Platform & Architecture Information Module

Below User defined module contains below methods as

CPU_Info_OS()	: Displays information of CPU depending on OS
Platform_Info()	: Display information of Platform (Operating System)
Boot_Info()	: Display boot time of machine
CPU_Info()	: Display all information of CPU
RAM_Usage()	: Display information of RAM usage
Disk_Info()	: Display information of Hard disk

```python
import psutil
import platform
from os import *;
from datetime import datetime

def CPU_Info_OS():
    print("---- Marvellous Infosystems CPU Info OS ----")
    if platform.system() == 'Windows':
        return platform.processor()
    elif platform.system() == 'Darwin':
        command = '/usr/sbin/sysctl -n machdep.cpu.brand_string'
        return popen(command).read().strip()
    elif platform.system() == 'Linux':
        command = 'cat /proc/cpuinfo'
        return popen(command).read().strip()
    return 'platform not identified'

def get_size(bytes, suffix="B"):
    factor = 1024
    for unit in ["", "K", "M", "G", "T", "P"]:
        if bytes < factor:
            return f"{bytes:.2f}{unit}{suffix}"
        bytes /= factor

def Platform_Info():
    print("---- Marvellous Infosystems System Information ----")
    uname = platform.uname()
    print(f"System: {uname.system}")
    print(f"Node Name: {uname.node}")
    print(f"Release: {uname.release}")
    print(f"Version: {uname.version}")
    print(f"Machine: {uname.machine}")
    print(f"Processor: {uname.processor}")

```

```
def Boot_Info():
    print("---- Marvellous Infosystems Boot Time ----")
    boot_time_timestamp = psutil.boot_time()
    bt = datetime.fromtimestamp(boot_time_timestamp)
    print(f"Boot Time: {bt.year}/{bt.month}/{bt.day} {bt.hour}:
        {bt.minute}:{bt.second}")

def CPU_Info():
    print("---- Marvellous Infosystems CPU Info ----")
    print("Physical cores:", psutil.cpu_count(logical=False))
    print("Total cores:", psutil.cpu_count(logical=True))

    cpufreq = psutil.cpu_freq()
    print(f"Max Frequency: {cpufreq.max:.2f}Mhz")
    print(f"Min Frequency: {cpufreq.min:.2f}Mhz")
    print(f"Current Frequency: {cpufreq.current:.2f}Mhz")

    print("CPU Usage Per Core:")
    for i, percentage in enumerate(psutil.cpu_percent(percpu=True)):
        print(f"Core {i}: {percentage}%")
    print(f"Total CPU Usage: {psutil.cpu_percent()}%")

def RAM_Usage():
    print("---- Marvellous Infosystems Memory Information ----")

    svmem = psutil.virtual_memory()
    print(f"Total: {get_size(svmem.total)}")
    print(f"Available: {get_size(svmem.available)}")
    print(f"Used: {get_size(svmem.used)}")
    print(f"Percentage: {svmem.percent}%")
    print("----SWAP----")

    swap = psutil.swap_memory()
    print(f"Total: {get_size(swap.total)}")
    print(f"Free: {get_size(swap.free)}")
    print(f"Used: {get_size(swap.used)}")
    print(f"Percentage: {swap.percent}%")

def Disk_Info():
    print("---- Marvellous Infosystems Disk Information ----")
    print("Partitions and Usage:")

    partitions = psutil.disk_partitions()
    for partition in partitions:
        print(f"=== Device: {partition.device} ===")
        print(f"  Mountpoint: {partition.mountpoint}")
        print(f"  File system type: {partition.fstype}")
        try:
            partition_usage = psutil.disk_usage(partition.mountpoint)
        except PermissionError:
            continue

    print(f"  Total Size: {get_size(partition_usage.total)}")
    print(f"  Used: {get_size(partition_usage.used)}")
    print(f"  Free: {get_size(partition_usage.free)}")
    print(f"  Percentage: {partition_usage.percent}%")
    disk_io = psutil.disk_io_counters()
    print(f"Total read: {get_size(disk_io.read_bytes)}")
    print(f"Total write: {get_size(disk_io.write_bytes)}")
```