


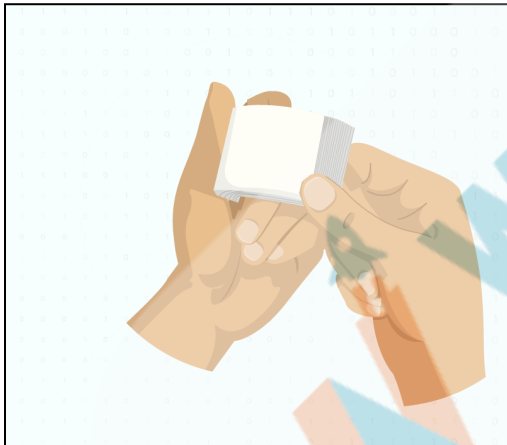


Topic	CREATING A VIDEO	
Class Description	The student will learn about the concept of how videos and webcam are accessed and saved using OpenCV. The student will create a video using multiple images with the help of OpenCV in Python.	
Class	PRO C105	
Class time	50 mins	
Goal	<ul style="list-style-type: none"> • Learn to access webcam using OpenCV. • Learn to access and save videos using OpenCV. • Create a Video from multiple images. 	
Resources Required	<ul style="list-style-type: none"> • Teacher Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code • Student Resources: <ul style="list-style-type: none"> ○ Laptop with internet connectivity ○ Webcam ○ Earphones with mic ○ Notebook and pen ○ Visual Studio Code 	
Class structure	Warm-Up Teacher-Led Activity 1 Student-Led Activity 1 Wrap-Up	10 mins 20 mins 15 mins 05 mins
Credit	OpenCV was started at Intel in 1999 by Gary Bradsky . The class uses Video by Anthony Shkraba from Pexels The images used for creating the Sunrise video are captured by Nikunj Patel from Pexels The class uses GIFs sourced from wiki commons.	

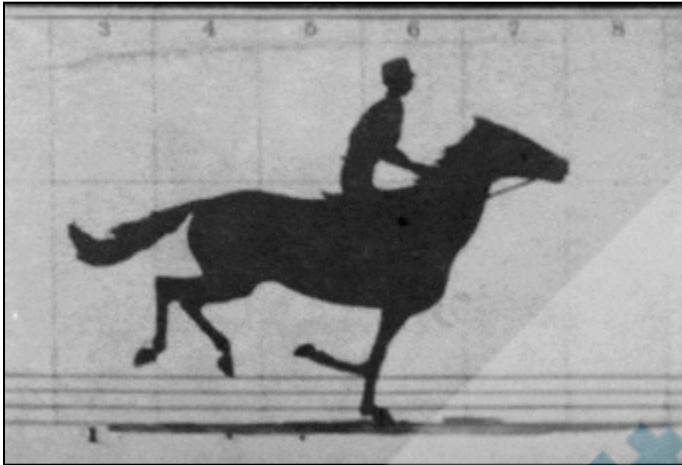
WARM-UP SESSION - 05 mins	
<div>  <p>Teacher Starts Slideshow Slide 1 to 5 Refer to speaker notes and follow the instructions on each slide.</p> </div>	
Teacher Action	Student Action
<p>Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today?</p> <p>Following are the WARM-UP session deliverables:</p> <ul style="list-style-type: none"> Greet the student. Revision of previous class activities. Quizzes. 	<p>ESR: Hi, thanks! Yes I am excited about it!</p> <p>Click on the slide show tab and present the slides</p>
WARM-UP QUIZ Click on In-Class Quiz	
<div>  <p>Continue WARM-UP Session Slide 6 to 16</p> </div>	
<p>Following are the session deliverables:</p> <ul style="list-style-type: none"> Appreciate the student. Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students. 	
<div>  <p>Teacher Ends Slideshow</p> </div>	
TEACHER-LED ACTIVITY - 20 mins	
Teacher Initiates Screen Share	

ACTIVITY

- Introduce the concept of how videos are created.
- Open webcam stream as a video using OpenCV.
- Experiment with fps of a video.

Teacher Action	Student Action
<p>Well done! You have answered both the questions well.</p> <p>Video:</p> <p>Before we begin, can you tell me what is a video?</p> <p>Let's understand it through the visual below.</p> <p><i>Note: Open Visual Aids to show this GIF to the student.</i></p>  <p>What is happening here?</p> <p>Correct!</p>	<p>ESR: Varied</p> <p>ESR: It is a flipbook, with many images. When the pages are flipped faster, it feels like a moving animation.</p>

It all started with a simple experiment in the year 1898, in Palo Alto, California. A photographer **Eadweard Muybridge** combined a series of 24 still photographs of a horse, which when appeared in sequence produced a motion picture of the horse galloping.



A video is a sequence of fast-moving images.

Understanding Frames Per Second:

The question that comes to mind is how fast are the pictures moving?

The measure of how fast the images are transitioning is given by a metric called **frames per second(fps)**.

Let's say a video has a **fps** of **40**, it simply means that 40 static images are being displayed every second.

ESR: Varied

A normal YouTube video that we watch runs at 30 fps, and a movie in a cinema hall runs at 24 fps.

The higher the frame rate, the smoother the video. This is because we have more images to show in one second.
For example, if you increase the YouTube video to 50/60 fps, you will feel the object in the videos are moving very smoothly.

Let us see how changing the fps changes the speed of the video:

Note: *Open Visual Aids to show this GIF to the student.*

- If we **increase fps to 50**; the video becomes much smoother.



- If we **decrease the fps to 5**, the video will lag, as now it has fewer images to show per second.



In today's class, we will learn to work with the video and display it using **OpenCV**.

Ready to get started?

ESR: Yes.

*The teacher downloads [Teacher Activity 1](#). Unzips the folder and saves it as C105. Opens in Visual Studio Code. Open a file named **Read_Video.py***

We can start with importing **OpenCV**.

*The teacher writes **import cv2**.*

Read Video:

In OpenCV, a video can be read:

- by using the feed from a camera connected to a computer
- by reading a video file

In both cases, the first step towards reading a video file is to create an object **VideoCapture** class. Its argument can be either the device index or the name of the video file to be read.

Syntax1:

```
cv2.VideoCapture(device_index )
```

Syntax2:

```
cv2.VideoCapture(video_path)
```

Device Index:

We will first read from the camera of a device. If you have only one camera connected to the system, we pass '0'.

It is important to remember that OpenCV only uses the camera connected to the computer.

When more than one camera is connected to the computer we can select:

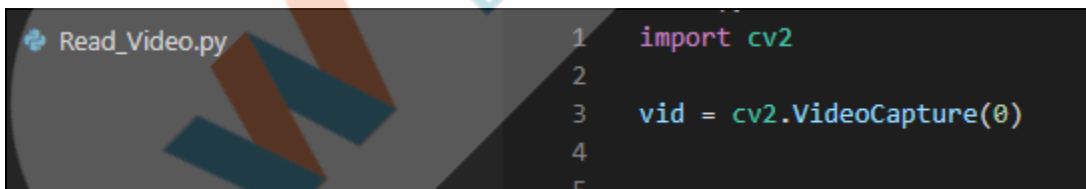
- The second camera by passing '1',
- The third camera by passing '2', and so on.

These numbers are known as **device indexes**.

*The teacher creates a variable, **vid** and assigns a **VideoCapture(0)** object to **vid**.*

IMP Note: If the device index 0 does not work with the systems having multiple cameras, try with index 1, 2, 3, 4 or 5. Else try the **cv2.CAP_ANY** flag along with the device index as **cv2.VideoCapture(0, cv2.CAP_ANY)**.

Reference: [OpenCV: Flags for video I/O](#)



```
1 import cv2
2
3 vid = cv2.VideoCapture(0)
4
5
```

Before we learn how to display the video. Let us check if the video is captured or not.

We can use the **if** condition to check if the camera works successfully.

We can use the **.isOpened()** method which returns **true** if **VideoCapture()** has worked successfully.

If the result is **false**, we can print a message to display “**Unable to read camera feed**”.

```
vid = cv2.VideoCapture(0)

if(vid.isOpened()==False):
    print("Unable to read the feed")
```

Next, let's move onto finding properties of our frames captured via camera's video.

Accessing Properties of Video:

Let us extract some properties from our capture video frames via webcam:

- Frame width
- Frame height
- Frames per second (fps).

