```
#########################################################################################################################################################
# This Image to Pdf Is Made by ""Himanshu Mahajan""
# Here You can Make A Pdf Using images with all the properties you want like editing,compressing,sizing etc.
#########################################################################################################################################################
from tkinter import *                           # importing for gui support
from tkinter.filedialog import *                # importing for dialogs like opening , saving etc.
from tkinter import ttk                         # importing for special themes and widgets in tkinter
from tkinter import messagebox,scrolledtext     # messagebox for showing some message and scrolledtext for showing instructions (Here)
from PIL import Image,ImageTk                   # Image for working with images and ImageTk for working with images for gui
import ttkthemes                                # for special themes in ttk
import os                                       # for working with operating system
from fpdf import FPDF                           # for converting images to pdf (Here)
import sys                                      # for working with system
#########################################################################################################################################################
class IMG_TO_PDF:                                       # using object oriented programming approach,class for main working
    def __init__ (self):                                # defining initiation function which will be called itself when a class instance is made
        self.win = Tk()                                 # main window for program
        self.win.config(bg="black")                     # black background for window
        self.screen_width = self.win.winfo_screenwidth()     # getting the system screen width
        self.screen_height = self.win.winfo_screenheight()   # getting the system screen height
        self.win.geometry(f"{self.screen_width}x{self.screen_height}") # defning the geometry for main window
        self.win.attributes('-alpha',0.98)              # making window transparent
        self.win.state('zoomed')                        # making window maximised
        self.win.title("Img To Pdf By - Himanshu Mahajan")   # title for main window
        Label(self.win,text="···"*100,font=('arial 35 bold'),bg='black',fg="red").place(x=0,y=self.screen_height//2+90) # label dividing half vertically
        Label(self.win,text="¦¦"*100,font=('arial 10 bold'),bg='black',fg="red",wraplength=1).place(x=self.screen_width//2,y=0) # divide half horizontally
        self.frame = Frame(self.win,bg = 'black',height = self.screen_height,width = self.screen_width//2) # frame on the right half for showing image
        self.frame.place(x = self.screen_width//2+10,y = 0)  # placing the frame
        self.image,self.image1 = Image.open(self.get_path("add_files.png")),Image.open(self.get_path("circle-cropped.png")) # opening images using a func
        self.add = ImageTk.PhotoImage(self.image)           # variable storing image
        self.add_resized = ImageTk.PhotoImage(self.image.resize((self.image.width//3,self.image.height//3))) # resizing the image for add files
        self.add_resized1 = ImageTk.PhotoImage(self.image1.resize((self.image.width-80,self.image.height-80)))# same for add folder
        self.add_label=Label(self.frame,image=self.add,bg='black',height=self.screen_height,width=self.screen_width//2) # label with add files image
        self.add_label.pack(fill= BOTH,expand= True)        # packing in the right frame fully
        self.style = ttkthemes.themed_style.ThemedStyle(theme="equilux") # setting theme for ttk widgets ,special("equilux")
        self.style.configure('Treeview',background = "black",foreground= "black",fieldbackground= "black",rowheight=20) # configuring style for treeview
        self.left_frame = Frame(self.win)                   # frame for packing treeview so that scroll bar could get attached to treeview
        self.left_frame.place(x= 0,y = 0)                   # placing the frame
        self.style.map("Treeview",background=[('selected',"gray99")]) # changing the selection colour(without this step bg,fg,fbg can't be seen changed)
        self.tree_view=ttk.Treeview(self.left_frame,columns=('Name','Location'),style='Treeview',height=10,selectmode='browse')# defining treeview
        self.tree_view.pack(fill =Y,side = LEFT)            # packing treeview on leftframe with side as left to ensure vacant space for scrollbar
        self.tree_view.column('#0',width = self.screen_width//30)      # defining columns for treeview (this is a inbuilt column for a treeview)
        self.tree_view.column('Name',width = self.screen_width//5)     # column for Name of the file
        self.tree_view.column('Location',width = self.screen_width//4) # column for Location of the file
        self.tree_view.heading('#0',text = "No.")                      # providing Headings to columns (This is inbuilt column )
        self.tree_view.heading('Name',text = "Name")                   # heading for name column
        self.tree_view.heading('Location',text = "Location")           # heading for location column
        self.scroll = ttk.Scrollbar(self.left_frame,orient = VERTICAL) # defining the scrollbar for treeview on left frame
        self.scroll.configure(command = self.tree_view.yview)          # configuring scrollbar for treeview's vertical(Y) scroll
        self.scroll.pack(fill = Y,side = RIGHT)                        # packing the scrollbar on left frame with side as right to the treeview
        self.tree_view.configure(yscrollcommand = self.scroll.set)     # setting the vertical scroll command for treeview
