15 FEB ASS

March 7, 2023

[ ]:
```
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 application into smaller threads that can run␣
 ↪independently.The operating system
then can allocate all these threads or processes to the processor to run them␣
 ↪parallely, thus improving the
overall performance and efficiency.
```

[ ]:
```
A Multiprocessing system can be represented as:
    - A system with more than a single centeral processor
    - A multi-core processor , a single computing unit with multiple␣
 ↪independent core processing units.
In  Multiprocessing , the system can divide and assign tasks to different␣
 ↪processors.
```

[ ]:
```

```

[ ]:
```

```

[ ]:
```
Q2. What are the differences between multiprocessing and multithreading?
```

[ ]:
```
ANS -
```

[ ]:
```
Multiprocessing:
                Multiprocessing refers to using multiple cpus/processors in a␣
 ↪single system. multiple cpu can act in
a parallel fashion and execute multiple processes that run on multiple␣
 ↪processors. when the task is over , the result from all
processors compiled together to provide the final output.multiprocessing␣
 ↪increase the computing power to a great extent.
```

[ ]:
```
Multithreading :
```

```
                Multithreading refers to multiple threads being executed by a␣
 ↪single cpu in such a way that each thread
is executed in a parallel fashion and cpu/processors is switched between them␣
 ↪using context switch.Multithreading is a technique to
increase the throughput of a processor. in multithreading, accessing memory␣
 ↪addresses is easy because all of the threads share the
same parent process.
```

[ ]:

[ ]:

[ ]:
```
Q3. Write a python code to create a process using the multiprocessing module.
```

[ ]:
```
ANS -
```

[4]:
```
import multiprocessing

def test():
    print("this is my multiprocessing prog")
if __name__ == "__main__":
    m = multiprocessing.Process(target = test)
    print("this is my main prog")
    m.start()
    m.join()
```

```
this is my main prog
this is my multiprocessing prog
```

[ ]:

[ ]:

[ ]:
```
Q4. What is a multiprocessing pool in python? Why is it used?
```

[ ]:
```
ANS -
```

[ ]:
```
Python Multiprocessing Pool can be used for parallel execution of a function␣
 ↪across multiple input data
across processes.
```

[3]:
```
import multiprocessing

def  square(n):
    return n **2
if __name__ == "__main__":
    with multiprocessing.Pool(processes = 5)as pool:
```

```
        out = pool.map(square , [2,2,3,4,5,5,45,6,7])
        print(out)
```

```
[4, 4, 9, 16, 25, 25, 2025, 36, 49]
```

[ ]:

[ ]:

[ ]:
```
Q5. How can we create a pool of worker processes in python using the␣
 ↪multiprocessing module?
```

[ ]:
```
ANS -
```

[5]:
```
import multiprocessing

def cube(n):
    return n**3
if __name__ == "__main__":
    with multiprocessing.Pool(processes = 6) as pool:
        out = pool.map(cube , [3,4,6,8,14,34,8,9])
        print (out)
```

```
[27, 64, 216, 512, 2744, 39304, 512, 729]
```

[ ]:

[ ]:

[ ]:
```
Q6. Write a python program to create 4 processes, each process should print a␣
 ↪different number using the
multiprocessing module in python.
```

[ ]:
```
ANS -
```

[ ]:
```
import multiprocessing

def sender(conn , msg):
    for i in msg:
        conn.send(i)
    conn.close()

def receive(conn):
    while True:
        try:
            msg = conn.recv()
        except Exception as e :
```

```
            print(e)
            break
        print(msg)

if __name__ == "__main__":
    msg = [1,2,3,4 , [5,6,7] , 8,9,10 , [11,12,13,14]]
    parent_conn , child_conn = multiprocessing.Pipe()
    m1 = multiprocessing.Process(target = sender , args = (child_conn, msg))
    m2 = multiprocessing.Process(target = receive , args = (parent_conn,))
    m1.start()
    m2.start()
    m1.join()
    child_conn.close()
    m2.join()
    parent_conn.close()
```

```
1
2
3
4
[5, 6, 7]
8
9
10
[11, 12, 13, 14]
```

[ ]:

[ ]:

[ ]: