

**NCR CAMPUS, MODINAGAR**

## Delhi-Meerut Road, Sikri Kalan, Modinagar – 201204, GHAZIABAD (U.P.)

**Python Programming (15IT322E)**

**Mini Project**

**(YOUTUBE DOWNLOADER**

**DESKTOP APPLICATION)**

**Name of Student : Abhya Sood**

**Reg. No : RA1711003030155**

**Degree/ Branch : B-Tech / CSE**

**Year/ Section : 3rd Year/ Sec C**

**Course Code : 15IT322E**

**Course Title : Python Programming**

**Faculty : Mr. Manikandan Rajagopal**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Chapter Name** | **Page- No.** | **Sign** |
| 1 | **Objective of the project** |  |  |
| 2 | **Introduction** |  |  |
| 3 | **Software Requirements** |  |  |
| 4 | **Methodology and modules used** |  |  |
| 5 | **Source code** |  |  |
| 6 | **Output and Screenshots** |  |  |
| 7 | **Conclusion** |  |  |

**Objective of the project**

The main objective of this project is to provide our user with the best downloading speed as well as in a good quality. This project might be very helpful for people who do not have access to high speed internet. The projects solve the problem of buffering in YouTube as rather than waiting for the video to load the user can simply download it in high quality within no time. The projects solve the problem of buffering in YouTube as rather than waiting for the video to load the user can simply download it in high quality within no time.

There are several places where people do not have any access to high speed internet. YouTube has now become a great source for learning for many individuals. From high level programming to basic cooking recipes YouTube has something for everyone but this learning can be interrupted because of low internet speed.

This situation can be tackled using YouTube Downloader Project which will not only allow you to download high quality videos from the site but also download them in blink of an eye. The download speed that this project provides is remarkable and also using it is very simple. The videos that are downloaded using YouTube downloader can be shared with anyone using offline media share and would be accessible to any device.

The Tkinter class provides graphics to the desktop application which makes the use even easier and user friendly. The pytube3 module does all the magic! It provides various streams to download the videos.

All the user has to do is to provide the URL of the video he wishes to download and the path where he wants to save the video in his computer.

**Introduction**

This project is made to download high quality video using their Uniform Resource Locator simply known as URL.

The idea behind the project was to download the videos from YouTube much faster than any other web site or any application. The final product would download any video from YouTube without any fuss and can be saved anywhere in the computer according to the user’s wish.

The main objective of this project is to provide our user with the best downloading speed as well as in a good quality. This project might be very helpful for people who do not have access to high speed internet. The projects solve the problem of buffering in YouTube as rather than waiting for the video to load the user can simply download it in high quality within no time. The Tkinter class provides graphics to the desktop application which makes the use even easier and user friendly. The pytube3 module does all the magic! It provides various streams to download the videos.

The videos that are downloaded using YouTube downloader can be shared with anyone using offline media share and would be accessible to any device.

The program simply checks all the streams available for download at that point of time and simply chooses the stream with highest quality and also which would not take much time to download. This would simply return the best stream available at that point of time.

Python uses various modules and classes like Tkinter and Pytube3 to download a high-quality video. The whole point of this project is to make streaming easier. This project will help the user to watch their favourite videos anytime and anywhere even if they do not have access to the internet.

**Software Requirements**

Any text editor which can be used to write programs in python can be used it to make this project but the only thing that we have to keep in mind is that our computer should have all the latest version of Pthon3.

Various IDEs that can be used are as follows:

* PyCharm
* Spyder
* Pydev
* Idle
* Wing

Various text editors are as follows:

* Sublime Text
* Atom
* Visual Studio Code

The one used in this project is **Visual Studio Code**

Visual Studio Code is an open source code editor which was developed mainly for the development and debugging of latest web and cloud projects.

It is capable of combining both editor and good development features very smoothly. It is one of the major choices for python developers.

**Best Features:**

1. It supports syntax highlighting and auto code complete feature with IntelliSense which completes syntax based on variable types, function definition etc.
2. It has a powerful debugger and the user can debug from the editor itself.
3. It has strong integration with GIT so that a user can perform GIT operations like push, commit straight from the editor itself.
4. Visual studio is highly extensible and customizable through which we can add languages, debuggers, themes etc.