        self.tree_view.tag_configure('oddrow' , background ="dodger blue")   # tagging the color for row,oddrow as tag name and coloured bg
        self.tree_view.tag_configure('evenrow',background = "medium purple") # same for evenrow as tag name
        self.add_image = Button(self.win,bd = 0,image = self.add_resized , height = 80,width = 80 , bg = "black",activebackground= "black",
            command= lambda :self.Add_image("event"))                  # defining button for adding image
        self.add_image.place(x = 0,y = self.screen_height//2-150)      # placing the same
        self.add_folder = Button(self.win,bd = 0,image = self.add_resized1 , height = 75,width = 75, bg = "white" ,activebackground= "black"
            ,command= self.Add_folder)                                 # defining button for adding folder
        self.add_folder.place(x = 100,y = self.screen_height//2-150)   # placing the same
        self.count = 0                                                 # variable for getting whether even or odd row in treeview
        self.inserted = 0                                              # variable for storing wether labels and buttons placed or not
        self.tree_view.focus_set()                                     # setting the focus to treeview
##-------------------------------------------## Binding Some of the Events ##-------------------------------------------------------##
        self.tree_view.bind("<ButtonRelease-1>",lambda e:self.show_image(e)) # show the image on right frame with release of left button on treeview
        self.tree_view.bind("<Up>",lambda up:self.show_image("up"))          # move up in the treeview,simultanously showing that image on right frame
        self.tree_view.bind("<Down>",lambda down:self.show_image("down"))    # same for move down
        self.add_label.bind("<ButtonRelease-1>",lambda e : self.Add_image(e))# add images with release of left button on right frame
##-----------------------------------------------------------------------------------------------------------------------------------##
        self.how_to_use()                                                     # calling this function for adding instructions to the scrolledtext
        self.win.iconbitmap(self.get_path("icon.ico"))                        # icon for main window
        self.win.mainloop()                                                   # calling the mainloop for gui window
#########################################################################################################################################################
# Defining Function for getting the path of the image used for buttons etc. after converting .py to exe
#########################################################################################################################################################
    def get_path(self,filename):                            # get_path will get the path of images used in script , called when program starts
        if hasattr(sys, "_MEIPASS"):                        # checking if the script is converted to exe or not
            return f'{os.path.join(sys._MEIPASS, filename)}' # if yes then returing the path of the asked image or file
        else:                                               # if not converted to exe
            return f'{filename}'                            # then simply returning the file name given above
#########################################################################################################################################################
# Defining Function for adding Images
#########################################################################################################################################################
    def Add_image(self,event):              # Add_image Will add the image to the treeview and called on event(ButtonRelease-1),add_image button press
        types = [('JPG files', '*.jpg'),('PNG files', '*.png'), ('JPEG files', '*.jpeg'), ('All files', '*')] # types to be opened
        self.ask = askopenfilenames(title = "Open File By - Himanshu Mahajan",filetypes = types) # dialog for opening files with types (returns list)
        if self.ask == "":                  # if the user canceled the process or not opened any file
            None                            # do nothing
        else:                               # if user added files to open
            self.insert("image")            # calling the insert function for inserting image to treeview
#########################################################################################################################################################
# Defining Function for adding a folder
#########################################################################################################################################################
    def Add_folder(self):                   # Add_folder will add all the items in a folder to the treeview,called on click of add_folder button
        try:                                # using try and except approach
            self.dir = askdirectory(title = "Open Folder By - Himanshu Mahajan") # asking the directory for the folder
            self.ask = os.listdir(self.dir) # getting all the items from the specified path
            self.insert("folder")           # calling the insert function for inserting folder to treeview
        except:                             # if the user canceled the process
            None                            # Do Nothing
#########################################################################################################################################################
# Defining  Function for inserting image and folder to the treeview
#########################################################################################################################################################
    def insert(self,what):                          # insert will add image or folder to treeview when ever add function called
        self.add_label.pack_forget()                # unpacking the add_label to clear the place for new image to show there
        self.placed = 0                             # variable for getting wether moveup,down and delete is placed or not
```

```
        for name in self.ask:                       # getting the items from the list turn wise turn
            if what =="folder":                     # if the user want to add a folder
                self.only_name=name                 # only name of the file
                name = f"{self.dir}/{name}"         # full location if the file
            else:                                   # if useer want to add images
                self.only_name = name.split("/")[-1] # only name of the file
            if self.count%2==0:                     # this insures even row
                self.tree_view.insert('','end',iid=self.count,text=self.count+1,values=(self.only_name,name),tag='evenrow')# insert to treeview(tag=even)
            else:                                   # else odd row
                self.tree_view.insert('','end',iid=self.count,text=self.count+1,values=(self.only_name,name),tag='oddrow') # insert to treeview(tag=odd)
            self.count+=1                           # increasing the count by one for next iteration
    ##-----------------------------------------## Labels And Buttons ##-------------------------------------------------------------##
        if self.inserted == 0:                          # if not inserted even once
            self.d = Image.open(self.get_path("delete_all.png")) # opening image for delete all button
            self.delete_img = ImageTk.PhotoImage(self.d.resize((self.d.width//2,self.d.height//2))) # resizing the image
            self.delete_all = Button(self.win,image = self.delete_img ,font ="gthoic 15 bold",bd = 0, height = 90,width = 90 ,fg = "red"
                ,bg = "black",activebackground= "black",command = lambda:self.remove("all"))        # button for deleting all the items in treeview
            self.delete_all.place(x=190,y = self.screen_height//2-150)                              # placing the same
            self.convert = Button(self.win,text = " CONVERT TO PDF ",font ="gthoic 15 bold",bd = 0,bg= "black",fg = "red"
                ,activebackground= "black",command = self.convert_to_pdf)                           # button for converting images to pdf
            self.convert.place(x= self.screen_width//4-100,y = self.screen_height//2+80)            # placing the same
            self.options = Label(self.win,text="OPTIONS",font=('gthoic 15 bold'),bg='black',fg="gray90") # label for options heading
            self.options.place(x=self.screen_width//4-50,y=self.screen_height//2-50)                # placing the same
            self.size = Label(self.win,text="SIZE",font=('gthoic 15 bold'),bg='black',fg="gray90")  # label for size heading
            self.size.place(x=90,y=self.screen_height//2-10)                                        # placing the same
            self.quality = Label(self.win,text="QUALITY",font=('gthoic 15 bold'),bg='black',fg="gray90")# label for quality heading
            self.quality.place(x=self.screen_width//4+75,y=self.screen_height//2-10)                # placing the same
            self.combo = ttk.Combobox(self.win,values = ('A3','A4','A5','Letter','Legal'),
            state = 'readonly',width = self.screen_width//27)                                       # defining combobox for choosing pdf size
            self.combo.current(1)                                                                   # making A4 as default highlight
            self.combo.place(x = 10,y= self.screen_height//2+20)                                    # placing combobox
            l = [int(x) for x in range(101)]                                                        # list of numbers from 0 to 100
            self.q = ttk.Combobox(self.win,values = tuple(l),state = 'readonly',width = self.screen_width//27) # defining combobox for quality
            self.q.place(x=self.screen_width//4+8,y = self.screen_height//2+20)                     # placing the same
            self.q.current(100)                                                                     # setting default quality to 100
            self.inserted = 1                                           # variable for storing wether labels and buttons placed or not
        else:
            None                                                                                    # Do Nothing
    ##-------------------------------------------------------------------------------------------------------------------------------##
    ###################################################################################################################################
    # Defining Function for Showing image to right frame and this will remove unsupported files from the treeview too
    ###################################################################################################################################
        def show_image(self,e):                     # show_image will show the image to the right frame called when event(ButtonRelease-1) will occur
            try:                                    # using try and except approach
                if self.tree_view.focus() == "":    # if nothing in treeview is selected
                    None                            # Do Nothing
                else:                               # if something is in treeview is selected
                    self.add_label.pack_forget()    # unpacking add_label
                    self.show = self.tree_view.focus() # storing the focus or selected item to a variable
                    if e == "up":                   # if the event is from up button of keyboard
