BASICS

PROGRAM

Definition - “**A program is a sequence of instructions written in programming languages**.”
Ex - Python , c++ etc

Example -
Google chrome > It is nothing but a .exe program where some instructions are written as to how the browser should work

PROCESS

Definition - “**A process is simply an instance of a program that is being executed**”

The process works along with the OS because it needs computer resources.
They can be **Shared Resources.**

Resources required >

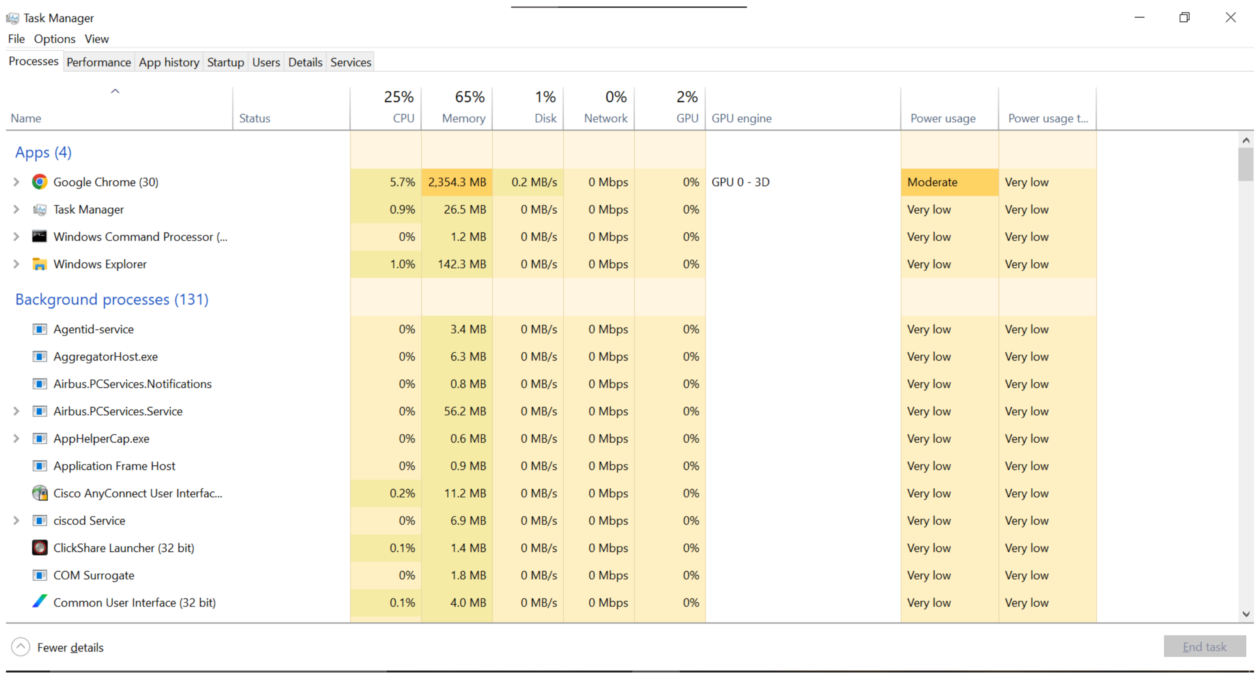
Code segment - The code / program will be available in this code segment.
Data Segment - It has a list of global and static variables.
Heap Memory - to dynamically allocate memory
Stack - make sure it has access to all local vars and function calls
Registers - smaller memory to store for a smaller period of time(temp memory)

Every process when we run it > It will have a separate memory space.
Because of this
>>**One process execution cannot corrupt another process.**
>> Increased execution time to switch between processes due to resource requirement.

Eg. Browser, Excel , Any App

PRACTICAL EXAMPLE

Open task manager to see list of processes running.


Task Manager

File Options View

Processes | Performance | App history | Startup | Users | Details | Services

Name	Status	25% CPU	65% Memory	1% Disk	0% Network	2% GPU	GPU engine	Power usage	Power usage t...
Apps (4)									
Google Chrome (30)		5.7%	2,354.3 MB	0.2 MB/s	0 Mbps	0%	GPU 0 - 3D	Moderate	Very low
Task Manager		0.9%	26.5 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Command Processor (...		0%	1.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Windows Explorer		1.0%	142.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Background processes (131)									
Agentid-service		0%	3.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AggregatorHost.exe		0%	6.3 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Airbus.PCServices.Notifications		0%	0.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Airbus.PCServices.Service		0%	56.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
AppHelperCap.exe		0%	0.6 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Application Frame Host		0%	0.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Cisco AnyConnect User Interfac...		0.2%	11.2 MB	0 MB/s	0 Mbps	0%		Very low	Very low
ciscod Service		0%	6.9 MB	0 MB/s	0 Mbps	0%		Very low	Very low
ClickShare Launcher (32 bit)		0.1%	1.4 MB	0 MB/s	0 Mbps	0%		Very low	Very low
COM Surrogate		0%	1.8 MB	0 MB/s	0 Mbps	0%		Very low	Very low
Common User Interface (32 bit)		0.1%	4.0 MB	0 MB/s	0 Mbps	0%		Very low	Very low

Fewer details

End task

THREAD

Definition - “**A thread is a unit of execution within a process**”

Eg : I have opened MS Paint application (This is Process)
Now If I create a rectangle box by clicking the option > I started a Thread
Now If I create a circle box by clicking the option > I started a Thread

A thread will have its own stack and registers but will share code segment, data segment & heap memory.

