Namespace

This lesson discusses the Namespace which is conceptually similar to a notice board for all cooperating processes.

Namespace

Namespace is a type that can be registered with a **`SyncManager`** for sharing between processes. It doesn't have public methods but we can add writeable attributes to it. Think of namespace as a bulletin board, where attributes can be assigned by one process, and read by others. This works for immutable values like strings and primitive types but changes to mutable objects aren't propagated.

Using namespace

```
from multiprocessing.managers import SyncManager
from multiprocessing import Process


def process1(ns):
    print(ns.item)
    ns.item = "educative"

def process2(ns):
    print(ns.item)
    ns.item = "educative is awesome !"

if __name__ == '__main__':

    # create a namespace
    manager = SyncManager(address=('', 55555))
    manager.start()
    shared_vars = manager.Namespace()  # manager.Namespace()
    shared_vars.item = "empty"
    # manager.register("get_namespace", callable=lambda: None)

    # create the first process
    p1 = Process(target=process1, args=(shared_vars,))
    p1.start()
    p1.join()

    # create the second process
    p2 = Process(target=process2, args=(shared_vars,))
    p2.start()
    p2.join()

    print(shared_vars.item)
```

```
from multiprocessing.managers import SyncManager
from multiprocessing import Process
import multiprocessing


def process1(ns):
    print(ns.item)
    ns.item = "educative"


def process2(ns):
    print(ns.item)
    ns.item = "educative is awesome !"


if __name__ == '__main__':
    multiprocessing.set_start_method("spawn")
    # create a namespace
    manager = SyncManager(address=('', 55555))
    manager.start()
    shared_vars = manager.Namespace()  # manager.Namespace()
    shared_vars.item = "empty"
    # manager.register("get_namespace", callable=lambda: None)

    # create the first process
    p1 = Process(target=process1, args=(shared_vars,))
    p1.start()
    p1.join()

    # create the second process
    p2 = Process(target=process2, args=(shared_vars,))
    p2.start()
    p2.join()

    print(shared_vars.item)
```

The only caveat is attaching an attribute starting with an underscore, e.g., _item will create an attribute on the namespace proxy and not on the referent. The attribute will not be visible to other processes. The below code shows this in practice and an error is raised:

```
from multiprocessing.managers import SyncManager
from multiprocessing import Process
import multiprocessing


def process1(ns):
    print(ns._item)

if __name__ == '__main__':
    multiprocessing.set_start_method("spawn")

    # create a namespace
    manager = SyncManager(address=('', 55555))
    manager.start()
    shared_vars = manager.Namespace()  # manager.Namespace()
    shared_vars._item = "empty"

    # create the first process
    p1 = Process(target=process1, args=(shared_vars,))
    p1.start()
    p1.join()
```

However, notice that if you change the start method to "fork" on **line#10**, the code will not throw an error anymore. Fork causes a copy of the parent process's memory image and the namespace object retains the underscore attribute in the child process.

```
from multiprocessing.managers import SyncManager
from multiprocessing import Process
import multiprocessing


def process1(ns):
    print(ns._item)

if __name__ == '__main__':
    multiprocessing.set_start_method("fork")

    # create a namespace
    manager = SyncManager(address=('', 55555))
    manager.start()
    shared_vars = manager.Namespace()  # manager.Namespace()
    shared_vars._item = "empty"

    # create the first process
    p1 = Process(target=process1, args=(shared_vars,))
    p1.start()
    p1.join()
```

Back

Working with Managers

Next

Quiz 1

Mark as Completed

Report an Issue

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