                        self.show=int(self.show)-1  # changing the index of item , lower by one for moving up
                        self.show = str(self.show)  # changing the value type to string
                    if e == "down":                 # if the event is from down button of keyboard
                        self.show=int(self.show)+1  # changing the index of item , higher by one for moving down
                        self.show = str(self.show)  # changing the value type to string
                    self.path = self.tree_view.item(self.show)['values'][1] # getting the path or location of selected file from dictonary of that item
                    image =  Image.open(self.path)  # opening the selected item(if image)
                    resize = image.resize((self.screen_width//2,self.screen_height)) # resizing the image to show on right side
                    self.i = ImageTk.PhotoImage(resize)# storing the image  to a variable after converting for gui
                    self.add_label.configure(image=self.i) # replacing the image on the label with one selected
                    self.add_label.image = self.i   # setting the image to label
                    self.add_label.pack()           # packing the label again
    ##-----------------------------------------## show function for opening image for edit ##---------------------------------------##
                    def show (e):                   # show will get called on event(Double-1) and show the image with inbuilt viewer in os
                        s = os.startfile(self.path)  # starting the file with default viewer in os
                        self.add_label.pack_forget() # unpacking the image from right frame so that changes can be observed ,if made any
                        self.tree_view.selection_remove(self.show) # removing the selection from the treeview
    ##-----------------------------------------## Binding and Unbinding required events ##-----------------------------------------------##
                    self.add_label.unbind("<ButtonRelease-1>")          # unbinding ButtonRelease-1 event from add_label so that Double-1 get binded
                    self.add_label.bind("<Double-1>",lambda e:show(e))    # Binding the Double-1 Event to add_label for calling the show function
                    self.tree_view.bind("<ButtonRelease-3>",lambda move:self.move("move")) # Binding this event for arranging order ,deleting selected image
                    self.win.bind("<Escape>",lambda e:self.remove_sel(e)) # binding escape for removing the selection by calling remove_sel function
    ##-------------------------------------------------------------------------------------------------------------------------------##
            except TclError:            # this error occured when the selected item is at top or bottom of treeview and user still press up and down button
                self.show_image("event") # again calling the same function (this ensure that we dont loose selection in treeview)
            except:                     # this error will occur when the selected item is not supported by Image for opening (this will remove other files )
                messagebox.showwarning("Img To Pdf By - Himanshu Mahajan","This File Is Not Supported") # showing warning
                self.tree_view.delete(self.tree_view.focus())                                       # deleting the file which is not supported
    ###################################################################################################################################
    # Defining Function for moving up,down,delete selected item from treeview when right click is done
    ###################################################################################################################################
        def move(self,event):                   # move will get call when event(ButtonRelease-3) occured and will offer rearranging and deleting selected
    ##-----------------------------------------## Up Function ##-------------------------------------------------------------------##
            def Up():                           # function for moving selected item up,called on click of up_button
                rows = self.tree_view.selection() # getting the selected item from treeview
                for row in rows :               # iterating over selected rows turn wise turn
                    self.tree_view.move(item=row,parent=self.tree_view.parent(item=row),index=self.tree_view.index(item=row)-1) # moving up by decreasing 1
    ##-----------------------------------------## Down Function ##-----------------------------------------------------------------##
            def Down():                         # function for moving the selected item down ,called on click of down_button
                rows = self.tree_view.selection() # getting the selected item from treeview
                for row in reversed(rows) :     # this time its reversed iteration for moving down
                    self.tree_view.move(item=row,parent=self.tree_view.parent(item=row),index=self.tree_view.index(item=row)+1) # moving down by increasing 1
    ##-------------------------------------------------------------------------------------------------------------------------------##
            if self.tree_view.focus() == "":    # if nothing is selected
                None                            # do nothing
            else:                               # if the user has selected an item
                try:                            # using try and except approach
                    self.tree_view.unbind("<ButtonRelease-3>")                                  # unbinding event(ButtonRelease-3) as work is done
                    if self.placed==0:                                                          # if these buttons are not placed
                        self.style.map("Treeview",background=[('selected',"navajo white")])     # changing the selected row color
