multithreading

June 19, 2023

[ ]:
```
#Q1
```

[ ]:
```
'multithreading is the process where the multiple programme runs in a single ␣
 ↪core of the processor'
' it is used as we can call the multiple programme in a single a core it help␣
 ↪in getting the time consume to get the output is very less'
import multithreading
```

[ ]:
```
#Q2
```

[ ]:
```
'as we cannnot call the multithreading directly we need to import the threading␣
 ↪module from the library . using this module we can combine the multiple␣
 ↪programme inside a single core
```

[ ]:
```
1.activeCount()-'the function that returns the number of thread that is␣
 ↪currently alive.it counts both the main thread and any active non thread␣
 ↪that have been created '
```

[ ]:
```
2. currentcount()-'the function return the thread object corresponding to the␣
 ↪current thread .
```

[ ]:
```
3. enumerate() - 'this function return a list of all thread object currently␣
 ↪alive . it may be main thread or an non active thread'
```

[ ]:
```
#Q3
```

[ ]:
```
1.run()-'start the execution of the thraed. to typically
'override this method in a subclass of thread class to define the behaviour of␣
 ↪the thread'
```

[2]:
```
import threading
import logging
logging.basicConfig(filename='run.txt',level=logging.INFO)
class mythread(threading.Thread):
    def run(self):
        logging.info('thread is running')
```

```
thread = mythread()

thread.start()
```

[ ]:
```
2.start()-'start a thread by invoking its run method .it create new thread of␣
↪execution and calls the run method in the seprate thread
```

[4]:
```
import threading
import logging
logging.basicConfig(filename='start.txt',level=logging.INFO)
def my_function():
    logging.info('thread is starting and running')

thread = threading.Thread(target=my_function)
thread.start()
```

[ ]:
```
3. join()-'this method wait the thread for the complete execution it block the␣
↪calling thread until the thread on which it is called finishes execution
```

[7]:
```
import threading
import logging
logging.basicConfig(filename='join.txt',level=logging.INFO)
def my_function():
    logging.info('thread is joining and running')

thread = threading.Thread(target=my_function)
thread.start()

thread.join()
```

[ ]:
```
4. isAlive() - 'this check whether the current thread is executing .it returns␣
↪true if the thread is alive otherwise false
```

[1]:
```
import threading
import logging
import time
logging.basicConfig(filename='alive.txt',level=logging.
↪INFO,format='%(asctime)s-%(message)s')
def my_function():
    time.sleep(2)

thread = threading.Thread(target=my_function)
thread.start()

logging.info('is the thread alive %s',thread.is_alive())
thread.join()
logging.info('is the thread is alive %s',thread.is_alive())
```

[ ]:
```
#Q4
```

[2]:
```
import threading
import logging
logging.basicConfig(filename='twothread.txt',level=logging.
 ↪INFO,format='%(asctime)s,%(levelname)s,%(message)s')

def logging_square():
    for i in range(1,6):
        logging.info('square of %d is %d',i,i**2)
def logging_cube():
    for i in range(1,6):
        logging.info('cube of %d is %d',i,i**3)
thread1= threading.Thread(target=logging_square)
thread2= threading.Thread(target=logging_cube)

thread1.start()
thread2.start()

thread1.join()
thread2.join()
```

[ ]:
```
#Q5
```

[ ]:
```
'advantages'
```

[ ]:
```
1.' it allows the multiple programme to execute simultaneously,enablling␣
 ↪execution of different part of program'
```

[ ]:
```
2.'thread enable share the memory space and resource with the process
```

[ ]:
```
3.'multithreading can enhace the responsive application by allowing the time␣
 ↪consuming operation on the background which keep the user interface␣
 ↪responsive and interactive application'
```

[ ]:
```
4.'multithreading can simplify the design and structure of the complex␣
 ↪application by dividing into the smaller which make the thread to operate␣
 ↪easily and focus on the specific task'
```

[ ]:
```
5.'multithreading enable parallel processing on systeme with multiple cpus core
```

[ ]:
```
'disadvantages'
```

[ ]:
```
1.'multithreading introduces the complexity to the code as need to cordinate␣
 ↪and synchronize their action properly '
2. 'this may include the problem like race conditions ,deadlocks etc'
```

```
3. 'thread share the same memory space which requires proper synchronization␣
 ↪mechanisms . managing shared data ensuring thread safety can be complex and␣
 ↪error'
4.' multithreading can improve the performace on system on the multiple core of␣
 ↪cpu .as there is a limited scalabilty
5. ' as the number of thread increases the system  efficiently may decrease '
```

```
[ ]: #Q6
```

```
[ ]: 'race condition and deadlock are the similar issue that occur in multithreading␣
      ↪programme'
     1. 'deadlock'
     .'dead lock refer to a situation where two or more thraed are blocked waiting␣
      ↪for each other to release the resources they hold'
     .'dead lock occur when both the thread are compete for shared resources and␣
      ↪acquire them in different orders'
     .'resource cannot be shared simultaneously'
     .' a thread holding  at least one resource is waiting to acqurie additional␣
      ↪resource held by other thread'
     2. 'race condition'
     . 'race condition occurs when the multiple resource are shared but the final␣
      ↪outcome depends upon the relative timing of their execution'
     . ' in race condition often result unperdictable result as in the order of the␣
      ↪exceution'
     . ' in race condition can  araise when thread perform non atomic operation of␣
      ↪not divisible'
     . ' in race condition can lead to the unexpected of leaving of data in the␣
      ↪interleaving so in middle of the execution'
     . 'eg fo race condition is include reading/writing or transfer of data with␣
      ↪proper synchronization
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```