These properties of captured video frames are defined in OpenCV, as these constant values:

Note: *There are many video properties in OpenCV, but we will limit our discussion with these three.*

- cv2.CAP_PROP_FRAME_WIDTH
- cv2.CAP_PROP_FRAME_HEIGHT
- cv2.CAP_PROP_FPS

Properties of any video can be read using the **get()** method of the **VideoCapture** class. The values of the properties returned by the **get()** method will be as **float** values, that is, values decimal points. We can convert these values to integer numbers using the **int()** method.

Syntax:

```
vid_obj.get(property_name)
```

Example:

```
vid.get( cv2.CAP_PROP_FRAME_WIDTH )
```

Now let's write the program is get the values of these properties of the captured video frames as integers:

1. Create a variable **height**, assign the value of property using the **get()** method, convert to integer using **int()** and print the **height** variable value.

```
height = int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT))  
print(height)
```

2. Create a variable **width**, assign the value of property using the **get()** method, convert to integer using **int()** and print the **width** variable value.

```
width = int(vid.get(cv2.CAP_PROP_FRAME_WIDTH))  
print(width)
```

3. Create a variable **fps** , assign the value of property using the **get()** method, convert to integer using **int()** and print the **fps** variable value.

```
fps = int(vid.get(cv2.CAP_PROP_FRAME_FPS))  
print(fps )
```

4. Run the code using **python Read_Video.py/py**
Read_Video.py.

```
height = int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT))  
print(height)  
width = int(vid.get(cv2.CAP_PROP_FRAME_WIDTH))  
print(width)  
fps = int(vid.get(cv2.CAP_PROP_FPS))  
print(fps)
```

OUTPUT(In the Terminal)

```
480.0  
640.0  
30.0  
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-s  
'::SourceReaderCB::~SourceReaderCB_terminating async callback
```

As we can see in the Terminal, the height of video is **480** px, width **640** px, and it runs **30** frames per second(fps).

After that it displays a warning as it is not able to display the video.

Display a Video:

We know that a video consists of multiple frames, so to show the video we need to first extract the frames of the video, then display the individual frame as an image.

What could we use here?

Yes. Correct!

We can use the **read()** method of the **VideoCapture** class to display a video. At the beginning of this process, it starts with the first frame. We will use a **while loop** so that the **vid.read()** method will load every frame from the video file.

Before we can understand how to read the video, let's quickly understand Tuples data type in Python.

Tuples are similar to list data types in every sense, except the value of a particular element cannot be reassigned!

To understand this open:

[Teacher Reference Document\(Understanding Tuples\)](#)

[Teacher Activity 2\(Colab Link\)](#)

Note: Run all the colab cells one-by-one to show and explain the operations on Tuples.

ESR: We can use a loop.

Now that we have understood how tuples work in Python, let's read the video frames using the **read()** method.

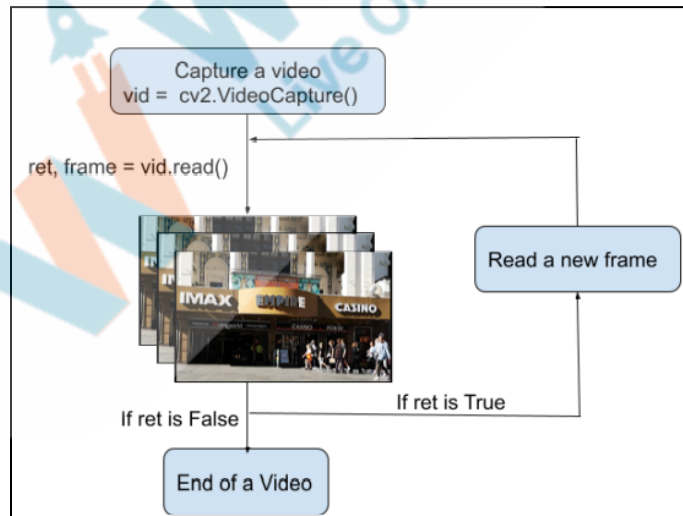
The **read()** method returns a **tuple** of two elements where:

- The first element is a boolean. (Variable **ret** in the image below)
- The second element is the actual video frame. (Variable **frame** in the image below)

Note: The variable **ret** and **frame** are user-defined. The value returned by the **read()** method is assigned to **ret**, **frame** separately.

When the first element is **True**, it indicates the video stream contains a frame to read. As long as the value of **ret** is **True** it keeps on reading the next frame. If the value is **False**, that means all the frames of the video have been read.

Note: The diagram below is for the Teacher Reference only to understand how a **while** loop works on Video.



