PYTHON PROJECT:
OS Info

Name: Muhd Fazil Istamar

Class: Centre for Cybersecurity (Batch – CFC240722)

Trainer Name: James Lim

Python Script File Name: S11_Python_Proj.py

A. Objective of Project

Create automation to display the operating system information as listed below:

1. Display the OS version – if Windows, display the Windows details; if executed on Linux, display the Linux details.

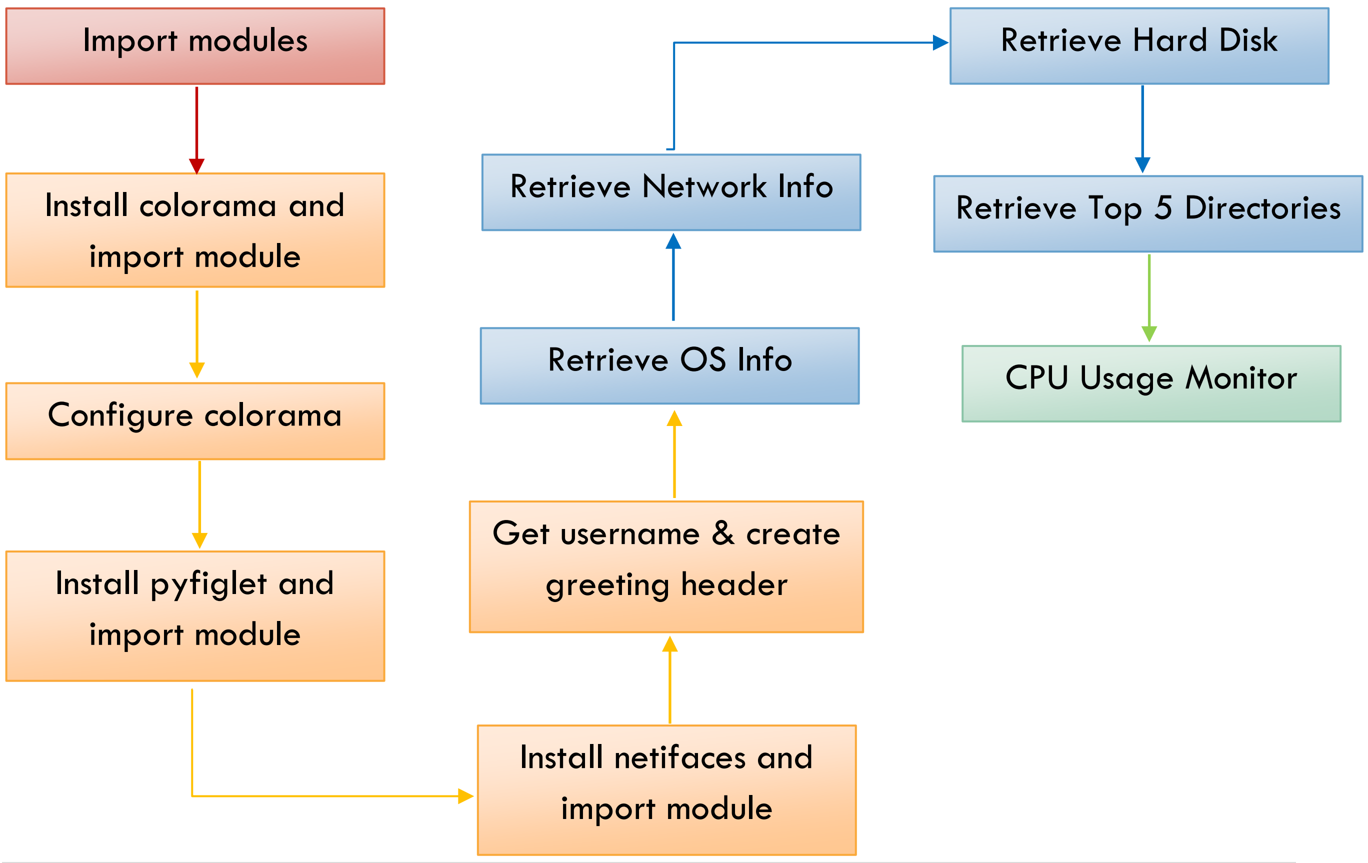
2. Display the private IP address, public IP address, and the default gateway.

