Creating Threads

This lesson demonstrates how threads can be created in Python.

Creating Threads

We can create threads in Python using the **`Thread`** class. The constructor for the thread class appears below:

Thread constructor

```
Thread(group=None, target=None, name=None, args=(), kwargs={},
 *, daemon=None)
```

The **`group`** argument is reserved for future extensions. **`target`** is the code that the thread being created will execute. It can be any callable object. A callable object can be a function or an object of a class that has the **`__call__`** method. Next, we can pass the arguments to the target either as a tuple **`args`** or as a dictionary of key value pairs using **`kwargs`**. Finally, the constructor takes a boolean specifying if the thread being created should be treated as a daemon thread.

(/learn)

The below snippet creates a thread using the **`Thread`** class's constructor:

Creating thread

```
from threading import Thread
from threading import current_thread


def thread_task(a, b, c, key1, key2):
    print("{0} received the arguments: {1} {2} {3} {4} {5}".fo
rmat(current_thread().getName(), a, b, c, key1, key2))


myThread = Thread(group=None,  # reserved
                  target=thread_task,  # callable object
                  name="demoThread",  # name of thread
                  args=(1, 2, 3),  # arguments passed to the t
arget
                  kwargs={'key1': 777,
                          'key2': 111},  # dictionary of keywo
rd arguments
                  daemon=None  # set true to make the threa
d a daemon
                  )

myThread.start()  # start the thread

myThread.join()  # wait for the thread to complete
```

The above code appears as a runnable snippet below:

```
from threading import Thread
from threading import current_thread


def thread_task(a, b, c, key1, key2):
    print("{0} received the arguments: {1} {2} {3} {4} {5}".format(current_t


myThread = Thread(group=None,  # reserved
                  target=thread_task,  # callable object
                  name="demoThread",  # name of thread
                  args=(1, 2, 3),  # arguments passed to the target
                  kwargs={'key1': 777,
                          'key2': 111},  # dictionary of keyword arguments
                  daemon=None  # set true to make the thread a daemon
                  )

myThread.start()  # start the thread

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

```

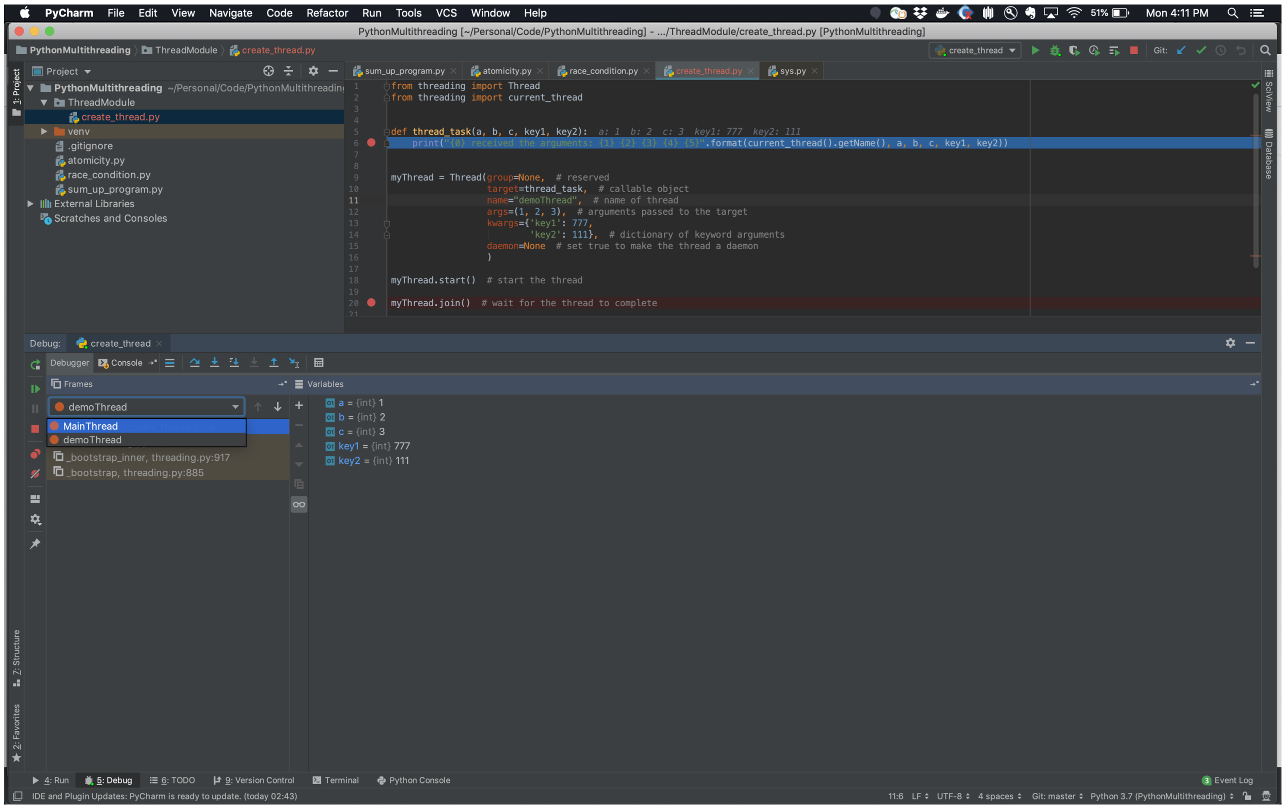
Creating and Running a Thread

Once the thread object is created, we must start it by invoking the `start()` method. Once started, the thread will execute the target.

Main Thread

The astute reader would realize that there has to be another thread that is actually executing the code we wrote to create a new thread. This ab initio thread is called the **main thread**. Under normal conditions, the main thread is the thread from which the Python interpreter was started. The below image captures the above script in debug mode. Notice the debugger dropdown displays two threads:

- Main Thread
- demoThread



```
from threading import Thread
from threading import current_thread


def thread_task(a, b, c, key1, key2):  a: 1  b: 2  c: 3  key1: 777  key2: 111
    print("{0} received the arguments: {1} {2} {3} {4} {5}".format(current_thread().getName(), a, b, c, key1, key2))


myThread = Thread(group=None,  # reserved
                  target=thread_task,  # callable object
                  name="demoThread",  # name of thread
                  args=(1, 2, 3),  # arguments passed to the target
                  kwargs={'key1': 777,
                          'key2': 111},  # dictionary of keyword arguments
                  daemon=None  # set true to make the thread a daemon
                  )

myThread.start()  # start the thread

myThread.join()  # wait for the thread to complete
```

```
demoThread
MainThread
demoThread
_bootstrap_inner, threading.py:917
_bootstrap, threading.py:885

a = {int} 1
b = {int} 2
c = {int} 3
key1 = {int} 777
key2 = {int} 111
```

Main and Spawned Thread in Debugger

Back

Moore's Law

Next

Subclassing Thread

✓ Mark as Completed

Report an Issue

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