                        self.delete = Button(self.win,text = "Delete 🗑" ,font ="gthoic 15 bold",bd = 0,fg = "gray99",bg = "black"
                        ,activebackground= "black",command = lambda:self.remove("e"))           # defining button for deleting selected item
                        self.delete.place(x=self.screen_width//2-120,y = self.screen_height//2-90) # placing the same
                        self.up_button = Button(self.win,text = "Move Up" ,font ="gthoic 15 bold",bd = 0,fg = "gray99",bg = "black"
                        ,activebackground= "black",command=Up)                                  # defining the button for moving up selected item
```

```
                self.up_button.place(x= self.screen_width//2-120,y = self.screen_height//2-150) # placing the same
                self.down_button = Button(self.win,text = "Move Down" ,font ="gthoic 15 bold",bd = 0,fg = "gray99",bg = "black"
                ,activebackground= "black",command=Down)                                    # defining the button for moving down selected item
                self.down_button.place(x=self.screen_width//2-120,y=self.screen_height//2-120)  # placing the same
                self.win.bind("<Delete>",lambda e :self.remove("e"))                          # Binding delete button with Delete button (keyboard) event
                self.placed = 1                                                             # changing value to 1 means placed
            else :                                                                          # if already placed
                None                                                                        # do nothing
        except:                                                                             # if any error encountered
            None                                                                            # Do Nothing
##########################################################################################################################################################
# Defining Function for Removing the selection highlight  from treeview
##########################################################################################################################################################
    def remove_sel(self,event):             # remove_sel Will Remove the selection Highlight from treeveiew,called when event(Escape) occured
        try:                                # using try and except approach
            self.placed = 0                 # means Buttons are not placed(moveup,down and delete)
            self.style.map("Treeview",background=[('selected',"gray99")]) # changing the selected row color (to last one)
            self.tree_view.selection_remove(self.tree_view.focus())       # removing the focus or selection highlight
            self.win.unbind("<Escape>")     # unbinding the event (Escape) because now nothing is selected in treeview
            self.win.unbind("<Delete>")     # unbinding event (Delete) Because nothing is selected
            self.add_label.pack_forget(),self.up_button.place_forget(),self.down_button.place_forget(),self.delete.place_forget()# unplacing required wid
        except:                             # if any error occured
            None                            # do nothing
##########################################################################################################################################################
# Defining Function for Reomving selected item from treeview or delete all the rows from treeview
##########################################################################################################################################################
    def remove(self,event):    # remove will get called when event(Delete) occurs(will remove selected item),when all is event parameter(will remove all)
        try:                   # using try and except approach
            if event == "all": # if the user want to delete all the rows from treeview at once
                self.placed = 0                                             # means Buttons are not placed(moveup,down and delete)
                self.tree_view.delete(*self.tree_view.get_children())       # deleting all the children items at once( * for getting all items at once)
                self.count = 0 # changing the value for count to zero so that next inserted row no. start from 1
                self.inserted = 0 # so that labels and buttons could place
                self.convert.place_forget(),self.delete_all.place_forget(),self.size.place_forget()
                self.quality.place_forget(),self.options.place_forget(),self.q.place_forget(),self.combo.place_forget() # unplacing the required buttons
                self.remove_sel("event")                                    # removing the selection highlight from treeview(by calling remove_sel func)
            if event=="e":                                                  # if the user want to remove the selected item from treeview
                self.tree_view.delete(self.tree_view.focus())               # deleting the selected item from treeview
            self.add_label.pack_forget()                                    # unpacking add_label Label (will remove image from showing)
            self.add_label['image'] = self.add                              # changing the image to add for add_label
            self.add_label.pack(fill= BOTH,expand= True)                    # again packing the add_label to show add image
            self.add_label.bind("<ButtonRelease-1>",lambda e : self.Add_image(e)) # again Binding event for adding images
            self.add_label.unbind("Double-1")                               # Unibnding event for opening image for editing
            if [*self.tree_view.get_children()] == []:                      # means nothing is present in treeview