3. Display the hard disk size; free and used space.

4. Display the top five (5) directories and their size.

5. Display the CPU usage; refresh every 10 seconds.

B. Project Script Flow

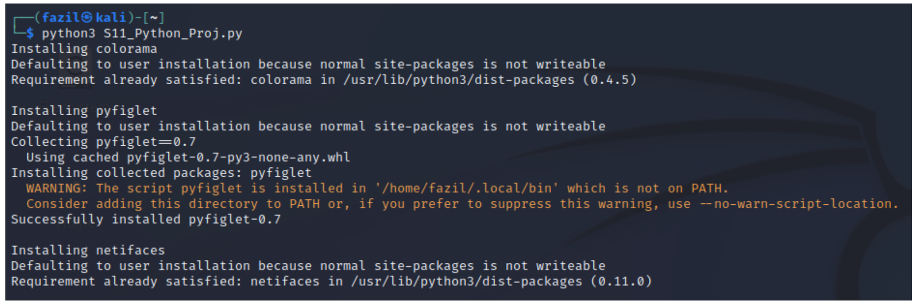

Import modules
Install colorama and import module
Configure colorama
Install pyfiglet and import module
Install netifaces and import module
Get username & create greeting header
Retrieve OS Info
Retrieve Network Info
Retrieve Hard Disk
Retrieve Top 5 Directories
CPU Usage Monitor

C. Python Script vs Output

Python Script

```python
# Import required modules
import os
import time
import psutil
import socket
import requests
import platform

# Install colorama and import module
print("Installing colorama")
os.system("python3 -m pip install colorama")
import colorama

# Configure colorama
from colorama import Fore, Back, Style
# Allow reset of font text colours when 'autoreset=True'
colorama.init(autoreset=True)
print()

# Install pyfiglet and import the module
print("Installing pyfiglet")
os.system("python3 -m pip install pyfiglet==0.7")
import pyfiglet
print()

# Install netifaces and import module
print("Installing netifaces")
os.system("python3 -m pip install netifaces")
import netifaces
print()
```

Output

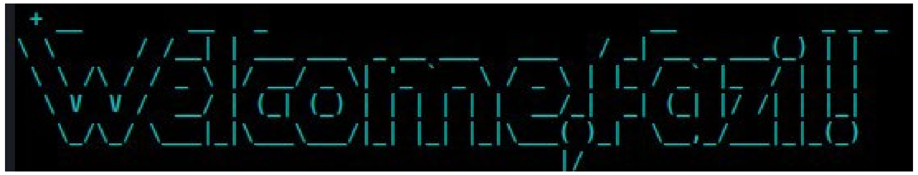


```
┌──(fazil㉿kali)-[~]
└─$ python3 S11_Python_Proj.py
Installing colorama
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: colorama in /usr/lib/python3/dist-packages (0.4.5)

Installing pyfiglet
Defaulting to user installation because normal site-packages is not writeable
Collecting pyfiglet==0.7
  Using cached pyfiglet-0.7-py3-none-any.whl
Installing collected packages: pyfiglet
  WARNING: The script pyfiglet is installed in '/home/fazil/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pyfiglet-0.7

Installing netifaces
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: netifaces in /usr/lib/python3/dist-packages (0.11.0)
```

Description

os.system("python3 -m pip install <package>")	*Install the latest version of package*
from colorama **import** Fore, Back, Style	*Eliminate the hassle of inputting colour code such as ' \033[31m 'for Red font colour.*
colorama.init(autoreset=**True**)	*Will allow reset of font colour to default setting (White) if {Fore.RESET} or {Back.RESET} is added prior to the text so that new font colour can be configured.*

Python Script	Output

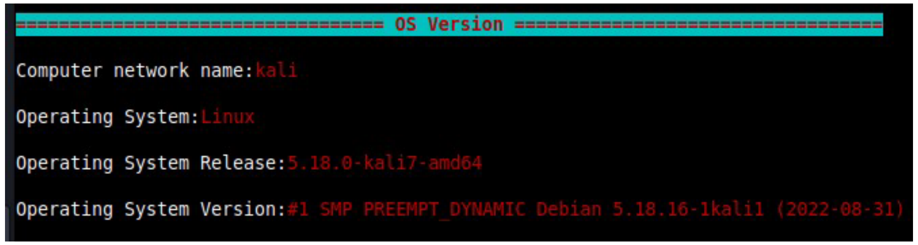
`# Retrieve username who login to the client.`
`# This username info will be used in greeting header for personal touch.`
`usr = os.getlogin()`
 	
`# Create a greetings header using pyfiglet`
`greet = pyfiglet.figlet_format( f"Welcome,{usr}!")`
`print(f"{Fore.CYAN}{Style.BRIGHT} + {greet}")`
`# Include a 3-second delay for each sections to allow user to read the contents ↵`
`before next set of information appears`
`time.sleep(3)`
	

Description	
greet = pyfiglet.figlet_format(f"Welcome,{usr}!")	*Use of f string command format to create a greeting in the format of Figlet (Linux) and address the user personally*
print(f"{Fore.CYAN}{Style.BRIGHT} + {greet}") -	*Use of f string command format to format the 'Welcome' text:* • *{Fore.CYAN} - foreground colour/ font colour = CYAN* • *{Style.BRIGHT} – Style of text display brighter than normal*

Python Script

```
print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}================================== OS
Version ==================================")
print()
# Modules required: colorama, platform, time

#Computer network name
print(f"Computer network name:{Fore.RED}{platform.node()}")
print()
#Operating System
print(f"Operating System:{Fore.RED}{platform.system()}")
print()
#Operating System Release
print(f"Operating System Release:{Fore.RED}{platform.release()}")
print()
#Operating System Version
print(f"Operating System Version:{Fore.RED}{platform.version()}")
print()
time.sleep(3)
```

