Report on Python Fundmentals

Student Name : Low Hong Jun
Student Code : S13

Objective

Create automation to display the operating system information

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

Contents

Python Fundmentals

```
14  #Importing all necessary modules for this script
15
16  import platform
17  import os
18  import time
19  import requests
20  import socket
21  import netifaces
22  import shutil
23  import psutil
24  import operator
25
26
```

1. Modules imported for the script and specific function of module used

platform.uname() -- Returns the a named tuple consisting of device, system, node, release, version, machine, and processor

os.chdir() -- changing of directory in python

os.getcwd() -- return the current working directory

os.listdir() -- return all entries in the given directory

os.path.isfile() -- checking if a path is a file, returns True if its a existing file

os.path.join() -- joining path component to create one full file path

os.path.getsize() -- return the size of the path indicated

time.sleep() -- adds time delay to the execution of the program

requests.get() -- send HTTP request and returns the value given

socket.socket() -- creating a socket object

s.connect() - connects to server

s.getsockname() - returns address of client socket

netifaces.gateways() - gets gateway information

shutil.disk_usage() - tells the disk usage statistics in 3 attribues, total, used and free

psutil.cpu_percent() - retrieve information on system process and ultilization

operator.itemgetter() - return a callable object, can set index to specific data wanted

```
26
27  print('\nWelcome, currently gathering information about your OS...\n')
28
29
30  OSInfo = (platform.uname())
31
32
33  a=(OSInfo[0])
34  b=(OSInfo[1])
35  c=(OSInfo[2])
36  d=(OSInfo[4])
37
38
39  time.sleep(5)
40
41
42
43  print('This is your operating system:', a)
44  print('This is the distro of the OS:', b)
45  print('This is your OS version:', c)
46  print('Bit processor information:', d)
47
```

Figure 1.1 – Python Code : OS Version

2. Breakdown of Python code

Our first task is the display our OS version

print(‘\nWelcome, currently gathering information about your OS...\n’) prints out a “Welcome, currently gathering information about your OS...” message to the user so they know the current process the program is executing.

\n in this case, creates a new line character so our output do not look cramped.

OSInfo = (platform.uname()) stores a named tuple consisting of device, system, node, release, version, machine, and processor in the variable OSInfo

The variable OSInfo is then ask to displayed the different values it is holding using the index method [0][1][2]... The desired values are then stored in variables we designated, ‘a to d’

For example, OSInfo[0], display the first value in the variable OSInfo which contains the platform.uname() module function. This is then stored into the variable ‘a’

time.sleep(5) adds a 5 seconds delay to the program so the the user will have time to read the previous message.

The informations are then print out using the print command.

```
Welcome, currently gathering information about your OS...

This is your operating system: Linux
This is the distro of the OS: kali
This is your OS version: 6.3.0-kali1-amd64
Bit processor information: x86_64
```

Figure 1.2 – Output of OS Version from terminal

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


print('\nGathering information about your IP Address...\n')



time.sleep(5)
```

```
#Getting the public IP through ipify.org API by calling the function as shown below.


def get_ip_address():
    url = 'https://api.ipify.org'
    response = requests.get(url)
    ip_address = response.text
    return ip_address

ip = get_ip_address()
print("This is your Public IP Address:", ip)
```

Figure 2.1 – Python Code : Public IP

3. Breakdown of Python code

On the second task, we are required to display our Public, Default and Internal IPs

We print out our message “Gathering information about your IP Address...” to inform user of current process

time.sleep(5) adds a time delay for the user to read the message

We then call a function def get_ip_address(), in this function we used the request module to acquire the information we want and return it to the function.

3. Breakdown of Python code (Continued)

url = ‘https://api.ipify.org’

In this code, we set url variable as the string https://api.ipify.org which is the domain where we will send a get request to retreieve our Public IP information.

response = requests.get(url)

Get request sent to url which contains our domain, the return value is stored in the reponse variables.

ip_address = response.txt
return ip_address

We then store response variables in to a text file and call it in to a ip_address variable, and we return this variable to the function get_ip_address.

We then set the variable ip to contain our function ip_get_address and have it print out our Public IP address.

