CIS 215 - Final project - Assignment 2

Final Project:
Live Dashboard for server stats at Starship Technologies

Team:
Only Domenico Galati due to limitations of being able to share info because of my NDA with Starship Technologies

Approach:
we have an internally designed Snowflake server in every office and it includes a nice little body with a screen that doesn't show anything, I reached out to our platform team and got approval to write a python program to get basic data printed to the screen to make use of it. Since ~80% of the server is already written in python, no new installations were needed, it was a simple plug and play addition of my program. I divided the program into five 'chunks', CPU usage, Memory usage, Storage usage, Network status, and Access Point usage. By splitting it into these chunks, it made getting incrementally more data back quite easy and left room for tons of improvement in the future.

In CPU usage, it simply will report the amount of cores in the system, and the current utilization percentage and then display a bar for a visual reference. In Memory usage, the program will return total RAM in the system, percentage of how much is being used, an d an accompanying bar just like the CPU usage for visual reference. The storage usage section is relatively similar to the Memory usage section, it will report total storage in system, percentage used along with another bar for visualization, and amount of space free. The Network status section will have two network interfaces listed, 'eno1.10' and 'eno1.20'. eno1.10 is the interface that ports all traffic outward to the real world and eno1.20 is the interface that handles all network trafficker internal to the private network. For each interface the program will show the name, if the interface is enabled or not, and the uplink speed of that interface. The access point usage section will differ from the previous four sections as it is not physically part of the server system. Each office has a different amount of access points along with devices that will connect to the server, IP addresses of each server, uptime, and a little bit more info that will only make sense for people internal to the company.

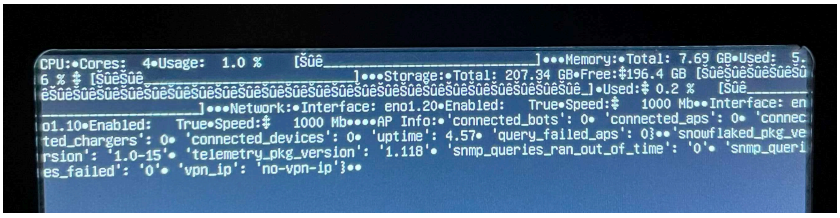
With all five of these different sections, this program should be very malleable for future updating and upkeep as the company scales, The other convenient piece of the program is all of the system stats are pulled using a library called 'plutil' which also means this program can be ran on any kind of device weather it be Mac, linux, or windows, all OSes can report their info with the

Final Product:

Once the final program was completed, the output was exactly as previously described, Each section outputs nicely organized to the console without issue as shown:

```
ubuntu@snowflake-nau:~$ python3 /opt/snowflake-telemetry/disp.py
CPU:
Cores:  4
Usage:  17.9 %    [██████____________________]


Memory:
Total: 7.69 GB
Used:  8.1 %     [███_______________________]


Storage:
Total: 207.34 GB
Free:   196.4 GB [████████████████████████████_]
Used:    0.2 %   [█___________________________]


Network:
Interface: eno1.20
Enabled:   True
Speed:     1000 Mb

Interface: eno1.10
Enabled:   True
Speed:     1000 Mb



AP Info:
connected_devices: 79
connected_aps: 4
query_failed_aps: 0
connected_chargers: 75
connected_bots: 0
uptime: 41.33 Hours
telemetry_pkg_version:1.118
snowflaked_pkg_version:1.0-15
snmp_queries_ran_out_of_time:0
snmp_queries_failed:0
```

The bigger issue I've had with the implementation of the program is that when printed to the visual console that contains the display screen, it doesn't show properly, as seen below



Raw Code:

```
#Import
import sys
import os
import psutil
import time

def __init__(self):
    print()

def printBar(remainder, total):
    print('[' , end ='')
    while total>=0:
        if remainder >= 0:
            print("", end ="")
            remainder = remainder -1
        else:
            print('_', end ="")
        total = total-1
    print(']')

def memUsage():
    print("Memory:")
    memory = psutil.virtual_memory()
    print("Total:", round((memory.total)/ (2**30), 2), "GB")
    print("Used: ", memory.percent, end =" % \t ")
    printBar(memory.percent/3.5, 28.57)
    print("\n")

def storageUsage():
    print("Storage:")
    SSD = psutil.disk_usage('/mnt/meta')
    print ("Total: " +str(round(SSD.total / (2**30),2)), "GB")
    print ("Free:\t" +str(round(SSD.free / (2**30),2)), end =" GB ")
    remainder = SSD.free/ (2**32.85)
    total = SSD.total/ (2**32.85)
    printBar(remainder, total)
    print ("Used:\t " +str(round(SSD.percent,2)), end =" %   ")
    printBar(SSD.percent/3.5, 28.57)
    print("\n")

def cpuUsage():
    print("CPU:")
    CPU = psutil.cpu_count()
    print("Cores: ", CPU)
    CPU = psutil.cpu_percent(.5)
    print("Usage: ", CPU, "%", end="    ")
    printBar(CPU/3.5, 28.57)
    print("\n")

def networkUsage():
    print("Network:")
    network = psutil.net_if_stats()
    for type in network:
        try: #Can be finalized later and try/except can be removed
            if type == 'eno1.10' or type == 'eno1.20':
                print('Interface:',type)
                print("Enabled:  ",network[type].isup)
                print("Speed:\t  ", network[type].speed, "Mb\n")
        except NameError:
            print("Missing 1 or both interfaces")
```

```
            break
    network = psutil.net_io_counters() #Maybe useful for data rates?
    print("\n")

def apUsage():
    data = open( '/tmp/snowflake-last-telemetry-for-dom')
    temp = True
    nums = ''
    stats = ''
    for tup in data:
        if temp == True:
            nums = tup
        else:
            stats = tup
        temp = False
    nums = nums.replace("{", "")
    nums = nums.replace("'", "")
    nums = nums.replace("}", "")
    nums = nums.replace(" ", "")
    nums = nums.split(',')
    print("AP Info:")
    for point in nums:
        key, pair = point.split(':')
        if(key == "uptime"):
            print(key, end =": ")
            pair = int(float(pair))/60
            pair= pair/60
            print(round(pair,2), "Hours")
        else:
            print(key, end =": ")
            print(pair)
    stats = stats.replace("{", "")
    stats = stats.replace(" ", "")
    stats = stats.replace("}", "")
    stats = stats.replace("'", "")
    stats = stats.split(',')
    for point in stats:
        key, pair = point.split(':')
        print(key, end =":")
        print(pair)

def main():
    while(True):
        cpuUsage()
        memUsage()
        storageUsage()
        networkUsage()
        apUsage()
        time.sleep(60)

if __name__ == "__main__":
    main()
```