Process

This lesson demonstrates how to create processes using the multiprocessing module.

Process

A process is a program in execution and operating systems provide different ways of creating new processes. Furthermore, each operating system has its own nuances when spawning new processes, which gets reflected in Python's APIs. The multiprocessing module offers the method **`set_start_method()`** to let the developer choose the way new processes are created. There are three of them:

- fork
- spawn
- fork-server

Example

The multiprocessing module exposes APIs very similarly to the threading module to create processes. The constructor of the **`Process`** class accepts a callable object that the spawned process then executes. Below is an example of creating three different processes:

Creating processes

```
from multiprocessing import Process
from multiprocessing import current_process
import os


def process_task():
    print("{0} has pid: {1} with parent pid: {2}".format(curre
nt_process().name, os.getpid(), os.getppid()))


process = [0] * 3

for i in range(0, 3):
    process[i] = Process(target=process_task, name="process-
{0}".format(i))
    process[i].start()

for i in range(0, 3):
    process[i].join()

print("{0} has pid: {1} ".format(current_process().name, os.ge
tpid()))
```

```
from multiprocessing import Process
from multiprocessing import current_process
import os


def process_task():
    print("{0} has pid: {1} with parent pid: {2}".format(current_process().name, os

if __name__ == "__main__":
    process = [0] * 3

    for i in range(0, 3):
        process[i] = Process(target=process_task, name="process-{0}".format(i))
        process[i].start()

    for i in range(0, 3):
        process[i].join()

    print("{0} has pid: {1} ".format(current_process().name, os.getpid()))
```

Note that in the above output, each process prints the pid (process id) that is assigned to it by the underlying operating system. The main process also prints its pid which is also the parent id of the spawned processes.

Passing arguments to processes

Similar to the threading module, we can pass arguments to a spawned process as a list of arguments or/and keyword dictionary. An example is given below:

Passing arguments to processes

```
from multiprocessing import Process
from multiprocessing import current_process
import os


def process_task(x, y, z, key1, key2):
    print("\n{0} has pid: {1} with parent pid: {2}".format(cur
rent_process().name, os.getpid(), os.getppid()))
    print("Received arguemnts {0} {1} {2} {3} {4}\n".format(x,
 y, z, key1, key2))


process = Process(target=process_task,
                  name="process-1",
                  args=(1, 2, 3),
                  kwargs={
                      'key1': 'arg1',
                      'key2': 'arg2'
                  })
process.start()

process.join()
```

```
from multiprocessing import Process
from multiprocessing import current_process
import os


def process_task(x, y, z, key1, key2):
    print("\n{0} has pid: {1} with parent pid: {2}".format(current_process().name,
    print("Received arguemnts {0} {1} {2} {3} {4}\n".format(x, y, z, key1, key2))


process = Process(target=process_task,
                  name="process-1",
                  args=(1, 2, 3),
                  kwargs={
                      'key1': 'arg1',
                      'key2': 'arg2'
                  })
process.start()

process.join()
```

In the above examples, we don't specify **`set_start_method()`** and let Python choose the default.

Back

Introduction

Next

Fork

Mark as Completed

Report an Issue

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