Module: Python

**Q1. Write code to convert the below Dictionary in Pandas Dataframe:**

      Dictionary:  `dct = [{'2022-03-31': {'A': 12323, 'B': 123123}},{'2021-03-31': {'A': 12, 'B': 123}}]`  
        
Expected Output:df is dataframe  
```         
>> df  
            Date         A         B

    0 2022-03-31 12323 123123

    1 2021-03-31     12         123  
```

**Code:**

import pandas as pd

dct = [{'2022-03-31': {'A': 12323, 'B': 123123}}, {'2021-03-31': {'A': 12, 'B': 123}}]

dataList = []

for entry in dct:

    date, valueDict = list(entry.items())[0]

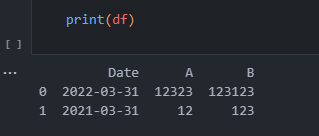
    valueDict['Date'] = date

    dataList.append(valueDict)

df = pd.DataFrame(dataList)

df = df[['Date', 'A', 'B']]

print(df)

**Output:   
**

**Q2. Write a Python code to count the Sum of positive and negative numbers that are below 40 in the below list**.

list = [10,-20,30,40,-50,60,12,-12,11,1,90,-20,-10,-5,-4]  
  
Expected Output:  
```  
The sum of positive elements in the list below 40: X  
The sum of negative elements in the list below is 40: Y  
```

**Code:**

list = [10, -20, 30, 40, -50, 60, 12, -12, 11, 1, 90, -20, -10, -5, -4]

positivesum = 0

negativesum = 0

for num in list:

    if num < 40:

        if num > 0:

            positivesum += num

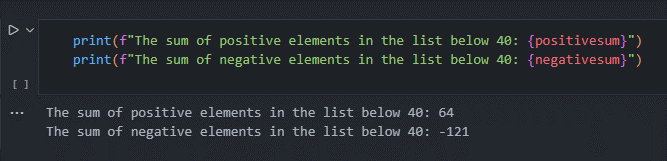
        elif num < 0:

            negativesum += num

print(f"The sum of positive elements in the list below 40: {positivesum}")

print(f"The sum of negative elements in the list below 40: {negativesum}")

**Output:**



Module: Linux

**Q1. Write a Shell script that will do the following.  
      i) Check Linux OS, which Linux distribution is present in the machine. E.g., Ubuntu, Cent OS  
      ii) Check whether application ‘httpd’ is installed or not on the Linux machine,  
      iii) If ‘httpd’ is not installed, install httpd  
      iv) check the status of the httpd application whether its status is running or not  
      v) If the status of the ‘httpd’ application is not running then start the application**

**Code:**

#!/bin/bash

# Check Linux OS distribution

if [ -f /etc/os-release ]; then

    source /etc/os-release

    echo "Linux Distro: $NAME"

    echo "Linux Distro Family: $ID\_LIKE"

else

    echo "Unknown Linux Distro"

fi

# Check if 'httpd' is installed

if command -v apache2 >/dev/null 2>&1 || command -v httpd >/dev/null 2>&1; then

    echo "httpd is already installed."

else

    echo "httpd is not installed. Installing httpd..."

    # Install HTTPD based on the OS family

    if [[ $ID\_LIKE == \*"debian"\* ]]; then

        # Debian-based distributions

        sudo apt-get update

        sudo apt-get install -y apache2

    elif [[ $ID\_LIKE == \*"rhel fedora"\* ]]; then

        # Red Hat-based distributions

        sudo yum update

        sudo yum install -y httpd

    else

        echo "Unsupported Linux distro. Please install 'httpd'/'apache2' manually."

        exit 1

    fi

fi

# Check the status of httpd application

if sudo systemctl is-active httpd >/dev/null 2>&1 || sudo systemctl is-active apache2 >/dev/null 2>&1; then

    echo "httpd is running."

else

    # Start httpd application if it's not running

    echo "httpd is not running. Starting httpd..."

    if sudo systemctl is-active httpd >/dev/null 2>&1; then

        sudo systemctl start httpd

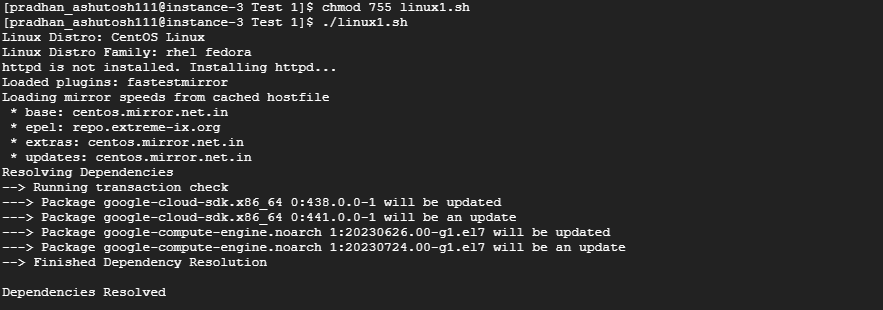
    elif sudo systemctl is-active apache2 >/dev/null 2>&1; then

