```
##########################################################################################################################################
# This youtube downloader is made by '''Himanshu Mahajan '''
# Here you can download any video from youtube just by typing its url
##########################################################################################################################################
from tkinter import *                            # importing tkinter for gui interface
from tkinter import messagebox                   # importing message box for showing some information
import pafy                                      # importing pafy for working with youtube
import socket                                    # importing socket for getting the info about secure network connection (Here)
import re                                        # importing regex for multiple spliting only (Here)
from PIL import Image, ImageTk                   # importing for working with images
from io import BytesIO                           # importing for reading the bytes of a image(thumbnail) (Here)
import requests                                  # importing for opening a site related to thumbnail (Here)
from tkinter.filedialog import askdirectory      # importing for getting the loction for saving (Here)
from tkinter import ttk                          # importing for checkbuttons only (Here)
from pytube import YouTube                       # importing for downloading captions and description of video (Here)
##########################################################################################################################################
class Main :                                                                                     # Main class for starting program
    def __init__(self):                                                                          # __init__ for initiating Main class
        try:                                                                                     # using try and except approach
            self.__win = Tk()                                                                    # this is main window
            self.__system_width = self.__win.winfo_screenwidth()                                 # getting screen width
            self.__system_height = self.__win.winfo_screenheight()                               # getting screen height
            self.__win.geometry(f"{self.__system_width}x{self.__system_height}")                 # giving geometry to main window
            self.__win.title("YouTube Downloader By - 'Himanshu Mahajan'")                       # title for main window
            self.__win.config (bg = 'black')                                                     # making background black
            self.__win.state("zoomed")                                                           # making window to open in maximize state
            self.__win.attributes('-alpha',0.9)                                                  # making window transparent
            Label(self.__win,text = 'Welcome To YouTube Downloader By - "Himanshu Mahajan"',
             bg = "black" ,fg = "red", font = ('MV Boli',25,'bold','underline')).pack(side = TOP)# label for giving heading to window
            self.__menubar = Menu(self.__win)                                                    # defining a menubar
            self.__about = Menu(self.__menubar,tearoff= 0)                                       # menu inside menubar
            self.__menubar.add_cascade(label='Made By - "Himanshu Mahajan"',menu = self.__about,state = DISABLED)# adding casacde just for showing name
            Label(self.__win,text = "Enter The Url Of Video Or 11 Character Video ID", bg = "black" ,fg = "red"
            , font = ('MV Boli',20,'bold')).place(x =self.__system_width/7 , y = self.__system_height/6) # label for ponting to entry box
            self.__url = Entry(self.__win, width = 49 , font = ('MV Boli',15))                           # entry box for pasting link for video
            self.__enter = Button (self.__win, text = "Done", fg = "red",bg = "black",command = lambda: self.__get_entry()
                , font = ('MV Boli',25,'bold'),bd = 0,width = 20,activebackground = 'black')     # button to press when pasted link in entry box
            self.__normal = Button (self.__win, text = "-->Normal", fg = "red",bg = "black" , command = lambda: self.__mode('normal')
                , font = ('MV Boli',25,'bold'),bd = 0,activebackground = 'black')                # button for normal video and audio both
            self.__video = Button (self.__win, text = "-->Only Video", fg = "red",bg = "black" , command = lambda: self.__mode('video')
                , font = ('MV Boli',25,'bold'),bd = 0,activebackground = 'black')                # button for only video to download
            self.__audio = Button (self.__win, text = "-->Only Audio", fg = "red",bg = "black" , command = lambda: self.__mode('audio')
                , font = ('MV Boli',25,'bold'),bd = 0,activebackground = 'black')                # button for only audio to download
            Label(self.__win,text = "!!! Make Sure That You Are Connected To A Fast Network Connection !!!\nOtherWise This Will Cause App To\
             Stop Responding",fg="red",bg="black" ,font=('MV Boli',18,'bold')).pack(side=BOTTOM)# labeling at bottom for surety in nework connection
            self.__enter.place(x =(self.__system_width/7)+110, y = (self.__system_height/6)+120) # placing enter button
            self.__url.focus()                                                                   # getting the focus to entry box
            self.__url.place(x =self.__system_width/7 , y = (self.__system_height/6)+50)         # placing the entry box
            self.__win.config(menu = self.__menubar)                                             # configuring the menubar for main window
            self.__win.mainloop()                                                                # calling the mainloop event
        except:                                                                                  # if any unknown error occured
            None                                                                                 # do nothing
##########################################################################################################################################
# Defining the function for getting the typed url from entry box which will be called when enter button is pressed
##########################################################################################################################################
