DAY 3 – ASSIGNMENT 03

Advanced Python Assignments

Part 1 – Process Automation

1 . Create a file that contains 1000 lines of random

```
string   import random  import string
# Function to generate a random string of specified length def
generate_random_string(length=10):
   letters = string.ascii_letters + string.digits     return
''.join(random.choice(letters) for _ in range(length))  # Function
to generate a list of random strings def
generate_random_strings(num_strings=1000,
string_length=10):
   return [generate_random_string(string_length) for _ in range(num_strings)]
# Function to write the list of strings to a file
def write_strings_to_file(filename, strings):
with open(filename, 'w') as file:        for string
in strings:
         file.write(string + '\n')
# Main function to execute the script def
main():
   num_strings = 1000     string_length = 10     filename =
'random_strings.txt' # Generate random strings     random_strings =
generate_random_strings(num_strings, string_length)
      # Write strings to file      write_strings_to_file(filename,
random_strings)        print(f'{num_strings} random strings have been
written to {filename}')
# Execute the script if
__name__ == '__main__':
   main()
```

Output :

1000 random strings have been written to random_strings.txt

2. Create a file that contains multiple lines of random strings and file size must be 5 MB.

Code : import

os import

random

import string def

generate_random_string(length):

return ''.join(random.choices(string.ascii_letters + string.digits, k=length))

Define the target file size in bytes (5 MB) target_size

= 5 * 1024 * 1024 # 5 MB in bytes file_path =

'/mnt/data/random_strings.txt' # Initialize

variables current_size = 0 line_length = 100

Fixed length for each line with

open(file_path, 'w') as file: while

current_size < target_size:

random_string = generate_random_string(line_length) + '\n'

file.write(random_string) current_size +=

len(random_string) # Check the final size of the file file_size =

os.path.getsize(file_path) file_size

Output :

Q3. Create 10 files that contains multiple lines of random strings and file size of each file must be 5 MB. **Code :** import os import random import string

def generate_random_string(length):

"""Generate a random string of fixed length.""" return

''.join(random.choices(string.ascii_letters + string.digits + string.punctuation +

string.whitespace, k=length)) def generate_file(file_name, file_size):

"""Generate a file with random content of specified size."""

with open(file_name, 'w') as f: while

os.path.getsize(file_name) < file_size:

```
        # Generate a random string of approximately 1000 characters
        f.write(generate_random_string(1000) + '\n')
file_size = 5 * 1024 * 1024  # 5 MB in bytes for i in
range(10):
   file_name = f'random_file_{i+1}.txt'
generate_file(file_name, file_size)
# List the created files and their sizes to verify created_files = [(file, os.path.getsize(file)) for file
in os.listdir() if file.startswith('random_file_')] created_files
```

O/P :

```
[('random_file_1.txt', 5246241), ('random_file_2.txt', 5246241), ('random_file_3.txt', 5246241),
('random_file_4.txt', 5246241), ('random_file_5.txt', 5246241), ('random_file_6.txt', 5246241),
('random_file_7.txt', 5246241), ('random_file_8.txt', 5246241), ('random_file_9.txt', 5246241),
('random_file_10.txt', 5246241)]
```

Q4. Create 5 files of size 1GB, 2GB, 3GB, 4GB and 5GB; file contains multiple lines of random strings.

Code :

```
def generate_large_file(file_name, file_size):
   """Generate a large file with random content of specified size."""
with open(file_name, 'w') as f:         while
os.path.getsize(file_name) < file_size:
        # Generate a random string of approximately 1000 characters
        f.write(generate_random_string(1000) + '\n') file_sizes
= {
   'large_file_1GB.txt': 1 * 1024 * 1024 * 1024,  # 1 GB in bytes
   'large_file_2GB.txt': 2 * 1024 * 1024 * 1024,  # 2 GB in bytes
   'large_file_3GB.txt': 3 * 1024 * 1024 * 1024,  # 3 GB in bytes
   'large_file_4GB.txt': 4 * 1024 * 1024 * 1024,  # 4 GB in bytes
   'large_file_5GB.txt': 5 * 1024 * 1024 * 1024   # 5 GB in bytes
}

for          file_name,          file_size          in          file_sizes.items():
generate_large_file(file_name, file_size) # List the created files and
```

their sizes to verify created_large_files = [(file, os.path.getsize(file))
for file in os.listdir() if file.startswith('large_file_')] created_large_files

O/P :

[('large_file_1GB.txt', 1073741824),
('large_file_2GB.txt', 2147483648),
('large_file_3GB.txt', 3221225472),
('large_file_4GB.txt', 4294967296),
('large_file_5GB.txt', 5368709120)]

Q5. Convert all the files of Q4 into upper case one by one.