Output

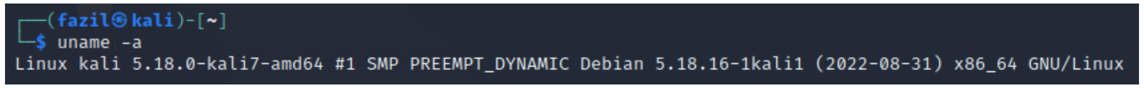


```
================================== OS Version ==================================

Computer network name:kali

Operating System:Linux

Operating System Release:5.18.0-kali7-amd64

Operating System Version:#1 SMP PREEMPT_DYNAMIC Debian 5.18.16-1kali1 (2022-08-31)
```

Description

{platform.node()}, {platform.system()}, {platform.release()}, {platform.version()},

These functions are similar in Linux by keying this command ‘uname -a’ and the details can be shown below:



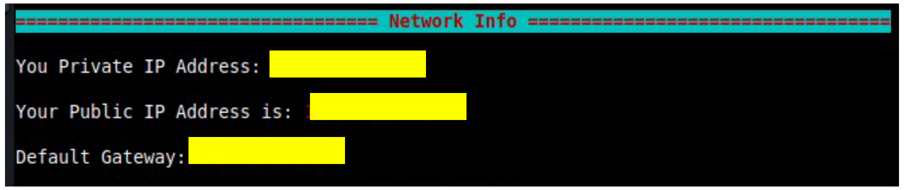
```
┌──(fazil㉿kali)-[~]
└─$ uname -a
Linux kali 5.18.0-kali7-amd64 #1 SMP PREEMPT_DYNAMIC Debian 5.18.16-1kali1 (2022-08-31) x86_64 GNU/Linux
```

It will display kernel name, hostname, kernel release, kernel version, hardware name and operating system.

Python Script

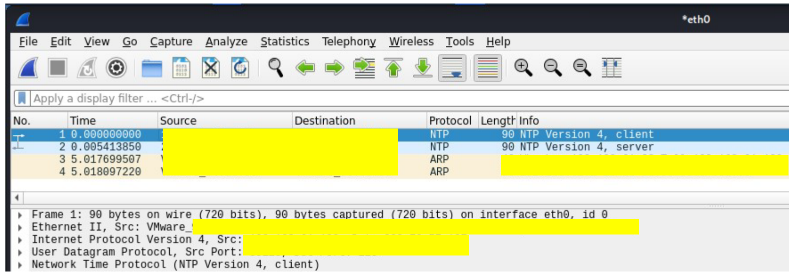
```
print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}================================== ↲
Network Info ==================================")
print()
# Modules required: colorama, netifaces, requests, socket, time

#Find Private / Internal IP Address
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8",80))
Int_IP = s.getsockname()[0]
print(f"You Private IP Address: {Fore.RED}{Int_IP}")
print()

#Find Public / External IP Address
b = requests.get("https://api.ipify.org?format=json")
Ext_IP = b.json()["ip"]

print(f"Your Public IP Address is: {Fore.RED}{Ext_IP}" )
print()

#Find Default Gateway
DG = netifaces.gateways()['default'][netifaces.AF_INET][0]
print(f"Default Gateway: {Fore.RED}{DG}")
print()
time.sleep(3)
```

Output



```
================================== Network Info ==================================

You Private IP Address: [redacted]

Your Public IP Address is: [redacted]

Default Gateway: [redacted]
```

Private IP Address (Wireshark)



*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time         Source      Destination   Protocol Length Info
  1 0.000000000   [redacted]                NTP          90 NTP Version 4, client
  2 0.005413850                             NTP          90 NTP Version 4, server
  3 5.017699507                             ARP             [redacted]
  4 5.018097220                             ARP