                self.count = 0                                              # resetting count to 0
                self.inserted = 0 # so that labels and buttons could place
                self.convert.place_forget(),self.delete_all.place_forget()  # unplacing required widgets
                if self.placed == 1:                                        # means Buttons are placed(moveup,down and delete)
                    self.up_button.place_forget(),self.down_button.place_forget(),self.delete.place_forget() # unplacing buttons
                self.combo.place_forget(),self.q.place_forget(),self.quality.place_forget()                  # same here
                self.size.place_forget(),self.options.place_forget(),self.progress_bar.place_forget()        # same here
        except:                                                             # if any error occured
            None                                                            # do nothing
##########################################################################################################################################################
# Defining Function for initiation of converting image to pdf process
##########################################################################################################################################################
    def convert_to_pdf(self):        # convert_to_pdf will get called when convert button is clicked
        self.directory = asksaveasfilename(title = "Choose Folder By - Himanshu Mahajan",initialfile = "Himanshu_Mahajan.pdf"
            ,filetype = [("PDF Document",'*.pdf')],defaultextension=[("PDF Document",'*.pdf')]) # asking to save pdf at desired location
        if self.directory=="":       # if user canceled the process or not choosed any location
            None                     # do nothing
        else:                        # if path for saving is available
            q = int(self.q.get())    # getting the selected item from combobox
            self.start(q)            # calling start with desired quality
##########################################################################################################################################################
# Defining Function For Starting Conversion process
##########################################################################################################################################################
    def start(self,q=100):           # start will called by convert_to_pdf Function(it will start with default quality(100),if not provided)
        self.z = str(self.combo.get())# getting the selected item from combobox
        if self.z == 'A3':           # if selected item is A3
            width,height = 3508,4960  # size for same in pixels
        if self.z == 'A4':           # if selected item is A4
            width,height = 2480,3508  # size for same in pixels
        if self.z == 'A5':           # if selected item is A5
            width,height = 1750,2480  # size for same in pixels
        if self.z == 'Letter':       # if selectd item is Letter
            width,height = 2550,3300  # size for same in pixels
        if self.z == 'Legal':        # if selected item is Legal
            width,height = 2550,4200  # size for same in pixels
        self.resize = (width,height)  # now variable storing size for resizing page of pdf in pixels(at 300 dpi)
        self.w,self.h = float(width)*0.084667,float(height)*0.084667 # same size in mm for resizing image(at 300 dpi)
        if self.placed == 1:         # means Buttons are  placed(moveup,down and delete)
            self.up_button.place_forget(),self.down_button.place_forget(),self.delete.place_forget() # unplacing buttons
        self.remove_sel("event")     # calling remove_sel for removing higlight
        self.convert.place_forget(),self.combo.place_forget(),self.q.place_forget(),self.quality.place_forget(),self.options.place_forget()# unplacing
        self.size.place_forget(),self.delete_all.place_forget(),self.add_folder.place_forget(),self.add_image.place_forget()              # same
        self.win.update()                # updating window
        items = [*self.tree_view.get_children()] # getting all items from treeview and making its list
        self.progress=0                  # initial progress variable
        self.progress_bar = ttk.Progressbar(self.win,orient = "horizontal",mode= "determinate",value=self.progress
            ,length =self.screen_width//2)    # defining progressbar for showing progerss of conversion
        self.progress_bar.place(x=0,y = self.screen_height//2+90) # placing the same
        self.length = len(items)         # length of the items in list of items in treeview
        l= []                            # empty list for appending copied images
        for i in items:                  # iterating for each item in list of items in treeview
            path = self.tree_view.item(i)['values'][1] # getting path of item
            try:                         # using try and except approach
                img = Image.open(path)   # opening the file(image,if not image then it will get deleted in except block)
                if img.mode == "RGBA":   # if image is transparent or have alpha mode in it
                    img = img.convert("RGB")  # changing image mode from RGBA to RGB(this will remove alpha transparancy from image)
                resized = img.resize(self.resize) # resizing the image to page size
                new = path.split(".")    # splitting the path of image from '.'(this will separate image extension)
                new.pop(-1)              # popping last item (this will remove exension of image from list )
                n= ""                    # an empty string (for renaming image with extension)
                for x in new:            # iterating for new name of image (will get words separated at ".")
