import urllib.request  
import urllib.error  
  
import re  
import sys  
import time  
import os  
  
from pytube import YouTube  
  
class progressBar:  
 def \_\_init\_\_(self, barlength=25):  
 self.barlength = barlength  
 self.position = 0  
 self.longest = 0

print\_progress(self, cur, total, start):  
 currentper = cur / total  
 elapsed = int(time.clock() - start) + 1  
 curbar = int(currentper \* self.barlength)  
 bar = '\r[' + '='.join(['' for \_ in range(curbar)]) # Draws Progress  
 bar += '>'  
 bar += ' '.join(['' for \_ in range(int(self.barlength - curbar))]) + '] ' # Pads remaining space  
 bar += bytestostr(cur / elapsed) + '/s ' # Calculates Rate  
 bar += getHumanTime((total - cur) \* (elapsed / cur)) + ' left' # Calculates Remaining time  
 if len(bar) > self.longest: # Keeps track of space to over write  
 self.longest = len(bar)  
 bar += ' '.join(['' for \_ in range(self.longest - len(bar))])  
 sys.stdout.write(bar)  
  
 def print\_end(self, \*args): # Clears Progress Bar  
 sys.stdout.write('\r{0}\r'.format((' ' for \_ in range(self.longest))))  
  
def getHumanTime(sec):  
 if sec >= 3600: # Converts to Hours  
 return '{0:d} hour(s)'.format(int(sec / 3600))  
 elif sec >= 60: # Converts to Minutes

return '{0:d} minute(s)'.format(int(sec / 60))  
 else: # No Conversion  
 return '{0:d} second(s)'.format(int(sec))  
  
def bytestostr(bts):  
 bts = float(bts)  
 if bts >= 1024 \*\* 4: # Converts to Terabytes  
 terabytes = bts / 1024 \*\* 4  
 size = '%.2fTb' % terabytes  
 elif bts >= 1024 \*\* 3: # Converts to Gigabytes  
 gigabytes = bts / 1024 \*\* 3  
 size = '%.2fGb' % gigabytes  
 elif bts >= 1024 \*\* 2: # Converts to Megabytes  
 megabytes = bts / 1024 \*\* 2  
 size = '%.2fMb' % megabytes

elif bts >= 1024: # Converts to Kilobytes  
 kilobytes = bts / 1024  
 size = '%.2fKb' % kilobytes  
 else: # No Conversion  
 size = '%.2fb' % bts  
 return size  
  
def getPageHtml(url):  
 try:  
 yTUBE = urllib.request.urlopen(url).read()  
 return str(yTUBE)  
 except urllib.error.URLError as e:  
 print(e.reason)  
 exit(1)  
  
def getPlaylistUrlID(url):  
 if 'list=' in url:  
 eq\_idx = url.index('=') + 1  
 pl\_id = url[eq\_idx:]  
 if '&' in url:  
 amp = url.index('&')  
 pl\_id = url[eq\_idx:amp]  
 return pl\_id   
 else:  
 print(url, "is not a youtube playlist.")  
 exit(1)  
  
def getFinalVideoUrl(vid\_urls):

elif bts >= 1024: # Converts to Kilobytes  
 kilobytes = bts / 1024  
 size = '%.2fKb' % kilobytes  
 else: # No Conversion  
 size = '%.2fb' % bts  
 return size  
  
def getPageHtml(url):  
 try:  
 yTUBE = urllib.request.urlopen(url).read()  
 return str(yTUBE)  
 except urllib.error.URLError as e:  
 print(e.reason)  
 exit(1)  
  
def getPlaylistUrlID(url):  
 if 'list=' in url:  
 eq\_idx = url.index('=') + 1  
 pl\_id = url[eq\_idx:]  
 if '&' in url:  
 amp = url.index('&')  
 pl\_id = url[eq\_idx:amp]  
 return pl\_id   
 else:  
 print(url, "is not a youtube playlist.")  
 exit(1)  
  
def getFinalVideoUrl(vid\_urls):

def download\_Video\_Audio(path, vid\_url, file\_no):  
 try:  
 yt = YouTube(vid\_url)  
 except Exception as e:  
 print("Error:", str(e), "- Skipping Video with url '"+vid\_url+"'.")  
 return  
  
 try: # Tries to find the video in 720p  
 video = yt.get('mp4', '720p')  
 except Exception: # Sorts videos by resolution and picks the highest quality video if a 720p video doesn't exist  
 video = sorted(yt.filter("mp4"), key=lambda video: int(video.resolution[:-1]), reverse=True)[0]  
  
 print("downloading", yt.filename+" Video and Audio...")  
 try:  
 bar = progressBar()  
 video.download(path, on\_progress=bar.print\_progress, on\_finish=bar.print\_end)  
 print("successfully downloaded", yt.filename, "!")  
 except OSError:  
 print(yt.filename, "already exists in this directory! Skipping video...")  
  
 try:  
 os.rename(yt.filename+'.mp4',str(file\_no)+'.mp4')  
 aud= 'ffmpeg -i '+str(file\_no)+'.mp4'+' '+str(file\_no)+'.wav'  
 final\_audio='lame '+str(file\_no)+'.wav'+' '+str(file\_no)+'.mp3'  
 os.system(aud)  
 os.system(final\_audio)  
 os.remove(str(file\_no)+'.wav')  
 print("sucessfully converted",yt.filename, "into audio!")  
 except OSError:  
 print(yt.filename, "There is some problem with the file names...")  
   
  
def printUrls(vid\_urls):  
 for url in vid\_urls:  
 print(url)  
 time.sleep(0.04)  
   
if \_\_name\_\_ == '\_\_main\_\_':  
 if len(sys.argv) < 2 or len(sys.argv) > 3:  
 print('USAGE: python ytPlaylistDL.py playlistURL OR python ytPlaylistDL.py playlistURL destPath')  
 exit(1)  
 else:  
 url = sys.argv[1]  
 directory = os.getcwd() if len(sys.argv) != 3 else sys.argv[2]  
   
 # make directory if dir specified doesn't exist  
 try:  
 os.makedirs(directory, exist\_ok=True)  
 except OSError as e:  
 print(e.reason)  
 exit(1)  
  
 if not url.startswith("http"):

url = 'https://' + url  
  
 playlist\_page\_content = getPageHtml(url)  
 vid\_urls\_in\_playlist = getPlaylistVideoUrls(playlist\_page\_content, url)  
  
 # downloads videos and audios  
 for i,vid\_url in enumerate(vid\_urls\_in\_playlist):  
 download\_Video\_Audio(directory, vid\_url, i)  
 time.sleep(1)