>> Single Threaded process

- **Definition:** The program executes one thread (sequence of instructions) at a time.
- **How it works:** Tasks are done one after another, sequentially.
- **Example:** When you run a program that loads a file, processes it, and then prints a result, all those steps happen one by one.

>> Multithreaded Process -

Definition: The program runs multiple threads concurrently.

How it works: Multiple threads can run simultaneously or be interleaved, allowing the program to perform multiple tasks at once.

Example: A web browser downloading images, processing user input, and playing audio all at the same time.

# MULTITHREADING PRACTICAL IMPLEMENTATION

When to use Multithreading?

>> I/O bound tasks - Tasks that spends more time executing I/O operations(File operations etc)
>>Concurrent Execution

Eg.

```python
import threading
import time

def print_numbers():
    for i in range(5):
        time.sleep(2)
        print(f"Number:{i}")

def print_letters():
    for letter in "abcde":
        time.sleep(2)
        print(f"Letters : {letter}")

#create 2 threads
t1=threading.Thread(target=print_numbers)
t2=threading.Thread(target=print_letters)



t = time.time()

#start thread
t1.start()
t2.start()
```

```
# wait for threads to complete
t1.join()
t2.join()

f_t = time.time() - t
print(f_t)


Output:
Letters : aNumber:0

Number:1Letters : b

Letters : cNumber:2

Letters : dNumber:3

Letters : eNumber:4

10.054944515228271
```

MULTIPROCESSING & ITS IMPLEMENTATION

What is Multiprocessing?

Allows to run processes in parallel.

When to use?

>>CPU Bound tasks - Heavy operations like math ops or data processing.
>>Parallel Execution in such a way that I use multiple cores of the CPU

```python
import multiprocessing
import time

def square_numbers():
    for i in range(5):
        time.sleep(1)
        print(f"Square :{i*i}")

def cube_numbers():
    for i in range(5):
        time.sleep(1.5)
        print(f"Cube :{i*i*i}")

if __name__ == "__main__":
    #Create two processes
    p1 = multiprocessing.Process(target = square_numbers)
    p2 = multiprocessing.Process(target = cube_numbers)

    t = time.time()
    #start the process
    p1.start()
    p2.start()

    # to join the main process
    p1.join()
    p2.join()

    f_t = time.time() - t
    print(f_t)
```

THREAD POOL EXECUTER

Python
```
from concurrent.futures import ThreadPoolExecutor
import time
```

```python
def print_numbers(number):
    time.sleep(1)
    return f"Number:{number}"

numbers = [1,2,3,4,5,6]

with ThreadPoolExecutor(max_workers=3) as executor:
    results = executor.map(print_numbers,numbers)

for result in results:
    print(result)
```

✅ Explanation:

1. `print_numbers` function

python
CopyEdit
```python
def print_numbers(number):
    time.sleep(1)
    return f"Number:{number}"
```

- Simulates a time-consuming task by pausing for 1 second.
- Returns a string like `"Number:3"`.

2. ThreadPoolExecutor

python
CopyEdit
```python
with ThreadPoolExecutor(max_workers=3) as executor:
    results = executor.map(print_numbers, numbers)
```

- Creates a **thread pool** with a maximum of 3 threads.

- `executor.map()` schedules `print_numbers(number)` to run **concurrently** for each number in `numbers`.
- Since there are 6 numbers and 3 workers, it runs **3 at a time**, then the next 3.

3. Output Loop

python

CopyEdit

```
for result in results:
    print(result)
```

- Prints the results **in the order of the input list**, not the order of completion.

⏱️ Runtime Insight:

- Without threads: 6 numbers × 1 second = **~6 seconds**.
- With 3 threads: it runs in **~2 seconds** (3 tasks in parallel per second).

Similar Topic : ProcessPoolExecutor

WEB SCRAPING USECASE

Python
```
import threading
import bs4
import requests

from bs4 import BeautifulSoup

urls = [

    "https://www.example.com",
```

```python
]

def fetch_contents(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.content,'html.parser')
    print(f'Fetched : {len(soup.text)} characters from {url} ')

threads = []


for url in urls:
    thread = threading.Thread(target =fetch_contents,args=(url,))
    threads.append(thread)
    thread.start()

for thread in threads:
    thread.join()

print("All web pages fetched")
```

REAL LIFE EXAMPLE - CPU BOUND TASKS