```

```
                n+=x                        # creating a new string from separated list(if more than one "." is present in name of image)
            c=n+"copy.jpg"                  # storing the full name(with path) with changed extension of image
            resized.save(c,dpi=(300,300),optimise = True,quality=q) # saving copy of that image with 300 dpi , quality depending on compression
            l.append(c)                     # appending image path to list for converting to pdf and deleting also(coming down)
        except:                             # if the file is not a image or not openable
            self.tree_view.delete(i)        # deleting that item from treeview
            continue                        # skipping this and go for next iteration
        self.progress+=100/self.length      # changing progressbar progress value
        self.progress_bar['value']=self.progress # change in prograssbar value
        self.progress_bar.update_idletasks()     # updating progressbar
        self.win.update()                        # updating window
    try:                                    # using try and except approach
        if [*self.tree_view.get_children()] == []:# if all the images in treeview are not supported
            messagebox.showwarning("Warning ! By - Himanshu Mahajan"," Deleted Files Are Not Supported \n Result : All Files Are Corrupt")# warning
            self.remove("None")                 # calling remove only to show add_label (here)
            self.progress_bar.place_forget()    # unplacing progressbar
            self.add_image.place(x=0,y=self.screen_height//2-150),self.add_folder.place(x=100,y=self.screen_height//2-150) # placing buttons
        else:                                   # if tree view is not empty
            pdf = FPDF(orientation = "P",unit = "mm",format = str(self.z)) # creating object of class FPDF with desired format
            pdf.set_auto_page_break(0)          # setting auto pagebreak to remove blank white pages after each image
            self.progress = 0                   # changing progress value again to 0
            self.progress_bar['value']=0        # change in progressbar value
            for image in l:                     # now iterating over list with copied images(done above)
                pdf.add_page()                  # adding page to pdf
                pdf.image(image,0,0,w=self.w,h=self.h) # printing image to page of pdf with size same as of page and coordinates as 0
                self.progress+=100/len(l)-0.2                   # changing progress value
                self.progress_bar['value']=self.progress        # chamge in progressbar value
                self.progress_bar.update_idletasks()            # updating progressbar
                self.win.update()                               # updating window
            messagebox.showinfo("Don't Panic By - Himanshu_Mahajan","Don't Panic If Window Gets Unresponsive") # show info(as here no value for prog)
            pdf.output(self.directory,"F")                      # this step will make pdf and its time and value depends upon no. of images and qual
            self.progress=100                                   # as pdf is made changing value for progress to 100 (means done)
            self.progress_bar['value']=self.progress            # change value in progressbar
            self.progress_bar.update_idletasks()                # updating progressbar
            self.win.update()                                   # updating window
            for copy in l:                                      # iterating over copied images(path) list
                os.remove(copy)                                 # removing copy of image made by program
            self.progress_bar.place_forget()                    # unplacing progressbar
            messagebox.showinfo("Successfull By - Himanshu Mahajan"," Successfully Done !") # showing successfully done
            self.delete_all.place(x=190,y=self.screen_height//2-150),self.add_folder.place(x=100,y=self.screen_height//2-150)  # placing buttons
            self.add_image.place(x=0,y=self.screen_height//2-150),self.convert.place(x=self.screen_width//4-100,y=self.screen_height//2+80) # same
            self.combo.place(x = 10,y= self.screen_height//2+20),self.quality.place(x=self.screen_width//4+75,y=self.screen_height//2-10)# same
            self.q.place(x=self.screen_width//4+8,y = self.screen_height//2+20),self.size.place(x=90,y=self.screen_height//2-10)         # same
            self.options.place(x=self.screen_width//4-50,y=self.screen_height//2-50) # label
            self.win.update()                                   # updating window
            os.startfile(self.directory)                        # sarting the pdf made
    except:                                                     # if any unklnown error occured
        None                                                    # Do Nothing
#############################################################################################################################################
# Defining Function for Showing instructions in scrolled text
#############################################################################################################################################
    def how_to_use(self):                                   # how_to_use will get called whenever program is started,will display instructions
        self.scrolled_text = scrolledtext.ScrolledText(self.win,width = self.screen_width//17,height = self.screen_height//54
            ,bg="black",fg= "gray99",wrap =WORD)             # defining scrolled text
        self.scrolled_text.place(x= 10,y = self.screen_height//2+130)  # placing the same
        with open(self.get_path("how_to_use.txt"),"r",encoding="utf-8",errors = "ignore") as f: # opening file where instructions are written
            for lines in f.readlines():                     # iterating over lines in file
                self.scrolled_text.insert("end",lines)      # inserting lines to scrolled text
        self.scrolled_text['state'] = DISABLED              # making state of scrolled text to disabled
#############################################################################################################################################
# MAIN STARTS HERE
#############################################################################################################################################
if __name__ == "__main__":   # in python __name__ is always equal to __main__ if no error occured before this statement
    try:                     # try and except approach
        start = IMG_TO_PDF() # instance of class IMG_TO_PDF
    except:                  # if any error occured
        messagebox.showerror("IMG_TO_PDF By - Himanshu Mahajan","AN ERROR HAS OCCURED !")  # showing error
#############################################################################################################################################
##-----------------------------------------------------------## FINISHED ##-----------------------------------------------------------##
#############################################################################################################################################
```