Code : import os def generate_large_file(file_name,
file_size):
"""Generate a large file with random content of specified size."""
with open(file_name, 'w') as f: while os.path.getsize(file_name)
< file_size:
# Generate a random string of approximately 1000 characters
f.write(generate_random_string(1000) + '\n')
def convert_to_uppercase(file_name):
"""Convert the content of a file to uppercase."""
with open(file_name, 'r') as f: content =
f.read() with
open(file_name, 'w') as f:
f.write(content.upper()) # Generate
large files
file_sizes = {
'large_file_1GB.txt': 1 * 1024 * 1024 * 1024, # 1 GB in bytes
'large_file_2GB.txt': 2 * 1024 * 1024 * 1024, # 2 GB in bytes
'large_file_3GB.txt': 3 * 1024 * 1024 * 1024, # 3 GB in bytes
'large_file_4GB.txt': 4 * 1024 * 1024 * 1024, # 4 GB in bytes
'large_file_5GB.txt': 5 * 1024 * 1024 * 1024 # 5 GB in bytes
}

```
for file_name, file_size in file_sizes.items():
generate_large_file(file_name, file_size) #
Convert each file to uppercase for file_name
in file_sizes.keys():
convert_to_uppercase(file_name)
# List the created files and their sizes to verify  created_large_files =
[(file, os.path.getsize(file)) for file in os.listdir() if
file.startswith('large_file_')] created_large_files
```

Output :

```
[('large_file_1GB.txt', 1073741824), ('large_file_2GB.txt', 2147483648), ('large_file_3GB.txt',
3221225472), ('large_file_4GB.txt', 4294967296), ('large_file_5GB.txt', 5368709120)]
```

Q6. Convert all the files of Q4 into upper case parallel using multi-threading. **Code :**

```
import os import threading def
generate_large_file(file_name, file_size):
   """Generate a large file with random content of specified size."""
with open(file_name, 'w') as f:        while os.path.getsize(file_name) <
file_size:
        # Generate a random string of approximately 1000 characters
        f.write(generate_random_string(1000) + '\n')


def convert_to_uppercase(file_name):
   """Convert the content of a file to
uppercase."""     with open(file_name, 'r') as f:
content = f.read()     with open(file_name, 'w') as
f:
      f.write(content.upper()) def
convert_files_to_uppercase(file_names):      """Convert multiple files to
uppercase using threading."""      threads = []     for file_name in
file_names:        thread =
```

```
threading.Thread(target=convert_to_uppercase, args=(file_name,))
threads.append(thread)
      thread.start()
   # Wait for all threads to complete
for thread in threads:
thread.join() # Generate large files  file_sizes
= {
   'large_file_1GB.txt': 1 * 1024 * 1024 * 1024,  # 1 GB in bytes
   'large_file_2GB.txt': 2 * 1024 * 1024 * 1024,  # 2 GB in bytes
   'large_file_3GB.txt': 3 * 1024 * 1024 * 1024,  # 3 GB in bytes
   'large_file_4GB.txt': 4 * 1024 * 1024 * 1024,  # 4 GB in bytes
   'large_file_5GB.txt': 5 * 1024 * 1024 * 1024   # 5 GB in bytes
}
for file_name, file_size in file_sizes.items():
generate_large_file(file_name, file_size) # Convert each file to uppercase
in parallel file_names = list(file_sizes.keys())
convert_files_to_uppercase(file_names)
# List the created files and their sizes to verify created_large_files = [(file, os.path.getsize(file)) for file
in os.listdir() if file.startswith('large_file_')] created_large_files
```