**Pros:**

1. It provides multi-language support and many other functionalities which the other languages don’t possess.
2. It has a good layout and smart interface.
3. It allows the use of many plugins which a developer can get from the VS code marketplace for its customization.
4. It supports the use of vertical orientation and multi-split window feature.

**Cons:**

1. Searching with visual studio code is very slow.
2. Initially, it takes an ample amount of time to launch.

**Methodology and modules used**

Various modules that have been used in making this project are:

1. pytube3

2. Tkinter

3 tkinter.filedialog

4. tkinter.messagebox

5. threading

**1.pytube :**

pytube is a very serious, lightweight, dependency-free Python library (and command-line utility) for downloading YouTube Videos.

YouTube is the most popular video-sharing platform in the world and as a hacker you may encounter a situation where you want to script something to download videos. For this I present to you pytube.

pytube is a lightweight library written in Python. It has no third party dependencies and aims to be highly reliable.

pytube also makes pipelining easy, allowing you to specify callback functions for different download events, such as on progress and on complete

**Features**

* Support for Both Progressive & DASH Streams
* Support for downloading complete playlist
* Command-line Interfaced Included
* Caption Track Support
* Outputs Caption Tracks to .srt format (SubRip Subtitle)
* Ability to Capture Thumbnail URL.
* Extensively Documented Source Code
* No Third-Party Dependencies

**Installation**

**pip install pytube**

**2. tkinter:**

The [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.)

Running python -m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that [tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter) is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

**3. Threading:**

This module constructs higher-level threading interfaces on top of the lower level [\_thread](https://docs.python.org/3/library/_thread.html#module-_thread) module. See also the [queue](https://docs.python.org/3/library/queue.html#module-queue) module.

Python threading allows you to have different parts of your program run concurrently and can simplify your design. If you’ve got some experience in Python and want to speed up your program using threads, then this tutorial is for you!

The [Thread](https://docs.python.org/3/library/threading.html#threading.Thread) class represents an activity that is run in a separate thread of control. There are two ways to specify the activity: by passing a callable object to the constructor, or by overriding the [run()](https://docs.python.org/3/library/threading.html#threading.Thread.run) method in a subclass. No other methods (except for the constructor) should be overridden in a subclass. In other words, only override the \_\_init\_\_() and [run()](https://docs.python.org/3/library/threading.html#threading.Thread.run) methods of this class. Once a thread object is created, its activity must be started by calling the thread’s [start()](https://docs.python.org/3/library/threading.html#threading.Thread.start) method. This invokes the [run()](https://docs.python.org/3/library/threading.html#threading.Thread.run) method in a separate thread of control. Once the thread’s activity is started, the thread is considered ‘alive’. It stops being alive when its [run()](https://docs.python.org/3/library/threading.html#threading.Thread.run) method terminates – either normally, or by raising an unhandled exception. The [is\_alive()](https://docs.python.org/3/library/threading.html#threading.Thread.is_alive) method tests whether the thread is alive. Other threads can call a thread’s [join()](https://docs.python.org/3/library/threading.html#threading.Thread.join) method. This blocks the calling thread until the thread whose [join()](https://docs.python.org/3/library/threading.html#threading.Thread.join) method is called is terminated. A thread has a name. The name can be passed to the constructor, and read or changed through the [name](https://docs.python.org/3/library/threading.html#threading.Thread.name) attribute.

**Source code:**

**Working with pytube**

from pytube import \*

url='https://www.youtube.com/watch?v=QiBeywmJoRY&list=RDQiBeywmJoRY&start\_radio=1'

path\_to\_save\_video='C:\\Users\\Abhya\\Desktop'

#creating youtube object with URL

ob=YouTube(url)

strms=ob.streams.all()

for s in strms:

    print(s)

strm=ob.streams.first()

print(strm)

print(strm.filesize)

print(strm.title)

print(ob.description)

#now downloading video

strm.download(path\_to\_save\_video)

print('done')

**Main code:**

from pytube import \*

from tkinter import \*

from tkinter.filedialog import \*

from tkinter.messagebox import \*

from threading import \*

#total size container

file\_size=0

#this fuction is used for updating percentage...

def progress(stream=None,file\_handle=None,remaining=None):

   #gets the percentage of the file that has been downloaded...

