Logging

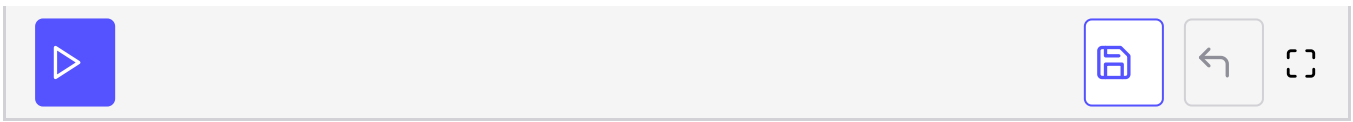
Logging processes is a little different than logging threads. The reason for this is that Python's logging packages doesn't use process shared locks, so it's possible for you to end up with messages from different processes getting mixed up. Let's try adding basic logging to the previous example. Here's the code:

```
import logging
import multiprocessing

from multiprocessing import Process, Lock


def printer(item, lock):
    """
    Prints out the item that was passed in
    """
    lock.acquire()
    try:
        print(item)
    finally:
        lock.release()

if __name__ == '__main__':
    lock = Lock()
    items = ['tango', 'foxtrot', 10]
    multiprocessing.log_to_stderr()
    logger = multiprocessing.get_logger()
    logger.setLevel(logging.INFO)
    for item in items:
        p = Process(target=printer, args=(item, lock))
        p.start()
```



The simplest way to log is to send it all to stderr. We can do this by calling the **log_to_stderr()** function. Then we call the **get_logger** function to get access to a logger and set its logging level to INFO. The rest of the code is the same. I will note that I'm not calling the **join()** method here. Instead, the parent thread (i.e. your script) will call **join()** implicitly when it exits.

When you do this, you should get output like the following:

```
[INFO/Process-1] child process calling self.run()
tango
[INFO/Process-1] process shutting down
[INFO/Process-1] process exiting with exitcode 0
[INFO/Process-2] child process calling self.run()
[INFO/MainProcess] process shutting down
foxtrot
[INFO/Process-2] process shutting down
[INFO/Process-3] child process calling self.run()
[INFO/Process-2] process exiting with exitcode 0
10
[INFO/MainProcess] calling join() for process Process-3
[INFO/Process-3] process shutting down
[INFO/Process-3] process exiting with exitcode 0
[INFO/MainProcess] calling join() for process Process-2
```

Let’s move on and learn about Pools.