▸ Frame 1: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface eth0, id 0
▸ Ethernet II, Src: VMware_[redacted]
▸ Internet Protocol Version 4, Src: [redacted]
▸ User Datagram Protocol, Src Port:
▸ Network Time Protocol (NTP Version 4, client)
```

Description

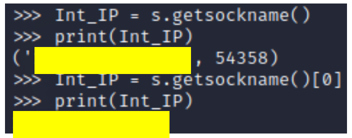
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

s.connect(("8.8.8.8",80))

Int_IP = s.getsockname()[0]

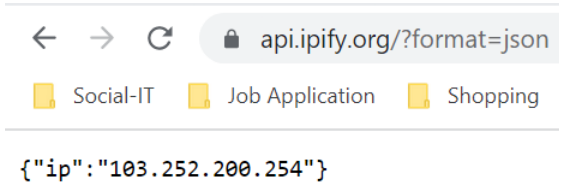
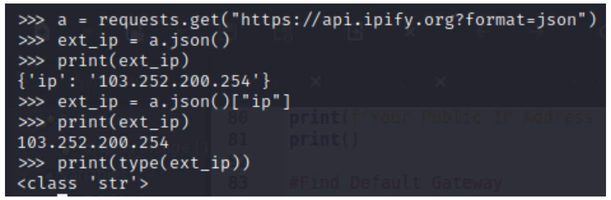
There are different types of sockets considered like communication endpoint will need to define the type. In this case, we want Internet socket (socket.AF_INET) and connection via UDP protocol (socket.SOCK_DGRAM).

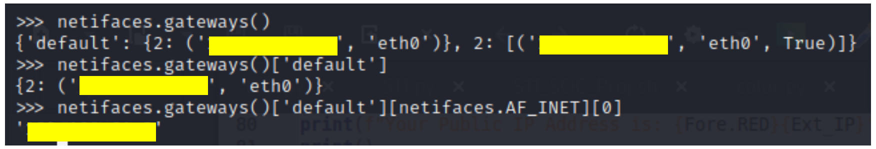
Attempt connection to public Google DNS server (8.8.8.8 port 80). In Wireshark, it seems that reverse ARP process was triggered to tell the host its IP address. 1st index [0] was chosen to retrieve the internal IP address of host.



```
>>> Int_IP = s.getsockname()
>>> print(Int_IP)
('[redacted]', 54358)
>>> Int_IP = s.getsockname()[0]
>>> print(Int_IP)
[redacted]
```

Description	

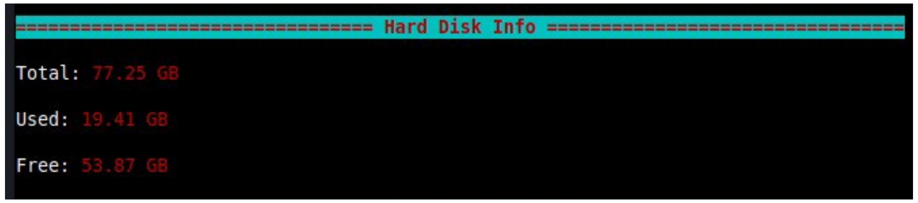
b = requests.get("https://api.ipify.org?format=json")
Ext_IP = b.json()["ip"]
	"ipify.org" is a website where public IP address can be retrieved in Plain text or JSON format or JSONP format. In this case, the public IP address was retrieved in JSON format which is 'printed' as a string.
	
	 api.ipify.org/?format=json
	Social-IT
	Job Application
	Shopping
	`{"ip":"103.252.200.254"}`
	
	`>>> a = requests.get("https://api.ipify.org?format=json")`
	`>>> ext_ip = a.json()`
	`>>> print(ext_ip)`
	`{'ip': '103.252.200.254'}`
	`>>> ext_ip = a.json()["ip"]`
	`>>> print(ext_ip)`
	`103.252.200.254`
	`>>> print(type(ext_ip))`
	`<class 'str'>`

DG = netifaces.gateways()['default'][netifaces.AF_INET][0]	*Below is the step-by step process of finding the default gateway in Python.*
	
	`>>> netifaces.gateways()`
	`{'default': {2: ('`[redacted]`', 'eth0')}, 2: [('`[redacted]`', 'eth0', True)]}`
	`>>> netifaces.gateways()['default']`
	`{2: ('`[redacted]`', 'eth0')}`
	`>>> netifaces.gateways()['default'][netifaces.AF_INET][0]`
	`'`[redacted]

Python Script

```python
print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}================================ Hard Disk Info ================================")
print()
# Modules required: colorama, psutil, time

# Variables for different requests of disk status
# Calculation = Total number of bytes ÷ (Size of kilobyte ^3)
hdd = psutil.disk_usage('/')
# TDS - Total Disk Space
TDS = hdd.total/1024**3
# UDS - Used Disk Space
UDS = hdd.used/1024**3
# FDS - Free Disk Space
FDS = hdd.free/1024**3

#Total HD Space
print (f'Total: {Fore.RED}{TDS:.2f} GB')
print()
#Used HD Space
print (f'Used: {Fore.RED}{UDS:.2f} GB')
print()
#Free HD Space
print (f'Free: {Fore.RED}{FDS:.2f} GB')
print()
time.sleep(3)
```

Output



```
=================================== Hard Disk Info ===================================

Total: 77.25 GB

Used: 19.41 GB

Free: 53.87 GB
```

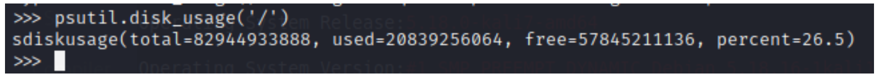
Description

hdd = psutil.disk_usage('/')

TDS = hdd.total/1024**3
UDS = hdd.used/1024**3

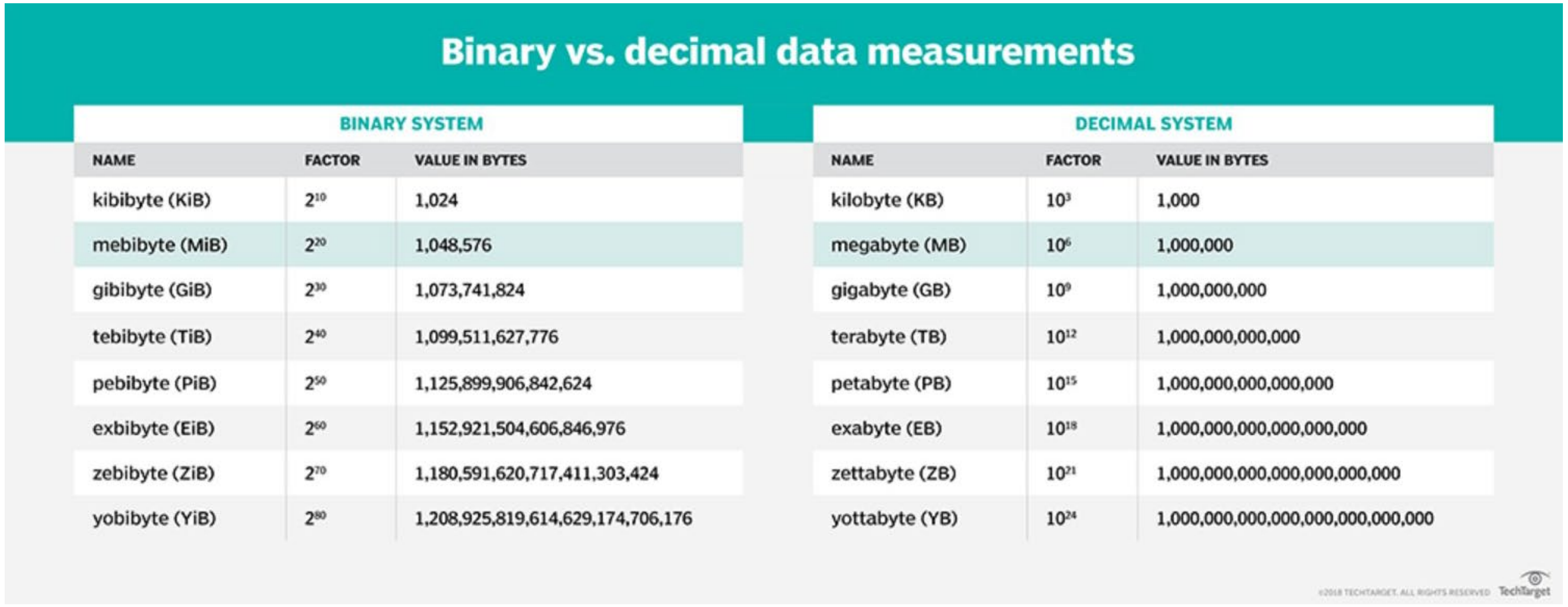
FDS = hdd.free/1024**3

psutil.disk_usage()returns a tuple that consists of total space, amount currently in use and the available remaining space as shown below:



```
>>> psutil.disk_usage('/')
sdiskusage(total=82944933888, used=20839256064, free=57845211136, percent=26.5)
>>> 
```

The values displayed are in bytes(b) and needs to be converted to gigabyte (GB) to express a short string of numbers which are more readable. As binary number system is the base of computer system which it is able to read, the total bytes are divided by 1024^3 = [Number of kilobytes in 1 Gigbyte].



Binary vs. decimal data measurements

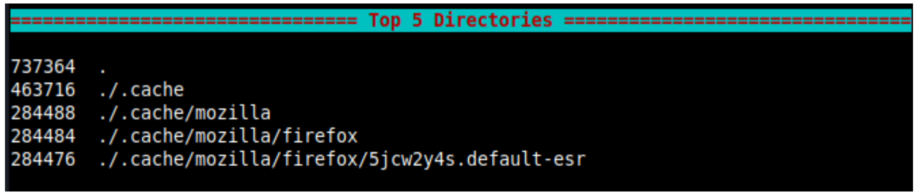
BINARY SYSTEM			DECIMAL SYSTEM		
NAME	FACTOR	VALUE IN BYTES	NAME	FACTOR	VALUE IN BYTES
kibibyte (KiB)	2^{10}	1,024	kilobyte (KB)	10^{3}	1,000
mebibyte (MiB)	2^{20}	1,048,576	megabyte (MB)	10^{6}	1,000,000
gibibyte (GiB)	2^{30}	1,073,741,824	gigabyte (GB)	10^{9}	1,000,000,000
tebibyte (TiB)	2^{40}	1,099,511,627,776	terabyte (TB)	10^{12}	1,000,000,000,000
pebibyte (PiB)	2^{50}	1,125,899,906,842,624	petabyte (PB)	10^{15}	1,000,000,000,000,000
exbibyte (EiB)	2^{60}	1,152,921,504,606,846,976	exabyte (EB)	10^{18}	1,000,000,000,000,000,000
zebibyte (ZiB)	2^{70}	1,180,591,620,717,411,303,424	zettabyte (ZB)	10^{21}	1,000,000,000,000,000,000,000
yobibyte (YiB)	2^{80}	1,208,925,819,614,629,174,706,176	yottabyte (YB)	10^{24}	1,000,000,000,000,000,000,000,000

(Source: https://www.techtarget.com/searchstorage/definition/gigabyte)

Python Script