   file\_downloaded=(file\_size-remaining)

   per=(file\_downloaded/file\_size)\*100

   dBtn.config(text="{:00.0f} % downloaded".format(per))

def startDownload():

    global file\_size

    try:

        url=urlField.get()

        #changing button text

        dBtn.config(text='Please Wait...')

        dBtn.config(state=DISABLED)

        path\_to\_save\_video=askdirectory()

        if path\_to\_save\_video is None:

            return

#creating youtube object with url

        ob=YouTube(url,on\_progress\_callback=progress)

        #strms=ob.streams.all()

        #for s in strms:

        #    print(s)'''

        strm=ob.streams.first()

        file\_size=strm.filesize

        vTitle.config(text=strm.title)

        vTitle.pack(side=TOP)

        strm.download(path\_to\_save\_video)

        print('done')

        dBtn.config(text='Start Download')

        dBtn.config(state=NORMAL)

        showinfo('Download Finished!','Downloaded Successfully')

        urlField.delete(0,END)

        vTitle.pack\_forget()

    except Exception as e:

        print(e)

        print("error!")

def startDownloadThread():

    #create thread...

    thread=Thread(target=startDownload)

    thread.start()

#starting gui building

main=Tk()

#setting the title

main.title('My Youtube Downloader')

#set the icon

main.iconbitmap('youtube\_icon.ico')

main.geometry("500x600")

#heading icon

file=PhotoImage(file='youtube.png')

headingIcon=Label(main,image=file)

headingIcon.pack(side=TOP,pady=15)

#url text field

urlField=Entry(main,font=("ariel",20),justify=CENTER)

urlField.pack(side=TOP,fill=X,padx=20,pady=10)

#download button

dBtn=Button(main,text='Start Download',font=("ariel",20),justify=CENTER,relief='raised',command=startDownloadThread)

dBtn.pack(side=TOP,pady=15)

#video title

vTitle=Label(main, text='Video Title')

#vTitle.pack(side=TOP)

        #print(strm)

        #print(strm.filesize)

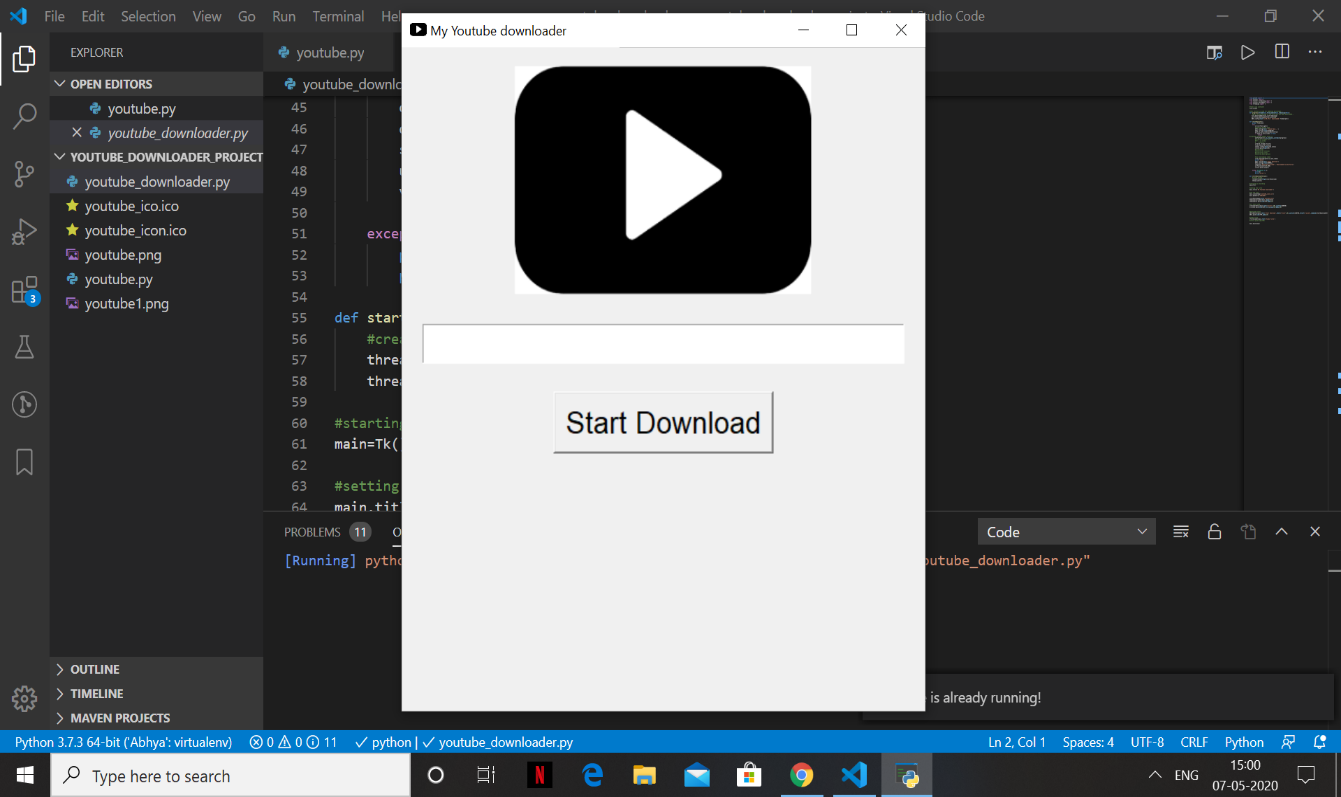
        #print(strm.title)

