

Python Fundamentals Project

Python Fundamental

STUDENT: MUHAMMAD FEROZ (S10)

Scope

A Python automation script gathering the following details;

- Display Linux OS version
 - Display Private/Public IP address and Default gateway
 - Display Hard disk size, free and used space.
 - Display the top 5 directories and their size.
 - Display CPU usage with a 10-sec refresh rate.
-

Content Page

[Scope](#)

[Content Page](#)

[Python Script Layout](#)

[Header](#)

[Match case and Variables](#)

[Cases](#)

[Case 1: Display of OS Version](#)

[Case 2: Display of IP addresses](#)

[Hostname](#)

[Public IP](#)

[Private IP](#)

[Default gateway IP](#)

[Case 3: Display of Hard Disk utilisation](#)

[Case 4: Display of list of top 5 directory](#)

[Case 5: Display of CPU usage](#)

[Script](#)

Python Script Layout

Header

The following selected modules are imported at the beginning of the script to allow functions to be utilized across the scripts.

```
#!/usr/bin/python3
import platform # retrieving information on os version
import os      # retrieving information on contents, changing and identifying the current directory, etc.
import socket  # retrieving information on ip address
import requests # retrieving information on ext_ipaddress
import shutil  # retrieving information on disk space
import psutil  # retrieving information on running processes and system utilization
import operator # retrieving information on itemgetter to sort dictionary
```

Match case and Variables

The match statement helps with matching of pattern against the user input and calling out the respective functions. This statement was selected due to ease of readability and expandability instead of complex nested if-else statements.

1. os_version()
2. dp_address()
3. hard_disk()
4. hard_disk()
5. hard_disk()

When a user inputs a string value between 1 to 5, the match statement steps in and calls out the respective user-defined function within a case, no parameters are passed or received at this point.

Values beyond the acceptable range are caught with the wild card parameter using the “_”, the script outputs a message and exits the script.



The match case statement is available after Python 3.10

```
user_input = input('Enter Menu number between 1-5 \nEnter 0 to exit\n')
match user_input:
    case '1':
        os_version()
    case '2':
        dp_address()
    case '3':
        hard_disk()
    case '4':
        list_dir()
    case '5':
        usage_cpu()
    case _:
        print ('Out of range, Exit Program')
        exit()
```

```

(kali㉿kali)-[~/Documents/Project_Python]
$ python3 s10_feroz_python.py
Project: Python Fundamentals
Name: Muhammad Feroz (S10)

1 for Display the OS version

2 for Display the private/public IP address

3 for Display the disk utilisation

4 Display the top five (5) directories

5 Display the CPU usage
Enter Menu number between 1-5
Enter 0 to exit
█

```

Credit :

<https://favtutor.com/blogs/python-switch-case>

<https://blog.enterprisedna.co/python-match-case/>

<https://www.udacity.com/blog/2021/10/python-match-case-statement-example-alternatives.html>

Cases

A total of 5 cases feed into the match statement and each case will execute its respective commands according to the scope.

Case 1: Display of OS Version

This case displays the Linux description and version of the host, it utilizes the **platform module** for functions such as `version()`, `system()` and `release()`.

`version()` - returns a string that displays the machine type.

`system()` - returns a string that displays the name of the operation system.

`release()` - return a string of the system's release.

```

def os_version():

    print("*****")
    print("\nName of the release version: ", platform.version()) #system's release version
    print("Name of the OS system: ", platform.system()) #get the name of the OS the system is running on
    print("Version of the operating system: ", platform.release()) #get the version of the operating system

```

```

Enter Menu number between 1-5
Enter 0 to exit
1
*****

Name of the release version: #1 SMP PREEMPT_DYNAMIC Debian 6.1.27-1kali1 (2023-05-12)
Name of the OS system: Linux
Version of the operating system: 6.1.0-kali9-amd64

(kali@kali)-[~/Documents/Project_Python]
$

```

Credit :

[#https://www.w3resource.com/python-exercises/python-basic-exercise-43.php](https://www.w3resource.com/python-exercises/python-basic-exercise-43.php)
[#https://www.adamsmith.haus/python/answers/how-to-get-the-running-os-in-python](https://www.adamsmith.haus/python/answers/how-to-get-the-running-os-in-python)
[#https://docs.python.org/3/library/platform.html](https://docs.python.org/3/library/platform.html)
[#https://www.geeksforgeeks.org/platform-module-in-python/](https://www.geeksforgeeks.org/platform-module-in-python/)

Case 2: Display of IP addresses

This case displays the private, public and default gateway of the host IP address, it utilizes the **socket**, **request** and **os** module for functions such as `get.hostname()`, `gethostbyname()`, `popen()` and `read()`.

`get.hostname()` - returns a string of the hostname.