Let's write a program to read and display video frames:

1. Start a **while** loop that reads the frames from our video continuously.
2. Take **ret** and **frame** variable to get **vid.read()** values:
 - **ret** value, a boolean data type that returns **True** if Python is able to read the **VideoCapture** object.
 - **frame** value
3. After reading a file, display the video frame by frame. A frame of a video is simply an image, and we display each frame the same way we display images, i.e., we use the function **imshow()**.

```
while(True):  
  
    # Capture the video frame  
    # by frame  
    ret, frame = vid.read()  
  
    cv2.imshow("Web cam", frame)
```

Remember, in last class we used the **waitKey()** post **imshow()** method?

In the case of an **image**, we pass **0** to the **waitKey()** method, but for playing a **video**, we need to pass a number **greater than 0** to the **waitKey()** method. Can you try to think why we will need a number greater than 0?

This is because **0** would pause the frame in the video for an infinite amount of time and in a video, we need each frame to be shown one after another after a certain interval of time.

ESR: Varied

ESR: Varied

The, **cv2.waitKey()** waits for a specific time in milliseconds before closing the output window until you press any button on the keyword in between.

If any key is pressed, the corresponding **ASCII code** is assigned to **waitKey()**. We can then take the desired action based on the key pressed.

Remember, in JavaScript, we learned about ASCII code for each key on the Keyboard, where we use 32 for the space key?

Note: The teacher can share & open the [Teacher Reference 1\(ASCII Chart\)](#).

4. Write a condition to break the **while** loop, if the user pressed a space key, by comparing **cv2.waitKey(1)** to 32.
5. Add **vid.release()** method to close the video.

ESR: Varied

```
if cv2.waitKey(1) ==32:
    break
vid.release()
```

OUTPUT:

BEFORE RUNNING THE CODE THE TEACHER WILL HAVE TO SWITCH OFF THEIR CAMERA IN THE CLASS, THE WEBCAM CAN BE ACCESSED BY ONE APPLICATION AT A TIME. THE OUTPUT WILL SHOW THE TEACHER'S FACE.

Note: The teacher can close the video by pressing a **space key** on the keyboard.

Let us try to view a video file similarly.

In the **VideoCapture()** method, instead of **0** now I will replace it with the name of a video.

Few points you should remember are:

- The video name should include an extension.
- If the video is stored in the same folder, give only the name of the file.
- If the video is stored in a different folder, provide the entire path.

We have our video stored in the same folder, so we will simply replace **0** with the **video name**.

```
import cv2  
  
vid = cv2.VideoCapture("AnthonyShkraba.mp4")  
  
if(vid.isOpened()==False):  
    print("Unable to read the feed")
```

After checking the output, it is good practice to close all windows.

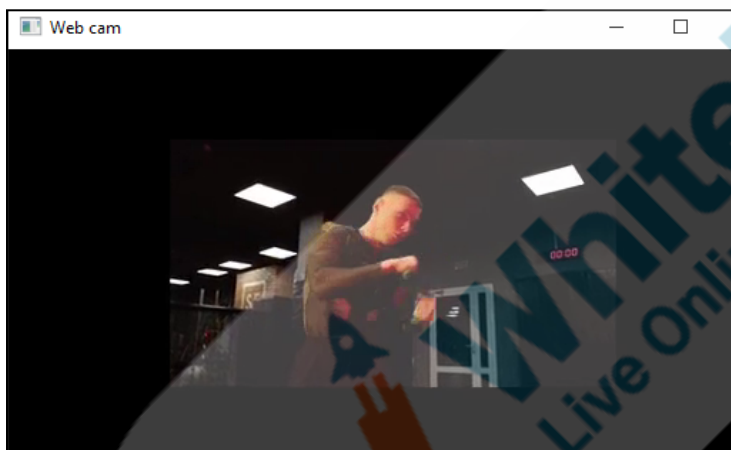
We can achieve that by using **cv2.destroyAllWindows()**. The windows will be automatically closed when **space** is pressed or when the video is over.

```
# Release the video capture object
vid.release()

# Closes all the frames
cv2.destroyAllWindows()
```

Run the code.

*The video will play; press 'q' to end the video OR it will be closed once the video is complete because of **cv2.destroyAllWindows()***



The video was very fast, we could hardly see any movements in the video.