        #print(ob.description)

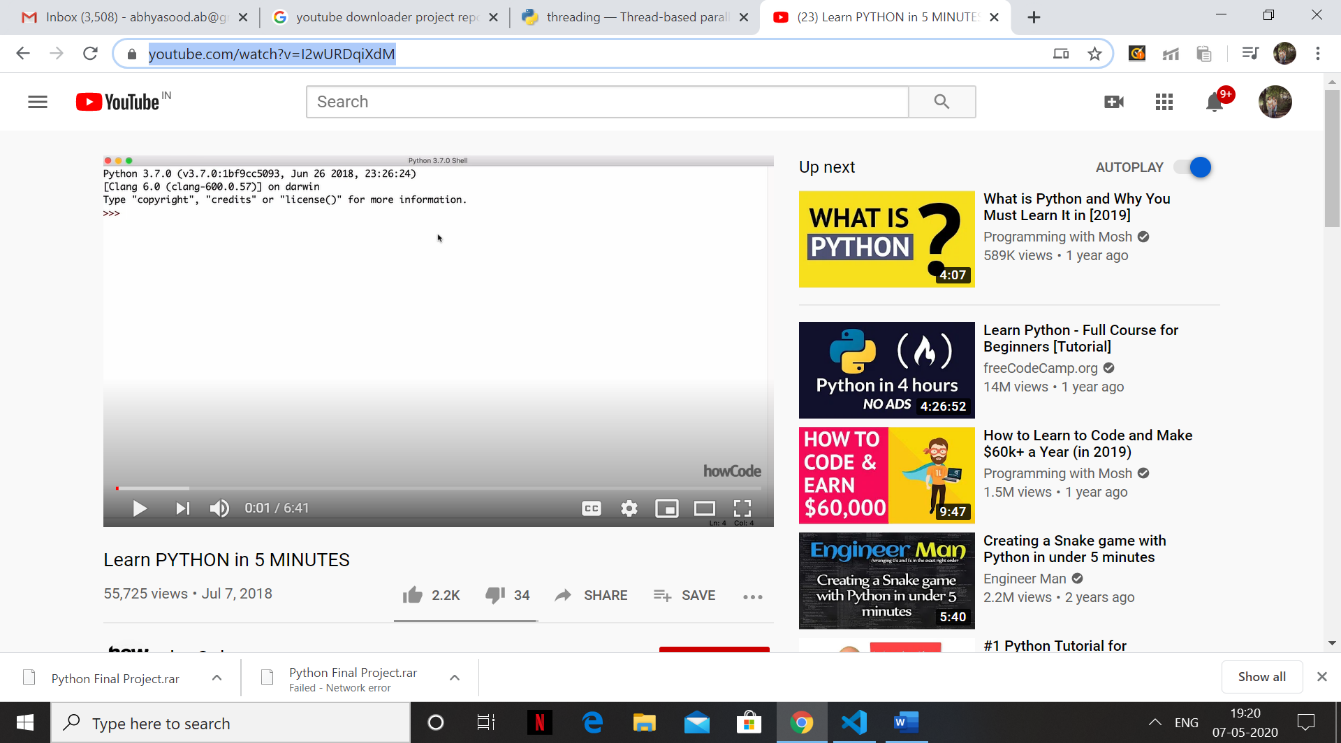
        #now downloading video

main.mainloop()

