Main thread exiting but Python program doesn't
```

```
Execution Timed Out!
```

In contrast, the runnable snippet below will exit even though the **daemon** thread is still executing.

```python
from threading import Thread
from threading import current_thread
import time

def daemon_thread_task():
    while True:
        print("{0} executing".format(current_thread().getName()))
        time.sleep(1)

regularThread = Thread(target=daemon_thread_task, name="daemonThread", daemon=True)
regularThread.start()  # start the daemon thread

time.sleep(3)

print("Main thread exiting and Python program too")
```

Run Save Reset

Output 3.7s

```
daemonThread executing
daemonThread executing
daemonThread executing
Main thread exiting and Python program too
```

Daemon threads are shut-down abruptly. Resources such as open files and database connections may not shut-down properly and daemon threads are not a good choice for such tasks. One final caveat to remember is that if you don't specify the `daemon` parameter in the constructor then the daemonic property is inherited from the current thread.

Back

Mark As Completed

Next

Subclassing Thread

Lock