```
print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}=============================== Top 5
Directories ===============================")
print()
# Modules required: colorama, os, time
os.system("du | sort -n -r| head -n 5")
print()
```

Output



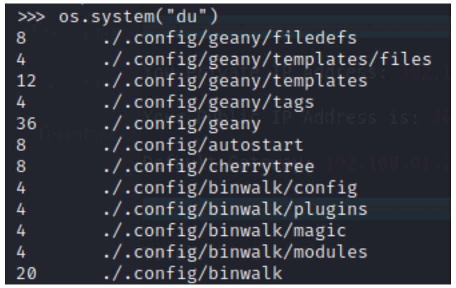
```
================================= Top 5 Directories =================================

737364  .
463716  ./.cache
284488  ./.cache/mozilla
284484  ./.cache/mozilla/firefox
284476  ./.cache/mozilla/firefox/5jcw2y4s.default-esr
```

Description

os.system("du -h | sort -n | head -n 5")

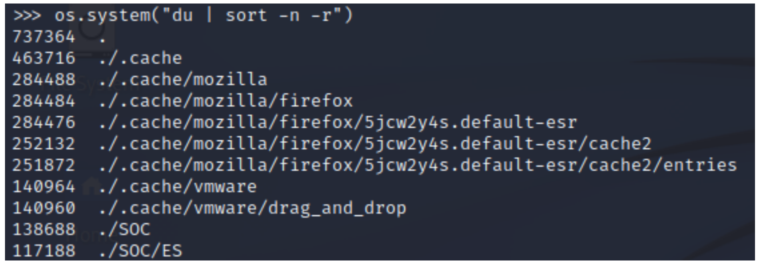
In order to run Linux external command from Python, os.system() command is used. 'du'(disk usage) is a Linux command to find current disk space occupied by files or directories. As it measures the current directory and all its sub-directories, this python script is run at '/home/<user>' due to access restriction at '/' (root directory).

1st: This command will list down all the directories with size in bytes (values only).

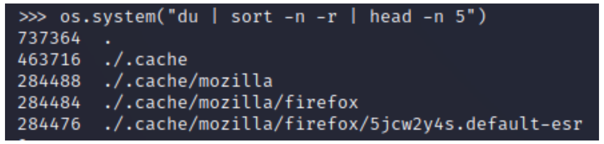


```
>>> os.system("du")
8       ./.config/geany/filedefs
4       ./.config/geany/templates/files
12      ./.config/geany/templates
4       ./.config/geany/tags
36      ./.config/geany
8       ./.config/autostart
8       ./.config/cherrytree
4       ./.config/binwalk/config
4       ./.config/binwalk/plugins
4       ./.config/binwalk/magic
4       ./.config/binwalk/modules
20      ./.config/binwalk
```

2nd: This command will sort the listing from greatest to least using option (-r)



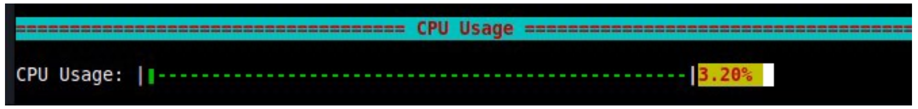
```
>>> os.system("du | sort -n -r")
737364  .
463716  ./.cache
284488  ./.cache/mozilla
284484  ./.cache/mozilla/firefox
284476  ./.cache/mozilla/firefox/5jcw2y4s.default-esr
252132  ./.cache/mozilla/firefox/5jcw2y4s.default-esr/cache2
251872  ./.cache/mozilla/firefox/5jcw2y4s.default-esr/cache2/entries
140964  ./.cache/vmware
140960  ./.cache/vmware/drag_and_drop
138688  ./SOC
117188  ./SOC/ES
```

Description	
os.system("du -h | sort -n | head -n 5")	*3rd: Final part of the command narrows the listing down to Top 5 entries which have been sorted according to the folder size in descending order.*  `>>> os.system("du | sort -n -r | head -n 5")` `737364  .` `463716  ./.cache` `284488  ./.cache/mozilla` `284484  ./.cache/mozilla/firefox` `284476  ./.cache/mozilla/firefox/5jcw2y4s.default-esr`

Python Script