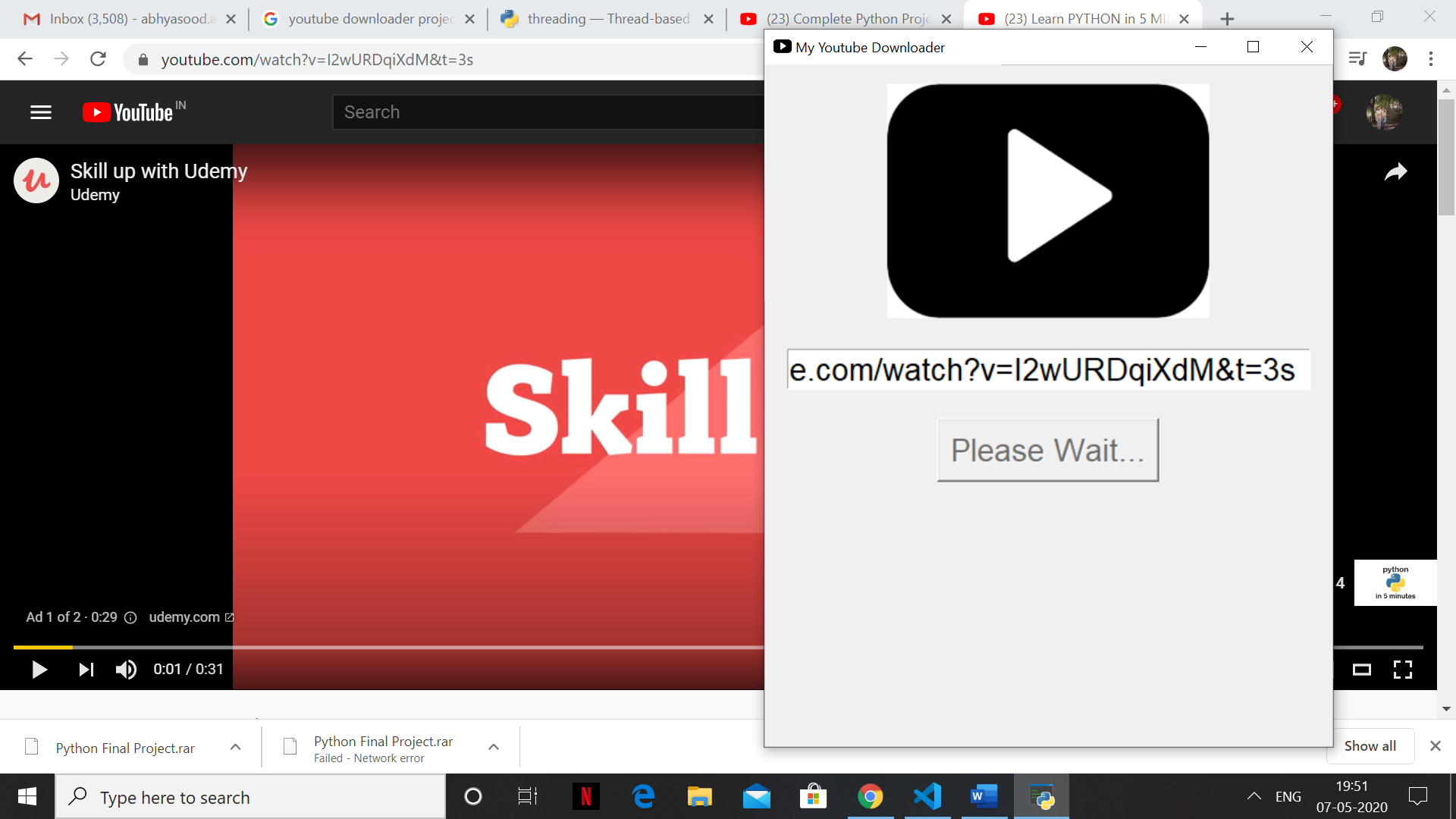
**Outputs and screenshots**

****