    def __get_entry(self):                                              # defining get_entry function with class parameter self
        self.__entry = self.__url.get()                                 # getting the entered url from entry box
        try:                                                            # using try and except approach
            if self.__entry == "":                                      # if nothing is written inside entry box
                messagebox.showwarning("YouTube Downloader By - 'Himanshu Mahajan'","Please Enter The Url For Video") # showing warning to add url
            else :                                                      # if url is given
                self.__enter['state'],self.__url['state'] = DISABLED ,DISABLED # making enter button and url box state as disabled
                Label(self.__win,text = "Mode", bg = "black" ,fg = "red", font = ('MV Boli',30,'bold')
                    ).place(x =100 , y = self.__system_height-470 )     # placing the label for showing modes
                self.__normal.place(x = 50,y=self.__system_height-400 ) # placing normal button defined above
                self.__video.place(x = 50,y =self.__system_height-345 ) # placing video button defined above
                self.__audio.place(x = 50,y =self.__system_height-290 ) # placing audio button defined above
        except:                                                         # if any unknown error occured
            None                                                        # do nothing
##########################################################################################################################################
# Defining the function for mode of video to download which will be called when the mode is selected for the video
##########################################################################################################################################
    def __mode(self,mode):                                             # defining mode function with class parameter self and mode to download
        self.__ipaddress = socket.gethostbyname(socket.gethostname())  # getting the state of network connection
        if self.__ipaddress == "127.0.0.1":                            # if the computer is not connected to a stable network connection
            messagebox.showwarning("YouTube Downloader By-'Himanshu Mahajan'","Make Sure You Are Connected To A Stable Network Connection")#show warning
            self.__enter['state'],self.__url['state'] = NORMAL,NORMAL  # making states normal
        else:                                                          # if computer is connected to a stable network connection
            try:                                                       # using try and except approach
                self.__d = False                                       # a variable for further use
                self.__v = pafy.new(self.__entry)                      # getting the streams of video
                self.__win1 = Toplevel(self.__win)                     # a toplevel window inside main window
                self.__win1.state("zoomed")                            # for always maximised screen
                self.__win1.geometry(f"{self.__win.winfo_screenwidth()}x{self.__win1.winfo_screenheight()}") # setting geometry for toplevel window
                self.__win1.attributes('-alpha',0.9)                   # making transparent
                self.__win1.config (bg = 'black')                      # black background
                if mode == 'normal':                                   # if normal type is needed
                    self.__media_type = 'NORMAL'                       # storing normal type to a variable
                    self.__stream = self.__v.streams                   # getting normal streams of video
                elif mode == 'video':                                  # if only video type is needed
                    self.__media_type = 'Only Video'                   # storing only video type to a variable
                    self.__stream = self.__v.videostreams              # getting only video streams
                elif mode == 'audio':                                  # if only audio type is needed
                    self.__media_type = 'Only Audio'                   # stroing only audio type to a variable
                    self.__stream = self.__v.audiostreams              # getting only audio streams
##------------------------------------------------------## Labels ##-------------------------------------------------------------------##
                Label(self.__win1,text = "|"*50, wraplength =1,font = ('arial',20), bg = "black"
                    ,fg = "red").place(x = self.__win.winfo_screenwidth()/2,y = 50)          # Label for dividing screen to half
                Label(self.__win1,text = "Details", font = ('MV Boli',30,'bold','underline')
                    , bg = "black" ,fg = "Darkorange").pack(side= TOP)                       # label for showing details as heading on top
                Label(self.__win1,text = f"Author     -> {self.__v.author}", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "red").place(x = 5,y = 60)                          # label for author name of video
                Label(self.__win1,text = f"Title      -> {self.__v.title[:25]}...", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "darkorange").place(x = 5,y = 90)                   # label for title of video
                Label(self.__win1,text = f"Duration   -> {self.__v.duration}", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "darkorange").place(x = 5,y = 120)                  # label for duration of video
                Label(self.__win1,text = f"Media Type -> {self.__media_type}", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "red").place(x = 5,y = 150)                         # label for media type of video
                Label(self.__win1,text = f"Views      -> {round(self.__v.viewcount*.001,2)} K", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "darkorange").place(x = 5,y = 180)                   # label for views of video
                Label(self.__win1,text = f"Likes      -> {round(self.__v.likes*.001,2)} K", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "red").place(x = 5,y = 210)                          # label for likes of video
                Label(self.__win1,text = f"Dislikes   -> {round(self.__v.dislikes*.001,2)} K", font = ('MV Boli',15,'bold')
                    , bg = "black" ,fg = "darkorange").place(x = 5,y = 240)                   # label for dislikes of video