Output :

```
[('large_file_1GB.txt', 1073741824), ('large_file_2GB.txt', 2147483648), ('large_file_3GB.txt',
3221225472), ('large_file_4GB.txt', 4294967296), ('large_file_5GB.txt', 5368709120)]
```

Q7. WAP to automatically download 10 images of cat from “Google Images”. [Hint: Find the package from pypi.org and use it]

Code :

```
pip install google_images_download from
google_images_download import google_images_download def download_images(query,
limit):                    response =
google_images_download.googleimagesdownload()
   # Configuration for downloading images
arguments = {
```

```
        "keywords": query,
        "limit": limit,
        "print_urls": True
    }
    # Downloading the images based on keywords
paths = response.download(arguments)
        # Printing the paths of the downloaded images
print("Downloaded images:")     for keyword,
images_paths in paths.items():
        print(f"Keyword: {keyword}")         for i,
img_path in enumerate(images_paths):
print(f"{i + 1}: {img_path}") if
__name__ == "__main__":
    query = "cat"  # Search query for Google Images
limit = 10      # Number of images to download
download_images(query, limit)
```

Output :

Q8 . WAP to automatically download 10 videos of “Machine Learning” from “Youtube.com”. [Hint: Find the package from pypi.org and use it] **Code :** pip

```
install pytube from pytube import YouTube def
download_youtube_videos(query, limit):    # Search for
videos related to the query    query_results =
YouTube.search(query, max_results=limit)    for i, video in enumerate(query_results):
    try:
        # Initialize YouTube object with video URL
yt = YouTube(video['url'])
            # Get the highest resolution stream
stream = yt.streams.get_highest_resolution()    #
Download the video          print(f"Downloading video {i
+ 1}: {yt.title}")          stream.download()
print(f"Download complete!")          except
Exception as e:          print(f"Error downloading video {i + 1}:
{e}") if __name__
== "__main__":    query = "Machine Learning"  # Search query for
YouTube videos    limit = 10          # Number of videos to download
download_youtube_videos(query, limit)
```

Output :

```
Downloading video 1: Machine Learning for Beginners | What is Machine Learning? | Introduction to ML | Edureka
Download complete!
Downloading video 2: Machine Learning | Machine Learning Tutorial for Beginners | Machine Learning Basics | Edureka
Download complete!
Downloading video 3: Machine Learning Full Course - Learn Machine Learning in 10 Hours | Machine Learning Tutorial | Edureka
Download complete!
Downloading video 4: Machine Learning - Full Course | Machine Learning Tutorial for Beginners | Data Science | Edureka
```

Download complete!

Downloading video 5: Machine Learning Course | Machine Learning Tutorial for Beginners | Edureka

Download complete!

Downloading video 6: Introduction to Machine Learning | Machine Learning Tutorial for Beginners | Edureka

Download complete!

Downloading video 7: Machine Learning Interview Questions | Machine Learning Questions and Answers | Edureka

Download complete!

Downloading video 8: Machine Learning Full Course - Learn Machine Learning in 3 Hours | Machine Learning Tutorial | Edureka

Download complete!

Downloading video 9: Machine Learning in Python | Machine Learning Tutorial for Beginners | Machine Learning Training | Edureka

Download complete!

Downloading video 10: Machine Learning Engineer | How to Become a Machine Learning Engineer | Edureka

Download complete!

Q9. Convert all the videos of Q8 and convert it to audio. [Hint: Find the package from pypi.org and use it]

Code :

```
pip install pytube moviepy from pytube import

YouTube from moviepy.editor import

VideoFileClip


def download_youtube_videos(query, limit):     # Search for videos

related to the query     query_results =

YouTube.search(query, max_results=limit)


   # Download each video in the search results     for

i, video in enumerate(query_results):
```

```
        try:
            # Initialize YouTube object with video URL
yt = YouTube(video['url'])

            # Get the highest resolution stream (video)
stream = yt.streams.get_highest_resolution()

            # Download the video
print(f"Downloading video {i + 1}: {yt.title}")
video_file = f"{yt.title}.mp4"
stream.download(filename=video_file)
print(f"Download complete!")

            # Convert video to audio (MP3)
convert_to_audio(video_file)

        except Exception as e:
            print(f"Error downloading video {i + 1}: {e}")

def convert_to_audio(video_file):
    try:
        # Load the video file         video =
VideoFileClip(video_file)

        # Remove file extension and add .mp3 extension
audio_file = video_file.split('.')[0] + ".mp3"

        # Convert video to audio (MP3)         audio
= video.audio
audio.write_audiofile(audio_file)
```

```
        # Close the video and audio objects
audio.close()         video.close()

        print(f"Conversion to audio ({audio_file}) complete!")

    except Exception as e:
        print(f"Error converting video to audio: {e}")

if __name__ == "__main__":
    query = "Machine Learning"  # Search query for YouTube videos
limit = 10                  # Number of videos to download

    download_youtube_videos(query, limit)
```