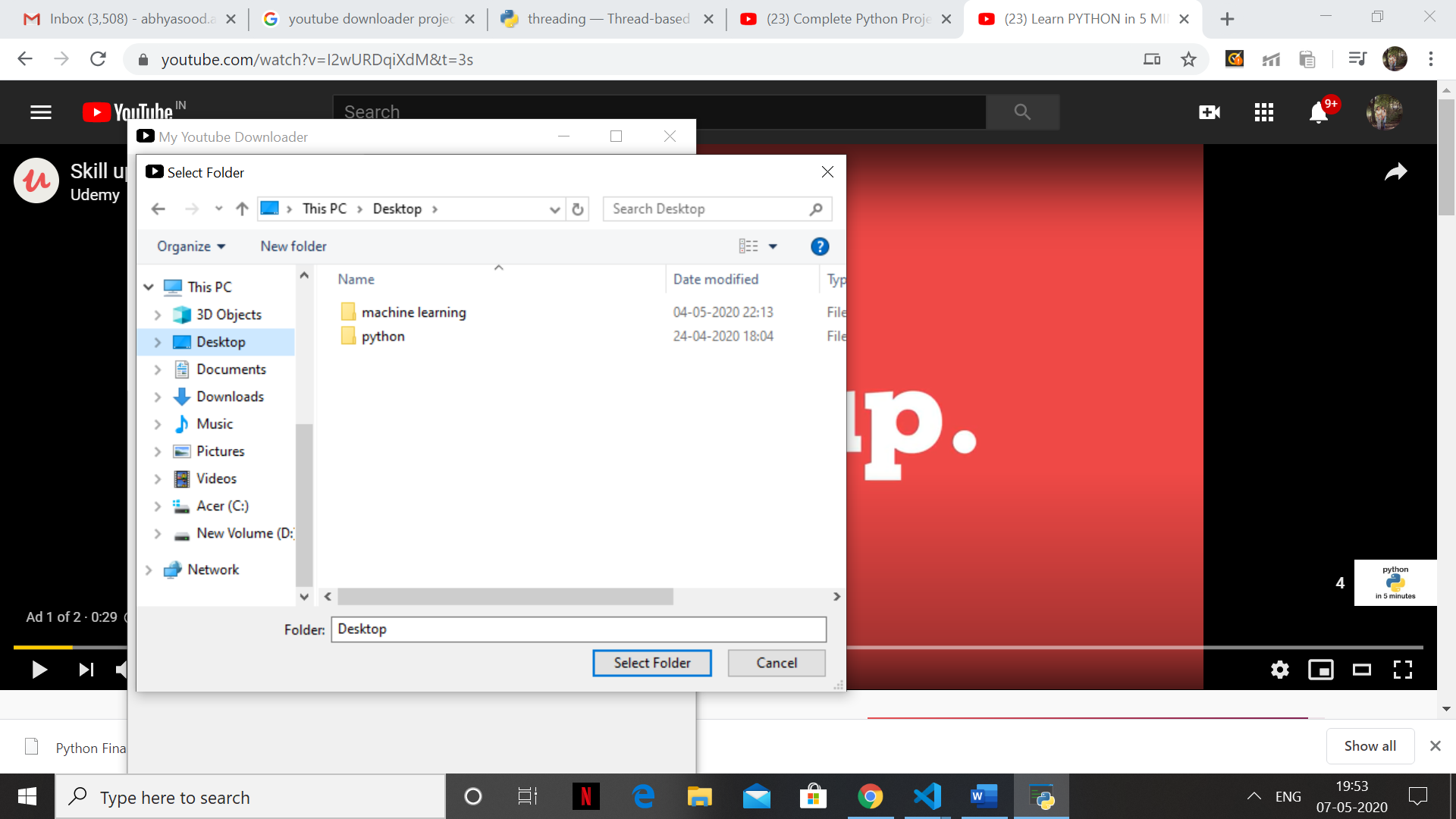
**The main window of out YouTube downloader application**