                self.__size = Label(self.__win1,text = "Size        -> ", font=('MV Boli',15,'bold'), bg = "black" ,fg = "red")# label for size of video
                self.__size.place(x = 5,y = 270)                                              # placing the size label
                Label(self.__win1,text = '-'*100, font = ('arial',30), bg = "black" ,fg = "red").place(x = 5,y = 300) # label for dividing screen to half
                Label(self.__win1,text = f" Quality ", font = ('MV Boli',25,'bold'), bg = "black" ,fg = "red").place(x = 150,y = 340) # label for quality
                self.__imagedata = requests.get(self.__v.bigthumbhd).content               # getting the image data in form of bytes from url of thumbnail
                self.__photo = ImageTk.PhotoImage(Image.open(BytesIO(self.__imagedata)))  # reading the image data from bytes to image
                Label(self.__win1,image = " Thumbnail ", font = ('MV Boli',30,'bold'), bg = "black"
                    ,fg = "red").place(x = (self.__win.winfo_screenwidth()/2)+120,y = 60 )# label for showing thumbnail heading
                Label(self.__win1,image = self.__photo,height = 350
                    ,width = 450).place(x= (self.__win.winfo_screenwidth()/2)+20,y = 130 )# label for showing the image(thumbnail)
##------------------------------------------------------------------------------------------------------------------------------------##
                self.__list =[]                                                         # empty list
                self.__list2 = []                                                       # empty list
                self.__list_variable = StringVar(value= self.__list)                    # variable for list box with value as list
                self.__list_box = Listbox(self.__win1,listvariable = self.__list_variable,width = 80,height = 13) # list box for showing video details
                for quality in self.__stream:                                           # iterating for getting quality
                    self.__list2.append(quality)                                        # appending quality to an empty list
                    qual = [str(x)for x in re.split('\:|@| ',str(quality))]            # splittng the quality from multiple splitters using regx
                    self.__list.append(f"{qual[0]}     ,     Extension    :     {qual[1]}     ,     Quality     :     {qual[2]}")# appending to list for listbox
                self.__list_box.bind('<<ListboxSelect>>',lambda e :self.__Quality (e))  # binding the event of selection for list box
                self.__include = None                                                   # variable for storing value for including video description
                self.__include1 = None                                                  # variable for storing value for including video captions
##-----------------------------------------## Function For Including Captions And Description ##----------------------------------------------##
                def clicked(a):                                         # function for changing variable values for desc and caption with a as choice
                    try:                                                # using try and except approach
                        if a == "ds":                                   # if choice is descripton
                            self.__include = True if self.__variable.get() == 1 else False    # changing variable value for descriptions
                        if a == "cm":                                   # if choice is caption
                            self.__include1 = True if self.__variable1.get() == 1 else False  # changing variable value for captions
                        if self.__include == True or self.__include1 == True:                 # if any of two is required
                            self.__download.place(x = 150,y = 650)                            # placing download button to download any of these
                        else:                                                                 # if nothing is needed
                            self.__download.place_forget()                                    # unplacing the download button
                    except:                                                                   # if any unknown error has occured
                        None                                                                  # do nothing
##-----------------------------------------------------------------------------------------------------------------------------------------##
                self.__variable = IntVar()                                                    # variable for description check button
                self.__variable1 = IntVar()                                                   # variable for captions check button
                self.__describe = ttk.Checkbutton(self.__win1,text ='Include Description',variable = self.__variable,width = 32
                    ,command = lambda :clicked("ds"))                                         # check buttton for description
                self.__describe.place(x = 20,y = 620)                                         # placing check button for description
                self.__caption = ttk.Checkbutton(self.__win1,text ='Include Captions',variable = self.__variable1,width = 37
                    ,command = lambda :clicked("cm"))                                         # check button for captions
                self.__caption.place(x = 257,y = 620)                                         # placing check button for captions
                self.__download = Button (self.__win1,text = " Download ", fg = "red",bg = "black" , command = lambda : self.__Download()
                    , font = ('MV Boli',25,'bold'),bd = 0,activebackground = 'black')         # defining download button
                if len(self.__entry)<12:                                                      # if the user has provided 11 digit code for video
                    self.__entry = f'https://youtube.com/watch?v={self.__entry}'              # creating url from 11 digit code
                self.__list_variable.set(self.__list)                                         # setting list to the listbox variable
                self.__list_box.place( x = 20,y= 400)                                         # placing the listbox
##-----------------------------------------------## Deefining Function for closing the window ##-------------------------------------------##
                def close(e):                                                    # function close with e as event parameter