`gethostbyname()` - returns a string of the IP address using the hostname.

`popen()` - opens a pipe to or from the command.

`read()` - returns the specified number of bytes from the file.

`getsockname()` - allows for more direct and precise retrieval of the local IP address.



Challenge faced: `socket.gethostbyname(socket.gethostname())` was giving a value of 127.0.1.1 as an static value do to configuration setting in the host.

The IP address `127.0.1.1` in the second line of this example may not be found on some other Unix-like systems. The Debian Installer creates this entry for a system without a permanent IP address as a workaround for some software (e.g., GNOME) as documented in the [bug #719621](#).

Credit : https://www.debian.org/doc/manuals/debian-reference/ch05.en.html#_the_hostname_resolution

```

def dp_address():
    #Display the private IP address, public IP address, and the default gateway

```

```

print("*****")
print("Host name is:",socket.gethostname()) #to get the hostname
#print("Private IP address is:",socket.gethostbyname(socket.gethostname()))
print("Public IP address is:",requests.get('https://api.ipify.org').text) #Use a external site's API to get IP
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80)) # temporary connection to 8.8.8.8 using port 80
print("Private IP address is:",s.getsockname()[0]) #get local IP address that initiate the connection
dgateway = os.popen('ip r | grep default | awk \'{print $3}\'' ).read() #pass Linux command as a subprocess
print("Default gate way:",dgateway)

```

```

Enter Menu number between 1-5
Enter 0 to exit
2
*****
Host name is: kali
Public IP address is: 39.109.180.228
Private IP address is: 192.168.154.128
Default gate way: 192.168.154.2

(kali) - [~/Documents/Project_Python] $

```

Hostname

This command displays the name of the host.

```

print("Host name is:",socket.gethostname())

```

Public IP

This command uses the request module to gather information from ipify.org's API (Official site provides the code sample) .

```

print("Public IP address is:",requests.get('https://api.ipify.org').text)

```

Private IP

This command retrieves the “local IP address of the device by establishing a temporary connection to the remote server with IP address 8.8.8.8 on port 80 using a UDP socket. It creates a socket object `s` with the appropriate address family `socket.AF_INET` and socket type `socket.SOCK_DGRAM` ” -(www.delftstack.com).

```

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("8.8.8.8", 80))
print("Private IP address is:",s.getsockname()[0])

```

Default gateway IP

This command uses the os module to run a subprocess in the host system and read the output of it.

```
dgateway = os.popen('ip r | grep default | awk \'{print $3}\'}').read()
print("Default gate way:",dgateway)
```

Credit :

[#https://www.geeksforgeeks.org/python-program-find-ip-address/](https://www.geeksforgeeks.org/python-program-find-ip-address/)
[#https://www.quora.com/How-can-you-get-your-public-IP-address-in-Python](https://www.quora.com/How-can-you-get-your-public-IP-address-in-Python)
[#https://www.youtube.com/watch?v=LbSaKAAYUgE](https://www.youtube.com/watch?v=LbSaKAAYUgE)
[#https://www.delftstack.com/howto/python/get-ip-address-python/](https://www.delftstack.com/howto/python/get-ip-address-python/)
[#https://pythonguides.com/python-get-an-ip-address/](https://pythonguides.com/python-get-an-ip-address/)
[#https://www.tutorialspoint.com/python/os_popen.htm](https://www.tutorialspoint.com/python/os_popen.htm)
[#ipify - A Simple Public IP Address API](#)
[#https://stackoverflow.com/questions/53902519/find-default-gateway-address-for-windows-and-linux-on-python](https://stackoverflow.com/questions/53902519/find-default-gateway-address-for-windows-and-linux-on-python)

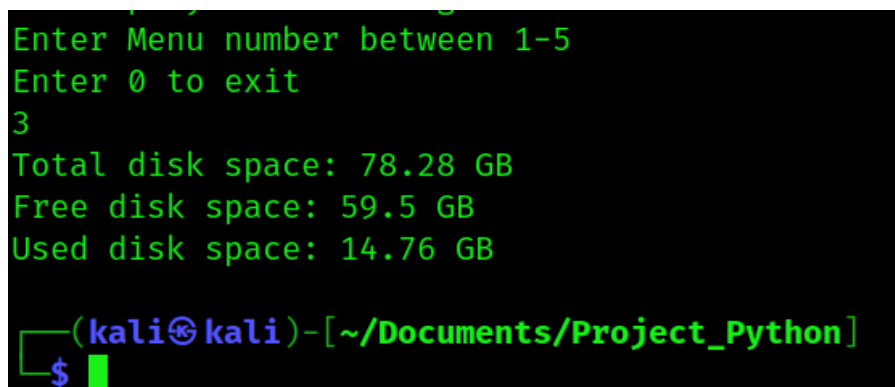
Case 3: Display of Hard Disk utilisation

This case displays the host's hard disk size, free and used space, it utilizes the **shutil module** to for functions such as `disk_usage()` .