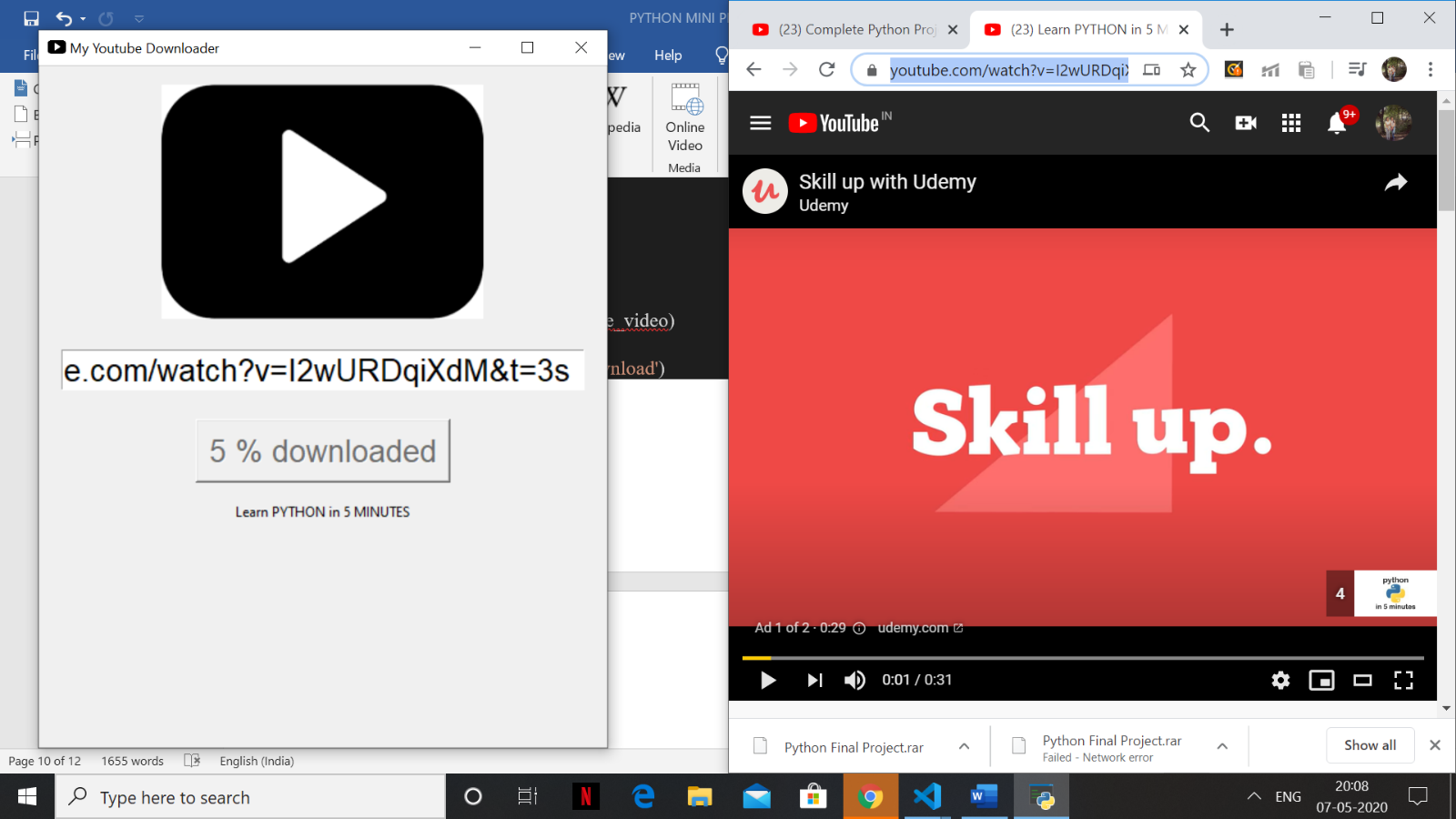
**Copying the desired URL from youtube**