                    self.__enter['state'],self.__url['state'] = NORMAL,NORMAL    # making states to normal for button and entry in main window
                    self.__win1.destroy()                                        # destroying the toplevel window
##-----------------------------------------------------------------------------------------------------------------------------------------##
                self.__win1.bind('<Alt-F4>',lambda e:close(e))                   # binding the toplevel window close option
                self.__win1.focus()                                              # setting the focus to toplevel window
                self.__win1.protocol("WM_DELETE_WINDOW",lambda :close("e"))      # binding cross option of window to close function
                self.__win1.mainloop()                                           # calling the mainloop for toplevel window
            except:                                                              # if any unknown error has occured
                messagebox.showwarning("YouTube Downloader By - 'Himanshu Mahajan'","Please Enter Valid Url Or Code") # showing warning
                self.__enter['state'],self.__url['state'] = NORMAL,NORMAL        # normaling the states of entry box and button
#############################################################################################################################################
# Defining the Quality function for finalising the selected quality which will be called whenever a item is selected in listbox
#############################################################################################################################################
    def __Quality (self,event):                      # function for finalising the quality with self as class parameter and event as event parameter
        try:                                         # using try and except approach
            self.__d =True                           # variable stored true here means downloading process is active
            self.__selected = self.__list_box.curselection()[0]  # getting the current selected item from listbox
            self.__quality = self.__stream[self.__selected]      # getting the item from sream list at selected index
            self.__size['text'] = f"Size        -> {round(self.__quality.get_filesize()*.000001,2)} MB" # showing the size of selected quality in MB
            self.__label_t = Label(self.__win1,text = '', font = ('MV Boli',15,'bold'), bg = "black" ,fg = "red") # label for total size of video
            self.__label_t.place(x = (self.__win1.winfo_screenwidth()/2)+10,y = 500)                 # placing the label
            self.__label_d = Label(self.__win1,text = '', font = ('MV Boli',15,'bold'), bg = "black" ,fg = "darkorange") # label for downlowded size
            self.__label_d.place(x = (self.__win1.winfo_screenwidth()/2)+10,y = 540)                 # placing the label
            self.__label_s = Label(self.__win1,text = '', font = ('MV Boli',15,'bold'), bg = "black" ,fg = "red") # label for speed
            self.__label_s.place(x = (self.__win1.winfo_screenwidth()/2)+10,y = 580)                 # placing the label
            self.__label_e = Label(self.__win1,text = '', font = ('MV Boli',15,'bold'), bg = "black" ,fg = "darkorange") # label for estimated time
            self.__label_e.place(x = (self.__win1.winfo_screenwidth()/2)+10,y = 620)                 # placing the label
            self.__label_p= Label(self.__win1,text = '', font = ('MV Boli',10,'bold'), bg = "black" ,fg = "red")  # label for percentage downloaded
            self.__download.place(x = 150,y = 650)                                                   # placing the download button
            self.__label_p.place(x = (self.__win1.winfo_screenwidth()/2)+10,y = 660)                 # placing the label
        except:                                                                                      # if any unknown error occured
            None                                                                                     # do nothing
###############################################################################################################################################
# Defining Download function for downloading the video finally which will be called when download button is pressed
###############################################################################################################################################
    def __Download(self):            # Download function for downloading the video with self as class parameter
        try:                         # using try and except approach
            self.__want = messagebox.askyesno(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'",
                message = "Start Downloading ? \n Once Started You Won't Be Able To Cancel It !")  # asking for download last time
            if self.__want == False: # if the user don't want to download
                None                 # doing nothing
            else:                    # else start downloading process
                self.__directory = askdirectory(parent = self.__win1,title ="Choose Location By - 'Himanshu Mahajan'" )  #asking the directory for saving
                self.__progressbar = ttk.Progressbar(self.__win1,orient = 'horizontal',length = 460,mode = 'determinate')# defining progressbar
                if self.__directory == "":      # if the directory is not selected
                    None                        # do nothing
                else:                           # else directory is defined
                    if self.__include == True:  # if the user want to include description
                        try:                    # using try and except approach
                            messagebox.showinfo(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'"
                                ,message = "Wait Downloading Video Description")      # showing that description is downloading
                            self.__desc = YouTube(self.__entry).description           # getting the description for the video
                            self.__content = [str(x)for x in self.__desc.split('\n')] # creating the list for lines in description
                            # opening the file made by program itself to write the content of description
                            with open (rf"{self.__directory}/description{self.__v.title[:5]}.txt" ,'w',encoding = 'utf-8',errors = 'ignore')as f:#opening
                                for lines in self.__content:                          # using iterator for getting each line from list of description
