# Subclassing Thread

This lesson demonstrates how we can subclass the Thread class to create threads.

# Subclassing Thread

Another way to create threads is to subclass the **`Thread`** class. As mentioned earlier, the **`threading`** module is inspired from Java and Java offers a similar way of creating threads by subclassing. Consider the snippet below:

## Creating threads by subclassing Thread class

```
class MyTask(Thread):

    def __init__(self):
        Thread.__init__(self, name="subclassThread", args=(2, 3))

    def run(self):
        print("{0} is executing".format(current_thread().getName()))
```

(/learn)

The important caveats to remember when subclassing the **`Thread`** class are:

- We can only override the **`run()`** method and the constructor of the **`Thread`** class.
- **`Thread.__init__()`** must be invoked if the subclass choses to override the constructor.
- Note that the **`args`** or **`kwargs`** don't get passed to the **`run`** method.

```
from threading import Thread
from threading import current_thread


class MyTask(Thread):

    def __init__(self):
        # The two args will not get passed to the overridden
        # run method.
        Thread.__init__(self, name="subclassThread", args=(2, 3))

    def run(self):
        print("{0} is executing".format(current_thread().getName()))


myTask = MyTask()

myTask.start()  # start the thread

myTask.join()  # wait for the thread to complete

print("{0} exiting".format(current_thread().getName()))

```

Back
Creating Threads

Next
Daemon Thread

Creating Threads

Daemon Thread

Mark as Completed

Report an Issue

Ask a Question
(https://discuss.educative.io/tag/subclassing-thread__threading-module__python-concurrency-for-senior-engineering-interviews)