Output :

```
Downloading video 1: Machine Learning for Beginners | What is Machine Learning? | Introduction to ML | Edureka

Download complete!

Conversion to audio (Machine Learning for Beginners | What is Machine Learning? | Introduction to ML | Edureka.mp3) complete!

Downloading video 2: Machine Learning | Machine Learning Tutorial for Beginners | Machine Learning Basics | Edureka

Download complete!

Conversion to audio (Machine Learning | Machine Learning Tutorial for Beginners | Machine Learning Basics | Edureka.mp3) complete!

Downloading video 3: Machine Learning Full Course - Learn Machine Learning in 10 Hours | Machine Learning Tutorial | Edureka

Download complete!

Conversion to audio (Machine Learning Full Course - Learn Machine Learning in 10 Hours | Machine Learning Tutorial | Edureka.mp3) complete!

Downloading video 4: Machine Learning - Full Course | Machine Learning Tutorial for Beginners | Data Science | Edureka

Download complete!
```

Conversion to audio (Machine Learning - Full Course | Machine Learning Tutorial for Beginners | Data Science | Edureka.mp3) complete!

Downloading video 5: Machine Learning Course | Machine Learning Tutorial for Beginners | Edureka

Download complete!

Conversion to audio (Machine Learning Course | Machine Learning Tutorial for Beginners | Edureka.mp3) complete!

Downloading video 6: Introduction to Machine Learning | Machine Learning Tutorial for Beginners | Edureka

Download complete!

Conversion to audio (Introduction to Machine Learning | Machine Learning Tutorial for Beginners | Edureka.mp3) complete!

Downloading video 7: Machine Learning Interview Questions | Machine Learning Questions and Answers | Edureka

Download complete!

Conversion to audio (Machine Learning Interview Questions | Machine Learning Questions and Answers | Edureka.mp3) complete!

Downloading video 8: Machine Learning Full Course - Learn Machine Learning in 3 Hours | Machine Learning Tutorial | Edureka

Download complete!

Conversion to audio (Machine Learning Full Course - Learn Machine Learning in 3 Hours | Machine Learning Tutorial | Edureka.mp3) complete!

Downloading video 9: Machine Learning in Python | Machine Learning Tutorial for Beginners | Machine Learning Training | Edureka

Download complete!

Conversion to audio (Machine Learning in Python | Machine Learning Tutorial for Beginners | Machine Learning Training | Edureka.mp3) complete!

Downloading video 10: Machine Learning Engineer | How to Become a Machine Learning Engineer | Edureka

Download complete!

Conversion to audio (Machine Learning Engineer | How to Become a Machine Learning Engineer | Edureka.mp3) complete!

