File Downloader
OS PROJECT

-BY
TEJASHREE-COE18B016
DHRUV SAWARKAR -CED18I017
HALDHAR DWIVEDI-CED18I022
HARINI.V-CED18I023
VIJAY MEENA-CED18I057

Note: Code of our project can be downloaded from github it contains video also which demonstrate the working of our project also to paste file link in gui of our project you can use CTRL+V instead of typing

MULTI-THREADED DOWNLOADER:

Description:-

Multi-threaded downloader is used to download any type of files(pdf,mp4,etc).

First of all, what Multi-Threading download does is, it creates multiple threads, which download the file from different starting positions, which tries to utilize maximum power of your internet connection.

"Threads" when downloading refers to the amount of synchronous downloads going on at one time. Three threads means that it's splitting the file in three parts, downloading those three parts at the same time and then merging them together once all the parts have finished downloading. This is useful for sites that limit your download speed.

Modules Used:-

Request

The requests module allows you to send HTTP requests using Python. The HTTP request returns a Response Object with all the response data (content, encoding, status, etc).

Code:-

```
from tkinter import ttk
from tkinter import *
import os
from tkinter import messagebox as m_box
import re
import requests
import threading
import time


n=15
```

```
histroy = dict()

#part_downloader function accepts the start and end byte index
def part_downloader(start,end,url,filename):

  #file_headers contains dictionary with the range of the bytes specified to be downloaded
  file_headers = {'Range':'bytes=%d-%d'%(start,end)}

  #request to get the response object  with specified headers
  r = requests.get(url,headers=file_headers,stream=True)

  #open the file for reading and writing in binary mode
  with open(filename,"r+b") as fp:
     #goto the required positon with seek
     fp.seek(start)
     #dump  the contents of the file in the file
     fp.write(r.content)


def download(url,thread_count):

  #send a head request to the url and save it in r which is the response object
  r = requests.head(url)

  filename =file_name_bfr_click.get()#get the filename from filename entry widget
```

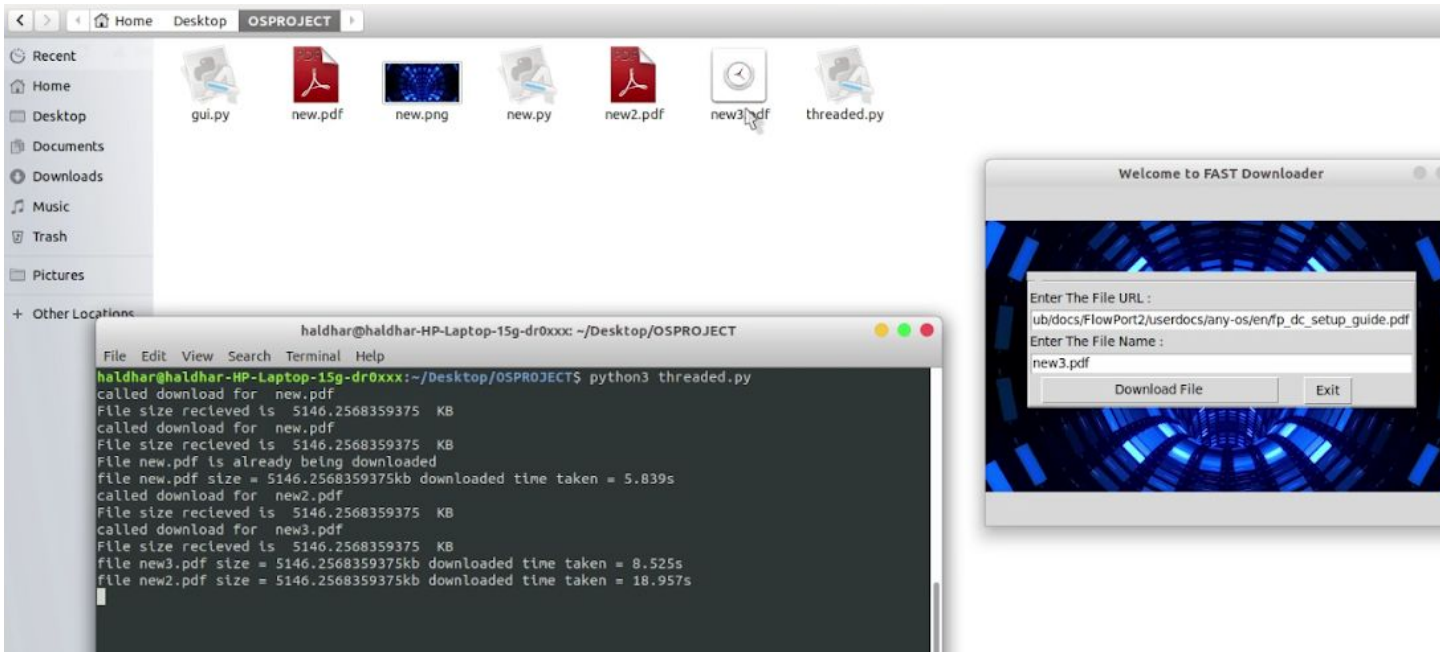
```
 #if the filename entry widget was left empty then use the default name of
file
 if(len(filename) == 0):
    filename = str(str(url).split('/')[-1])
 print("called download for",filename,"from",url)

 #from the headers get the content-length
 try:
    size = int(r.headers['content-length'])
    print("File size recieved is ",str(size/1024)," KB")
 except:
    print("Invalid url or File cannot be downloaded")
    return
 #check if we are not downloading the same file under the same name
 #so that we dont have any conflicts
 try:
    if histroy[filename] is 'downloading':
       print("File %s is already being downloaded"%(filename))
       return
 except:
    histroy[filename] = 'downloading'

 #calculate the fraction of bytes which we need for each thread
 fraction = int(size/int(thread_count))

 #create file and write empty characters in each and every
 #byte for the required size
 with open(filename,"w") as fp:
    fp.write("\0" * size)
    fp.close
```

```
  start_time = time.time()

  #create list for holding the threads which we created
  threadlist = list()

  #create threads for downloading the parts of the file
  for i in range(int(thread_count)):

     start = i*fraction
     end = start + fraction
     #create thread with part_downloader function and
     #give the url and the parameters for the file headers to download
     t =
threading.Thread(target=part_downloader,kwargs={'start':start,'end':end,
'url':url,'filename':filename})
     t.start()
     threadlist.append(t)

  #wait for all threads to finish
  for t in threadlist:
     t.join()
  #marks the file as downloaded
  histroy[filename] = 'downloaded'

  print("file " + str(filename)+" size = "+str(size/(1024))+"kb threads
used",
  str(thread_count)+ " downloaded time taken = " +
str(round(time.time()-start_time,3)) +"s")
```