What should we do to run the video slowly to observe and learn from his moves?

We can increase the time in **waitkey()**. This will allow the video to wait before switching to the new frame. Let us change it to **25** and see the difference.

ESR: Varied


```
if cv2.waitKey(25) < 0:
    break
```

Run the code: *The same video will play slowly.*

By increasing the **waitKey(25)** we can slow down the display of the video frame, but it does not change anything in the original video.

Write a Video:

What if we want to save the video with this speed to view it again later?

Similar to **cv2.imwrite()**, which allows us to save images. **OpenCV** has a method named **cv2.VideoWriter()** to save the video. It also allows us to save the video in the local system.

Syntax:

```
cv2.VideoWriter(filename, fourcc, fps, frameSize, isColor)
```

Note: Open [Teacher Reference 2](#) to check detailed syntax.

Parameters of VideoWriter():

- **filename:** The first parameter is the name video file to be saved as the output. Specify the **filename** with **extension** of the file.

ESR: We can use **cv2.imwrite()**

- **fourcc**: The second parameter is the **fourcc code**(*four-character code*). It is a 4-byte code which is used to compress the frames and to specify the **video codec**.

A **video codec** is software that compresses a digital video so that it can be easily stored and can be played again.

Some of the most common codecs are: **DIVX**, **XVID**, **X264**, **MJPEG** and others. Video file formats such as **AVI** (.avi), **MP4**(.mp4) and **Windows Media Video**(.wmv) are most commonly used to store digital video data.

- **fps**: The third parameter is a number of frames per second (**fps**) which defines how many frames you want to store per second in the output video file.
- **frameSize**: The fourth argument is the **size**, which is the **width** and the **height** of our video frame. In our code, we are going to provide this argument in the form of a **tuple (640, 480)**.
- **isColor**:The last parameter is optional; it controls whether or not we are writing colored frames to the file. A value of **True** indicates that we are writing color frames. Supplying **False** indicates we are not writing colored frames (Default is **True**)

Let us declare a variable named **out**, before the **while** loop, and set the **VideoWriter()** parameters as to initialize the object:

- **filename:** 'Boxing.mp4'
- **fourcc:** `cv2.VideoWriter_fourcc(*'DIVX')`
- **fps:** 30
- **frameSize:** (width, height)

```
height = int(vid.get(cv2.CAP_PROP_FRAME_HEIGHT))
print(height)
width = int(vid.get(cv2.CAP_PROP_FRAME_WIDTH))
print(width)
fps = int(vid.get(cv2.CAP_PROP_FPS))
print(fps)

out = cv2.VideoWriter('Boxing.mp4', cv2.VideoWriter_fourcc(*'DIVX'), 30, (width, height))

while(True):|
```

Once the **VideoWriter** object is initialised by specifying the arguments, we are now ready to add frames to the **out** variable we just created.

The output frame is written to the file using the **write** method of the **cv2.VideoWriter()**.

We will keep adding frames to the **out** variable. Hence, add inside **while loop** - **out.write(frame)** and then close the video by **out.release()**

```
while(True):

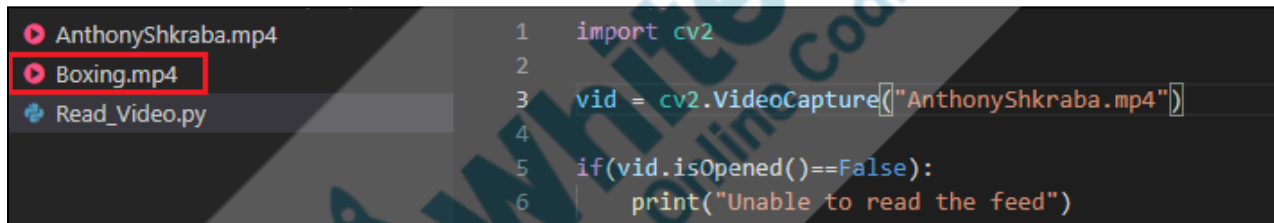
    # Capture the video
    # by frame
    ret, frame = vid.re

    cv2.imshow("Web cam
    out.write(frame)
    if cv2.waitKey(25)
        break

vid.release()
out.release()
```

OUTPUT:

The video will play, and a new video named **Boxing.mp4** will be added in the same folder.



As you can see, the output video is slower than the original one. You can further reduce the speed by decreasing **fps**, or you can run it faster by increasing **fps**.