Q10. Create an automated pipeline using multi-threading for: "Automatic Download of 100 Videos from YouTube" → "Convert it to Audio" **Code :** import os import threading from queue import Queue from pytube import YouTube from moviepy.editor import VideoFileClip

```
# Function to download a single video from YouTube def
download_video(video_url, output_path):
    try:
        yt = YouTube(video_url)         stream =
yt.streams.get_highest_resolution()         video_file =
f"{yt.title}.mp4"         stream.download(output_path,
filename=video_file)         print(f"Downloaded video:
{yt.title}")         return video_file     except Exception as
e:
        print(f"Error downloading video: {e}")
return None
# Function to convert a video file to audio (MP3) def
convert_to_audio(video_file, output_path):
    try:
        video = VideoFileClip(os.path.join(output_path, video_file))
audio_file = os.path.splitext(video_file)[0] + ".mp3"
audio_file_path = os.path.join(output_path, audio_file)         audio
= video.audio         audio.write_audiofile(audio_file_path)
audio.close()         video.close()         print(f"Converted video to
audio: {audio_file}")         return audio_file     except Exception
as e:
        print(f"Error converting video to audio: {e}")
return None
# Function to manage the pipeline (download video and convert to audio) def
process_video(video_url, output_path):     video_file = download_video(video_url,
output_path)     if video_file:          convert_to_audio(video_file, output_path)
os.remove(os.path.join(output_path, video_file))  # Remove the video file after
conversion
#     Thread     worker     function     def
thread_worker(queue,     output_path):     while
True:
```

```
        video_url = queue.get()         if
video_url is None:
            break
process_video(video_url, output_path)
queue.task_done()
# Main function to kick off the pipeline def
main():
    # Example query for YouTube search
query = "Machine Learning"     limit = 100  #
Number of videos to download # Create a
queue to hold video URLs     video_queue =
Queue()
        # Search for videos related to the query     query_results
= YouTube.search(query, max_results=limit)     for video in
query_results:          video_queue.put(video['url'])
# Output path for downloaded and converted files
output_path = "./output"     os.makedirs(output_path,
exist_ok=True)
        # Number of threads to use     num_threads
= 5 # Create and
start threads     threads = []     for _ in range(num_threads):          thread =
threading.Thread(target=thread_worker, args=(video_queue, output_path))
        thread.start()
threads.append(thread)
        # Wait for all threads to complete
video_queue.join()
# Stop threads by placing None in queue
for _ in range(num_threads):
video_queue.put(None)         # Join threads
for thread in threads:
```

```
    thread.join() if
__name__ == "__main__":
  main()
```

Output :

```
Downloaded video: Machine Learning for Beginners | What is Machine Learning? | Introduction to ML | Edureka

Converted video to audio: Machine Learning for Beginners | What is Machine Learning? | Introduction to ML | Edureka.mp3

Downloaded video: Machine Learning | Machine Learning Tutorial for Beginners | Machine Learning Basics | Edureka

Converted video to audio: Machine Learning | Machine Learning Tutorial for Beginners | Machine Learning Basics | Edureka.mp3

Downloaded video: Machine Learning Full Course - Learn Machine Learning in 10 Hours | Machine Learning Tutorial | Edureka

Converted video to audio: Machine Learning Full Course - Learn Machine Learning in 10 Hours | Machine Learning Tutorial | Edureka.mp3

Downloaded video: Machine Learning - Full Course | Machine Learning Tutorial for Beginners | Data Science | Edureka

Converted video to audio: Machine Learning - Full Course | Machine Learning Tutorial for Beginners | Data Science | Edureka.mp3

Downloaded video: Machine Learning Course | Machine Learning Tutorial for Beginners | Edureka

Converted video to audio: Machine Learning Course | Machine Learning Tutorial for Beginners | Edureka.mp3

Downloaded video: Introduction to Machine Learning | Machine Learning Tutorial for Beginners | Edureka

Converted video to audio: Introduction to Machine Learning | Machine Learning Tutorial for Beginners | Edureka.mp3

Downloaded video: Machine Learning Interview Questions | Machine Learning Questions and Answers | Edureka

Converted video to audio: Machine Learning Interview Questions | Machine Learning Questions and Answers | Edureka.mp3

Downloaded video: Machine Learning Full Course - Learn Machine Learning in 3 Hours | Machine Learning Tutorial | Edureka

Converted video to audio: Machine Learning Full Course - Learn Machine Learning in 3 Hours | Machine Learning Tutorial | Edureka.mp3
```

Downloaded video: Machine Learning in Python | Machine Learning Tutorial for Beginners | Machine Learning Training | Edureka

Converted video to audio: Machine Learning in Python | Machine Learning Tutorial for Beginners | Machine Learning Training | Edureka.mp3

Downloaded video: Machine Learning Engineer | How to Become a Machine Learning Engineer | Edureka

Converted video to audio: Machine Learning Engineer | How to Become a Machine Learning Engineer | Edureka.mp3