```
#Getting the default gateway through the netiface module


gateways = netifaces.gateways()
default_gateway = gateways['default'][netifaces.AF_INET][0]

print('This is your Default Gateway IP Address:', default_gateway)
```

Figure 2.2 – Python Code : Default Gateway IP

4. Breakdown of Python code

Using the netifaces module, we will can retrieve our device default gateway

In our example, we set gateways variable to contain the netifaces.gateway() function

netifaces.gateway() display all available gateway on our device

If we just print netifaces.gateway() it should show some thing like this

{'default': {2: ('192.168.***.2', 'eth0')}, 2: [('192.168.***.2', 'eth0', True)]}

Format : address , interface, is_default
If it shows True, that is our default gateway

4. Breakdown of Python code (Continued)

We then filter off/select the information we want from this output using []

For example, gateways[‘default’] will print out {2: ('192.168.***.2', 'eth0')}, we then further filter it by adding gateways[‘default’][netifaces.AF_INET] which will give us ('192.168.***.2', 'eth0')

From here we can select the information we want by its index, the IP address is in [0], so we adjust our code gateways[‘default’][netifaces.AF_INET][0] and store this in default_gateway variable.

default_gateway = gateways[‘default’][netifaces.AF_INET][0]
print(‘This is your Default Gateway IP Address:’)

We then print out the default gateway as information for the user to see.

```
#Getting our internal ip address


s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))

print('Your Internal IP Address is:', s.getsockname()[0])
```

Figure 2.3 – Python Code : Internal IP

5. Breakdown of Python code

The socket modules allows communnication between machines over a network

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

socket.AF_INET means we are using IPV4 family address
socket.SOCK_DGRAM means we are using UDP to send and receive data

This allow us to create a socket that can send and receive data using the IPV4 family and UDP protocol. This socket is store into the variable ‘s’

This new variable socket is then use to connect to the 8.8.8.8, port 80 server so we can communicate with each other. This is the Google DNS server, which is very reliable.

We then s.getsockname() to get something like this '192.168.155.128', 52546

But since we only need the ip address, we indicate [0]

s.getsockname()[0] which will only display 192.168.155.128, we then print it out to the terminal using the print command.

```
Gathering information about your IP Address...

This is your Public IP Address: 103.252.200.35
This is your Default Gateway IP Address: 192.168.155.2
Your Internal IP Address is: 192.168.155.128
```

Figure 2.4 – Output of IP Addresses from Terminal

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

print('\nCalculating disk usage...\n')


time.sleep(5)

total, used, free = shutil.disk_usage("/")

print("Total: %d GiB" % (total // (2**30)))
print("Used: %d GiB" % (used // (2**30)))
print("Free: %d GiB" % (free // (2**30)))
```

Figure 3.1 – Python Code : Disk Usage

6. Breakdown of Python code

print(‘\nCalculating disk usage...\n’) output a message for the user to know the current process

Calculating disk usage...

time.sleep(5) delays the execution of program for 5 seconds so that the user will have time to read the message and allow a progressive output of information

total, used, free = shutil.disk_usage(“/”)

For this task to display disk usage, we will be using the shutil module, disk_usage function and output a tuple of 3 attributes, total, used, free. We then store these in the same named variable. Total, used, free.

In the shutil.disk_usage(), we indicate the path folder we want to check. In our case, the root folder “/”

We then output the variables using the print command to show the data. As the datas are shown in bytes, it might be hard to read. So we will divide the data by (2**30) which is close to 1Gigabyte, to convert the information in to Gigabytes.

```
Calculating disk usage...

Total: 78 GiB
Used: 14 GiB
Free: 60 GiB
```

Figure 3.2 – Output of Disk Usage from Terminal

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

print('\nCalculating the top 5 directories and their size...\n')


time.sleep(5)

os.chdir("/")

dir_name = os.getcwd()

file_sizes = {}

for filename in os.listdir(dir_name):
    if os.path.isdir(os.path.join(dir_name, filename)):
        file_sizes[filename] = os.path.getsize(os.path.join(dir_name, filename))

sorted_file_sizes = sorted(file_sizes.items(), key=operator.itemgetter(1), reverse=True)


for i in range(5):

    filename, size = sorted_file_sizes[i]