Are you enjoying working with OpenCV?

ESR: Yes

As we just saw, for smoother video, multiple frames need to run per second. We will give you around 90 frames. Would you like to create a video with these frames using OpenCV?

ESR: Yes

Share your screen and I will guide you.

Teacher Stops Screen Share	
<div>  </div> <p>Teacher Starts Slideshow Slide 17 to 18</p> <p>Refer to speaker notes and follow the instructions on each slide.</p>	
<p>We have one more class challenge for you. Can you solve it?</p> <p>Let's try. I will guide you through it.</p>	
<div>  </div> <p>Teacher Ends Slideshow</p>	
STUDENT-LED ACTIVITY - 15 mins	
<ul style="list-style-type: none"> • Ask the student to press the ESC key to come back to the panel. • Guide the student to start Screen Share. • The teacher gets into Fullscreen. 	
<p>ACTIVITY</p> <ul style="list-style-type: none"> • Create a video using given images • Save the video on a local device. 	
Teacher Action	Student Action
<p><i>The teacher guides the student to download Student Activity 1.</i></p> <p>Note: <i>The student activity folder includes 92 images sorted and named in the sequence. The boilerplate includes code to read and check image files and append to a list (similar to what was covered in C102).</i></p>	<p><i>The Student downloads Student Activity 1. Unzip the folder and open it in VSC.</i></p>

As you can see, there are more than 90 images given in the images folder. Can you open a few images and tell me what kind of video we want you to create?

ESR: Varied

The images show the sun at different positions. We can create a Sunset or a Sunrise video based on the sequence that we will follow while adding frames in **cv2.VideoWriter()**.

Sunset Image Sequence:

If you add **110.jpg** first and then keep adding till **201.jpg**, you can create a video showing Sunset.

Sunrise Image Sequence:

Similarly, if you start from **201.jpg** and then keep going down till **110.jpg**. You can create a video showing the Sunrise.

ESR: Varied

So, what do you want to show in the video, Sunset or Sunrise?

***Note:** The guidelines given in this doc are for generating the Sunrise video. (Hint is given below to generate a video for Sunset)*

***Note:** The below steps are similar to what we learned in a few classes earlier while moving images files from the download folder to another folder.*

Let's take a walkthrough through the program given **CreateVideo.py**.

1. We are importing **cv2** and **os** module
2. We set a **path** for the **Images** folder.
3. Created a **list** variable named **Images = []**.
4. Using a **for loop** to check each file in the folder using **os.listdir(path)**

Now, for each file name, we are using `os.splitext(file)` to separate the name and extension from a file name. We are doing this step to pick up only image files from the folder.

***Note:** In this folder, we are sure that we only have images. But when you want to run the same code on a different folder, It might have other files. So this step and condition will help us to pick up image files only.*

5. Create a variable **file_name** by concatenating the path “/” and file name(includes both name and extension).
6. Use **print(file_name)** to make sure we are getting the correct result.
7. Add each file in the **images** list using **.append()** method
8. Print the length of images to check how many images are added in the list using **print(len(images))**.

```
import os
import cv2

path = "Images"
images = []

for file in os.listdir(path):
    name, ext = os.splitext(file)

    if ext in ['.gif', '.png', '.jpg', '.jpeg', '.jfif']:
        file_name = path+"/"+file

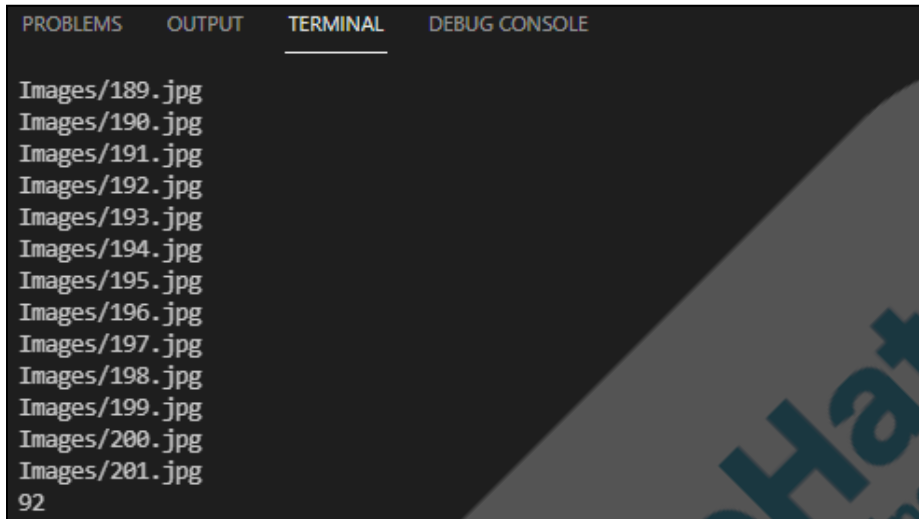
        print(file_name)

        images.append(file_name)

print(len(images))
count = len(images)
```