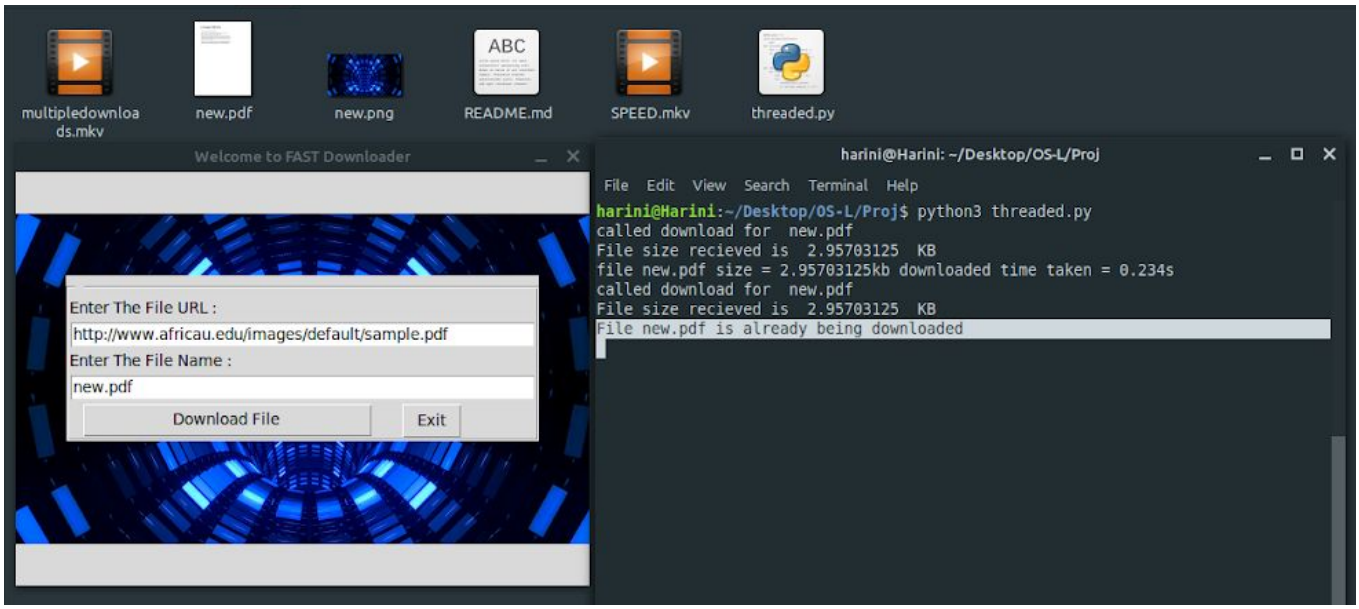
```
#runner is here
#get the url from entry and give it to download function
#with the number of threads as n
def runner_download() :
    file_url = url.get()
    download(file_url,n)

#when clicking the download button create
#new thread and call the runner download function
def onClick():
  t = threading.Thread(target =runner_download)
  t.start()

#function for exiting the program
def Exit():
  print("Exiting...")
  win.destroy()
  sys.exit()

#Create the main frame
win = Tk()
win.title("Welcome to FAST Downloader")
bgimage = PhotoImage(file ="new.png")
Label(win,image=bgimage).place(relwidth=1,relheight=1)
win.geometry("500x360")
win.minsize(500,360)
win.maxsize(500,360)

#Create inner Frame for inputs and button
frame = ttk.LabelFrame(win, width=290)
```

```
frame.pack(padx=30, pady=90)

urlLB = ttk.Label(frame, text="Enter The File URL : ")
urlLB.grid(row=0, column=0,sticky=W)
url = StringVar()
#Entry widget for url
url_entry = ttk.Entry(frame, width=50, textvariable=url)
url_entry.grid(row=1, columnspan=4, padx=2, pady=3)

fileNameLB = ttk.Label(frame, text="Enter The File Name : ")
fileNameLB.grid(row=2, column=0,sticky=W)

file_name_bfr_click = StringVar()
file_name = ttk.Entry(frame, width=50, textvariable=file_name_bfr_click)
file_name.grid(row=3, columnspan=4, padx=2, pady=3)
#Create Download Button with command newThread which creates new
thread and starts downloading the file from url
DownloadBT = ttk.Button(frame, width=30, text="Download File",
command=onClick)
DownloadBT.grid(row=4,column=0)

#Button for Exiting the application
ExitB = Button(frame,text='Exit',command = Exit)
ExitB.grid(row=4,column=1)


win.mainloop()
```