    print(f"{i+1}. {filename} - {size} bytes")
```

Figure 4.1 – Python Code : Top 5 Directories In Size

7. Breakdown of Python code

In this task, we are interested in looking for the top 5 files in the system. Hence, the path folder we are interested in will be the root folder.

os.chdir("/") changes our current working directory to the root directory.

We then print out the current working directory name using the *os.getcwd()* module and store the data in the variable dir_name.

We set *file_sizes = {}* dictionary to be use later in the code

7. Breakdown of Python code (Continued)

We then use a For loop to help us repeat the task of listing out all the files in the root directory.

for filename in os.list.dir(dir_name):

os.listdir(dir_name) list all the files in the directory 'dir_name' which is the current working directory, which should be root in our case.

If os.path.isdir(os.path.join(dir_name, filename)):

This line create a IF statement, that if os.listdir(dir_name) output is a directory, os.path.join will join the 'dir_name' with the filename creating a full folder path.

file_sizes[filename] = os.path.getsize(os.path.join(dir_name, filename))

The next set of command will only execute if the previous result is a file. This command ***file_sizes[filename]*** *w*ill store the output of ***os.path.getsize(os.path.join(dir_name, filename))*** into the filename in the file_sizes dictionary.

os.path.getsize(os.path.join(dir_name, filename))

This module function get the size in bytes of the file output by (os.path.join(dir_name, filename) which is the full folder path of the specfic file

sorted_file_sizes = sorted(file_sizes.items(), key=operator.itemgetter(1), reverse=True)

In this command we make use of the operator module to sort our results and reverse it in descending order

sorted **-**sort command, to sort according to the condition we input
file_sizes.items() **-** return new of dictionary item (key-value pairs)
key=operator.itemgetter(1) – sorting arguement, we use the information in index 1 which is our size to sort accordingly
reverse=True – Reverse the sort result, in descending order

The output of this whole command will give us the biggest files at the very top

We then use a for loop to loop 5 times so it shows the top 5 biggest directories

```
for i in range(5):
filename, size = sorted_file_sizes[i]
print(f"{i+1}. {filename} - {size} bytes")
```

We save the key(filename) and value(size) in to their variables then print it out accordingly.

The f-string in the print command is used to format the text and {i+1} is to print our the ranking of the files.

```
Calculating the top 5 directories and their size...

1. swapfile - 1073741824 bytes
2. initrd.img - 76453411 bytes
3. initrd.img.old - 76453411 bytes
4. vmlinuz - 8983392 bytes
5. vmlinuz.old - 8983392 bytes
```

Figure 4.2 – Output : Top 5 Directories In Size

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

print('\nDisplaying your CPU Usage in percentage % ...\n')

while True:
    try:
        print('The CPU usage is: ', psutil.cpu_percent(10),
        '%, information will refresh infinitely every 10 secs')
        print('Ctrl + C to exit program\n')

    except KeyboardInterrupt:
        print('\nUser exited through keyboard input')
        break
```

Figure 5.1 – Python Code : CPU Usage

8. Breakdown of Python code

To get CPU Usage, we can use the psutil module to get information on cpu ultilization in percentage.

We will be using a WHILE loop to create a infinite loop as it is always True unless a condition is met. In this case, a KeyboardInterrupt (Control + C input) will disrupt the infinite loop.

```
while True:
   try:
      print('The CPU usage is: ', psutil.cpu_percent(10),
      print('Ctrl + C to exit program\n')

   except KeyboardInterrupt:
      print('\nUser exited through keyboard input')
      break
```

psutil.cpu_percent(10) indicates that the cpu usage will display itself every 10 seconds.

TRY and EXCEPT is use in our case as IF and ELSE is more suitable for logical condition and a making decision.

```
The CPU usage is:  1.9 %, information will refresh infinitely every 10 secs
Ctrl + C to exit program

The CPU usage is:  1.9 %, information will refresh infinitely every 10 secs
Ctrl + C to exit program

The CPU usage is:  2.2 %, information will refresh infinitely every 10 secs
Ctrl + C to exit program

The CPU usage is:  4.0 %, information will refresh infinitely every 10 secs
Ctrl + C to exit program

^C
User exited through keyboard input
```

Figure 5.2 – Output : CPU Usage

After the user exit using Control + C, the text message will inform the user that they exited.

If undisturbed, the CPU Usage will be displayed every 10 seconds infinitely.

This is the end of the program