OUTPUT(In the Terminal)

All the image names will be printed with a total number of images.



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

Images/189.jpg
Images/190.jpg
Images/191.jpg
Images/192.jpg
Images/193.jpg
Images/194.jpg
Images/195.jpg
Images/196.jpg
Images/197.jpg
Images/198.jpg
Images/199.jpg
Images/200.jpg
Images/201.jpg
92
```

Great! So the list of images is ready. What should we do next?

Superb!

We will use **cv2.VideoWriter()** to create a video. One of the parameters is **frameSize**. We need to pass the size of a video, for this we will need the size of each frame.

Since each frame is an image, we can use the **shape** property of the image. The shape includes the values of:

- number of rows(image height),
- number of columns(image width) and
- channel or color band

The shape property values are returned as tuple of (**rows, cols, channels**)

ESR: We can use **VideoWriter()** and add each image in that.

The value of channels is only returned for the colored image.

For the **grayscale** images, the shape returns only (**rows, cols**)
Now let's write the program to get the size a frame:

1. Create a variable named **frame** to read the first image from the **images** list.

frame = cv2.imread(images[0])
2. Take three variables: **width, height** and **channels**.
3. Assign the values to variables using **frame.shape** to get **width, height & channels**
4. Create a tuple variable **size** using **width, height**.
5. Use **print(size)** to check the result.

```
print(len(images))  
count = len(images)  
  
frame = cv2.imread(images[0])  
height, width, channels = frame.shape  
size = (width,height)  
print(size)
```

Run the code OUTPUT:

```
Images/201.jpg  
92  
(1280, 720)
```

The size of image is 1280 px width & 720 px height

We will use this size while creating a video, along with the following parameters.

Let us declare a variable named **out**, before the **while** loop, and set the **VideoWriter()** parameters as to initialize the object:

- **filename:** project.mp4'
- **fourcc:** cv2.VideoWriter_fourcc(*'DIVX')
- **fps:** 5
- **frameSize:** size

```
out = cv2.VideoWriter('project.mp4',cv2.VideoWriter_fourcc(*'DIVX'), 5, size)
```

What shall we do next?

We have stored images in a list format, and to traverse from the list we need to use a **for** loop.

For creating a video showing the sunrise, we will start from the last image and move towards the first.

For that, we will use the **range()** function. Do you remember how the **range()** function works?

The **range()** function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Similarly, for counting downwards we start with the last number till zero, reducing by 1.

ESR: Write each image from a list of **images** into the **out** variable using a loop.

ESR: Varied

Creating/Writing a Video:

1. Create a **for** loop to traverse through **range(count, 0, -1)**:

As the elements in the list are numbered from 0 and the value of **count variable** used to assign the value of **length of images list** above is **92**.

Range will start from 91(count) till it reaches 0, reducing the number by -1 in each loop.

***Note:** The teacher can ask the student to comment out all previous **print()** statements used till now.*

2. Read each frame using the **imread()** method and assign it to the **frame** variable.
3. Write the frame in the **out** variable using **write()** method
4. Close the video after for loop using **release()** method
5. Print a message to know the video is complete

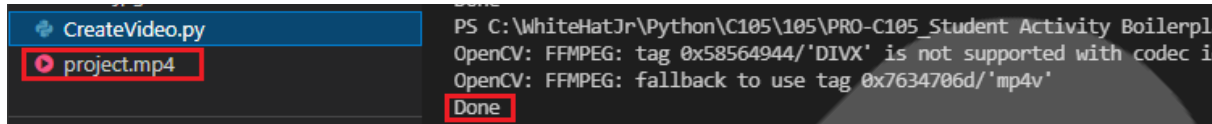
```
out = cv2.VideoWriter('project.mp4',cv2.VideoWriter_fourcc(*'DIVX'), 5, size)