```
print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}==================================== CPU    ↲
Usage ====================================")
print()
# Modules required: colorama, psutil, time

def display_usage(cpu_usage, bars=50):
    # Convert CPU percentage into decimal value to allow calculation on the numbers    ↲
    of bar characters to display.
    cpu_dp = (cpu_usage / 100.0)
    # Calculate the number of bar '❚' vs number of dash '-' to occupy a fixed space.
    # Shift + Ctrl + U + 275A >> ❚
    cpu_bar = '❚' * int(cpu_dp * bars) + '-' * (bars - int(cpu_dp * bars))
    # To display the the CPU Usage monitor with its value (as 2 decimal places).
    print(f"\rCPU Usage:                                                              ↲
    |{Fore.GREEN}{Style.BRIGHT}{cpu_bar}{Fore.RESET}{Back.RESET}|{Fore.RED}{Back.YELLO ↲
    W}{cpu_usage:.2f}% ", end="")

# Create a loop that refreshes CPU usage percentage every 10 seconds
while True:
    display_usage(psutil.cpu_percent(), 50)
    time.sleep(10)
```

Output



```
==================================== CPU Usage ====================================

CPU Usage: |❚-------------------------------------------------|3.20%
```

Description

```
def display_usage(cpu_usage, bars=50):
    cpu_dp = (cpu_usage / 100.0)
    cpu_bar = '❚' * int(cpu_dp * bars) + '-' * (bars - int(cpu_dp * bars))
    print(f"\rCPU Usage:
    |{Fore.GREEN}{Style.BRIGHT}{cpu_bar}{Fore.RESET}{Back.RESET}|{Fore.RED}{Back.YELLO
    W}{cpu_usage:.2f}% ", end="")

while True:
    display_usage(psutil.cpu_percent(), 50)
    time.sleep(10)
```

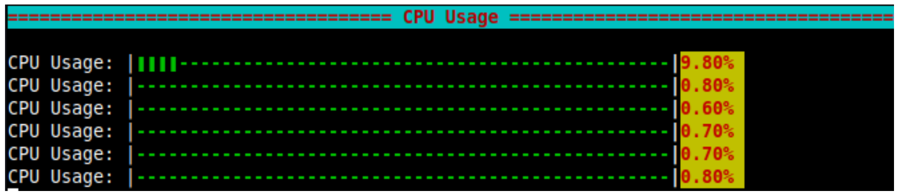
Part 1: Creation of CPU Usage Monitor

- *Define a new function 'display_usage(cpu_usage, bars = 50)' with the number bars to display. 'cpu_usage' will have a return value from psutil.cpu_percent() in the form of %.*
- *Variable cpu_dp will help to convert cpu_usage value from % to float data.*
- *Variable cpu_bar will help to calculate the number of '❚' (represents the psutil.cpu_percent() value) and '-' (fill up empty void spaces).*
- *f string format syntax is used to print the cpu_bar and report the value to 2 decimal places denoted by '.2f'.*

Description

```python
def display_usage(cpu_usage, bars=50):
    cpu_dp = (cpu_usage / 100.0)
    cpu_bar = '▌' * int(cpu_dp * bars) + '-' * (bars - int(cpu_dp * bars))
    print(f"\rCPU Usage: |{Fore.GREEN}{Style.BRIGHT}{cpu_bar}{Fore.RESET}{Back.RESET}|{Fore.RED}{Back.YELLOW}{cpu_usage:.2f}% ", end="")

while True:
    display_usage(psutil.cpu_percent(), 50)
    time.sleep(10)
```

- *'\r' is important to ensure CPU Usage monitor and its value gets cleared for every 10 second refresh.*
- *end="" – This will prevent from seeing new lines entry appearing instead of getting refreshed.*



```
=================================== CPU Usage ===================================

CPU Usage: |▌▌▌▌----------------------------------------------|9.80%
CPU Usage: |--------------------------------------------------|0.80%
CPU Usage: |--------------------------------------------------|0.60%
CPU Usage: |--------------------------------------------------|0.70%
CPU Usage: |--------------------------------------------------|0.70%
CPU Usage: |--------------------------------------------------|0.80%
```

<u>Part 2: Loop creation and Refresh</u>

Syntax for this portion will run infinitely with no condition set. At the same time, the CPU Usage monitor gets refreshed once every 10 seconds using time.sleep() function.

D. Credits / References

YouTube Videos

- Oğuzhan Kır- Get disk usage with python
  https://youtube.com/shorts/0oR1Q2EP3PM?feature=share

- Misha Sv – How to Get System and Hardware Information using Python
  https://youtu.be/wt8Aqm256NI

- NeuralNine - CPU & RAM Usage Monitor in Python
  https://youtu.be/rdxt6ntfX24

- NeuralNine – Python Sockets Simply Explained
  https://youtu.be/YwWfKitB8aA

- Tech With Time – How to Print Colored Text in Python (Colorama Tutorial)
  https://youtu.be/u51ZjlnuI4Y

Online

- Socket Programming in Python (Guide)
  URL - https://realpython.com/python-sockets/

- Compart – Unicode Character
  URL - https://www.compart.com/en/unicode/U+275A

- Stack Overflow – Socket IO returns 127.0.0.1 as host address and not 192.168.0.* on my device
  URL - https://stackoverflow.com/q/72331707

- Pyfiglet -PYPI
  URL - https://pypi.org/project/pyfiglet/0.7/

- Geeks for Geeks - How to get the current username in Python
  URL - https://www.geeksforgeeks.org/how-to-get-the-current-username-in-python/

GitHub

- Techwithtim/ ColoredTextInPython
  URL - https://github.com/techwithtim/ColoredTextInPython

Book(s)

- Linux Pocket Guide Essential Commands, 3rd Edition, Daniel J. Barrett