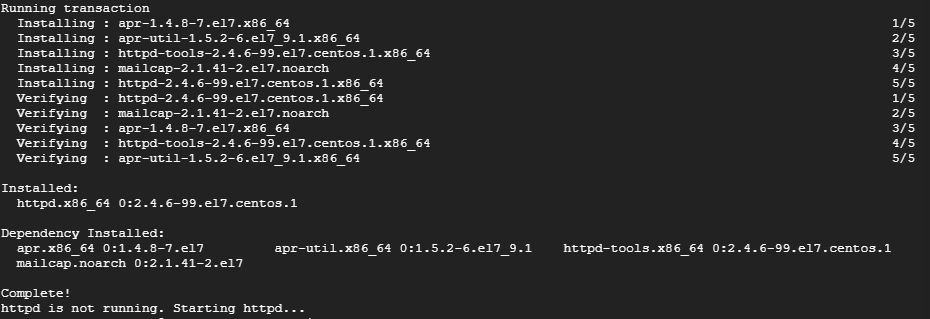
        sudo systemctl start apache2

    fi

fi

**Output:**



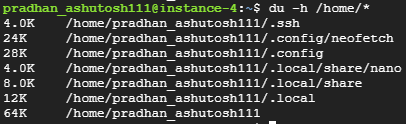


**Q2. i) Write Linux commands to check the storage of each folder present in ‘/home’ directory.  
      ii) Write a Linux command to check CPU Utilization, RAM, and Number of CPU Cores.**  
**Code:**

**i)**

du -h /home/\*

This will show the size of all the folders ( including hidden ones ) .



**ii)**

# CPU Utilization (

top -n 1

# RAM Usage

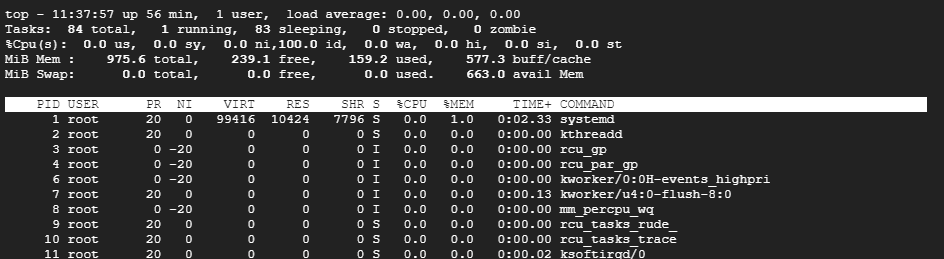
free -h

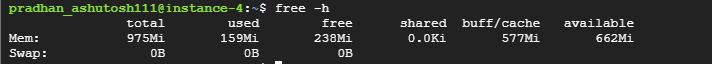
# Number of CPU Cores

Nproc

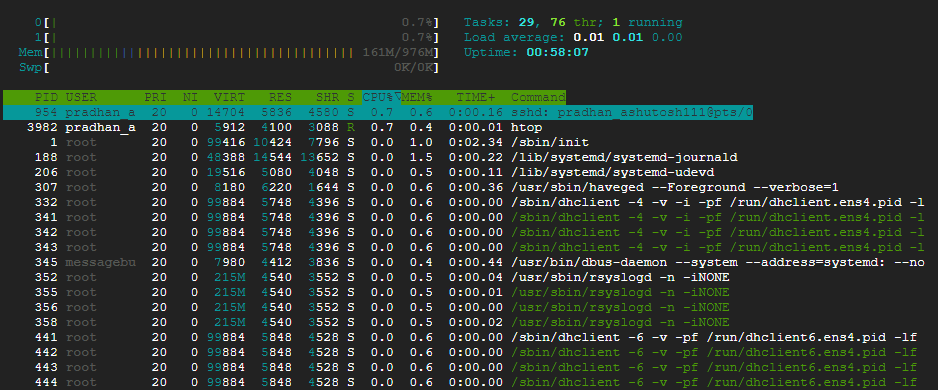
# For Live Monitoring , we can also use htop.

htop









Module: Kubernetes or Swarm

Git

**Q1. Write a git command to clone a particular branch in a local machine from GitLab (or GitHub).**

git clone -b <branch\_name> <repository\_url>

**Example:**

**Q2. Write git commands step by step to push the local git branch into the GitLab  (or GitHub) repository.**

Assuming the repository has been initialized ( if not then `git init`), we have to use the following commands.

git add -A  # Stage all changes in the project directory

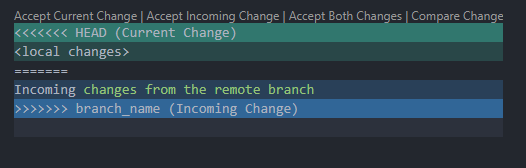
git commit -m 'commit message'  # Create a commit with the staged changes with the commit message

git remote add origin <url> # Add the remote repository with the given URL as 'origin'

git remote -v # Show the list of remote repositories

git push origin <branch\_name> # Push the committed changes

**Q3. How to resolve merge conflicts in GitLab (or GitHub)?**

Whenever we are merging or pulling from a branch, git notifies about the conflicting files. The conflicting parts are marked as follows:   


We can make changes as required.

git add resolved\_file1 resolved\_file2

Then we can commit the changes.

git commit -m "Commit Message"

Then we can push the merged changes to the remote branch.

git push origin <branch\_name>

**Q.4. How to create local branch from test branch present in gitlab or Github.**

git checkout -b new\_local\_branch origin/test\_branch

**Q.5 How to push local branch in our computer to GitLab or GitHub by changing the name of local branch to master branch in GitLab or GitHub.**

We have to first push the local branch into remote repo and then set the upstream branch.

git push -u origin your\_local\_branch:master