NAME : DHEVISHREE GN REG NUM : 22MID0136 LAB ASSESSMENT : 11

In [1]:
```
 # TASK-1
 # Threading significantly speed up the program, but it depends on the task that you are doing
 import time
 start = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
 def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
 do_something()
 finish = time.perf_counter() #using to find the time the entirer sequecnce takes from here
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second...
Done Sleeping
Finished in 1.0 seconds
```

In [2]:
```
 # TASK-2
 # Run the function twice
 import time
start = time.perf_counter() #to find the time the entrire sequecnce takes from here
 def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
 do_something()
 do_something()
 finish = time.perf_counter() #to find the time the entrire sequecnce till here
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Finished in 2.0 seconds
```

In [3]:
```
 # TASK-3
 import threading     #its a existing python package, this is the traditional way of doing threading, more effe
 import time
 start = time.perf_counter()
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
 #instead the running the two functions like this, threads are used both of these
 #do_something()
 #do_something()
 t1 = threading.Thread(target = do_something) #do not pass the function with (), as we dont intend to run the f
 t2 = threading.Thread(target = do_something)
 finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

```
Finished in 0.0 seconds
```

In [4]:
```
 # TASK-4
 import threading     #its a existing python package, this is the traditional way of doing threading, more effe
 import time
 start = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
 def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
 #instead the running the two functions like this, threads are used both of these
 #do_something()
 #do_something()
 t1 = threading.Thread(target = do_something) #do not pass the function with (), as we dont intend to run the f
 t2 = threading.Thread(target = do_something)
 t1.start()
 t2.start()
 finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second...Sleeping 1 second...

Finished in 0.01 seconds
Done Sleeping
Done Sleeping
```

In [6]:
```
 #TASK-5
 import threading
import time
 start = time.perf_counter()
```

```
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
 #instead the running the two functions like this, threads are used both of these
 #do_something()
 #do_something()
 t1 = threading.Thread(target = do_something) #do not pass the function with (), as we dont intend to run the fu
 t2 = threading.Thread(target = do_something)
 t1.start()
 t2.start()
 t1.join()
 t2.join()
 finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second...
Sleeping 1 second...
Done Sleeping
Done Sleeping
Finished in 1.01 seconds
```

In [14]:
```
 import threading     #no need to install, its already a part of python package, this is the traditional way of
 import time
 start = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
 def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
 for _ in range(10):      # underscore variable is a throwaway variable to simply loop for 10 times and we are n
    t = threading.Thread(target = do_something)
    t.start()            # We cant use join() within the loop as it will join on the thread before looping throu
                         # To do this we can create a list of threads and perform join()
 finish = time.perf_counter() #using to find the time the entrirer sequecnce takes from here
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Sleeping 1 second...
Finished in 0.01 seconds
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
```

In [20]:
```
#TASK-7
# Threading effect with 10 calls
import threading
import time
start = time.perf_counter()
def do_something():
    print('Sleeping 1 second...')
    time.sleep(1)
    print('Done Sleeping')
threads = []

for _ in range(10):
    t = threading.Thread(target = do_something)
    t.start()
    threads.append(t)
    for thread in threads:
        thread.join()


finish = time.perf_counter()
print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Sleeping 1 second...
Done Sleeping
Finished in 10.06 seconds
```

In [23]:

```
 #TASK-8
 # Threading fucntions with arguments
 # Threading effect with 10 calls
 import threading
import time
 start = time.perf_counter()
def do_something(seconds):                    #argument passed here
     print(f'Sleeping {seconds} second(s)...') #fstring
     time.sleep(seconds)
     print('Done Sleeping')
 threads = []

for _ in range(10):
    t = threading.Thread(target = do_something, args = [1.5]) #same argument for all threads in the list
    t.start()
    threads.append(t)
 for thread in threads:
    thread.join()

finish = time.perf_counter()
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1.5 second(s)...Sleeping 1.5 second(s)...

Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Sleeping 1.5 second(s)...
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Done Sleeping
Finished in 1.51 seconds
```

In [40]:

```
import concurrent.futures
# import threading    - Not required
import time
start = time.perf_counter()
def do_something(seconds):
    print(f'Sleeping {seconds} second(s)...')
    time.sleep(seconds)
    #print('Done Sleeping')
    return 'Done Sleeping...'
with concurrent.futures.ThreadPoolExecutor() as executor:
    f1 = executor.submit(do_something, 1)  #submit function will schedule the execution of function and returns
    print(f1.result())

finish = time.perf_counter()

print(f'Finished in {round(finish-start,2)} seconds')
import threading
import time
import random
def print_names():
    for name in ('John', 'Mark', 'Elon', 'Callahan'):
        print (name)
        time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
    for _ in range(4):
        print(random.randint(20,50))

        time.sleep(random.uniform(0.5,1.5))
t1 = threading.Thread(target = print_names)
t2 = threading.Thread(target = print_ages)
# The above threads are not doing anything yet. To do that, use start()
t1.start()
t2.start()
# t1.join()
# t2.join()
```

```
Sleeping 1 second(s)...
Done Sleeping...
Finished in 1.01 seconds
John
26
Mark
24
Elon
42
Callahan
34
```

In [42]:

```
 import concurrent.futures
 # import threading    - Not required
 import time
 start = time.perf_counter()
def do_something(seconds):
    print(f'Sleeping {seconds} second(s)...')
    time.sleep(seconds)
    #print('Done Sleeping')
    return 'Done Sleeping...'
 with concurrent.futures.ThreadPoolExecutor() as executor:
    f1 = executor.submit(do_something, 1)  #submit function will schedule the execution of function and returns
    f2 = executor.submit(do_something, 1)
    print(f1.result())
    print(f2.result())


finish = time.perf_counter()
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Done Sleeping...
Done Sleeping...
Finished in 1.01 seconds
```

In [43]:

```
 #TASK-11
 import concurrent.futures
 import time
 start = time.perf_counter()
def do_something(seconds):
    print(f'Sleeping {seconds} second(s)...')
    time.sleep(seconds)
    return 'Done Sleeping...'
 with concurrent.futures.ThreadPoolExecutor() as executor:
    results = [executor.submit(do_something, 1) for _ in range(10)] #list comprehension, alternative to loop
    for f in concurrent.futures.as_completed(results):
        print(f.result())


finish = time.perf_counter()
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Sleeping 1 second(s)...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Done Sleeping...
Finished in 1.01 seconds
```

In [44]:

```python
 #TASK-12
 import concurrent.futures
 import time
 start = time.perf_counter()
def do_something(seconds):
     print(f'Sleeping {seconds} second(s)...')
     time.sleep(seconds)
     return f'Done Sleeping...{seconds}'
 with concurrent.futures.ThreadPoolExecutor() as executor:
     s = [5,4,3,2,1] #different sleeping time for threads
     results = [executor.submit(do_something, s) for sec in s] #list comprehension, alternative to loop
     for f in concurrent.futures.as_completed(results):
         print(f.result())


finish = time.perf_counter()
 print(f'Finished in {round(finish-start,2)} seconds')
```

```
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
Sleeping [5, 4, 3, 2, 1] second(s)...
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[44], line 13
     11     results = [executor.submit(do_something, s) for sec in s] #list comprehension, alternative to loop
     12     for f in concurrent.futures.as_completed(results):
---> 13         print(f.result())
     16 finish = time.perf_counter()
     17 print(f'Finished in {round(finish-start,2)} seconds')

File C:\ProgramData\anaconda3\Lib\concurrent\futures\_base.py:449, in Future.result(self, timeout)
    447     raise CancelledError()
    448 elif self._state == FINISHED:
--> 449     return self.__get_result()
    451 self._condition.wait(timeout)
    453 if self._state in [CANCELLED, CANCELLED_AND_NOTIFIED]:

File C:\ProgramData\anaconda3\Lib\concurrent\futures\_base.py:401, in Future.__get_result(self)
    399 if self._exception is not None:
    400     try:
--> 401         raise self._exception
    402     finally:
    403         # Break a reference cycle with the exception in self._exception
    404         self = None

File C:\ProgramData\anaconda3\Lib\concurrent\futures\thread.py:59, in _WorkItem.run(self)
     56     return
     58 try:
---> 59     result = self.fn(*self.args, **self.kwargs)
     60 except BaseException as exc:
     61     self.future.set_exception(exc)