`disk_usage()` - returns a named tuple with the attributes `total` , `used` and `free` which are the amount of total, used and free space, in bytes

Output of bytes are divided by 1024 (2¹⁰ - 3 times) to convert into GB for easy readability.

```
print("Total disk space:",round(shutil.disk_usage('/').total/ 1024 / 1024 / 1024,2),"GB") # Total space in GB
print("Free disk space:",round(shutil.disk_usage('/').free/ 1024 / 1024 / 1024,2),"GB") # Free space in GB
print("Used disk space:",round(shutil.disk_usage('/').used/ 1024 / 1024 / 1024,2),"GB") # Used space in GB
```

A terminal window with a black background and green text. It shows a menu prompt 'Enter Menu number between 1-5' and 'Enter 0 to exit'. The user has entered '3'. The output shows 'Total disk space: 78.28 GB', 'Free disk space: 59.5 GB', and 'Used disk space: 14.76 GB'. At the bottom, the prompt is '(kali@kali)-[~/Documents/Project_Python]' followed by a '\$' and a cursor.

```
Enter Menu number between 1-5
Enter 0 to exit
3
Total disk space: 78.28 GB
Free disk space: 59.5 GB
Used disk space: 14.76 GB

(kali@kali)-[~/Documents/Project_Python]
$
```

Credit :

[#https://www.geeksforgeeks.org/python-shutil-disk_usage-method/](https://www.geeksforgeeks.org/python-shutil-disk_usage-method/)

<https://gist.github.com/rikonor/74ac150b746dfa09190515268c5bfba4>

<https://java2blog.com/format-a-float-to-two-decimal-places/>

Case 4: Display of list of top 5 directory

This case displays the host's top 5 directories and their sizes, it utilizes the **os** and **operator module** to for functions such as `listdir()`, `path.join()`, `isdir()`, `walk()`, `sort()`, `itemgetter()`.

`listdir()` - list directory of a provided path

`path.join()` - joins the path segments into one complete path

`isdir()` - check whether the specified path is directory

`walk()` - generates the file names/ directories of a provided path

`sort()` - sorts the items of a list in ascending or descending order

`itemgetter()` - fetches an item from an object. It corresponds to specifying an index

1. Set a defined path
2. Declare a dictionary to save both Directory name and Size to manipulate at late stage.
3. For-loop by listing directories within the given path, not accounting for the files.
 - a. Followed by joining the directory and path name.
4. If-statement to check the full path is a directory itself.
5. Based on the full path provided, the for-loop is set up to walk into its sub-directories to calculate the file sizes within it. This will determine the top 5 largest directories.
 - a. to calculate the size of each file, a new full path name is determined.
 - b. The size of the file within the same directory is stored into a variable similar to a counter.
6. The dictionary is updated with the name of the parent directory and the total number of files sizes within it.
7. A new dictionary is created with the sorted file size in ascending order.

```
path='/home/kali/Desktop' #set a fix path
dic_dir = {}              #dictionary to contain dir name and size
size1=0                   #store dir size
for p in os.listdir(path): #step in each dir of the given path
    full_path = os.path.join(path, p) #require full path of walk()
    if os.path.isdir(full_path): #check to ensure no files, onld dir
        #print(p,os.path.getsize(full_path))
        for dirpath, dirnames, filenames in os.walk(full_path): #step into each dir, to calualte size
            for f in filenames:
                fp = os.path.join(dirpath, f) #require full path of getsize()
                size1 += int(os.path.getsize(fp)) #within a dir caluculate all the file size
            dic_dir.update({p:size1}) #store the dir name (only parent folder) and size only

newdic_dir = dict(sorted(dic_dir.items(), key=operator.itemgetter(1), reverse=True)[:5])
print(newdic_dir) # sort the dic values (1) in accending order but only the to 5
```

```

Enter Menu number between 1-5
Enter 0 to exit
4
{'tor_dir': 285782560, 'ubuntu_export': 285406995, 'NetworkResearch2': 28538698
3, 'B0m8': 38883321, 'cfc20230819': 38869320}

(kali㉿kali)-[~/Documents/Project_Python]
$

```

Credit :

<https://stackoverflow.com/questions/7138379/directory-sizes-and-extensions>

<https://www.geeksforgeeks.org/how-to-get-size-of-folder-using-python/>

Case 5: Display of CPU usage

This case displays the host's CPU usage with a refresh rate of every 10 seconds, it utilizes the **os** and **psutil module** to for functions such as `cpu_percent()` .

`cpu_percent()` - return the current system-wide CPU utilization as a percentage.