#For SUNSET
#for i in range(0,count-1):

#For SUNRISE
for i in range(count-1,0,-1):
    frame = cv2.imread(images[i])
    out.write(frame)

out.release() # releasing the video generated
print("Done")
```

Output in Terminal shows **Done** and a file gets added in the same folder named as **project.mp4**



Note: The teacher can guide students to open the folder structure and run the video from there to check the result. If time permits, change the fps values to create faster or slower video output.

The result for [Sunrise Output Reference](#) with fps 5.

The result for [Sunset Output Reference](#) with fps 27.

Wow, the sunrise & sunset looks beautiful.

Remember to have a smoother video you need more images capturing immediate actions.

You can now create a video for your friends and families using the photos you already have or taking from Google using Python.

Isn't that exciting?

With this, we will stop for today.

ESR: Yes

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Teacher Starts Slideshow
Slide 19 to 24



Activity Details:

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ

Click on In-Class Quiz

Continue WRAP-UP Session
 Slide 25 to 30




Activity Details:



Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate the student for his/her efforts in the class.**
- **Ask the student to make notes for the reflection journal along with the code they wrote in today's class.**

Teacher Action	Student Action
<p>You get Hats off for your excellent work!</p> <p>In the next class, we will learn how we can use OpenCV to detect the face and highlight it using a rectangle.</p>	<p><i>Make sure you have given at least 2 Hats Off during the class for:</i></p> <div> Creatively Solved Activities  </div>

	<div style="background-color: #0072bc; color: white; padding: 5px; margin-bottom: 5px; display: flex; justify-content: space-between; align-items: center;"> Great Question  +10 </div> <div style="background-color: #0072bc; color: white; padding: 5px; display: flex; justify-content: space-between; align-items: center;"> Strong Concentration  +10 </div>
PROJECT OVERVIEW DISCUSSION Refer the document below in Activity Links Sections	
Teacher Clicks	<div style="background-color: #ff0000; color: white; padding: 10px 20px; border-radius: 15px; display: inline-block;"> ✕ End Class </div>

ACTIVITY LINKS		
Activity Name	Description	Link
Teacher Activity 1	Teacher Boilerplate	https://github.com/procodingclass/PRO-C105-Teacher-Boilerplate
Teacher Activity 2	Tuple Operations	https://colab.research.google.com/drive/1yUo_Z5ldMBAQ7VSdfh2ICRbp68YiKnTE?usp=sharing
Teacher Activity 3	Reference Code for Teacher Activity	https://github.com/procodingclass/PRO-105-Reference-Code-Teacher-Activity
Teacher Activity 4	Reference Code for Student Activity	https://github.com/procodingclass/PRO-105-Reference-Code-Student-Activity
Student Activity 1	Student Boilerplate	https://github.com/procodingclass/PRO-C105-Student-Boilerplate
Teacher Reference 1	ASCII Chart	https://commons.wikimedia.org/wiki/File:ASCII-Table-wide.svg

Teacher Reference 2	The cv2.videoWriter() Syntax	https://docs.opencv.org/3.4/dd/d9e/class_cv_1_1VideoWriter.html
Teacher Reference 3	Understanding Tuples	https://s3-whjr-curriculum-uploads.whjr.online/e45fd28f-119b-4699-bf7d-10f57fe00841.pdf
Teacher Reference 4	Sunrise Output	https://s3-whjr-curriculum-uploads.whjr.online/4d3bd681-7b8d-4b2b-96e4-091f2e75a9f9.gif
Teacher Reference 5	Sunset Output	https://s3-whjr-curriculum-uploads.whjr.online/733f647f-dd63-46e9-b757-150c4bfcf976.mp4
Teacher Reference 6	Project Document	https://s3-whjr-curriculum-uploads.whjr.online/f46d8a6d-039d-47ab-bf7e-277319a91187.pdf
Teacher Reference 7	Project Solution	https://github.com/procodingclass/PRO-104-Project-Solution
Teacher Reference 8	Visual Aid Link	https://s3-whjr-curriculum-uploads.whjr.online/61b25722-954d-4dcf-9520-550e972e59d5.html
Teacher Reference 9	In Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/7b6320b0-377c-4f80-a562-89acdad1711f.pdf