Shared Data

The examples we have seen so far have the processes completely independent of each other. In many problems the processes need to coordinate by sharing some of the variables.

The multiprocessing module has a class for this: multiprocessing.ctypes.RawValue( )

The names here are getting really long. To simplify this, we will import the modules in a different way.

If we say

```
import multiprocessing
```

then we can use things in the multiprocessing module, but we need to prefix the multiprocessing name:

```
multiprocessing.Process( )
```

If we say

```
from multiprocessing import *
```

we can use them without prefixing the module name.

In the past when we have used the random module we said

```
import random
```

and used it as

```
x = random.randint(0, 10)
```

If we said instead

```
from random import *
```

we could use this as

```
x = randint(0, 10)
```

To make a shared variable that represents an integer, we use

```
        r = RawValue( "i", <integer value> )
```

such as

```
        r = RawValue( "i", 0 )
```

RawValues have an instance variable self. value that represents the current value. For example, to set r to 25 we would say

```
        r.value = 25.
```

To print r we would say

```
        print( r.value)
```

RawValues need to be created outside of the function the processes will work on, and passed as an argument to them.

<FirstSharedExample.py>

You need to be careful about how you work with shared data because the processes can modify it asynchronously.

<SecondSharedExample.py>

<ThirdSharedExample.py>