Script for Monitoring Server

Examples :

CPU

```
>>> import psutil
>>> # blocking
>>> psutil.cpu_percent(interval=1)
2.0
>>> # non-blocking (percentage since last call)
>>> psutil.cpu_percent(interval=None)
2.9
>>> # blocking, per-cpu
>>> psutil.cpu_percent(interval=1, percpu=True)
[2.0, 1.0]
>>>
```

```
>>> import psutil
>>> psutil.cpu_count()
4
>>> psutil.cpu_count(logical=False)
2
```

```
>>> len(psutil.Process().cpu_affinity())
1
```

```
>>> import psutil
>>> psutil.cpu_stats()
scpustats(ctx_switches=20455687, interrupts=6598984, soft_interrupts=2134212, syscalls=0)
```

```
>>> import psutil
>>> psutil.cpu_freq()
scpufreq(current=931.42925, min=800.0, max=3500.0)
>>> psutil.cpu_freq(percpu=True)
[scpufreq(current=2394.945, min=800.0, max=3500.0),
 scpufreq(current=2236.812, min=800.0, max=3500.0),
 scpufreq(current=1703.609, min=800.0, max=3500.0),
 scpufreq(current=1754.289, min=800.0, max=3500.0)]
```

```
>>> import psutil
>>> psutil.getloadavg()
(3.14, 3.89, 4.67)
>>> psutil.cpu_count()
10
>>> # percentage representation
>>> [x / psutil.cpu_count() * 100 for x in psutil.getloadavg()]
[31.4, 38.9, 46.7]
```

Memory

```
>>> import psutil
>>> mem = psutil.virtual_memory()
>>> mem
svmem(total=10367352832, available=6472179712, percent=37.6, used=8186245120, free=2181107712,
active=4748992512, inactive=2758115328, buffers=790724608, cached=3500347392, shared=787554304,
slab=199348224)
>>>
>>> THRESHOLD = 100 * 1024 * 1024  # 100MB
>>> if mem.available <= THRESHOLD:
...     print("warning")
...
>>>
```

swap_memory

```
>>> import psutil
>>> psutil.swap_memory()
sswap(total=2097147904L, used=886620160L, free=1210527744L, percent=42.3, sin=1050411008,
sout=1906720768)
```

Disks

```
>>> import psutil
>>> psutil.disk_partitions()
[sdiskpart(device='/dev/sda3', mountpoint='/', fstype='ext4', opts='rw,errors=remount-ro', maxfile=255,
maxpath=4096),
 sdiskpart(device='/dev/sda7', mountpoint='/home', fstype='ext4', opts='rw', maxfile=255, maxpath=4096)]
```

```
>>> import psutil
>>> psutil.disk_usage('/')
sdiskusage(total=21378641920, used=4809781248, free=15482871808, percent=22.5)
```

CPU and System Load

```
import os
import subprocess
import re

statistics = {}

# Get Physical and Logical CPU Count
physical_and_logical_cpu_count = os.cpu_count()
statistics['physical_and_logical_cpu_count'] = physical_and_logical_cpu_count
"""
# Load average
# This is the average system load calculated over a given period of time of 1, 5 and 15 minutes.
# In our case, we will show the load average over a period of 15 minutes.

# The numbers returned by os.getloadavg() only make sense if
# related to the number of CPU cores installed on the system.

# Here we are converting the load average into percentage. The higher the percentage the higher the load
"""

cpu_load = [x / os.cpu_count() * 100 for x in os.getloadavg()][-1]
statistics['cpu_load'] = cpu_load
```

Memory(RAM) usage.

```
 import subprocess
import re


statistics = {}
matcher = re.compile('\d+')


# Memory usage
total_ram = subprocess.run(['sysctl', 'hw.memsize'],
stdout=subprocess.PIPE).stdout.decode('utf-8')
vm = subprocess.Popen(['vm_stat'],
stdout=subprocess.PIPE).communicate()[0].decode('utf-8')
vmLines = vm.split('\n')


wired_memory = (int(matcher.search(vmLines[6]).group()) * 4096) / 1024 ** 3
free_memory = (int(matcher.search(vmLines[1]).group()) * 4096) / 1024 ** 3
active_memory = (int(matcher.search(vmLines[2]).group()) * 4096) / 1024 ** 3
inactive_memory = (int(matcher.search(vmLines[3]).group()) * 4096) / 1024 ** 3


# Used memory = wired_memory + inactive + active


statistics['ram'] = dict({
    'total_ram': int(matcher.search(total_ram).group())/1024**3,
    'used_ram': round(wired_memory+active_memory+inactive_memory, 2),
})
```

