15th Feb Assignment

February 23, 2023

1 Assignment 15

Q1. What is multiprocessing in python? Why is it useful?

Ans.Multiprocessing in Python is a built-in package that allows the system to run multiple processes simultaneously. It will enable the breaking of applications into smaller threads that can run independently

The multiprocessing package offers both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using subprocesses instead of threads. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine.

Q2. What are the differences between multiprocessing and multithreading?

Ans. The differences in both are:

- In Multiprocessing, CPUs are added for increasing computing power.While In Multithreading, many threads are created of a single process for increasing computing power.
- In Multiprocessing, Many processes are executed simultaneously.While in multithreading, many threads of a process are executed simultaneously.
- Multiprocessing are classified into Symmetric and Asymmetric.While Multithreading is not classified in any categories.
- In Multiprocessing, Process creation is a time-consuming process.While in Multithreading, process creation is according to economical.
- In Multiprocessing, every process owned a separate address space. While in Multithreading, a common address space is shared by all the threads.

Q3. Write a python code to create a process using the multiprocessing module.

[2]:
```python
#### create a process using multiprocessing module
import multiprocessing

def test():
    print('This is my multiprocessing program')

if __name__=="__main__":
    m=multiprocessing.Process(target=test)
    print('This is my main program')
    m.start()
    m.join()
```

```
This is my main program
This is my multiprocessing program
```

Q4. What is a multiprocessing pool in python? Why is it used?

Ans.Pool can be used for parallel execution of a function across multiple input values, distributing the input data across processes.

[3]:
```
###multiprocessing pool

def square(i):
    return i*i

if __name__=="__main__":
    m=multiprocessing.Pool(processes=5)
    output=m.map(square,[1,2,3,4,5,6,7,8,9,0])
    #this divide the data into 5 processes this is going to give me pool
    # of outcomes
    print(output)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 0]
```

Q5. How can we create a pool of worker processes in python using the multiprocessing module?

[4]:
```
##create a pool of worker process in python
from multiprocessing import Pool, TimeoutError
import time
import os

def f(x):
    return x*x

if __name__ == '__main__':
    # start 4 worker processes
    with Pool(processes=4) as pool:

        # print "[0, 1, 4,..., 81]"
        print(pool.map(f, range(10)))

        # print same numbers in arbitrary order
        for i in pool.imap_unordered(f, range(10)):
            print(i)

        # evaluate "f(20)" asynchronously
        res = pool.apply_async(f, (20,))      # runs in *only* one process
        print(res.get(timeout=1))             # prints "400"

        # evaluate "os.getpid()" asynchronously
        res = pool.apply_async(os.getpid, ()) # runs in *only* one process
        print(res.get(timeout=1))             # prints the PID of that process

        # launching multiple evaluations asynchronously *may* use more processes
        multiple_results = [pool.apply_async(os.getpid, ()) for i in range(4)]
        print([res.get(timeout=1) for res in multiple_results])

        # make a single worker sleep for 10 seconds
        res = pool.apply_async(time.sleep, (10,))
        try:
            print(res.get(timeout=1))
        except TimeoutError:
            print("We lacked patience and got a multiprocessing.TimeoutError")

        print("For the moment, the pool remains available for more work")

    # exiting the 'with'-block has stopped the pool
    print("Now the pool is closed and no longer available")
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
0
1
9
4
25
16
36
64
81
49
400
1730
[1727, 1728, 1729, 1728]
We lacked patience and got a multiprocessing.TimeoutError
For the moment, the pool remains available for more work
Now the pool is closed and no longer available
```

Q6. Write a python program to create 4 processes, each process should print a different number using the multiprocessing module in python.

[7]:
```
#program
def out(i):
    return i
if __name__=="__main__":
    m=multiprocessing.Pool(processes=4)
    output=m.map(out,[1,2,3,4,5,6,7,8,9,0])
    #this divide the data into 5 processes this is going to give me pool
    # of outcomes

for i in output:
    print(i)
```

```
1
2
3
4
5
6
7
8
9
0
```