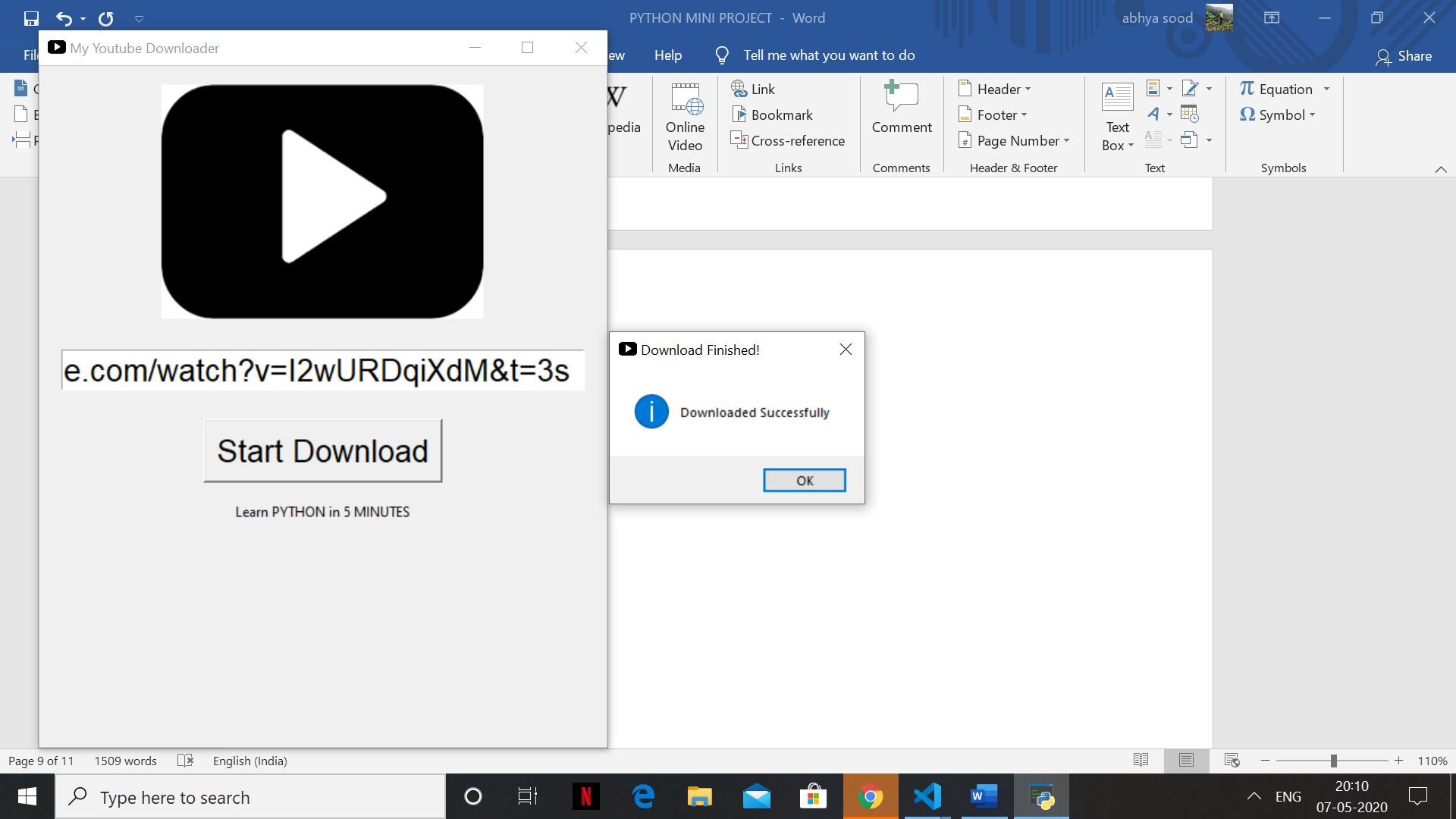
**Link is processed by the application**

****

**Directory asks for the path to store the file**

****

**Video title and the percentage of download is visible while the video is being downloaded**

****

**Finally download successful message is displayed**

**Conclusion**

YouTube downloader desktop application has been implemented in Python Programming Language. The interface is user friendly and all user has to do is to provide the link or the UR of the video and the path where the downloaded video would be stored and the rest is done by the program itself. This application is something that can be used by anyone regardless of their age or technical understanding.

**Features:**

* Simple and easy to use
* User friendly
* Less storage is required
* High speed download
* User friendly GUI

**References:**

* <https://docs.python.org/3/library/tkinter.html>
* <https://pypi.org/project/pytube/>
* <https://docs.python.org/3/tutorial/errors.html>