```python
#!/usr/bin/python3

# Name of Student (Code): Muhd Fazil Istamar (S11)
# Class Code: CFC240722
# Name of Trainer: James Lim
# Filename: S11_Python_Proj.py

# Import required modules
import os
import time
import psutil
import socket
import requests
import platform

# Install colorama and import module
print("Installing colorama")
os.system("python3 -m pip install colorama")
import colorama

# Configure colorama
from colorama import Fore, Back, Style
# Allow reset of font text colours when 'autoreset=True'
colorama.init(autoreset=True)
print()

# Install pyfiglet and import the module
print("Installing pyfiglet")
os.system("python3 -m pip install pyfiglet==0.7")
import pyfiglet
print()

# Install netifaces and import module
print("Installing netifaces")
os.system("python3 -m pip install netifaces")
import netifaces
print()

# Retrieve username who login to the client.
# This username info will be used in greeting header for personal touch.
usr = os.getlogin()
```

```
# Create a greetings header using pyfiglet
greet = pyfiglet.figlet_format( f"Welcome,{usr}!")
print(f"{Fore.CYAN}{Style.BRIGHT} + {greet}")
# Include a 3-second delay for each sections to allow user to read the contents before next set of information appears
time.sleep(3)

print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}================================== OS Version ==================================")
print()
# Modules required: colorama, platform, time

#Computer network name
print(f"Computer network name:{Fore.RED}{platform.node()}")
print()
#Operating System
print(f"Operating System:{Fore.RED}{platform.system()}")
print()
#Operating System Release
print(f"Operating System Release:{Fore.RED}{platform.release()}")
print()
#Operating System Version
print(f"Operating System Version:{Fore.RED}{platform.version()}")
print()
time.sleep(3)

print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}==================================
Network Info ==================================")
print()
# Modules required: colorama, netifaces, requests, socket, time

#Find Private / Internal IP Address
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8",80))
Int_IP = s.getsockname()[0]
print(f"You Private IP Address: {Fore.RED}{Int_IP}")
print()

#Find Public / External IP Address
b = requests.get("https://api.ipify.org?format=json")
Ext_IP = b.json()["ip"]
```

```python
print(f"Your Public IP Address is: {Fore.RED}{Ext_IP}" )
print()

#Find Default Gateway
DG = netifaces.gateways()['default'][netifaces.AF_INET][0]
print(f"Default Gateway: {Fore.RED}{DG}")
print()
time.sleep(3)

print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}================================= Hard Disk Info =================================")
print()
# Modules required: colorama, psutil, time

# Variables for different requests of disk status
# Calculation = Total number of bytes ÷ (Size of kilobyte ^3)
hdd = psutil.disk_usage('/')
# TDS - Total Disk Space
TDS = hdd.total/1024**3
# UDS - Used Disk Space
UDS = hdd.used/1024**3
# FDS - Free Disk Space
FDS = hdd.free/1024**3

#Total HD Space
print (f'Total: {Fore.RED}{TDS:.2f} GB')
print()
#Used HD Space
print (f'Used: {Fore.RED}{UDS:.2f} GB')
print()
#Free HD Space
print (f'Free: {Fore.RED}{FDS:.2f} GB')
print()
time.sleep(3)

print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}================================ Top 5 Directories ================================")
print()
# Modules required: colorama, os, time
os.system("du | sort -n -r| head -n 5")
print()
```

```python
time.sleep(3)

print(f"{Fore.RED}{Back.CYAN}{Style.BRIGHT}==================================== CPU Usage ====================================")
print()
# Modules required: colorama, psutil, time

def display_usage(cpu_usage, bars=50):
    # Convert CPU percentage into decimal value to allow calculation on the numbers of bar characters to display.
    cpu_dp = (cpu_usage / 100.0)
    # Calculate the number of bar '❚' vs number of dash '-' to occupy a fixed space.
    # Shift + Ctrl + U + 275A >> ❚
    cpu_bar = '❚' * int(cpu_dp * bars) + '-' * (bars - int(cpu_dp * bars))
    # To display the the CPU Usage monitor with its value (as 2 decimal places).
    print(f"\rCPU Usage: |{Fore.GREEN}{Style.BRIGHT}{cpu_bar}{Fore.RESET}{Back.RESET}|{Fore.RED}{Back.YELLOW}{cpu_usage:.2f}% ", end="")

# Create a loop that refreshes CPU usage percentage every 10 seconds
while True:
    display_usage(psutil.cpu_percent(), 50)
    time.sleep(10)

```