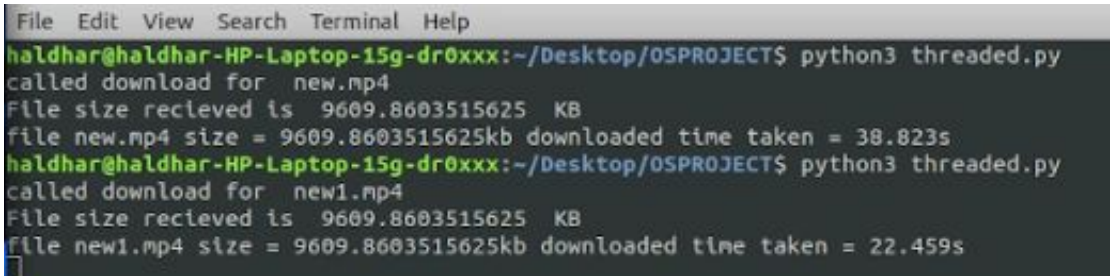
OUTPUT:-



As you can see,we can download multiple files simultaneously using multiple threads.



And our project's uniqueness is that ,a message will be popped that it's already been downloaded if we try to download again.



As you can see, when no of threads is 10 , time taken to download a file named new.mp4 is 38.823s
Whereas, when no of threads is 15 , time taken for downloading the same file is 22.459s
From here,we can conclude that
Multi-threading **does not** speed anything up. If a routine requires X clock-cycles to run: it will take X clock-cycles; whether on 1 thread or many threads.
What multithreading does do is it just allows tasks to run concurrently (at the same time).

Conclusion:-

Thus we can conclude that though many multi-threaded downloaders are available ,Our project's uniqueness is that a message will be popped when we try to download an already existing file and multiple files are downloaded simultaneously using multiple threads.