The interval of value 10 is set as a loop, to get the latest CPU utilization after every 10 seconds.

```

def usage_cpu():
    #Display the CPU usage; refresh every 10 seconds

    counter=0
    while counter<10: # maxed at 10 counts,to prevent cont loop.
        print('The CPU usage is (every 10s): ', psutil.cpu_percent(10)) #10sec interval
        counter+=1

```

```

Enter Menu number between 1-5
Enter 0 to exit
5
The CPU usage is (every 10s): 3.8
The CPU usage is (every 10s): 3.7
The CPU usage is (every 10s): 11.0
The CPU usage is (every 10s): 5.3
The CPU usage is (every 10s): 14.8
The CPU usage is (every 10s): 8.5
The CPU usage is (every 10s): 5.7

```

Credit :

<https://www.geeksforgeeks.org/how-to-get-current-cpu-and-ram-usage-in-python/>

Script

```
#!/usr/bin/python3
import platform # retrieving information on os version
import os # retrieving information on contents, changing and identifying the current directory, etc.
import socket # retrieving information on ip address
import requests # retrieving information on ext_ipaddress
import shutil # retrieving information on disk space
import psutil # retrieving information on running processes and system utilization
import operator # retrieving information on itemgetter to sort dictionary

def os_version():
    #Display os details
    print("*****")
    print("\nName of the release version: ", platform.version()) #system's release version
    print("Name of the OS system: ", platform.system()) #get the name of the OS the system is running on
    print("Version of the operating system: ", platform.release()) #get the version of the operating system

def dp_address():
    #Display the private IP address, public IP address, and the default gateway
    print("*****")
    print("Host name is:",socket.gethostname()) #to get the hostname
    #print("Private IP address is:",socket.gethostbyname(socket.gethostname()))
    print("Public IP address is:",requests.get('https://api.ipify.org').text) #Use a external site's API to get IP
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.connect(("8.8.8.8", 80)) # temporary connection to 8.8.8.8 using port 80
    print("Private IP address is:",s.getsockname()[0]) #get local IP address that initiate the connection
    dgateway = os.popen('ip r | grep default | awk \'{print $3}\'}').read() #pass Linux command as a subprocess
    print("Default gate way:",dgateway)

def hard_disk():
    #Display the hard disk size; free and used space.
    print("Total disk space:",round(shutil.disk_usage('/').total/ 1024 / 1024 / 1024,2),"GB") # Total space in GB
    print("Free disk space:",round(shutil.disk_usage('/').free/ 1024 / 1024 / 1024,2),"GB") # Free space in GB
    print("Used disk space:",round(shutil.disk_usage('/').used/ 1024 / 1024 / 1024,2),"GB") # Used space in GB

def list_dir():
    #Display the top five (5) directories and their size.

    path='/home/kali/Desktop' #set a fix path
    dic_dir ={} #dictory to contain dir name and size
    size1=0 #store dir size
    for p in os.listdir(path): #step in each dir of the given path
        full_path = os.path.join(path, p) #require full path of walk()
        if os.path.isdir(full_path): #check to ensure no files, onld dir
            #print(p,os.path.getsize(full_path))
            for dirpath, dirnames, filenames in os.walk(full_path): #step into each dir, to calualte size
                for f in filenames:
                    fp = os.path.join(dirpath, f) #require full path of getsize()
                    size1 += int(os.path.getsize(fp)) #within a dir caluculate all the file size
                    dic_dir.update({p:size1}) #store the dir name (only parent folder) and size only

    newdic_dir = dict(sorted(dic_dir.items(), key=operator.itemgetter(1), reverse=True) [:5])
    print(newdic_dir) # sort the dic values (1) in accending order but only the to 5

def usage_cpu():
    #Display the CPU usage; refresh every 10 seconds
```

```

#https://www.geeksforgeeks.org/how-to-get-current-cpu-and-ram-usage-in-python/
counter=0
while counter<10: # maxed at 10 counts,to prevent cont loop.
    print('The CPU usage is (every 10s): ', psutil.cpu_percent(10)) #10sec interval
    counter+=1
print ('Project: Python Fundamentals\n','Name: Muhammad Feroz (S10)')
print ('\n 1 for Display the OS version')
print ('\n 2 for Display the private/public IP address')
print ('\n 3 for Display the disk utilisation')
print ('\n 4 Display the top five (5) directories')
print ('\n 5 Display the CPU usage')

user_input = input('Enter Menu number between 1-5 \nEnter 0 to exit\n')
match user_input:
    case '1':
        os_version()
    case '2':
        dp_address()
    case '3':
        hard_disk()
    case '4':
        list_dir()
    case '5':
        usage_cpu()
    case _:
        #wild-card catch
        print ('Out of range, Exit Program')
        exit()

```