Cell In[44], line 7, in do_something(seconds)
      5 def do_something(seconds):
      6     print(f'Sleeping {seconds} second(s)...')
----> 7     time.sleep(seconds)
      8     return f'Done Sleeping...{seconds}'

TypeError: 'list' object cannot be interpreted as an integer
```

In [45]:
```
#TASK-12 (CORRECTED CODE)
import concurrent.futures
import time
start = time.perf_counter()
def do_something(seconds):
   print(f'Sleeping {seconds} second(s)...')
   time.sleep(seconds)
   return f'Done Sleeping...{seconds}'
with concurrent.futures.ThreadPoolExecutor() as executor:
   s = [5, 4, 3, 2, 1]  # different sleeping time for threads
   results = [executor.submit(do_something, sec) for sec in s]
   for f in concurrent.futures.as_completed(results):
       print(f.result())
finish = time.perf_counter()
print(f'Finished in {round(finish - start, 2)} seconds')
```

```
Sleeping 5 second(s)...
Sleeping 4 second(s)...
Sleeping 3 second(s)...
Sleeping 2 second(s)...
Sleeping 1 second(s)...
Done Sleeping...1
Done Sleeping...2
Done Sleeping...3
Done Sleeping...4
Done Sleeping...5
Finished in 5.01 seconds
```

In [50]:
```
#TASK-13
import threading
import time

import random
def print_names():
    for name in ('John', 'Mark', 'Elon', 'Callahan'):
        print (name)
        time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
    for _ in range(4):
        print(random.randint(20,50))
        time.sleep(random.uniform(0.5,1.5))
print_names()
print_ages()
 # without using thread concept, simply calling the function
```

```
John
Mark
Elon
Callahan
26
30
43
23
```

In [52]:
```
#TASK-14
import threading
import time
import random
def print_names():
   for name in ('John', 'Mark', 'Elon', 'Callahan'):
       print (name)
       time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
   for _ in range(4):
       print(random.randint(20,50))
       time.sleep(random.uniform(0.5,1.5))

t1 = threading.Thread(target = print_names)
 t2 = threading.Thread(target = print_ages)
 # The above threads are not doing anything yet. To do that, use start()
 t1.start()
 t2.start()
 # t1.join()
 # t2.join()
 # JOIN() NOT used - So, it will not make sure that the threads complete before moving on to the next part of t
```

```
John
49
Mark
42
23
Elon
Callahan
44
```

In [53]:
```
#TASK-15
import threading
import time
import random
def print_names():
   for name in ('John', 'Mark', 'Elon', 'Callahan'):
       print (name)
       time.sleep(random.uniform ( 0.5, 1.5))
def print_ages():
   for _ in range(4):
       print(random.randint(20,50))
       time.sleep(random.uniform(0.5,1.5))
t1 = threading.Thread(target = print_names)
t2 = threading.Thread(target = print_ages)
# The above threads are not doing anything yet. To do that, use start()
t1.start()
t2.start()
t1.join()
t2.join()
# JOIN() is used - It makes sure that the threads complete before moving on to the next part of the code
```

```
John
47
Mark
48
Elon
27
Callahan
29
```

In [57]:
```
#TASK-16
import threading
import requests
from pathlib import Path
# Create the Downloads directory if it doesn't exist
Path("Downloads").mkdir(exist_ok=True)
def download_file(url, filename):
   print(f'Downloading {url} to {filename}')
   try:
       response = requests.get(url)
       response.raise_for_status() # Raise an exception for bad status codes (4xx or 5xx)
       Path(filename).write_bytes(response.content)
       print(f'Finished Downloading {filename}')
   except requests.exceptions.RequestException as e:
       print(f'Error downloading {url}: {e}')
# Replace these URLs with your GitHub raw file URLs
urls = [
   'https://github.com/DheviSri/python-lab/blob/main/Lab_Activity_10_team_devishree.pdf',

'https://github.com/DheviSri/python-lab/blob/main/lab7_activity.ipynb',
]
threads = []
for url in urls:
# Use the last part of the URL as the filename
    filename = Path("Downloads") / url.split("/")[-1]
t = threading.Thread(target=download_file, args=(url, filename))
t.start()
threads.append(t)
# Wait for all threads to complete
[t.join() for t in threads]
print("All downloads complete.")
```