                                    f.write(lines+'\n')                               # writing the line to opened file and inserting a new line at end
                        except:                                                       # if the description is not available
                            messagebox.showinfo(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'"
                                ,message = "This Video Has A Problem With Description")# showing info that description is not available
                    if self.__include1 == True:             # if the user want to download the captions
                        try:                                # using try and except approach
                            messagebox.showinfo(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'"
                                ,message = "Wait Downloading Video Captions") # showing message that captions are downloading
                            self.__cap = YouTube(self.__entry).captions.get_by_language_code('en').generate_srt_captions()# generating captions for video
                            # opening the file made by program itself to write the content of captions
                            with open (rf"{self.__directory}/captions{self.__v.title[:5]}.txt" ,'w',encoding = 'utf-8',errors = 'ignore')as f: # opening
                                for lines in self.__cap:    # getting the lines from the list of captions
                                    f.write(lines)          # writing the lines to the opened file
                        except:                             # if the captions are not available for the video
                            messagebox.showinfo(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'"
                                ,message = "This Video Has No Inbuilt Captions") # showing message that captions are not available for the video
                    if self.__d == True:                    # if thr downloading process variable is true
                        messagebox.showinfo(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'"
                            ,message = " Downloading Video ") # showing message that video downaload is started
                        self.__win1.update_idletasks()        # updating the window
                        self.__progressbar.place(x = (self.__win1.winfo_screenwidth()/2)+50 ,y = 660)   # placing the progressbar
                        self.__pval = 0                       # setting the initial value to 0
                        self.__progressbar['value'] = self.__pval # changing the value for progressbar
                        self.__progressbar.update_idletasks()     # updating the idle tasks for progressbar
                        self.__win1.update_idletasks()            # updating idle tasks for window
                        self.__quality.download(filepath=self.__directory,quiet=True,callback=self.__progress ) # started download process by callbacking
                        messagebox.showinfo(parent = self.__win1,title = "YouTube Downloader By - 'Himanshu Mahajan'"
                            ,message = "Video Downloaded")        # showing info when the video is downloaded
                        self.__enter['state'],self.__url['state'] = NORMAL,NORMAL # normaling the states for entry box and enter button
                        # changing the label texts to empty strings
                        self.__label_t['text'],self.__label_d['text'],self.__label_s['text'],self.__label_e['text'],self.__label_p['text']='','','','',''
                        self.__progressbar.destroy()              # destroying the progressbar
                        self.__win1.update_idletasks()            # updating idle tasks for window
                        self.__d = False                          # setting downloading process to False
                    messagebox.showinfo(parent=self.__win1,title="YouTube Downloader By - 'Himanshu Mahajan'",message="Done ! ") # showing Done message
        except:                                                   # if any error occured
            None                                                  # do nothing
################################################################################################################################################
# Defining progress function for showing the download process it is a callback function called after each change in download process
################################################################################################################################################
    def __progress(self,total,downloaded,ratio,speed,time): # progress function with parameters as default callback options
        try:                                                # using try and except approach
            self.__win1.update_idletasks()                  # updating the window
            self.__progressbar.update_idletasks()           # updating the progerssbar
            self.__pval = int(ratio*100)                    # downloading percentage
            self.__progressbar['value'] = self.__pval       # changing the value for progressbar
            # changing the text for all the labels to show change in progress
            self.__label_t['text'],self.__label_d['text']=f"Total  -> {round((total*.000001),2)} MB",f"Downloaded -> {round((downloaded*.000001),2)} MB"
            self.__label_s['text'],self.__label_e['text']=f"Speed   -> {round((speed),0)} Kbps",f"Estimated Time   -> {time} secs"
            self.__label_p['text']=f"{int(ratio*100)}%"
            self.__progressbar.update_idletasks()           # updating the progressbar
            self.__win1.update_idletasks()                  # updating the window
        except:                                             # if any unknown error has occured
            None                                            # do nothing
#################################################################################################################################################
# Main Program Starts Here
#################################################################################################################################################
if __name__ == "__main__":   # __name__ is always equal to __main__ in python if no error has occured
    try:                     # using try and except approach
        window = Main()      # creating the object of the class
    except:                  # if any error has occured
        None                 # do nothing
#################################################################################################################################################
```