Disk Usage.

```
import shutil
import subprocess
 statistics = {}
 # Top command on mac displays and updates sorted information about processes.
 top_command = subprocess.run(['top', '-l 1', '-n 0'],
stdout=subprocess.PIPE).stdout.decode('utf-8').split('\n')
 # Disk usage
# Get total disk size, used disk space, and free disk
 total, used, free = shutil.disk_usage("/")
 # Number of Read and write operations
# from the top command, the read written result will be as follows
# 'Disks: XXXXXX/xxG read, XXXX/xxG written.'
# we thus need to extract the read and written from this.
read_written = top_command[9].split(':')[1].split(',')
read = read_written[0].split(' ')[1]
written = read_written[1].split(' ')[1]
 statistics['disk'] = dict(
   {
      'total_disk_space': round(total / 1024 ** 3, 1),
      'used_disk_space': round(used / 1024 ** 3, 1),
      'free_disk_space': round(free / 1024 ** 3, 1),
      'read_write': {
         'read': read,
         'written': written
      }
   }
)
```

Combined into One File

```
import os
import subprocess
import re
import shutil


 def get_statistics():
    statistics = {}
    matcher = re.compile('\d+')
    # Top command on mac displays and updates sorted information about processes.


    top_command = subprocess.run(['top', '-l 1', '-n 0'],
stdout=subprocess.PIPE).stdout.decode('utf-8').split('\n')


    # Get Physical and Logical CPU Count
    physical_and_logical_cpu_count = os.cpu_count()
    statistics['physical_and_logical_cpu_count'] = physical_and_logical_cpu_count
    """
    # Load average
    # This is the average system load calculated over a given period of time of 1, 5 and 15
minutes.
    # In our case, we will show the load average over a period of 15 minutes.


    # The numbers returned by os.getloadavg() only make sense if
    # related to the number of CPU cores installed on the system.
```

```
 # Here we are converting the load average into percentage. The higher the percentage the higher the load
    """
     cpu_load = [x / os.cpu_count() * 100 for x in os.getloadavg()][-1]
    statistics['cpu_load'] = round(cpu_load)


    # Memory usage
    total_ram = subprocess.run(['sysctl', 'hw.memsize'], stdout=subprocess.PIPE).stdout.decode('utf-8')
    vm = subprocess.Popen(['vm_stat'], stdout=subprocess.PIPE).communicate()[0].decode('utf-8')
    vmLines = vm.split('\n')


    wired_memory = (int(matcher.search(vmLines[6]).group()) * 4096) / 1024 ** 3
    free_memory = (int(matcher.search(vmLines[1]).group()) * 4096) / 1024 ** 3
    active_memory = (int(matcher.search(vmLines[2]).group()) * 4096) / 1024 ** 3
    inactive_memory = (int(matcher.search(vmLines[3]).group()) * 4096) / 1024 ** 3


    # Used memory = wired_memory + inactive + active


    statistics['ram'] = dict({
        'total_ram': int(matcher.search(total_ram).group()) / 1024 ** 3,
        'used_ram': round(wired_memory + active_memory + inactive_memory, 2),
    })
```

```
    # Disk usage
    # Get total disk size, used disk space, and free disk

    total, used, free = shutil.disk_usage("/")
     # Number of Read and write operations
    # from the top command, the read written result will be as follows
    # 'Disks: XXXXXX/xxG read, XXXX/xxG written.'
    # we thus need to extract the read and written from this.
    read_written = top_command[9].split(':')[1].split(',')
    read = read_written[0].split(' ')[1]
    written = read_written[1].split(' ')[1]

    statistics['disk'] = dict(
       {
          'total_disk_space': round(total / 1024 ** 3, 1),
          'used_disk_space': round(used / 1024 ** 3, 1),
          'free_disk_space': round(free / 1024 ** 3, 1),
          'read_write': {
             'read': read,
             'written': written
          }
       }
    )
     return statistics

statistics = get_statistics()
```