```
Downloading https://github.com/DheviSri/python-lab/blob/main/lab7_activity.ipynb to Downloads\lab7_activity.ipyn
b
Finished Downloading Downloads\lab7_activity.ipynb
All downloads complete.
```

In [60]:
```
pip install nbconvert
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: nbconvert in c:\programdata\anaconda3\lib\site-packages (7.16.6)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (4.
12.3)
Requirement already satisfied: bleach!=5.0.0 in c:\programdata\anaconda3\lib\site-packages (from bleach[css]!=5.
0.0->nbconvert) (6.2.0)
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (3.1.6
)
Requirement already satisfied: jupyter-core>=4.7 in c:\programdata\anaconda3\lib\site-packages (from nbconvert)
(5.7.2)
Requirement already satisfied: jupyterlab-pygments in c:\programdata\anaconda3\lib\site-packages (from nbconvert
) (0.3.0)
Requirement already satisfied: markupsafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (3
.0.2)
Requirement already satisfied: mistune<4,>=2.0.3 in c:\programdata\anaconda3\lib\site-packages (from nbconvert)
(3.1.2)
Requirement already satisfied: nbclient>=0.5.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0
.10.2)
Requirement already satisfied: nbformat>=5.7 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.1
0.4)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (24.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconver
t) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (2
.19.1)
Requirement already satisfied: traitlets>=5.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.
14.3)
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-packages (from bleach!=5.0.0->b
leach[css]!=5.0.0->nbconvert) (0.5.1)
Requirement already satisfied: tinycss2<1.5,>=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from bleach[c
ss]!=5.0.0->nbconvert) (1.4.0)
Requirement already satisfied: platformdirs>=2.5 in c:\programdata\anaconda3\lib\site-packages (from jupyter-cor
e>=4.7->nbconvert) (4.3.7)
Requirement already satisfied: pywin32>=300 in c:\programdata\anaconda3\lib\site-packages (from jupyter-core>=4.
7->nbconvert) (308)
Requirement already satisfied: jupyter-client>=6.1.12 in c:\programdata\anaconda3\lib\site-packages (from nbclie
nt>=0.5.0->nbconvert) (8.6.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\programdata\anaconda3\lib\site-packages (from jupyte
r-client>=6.1.12->nbclient>=0.5.0->nbconvert) (2.9.0.post0)
Requirement already satisfied: pyzmq>=23.0 in c:\programdata\anaconda3\lib\site-packages (from jupyter-client>=6
.1.12->nbclient>=0.5.0->nbconvert) (26.2.0)
Requirement already satisfied: tornado>=6.2 in c:\programdata\anaconda3\lib\site-packages (from jupyter-client>=
6.1.12->nbclient>=0.5.0->nbconvert) (6.5.1)
Requirement already satisfied: fastjsonschema>=2.15 in c:\programdata\anaconda3\lib\site-packages (from nbformat
>=5.7->nbconvert) (2.20.0)
Requirement already satisfied: jsonschema>=2.6 in c:\programdata\anaconda3\lib\site-packages (from nbformat>=5.7
->nbconvert) (4.23.0)
Requirement already satisfied: attrs>=22.2.0 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.6
->nbformat>=5.7->nbconvert) (24.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\programdata\anaconda3\lib\site-package
s (from jsonschema>=2.6->nbformat>=5.7->nbconvert) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in c:\programdata\anaconda3\lib\site-packages (from jsonschem
a>=2.6->nbformat>=5.7->nbconvert) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\programdata\anaconda3\lib\site-packages (from jsonschema>=2.
6->nbformat>=5.7->nbconvert) (0.22.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8
.2->jupyter-client>=6.1.12->nbclient>=0.5.0->nbconvert) (1.17.0)
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsoup4-
>nbconvert) (2.5)
Note: you may need to restart the kernel to use updated packages.
```

In [61]:
```
conda install nbconvert
```

```
Note: you may need to restart the kernel to use updated packages.Jupyter detected...
3 channel Terms of Service accepted
Retrieving notices: done
```

```
EnvironmentNotWritableError: The current user does not have write permissions to the target environment.
  environment location: C:\ProgramData\anaconda3
```

In [ ]: