



<b>Subject: Programming With Python (01CT1309)</b>	<b>Aim:</b> Exploring Convolution and Correlation in Discrete-Time Signals	
<b>Experiment No: 18</b>	<b>Date:</b>	<b>Enrollment No: 92510133011</b>

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**IDE:**

**What is Convolution?**

**Convolution** is a mathematical operation that combines two functions (or signals) to produce a third function, showing how one signal modifies or filters another. It essentially describes how the shape of one signal is altered by another signal. In the context of **signal processing**, convolution is used to analyze the effect of a system (such as a filter) on an input signal.

Mathematically, the **convolution** of two discrete-time signals  $x[n]$  and  $h[n]$  is given by:

$$x[n] = (x * h)[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k]$$

In simple terms, the convolution between two signals involves "sliding" one signal over another, multiplying overlapping values, and summing the results to produce a new signal.

**Types of Convolution:**

1. **Linear Convolution:** This computes the output for the entire range of the input signals and doesn't assume periodicity.
2. **Circular Convolution:** This assumes the signals are periodic and wraps the values around when they go beyond the signal length.

**Python Implementation**

Example 1:

```
import numpy as np
# Define two input signals
x = [1, 2, 3, 4]
h = [1, 0, -1]
# Perform linear convolution using numpy
linear_conv = np.convolve(x, h)
print("Linear Convolution: ", linear_conv)
```

```
Linear Convolution:  [ 1  2  2  2 -3 -4]
PS E:\python codes>
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**Experiment No: 18**

**Date:**

**Enrollment No: 92510133011**

### Circular Convolution

Circular convolution can be implemented by first padding the shorter sequence to the length of the longer one and then using the Fast Fourier Transform (FFT) for efficient computation.

Example 2:

```
import numpy as np
```

```
# Define two input signals
```

```
x = [1, 2, 3, 4]
```

```
h = [1, 0, -1]
```

```
# Determine the length for circular convolution
```

```
N = max(len(x), len(h))
```

```
# Zero-pad the shorter sequence to match the length of the longer one
```

```
x_padded = np.pad(x, (0, N - len(x)), mode='constant')
```

```
h_padded = np.pad(h, (0, N - len(h)), mode='constant')
```

```
# Perform circular convolution using FFT
```

```
X_fft = np.fft.fft(x_padded)
```

```
H_fft = np.fft.fft(h_padded)
```

```
circular_conv = np.fft.ifft(X_fft * H_fft)
```

```
# Only the real part is considered (since the imaginary part should be zero)
```

```
circular_conv = np.real(circular_conv)
```

```
print("Circular Convolution: ", circular_conv)
```

```
Circular Convolution: [-2. -2. 2. 2.]
```

```
PS E:\python codes>
```

### Key Points:

- **Linear convolution:** Extends beyond the input lengths.
- **Circular convolution:** Assumes periodicity, so the result has the same length as the longest sequence.

### What is Correlation?

Correlation is a statistical measure that describes the extent to which two signals or datasets are related. It measures how much two variables change together. In signal processing, correlation is used to find similarities between two signals or between different parts of the same signal. Correlation helps in detecting patterns, shifts, or commonalities in signals.

 <b>Marwadi</b> <b>University</b> <small>Marwadi Chandarana Group</small>	<b>NAAC</b> 	<b>Marwadi University</b> <b>Faculty of Engineering &amp; Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Programming With Python (01CT1309)</b>	<b>Aim:</b> Exploring Convolution and Correlation in Discrete-Time Signals	
<b>Experiment No: 18</b>	<b>Date:</b>	<b>Enrollment No: 92510133011</b>

### Types of Correlation:

1. **Cross-Correlation:** Measures the similarity between two different signals as a function of time-lag.
2. **Autocorrelation:** Measures the similarity of a signal with a delayed version of itself, also as a function of time-lag.

#### Cross-Correlation

Cross-correlation is a measure of similarity between two different signals. It shows how one signal correlates with another signal over time, meaning how much one signal resembles a shifted version of another.

Mathematically, the cross-correlation of two discrete signals  $x[n]$  and  $y[n]$  is defined as:

$$R_{x,y}[k] = \sum_n x[n]y[n+k]$$

Where  $x[n]$  and  $y[n]$  are the two signals.  $k$  is the time-lag.  $R_{x,y}[k]$  is the cross-correlation at lag  $k$ .

Cross-correlation shifts one signal relative to the other and calculates the dot product (similarity) for each shift. The peak in the cross-correlation function indicates the time-shift where the signals are most similar.

#### Example

```
import numpy as np
import matplotlib.pyplot as plt
```

```
# Define two signals
x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 3, 4, 5, 6])

# Perform cross-correlation using numpy
```



**Subject: Programming With Python (01CT1309)**

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**Experiment No: 18**

**Date:**

**Enrollment No: 92510133011**

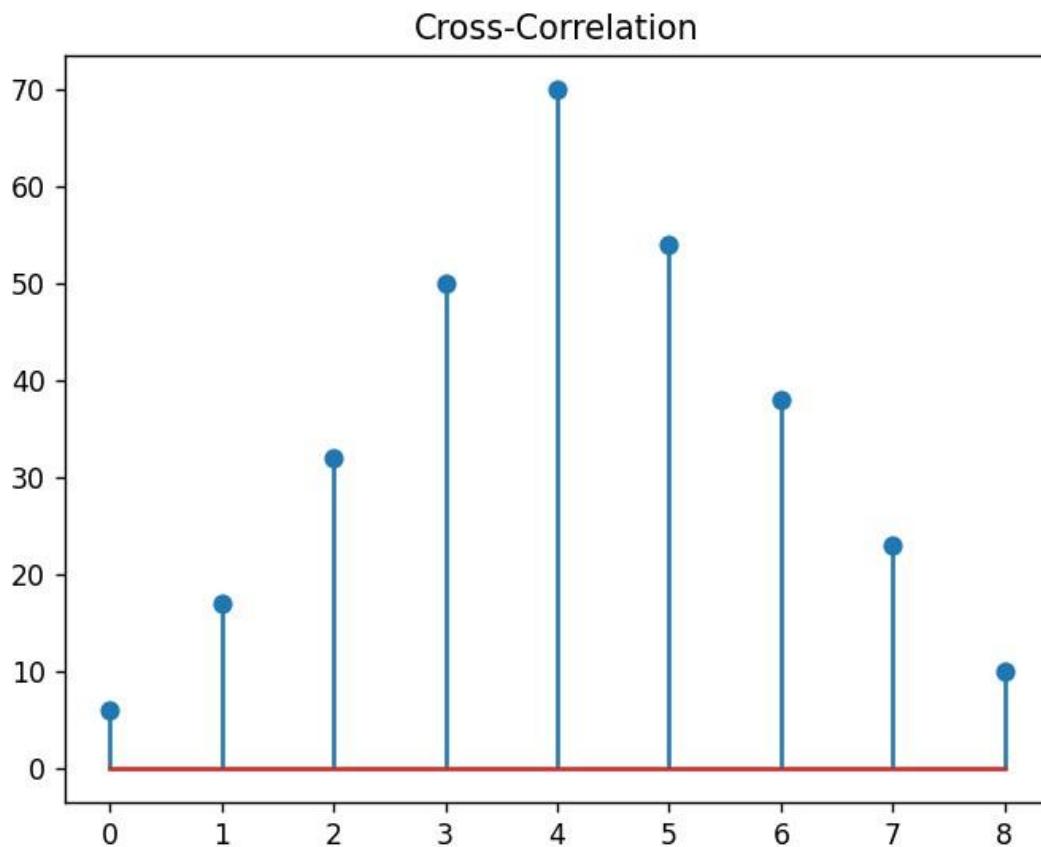
```
cross_corr = np.correlate(x, y, mode='full')
```

```
# Plot cross-correlation
```

```
plt.stem(cross_corr)
```

```
plt.title('Cross-Correlation')
```

```
plt.show()
```



### Autocorrelation

Autocorrelation is a special case of cross-correlation, where a signal is correlated with a delayed version of itself. It shows how much a signal resembles its own shifted version.

 <b>Marwadi</b> <b>University</b> <small>Marwadi Chandarana Group</small>	<b>NAAC</b> 	<b>Marwadi University</b> <b>Faculty of Engineering &amp; Technology</b> <b>Department of Information and Communication Technology</b>
<b>Subject: Programming With Python (01CT1309)</b>	<b>Aim:</b> Exploring Convolution and Correlation in Discrete-Time Signals	
<b>Experiment No: 18</b>	<b>Date:</b>	<b>Enrollment No: 92510133011</b>

Mathematically, the autocorrelation of a signal  $\{x[n]\}$  and  $\{x[n-k]\}$  is defined as:

$$R_{x,x}[k] = \sum_n x[n]x[n+k]$$

Where  $\{x[n]\}$  signal.  $k$  is the time-lag.  $R_{x,x}[k]$  is the auto-correlation at lag  $k$ .

Autocorrelation measures the internal structure of a signal by checking how it correlates with itself over time. A high autocorrelation at some lag indicates a repeating or periodic pattern in the signal.

```
# Define a signal
```

```
x = np.array([1, 2, 3, 4, 5])
```

```
# Perform autocorrelation using numpy
```

```
auto_corr = np.correlate(x, x, mode='full')
```

```
# Plot autocorrelation
```

```
plt.stem(auto_corr)
```

```
plt.title('Autocorrelation')
```

```
plt.show()
```



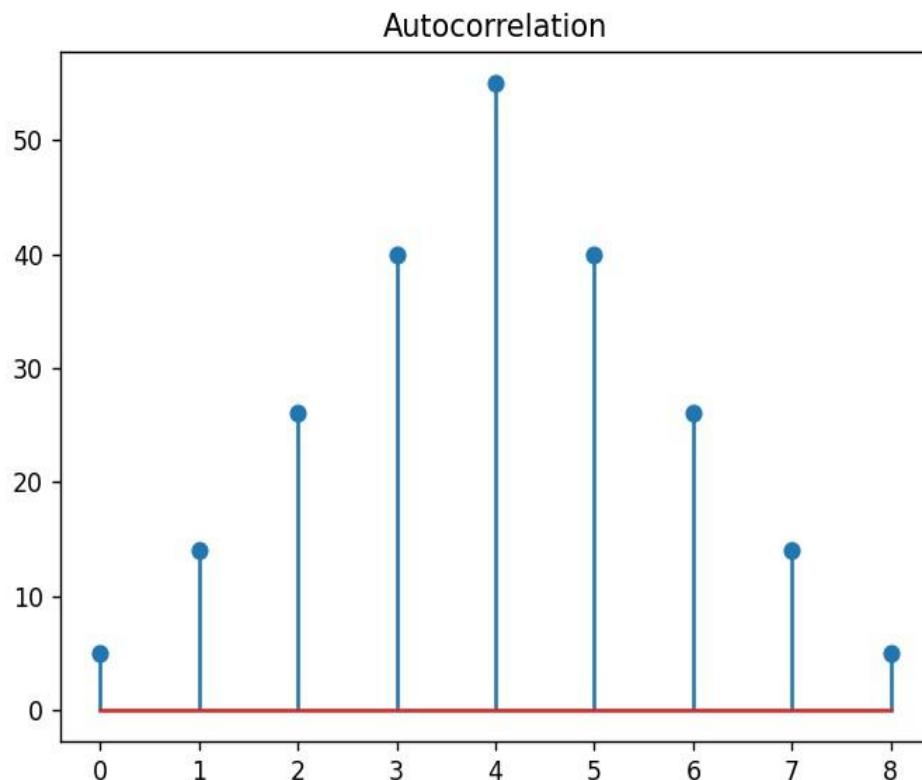
**Subject: Programming With Python (01CT1309)**

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**Experiment No: 18**

**Date:**

**Enrollment No: 92410133034**



**Post Lab Exercise:**

- Use Python to create a script that performs both **linear** and **circular convolution** on an audio file with an impulse response. Compare and visualize the results.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
import librosa

#audio file
audio_file, _ = librosa.load(r'C:\Users\HP\OneDrive\Desktop\Python\file_example_WAV_1MG.wav')
audio_data = np.array(audio_file, dtype=np.float32)
#impulse response
impulse_response_file, _ = librosa.load(r'C:\Users\HP\OneDrive\Desktop\Python\sunflower-street-drumloop-85bpm-163900.mp3')
impulse_response_data = np.array(impulse_response_file, dtype=np.float32)
```

**Subject: Programming With Python (01CT1309)**

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**Experiment No: 18**

**Date:**

**Enrollment No: 92510133011**

```
# Perform linear convolution
```

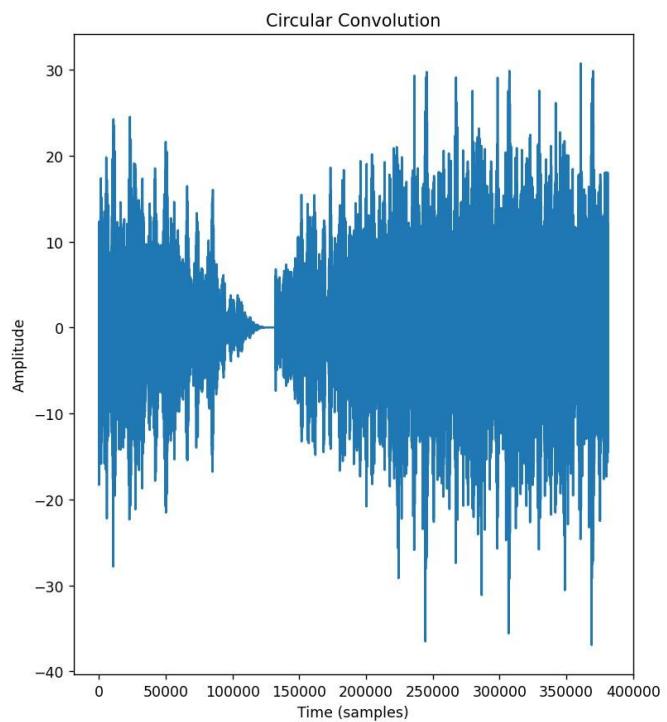
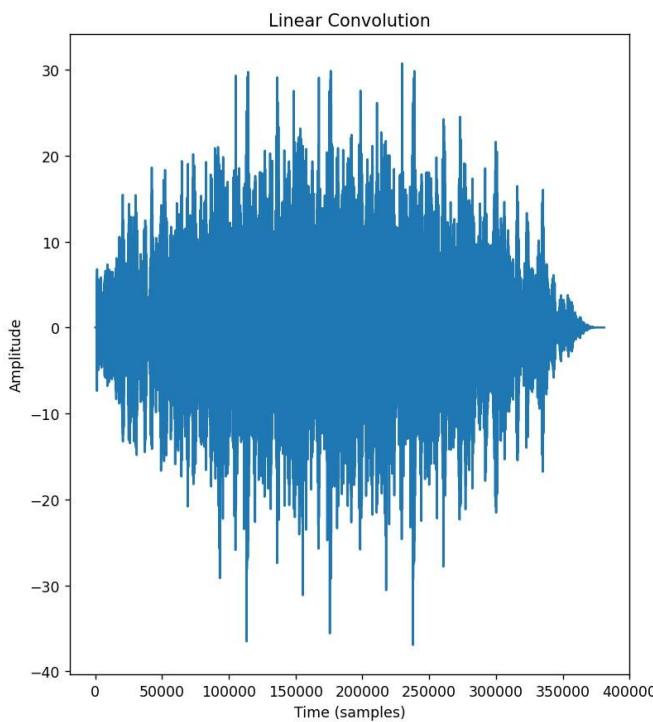
```
linear_convolved_data = np.convolve(audio_data, impulse_response_data, mode='full')
```

```
# Perform circular convolution
```

```
circular_convolved_data = np.roll(np.convolve(audio_data, impulse_response_data, mode='full'), -len(impulse_response_data) + 1)
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(linear_convolved_data)
plt.title('Linear Convolution')
plt.xlabel('Time (samples)')
plt.ylabel('Amplitude')
plt.subplot(1, 2, 2)
plt.plot(circular_convolved_data)
plt.title('Circular Convolution')
plt.xlabel('Time (samples)')
plt.ylabel('Amplitude')
plt.show()
```

Output:





<b>Subject: Programming With Python (01CT1309)</b>	<b>Aim:</b> Exploring Convolution and Correlation in Discrete-Time Signals	
<b>Experiment No: 18</b>	<b>Date:</b>	<b>Enrollment No: 92510133011</b>

- Use Python to implement both **cross-correlation** and **autocorrelation** on a set of audio files (clean\_audio.wav, noisy\_audio.wav, periodic\_audio.wav). Visualize and compare the results.

Code:

```
import librosa
import numpy as np
import matplotlib.pyplot as plt
clean_audio, _ = librosa.load(r'C:\Users\HP\OneDrive\Desktop\Python\sunflower-street-drumloop-85bpm-163900.mp3')
noisy_audio, _ = librosa.load(r'C:\Users\HP\OneDrive\Desktop\Python\babble.wav')
periodic_audio, _ = librosa.load(r'C:\Users\HP\OneDrive\Desktop\Python\file_example_WAV_1MG.wav')
def autocorrelate(signal):
    return np.correlate(signal, signal, mode='full')

clean_autocorrelation = autocorrelate(clean_audio)
noisy_autocorrelation = autocorrelate(noisy_audio)
periodic_autocorrelation = autocorrelate(periodic_audio)
def cross_correlate(signal1, signal2):
    return np.correlate(signal1, signal2, mode='full')

clean_noisy_crosscorrelation = cross_correlate(clean_audio, noisy_audio)
clean_periodic_crosscorrelation = cross_correlate(clean_audio, periodic_audio)
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)
plt.plot(clean_autocorrelation)
plt.title('Autocorrelation of Clean Audio')

plt.subplot(1, 3, 2)
plt.plot(noisy_autocorrelation)
plt.title('Autocorrelation of Noisy Audio')

plt.subplot(1, 3, 3)
plt.plot(periodic_autocorrelation)
plt.title('Autocorrelation of Periodic Audio')

plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
```



**Marwadi University**  
**Faculty of Engineering & Technology**  
**Department of Information and Communication Technology**

**Subject: Programming With Python (01CT1309)**

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**Experiment No: 18**

**Date:**

**Enrollment No: 92510133011**

```
plt.subplot(1, 2, 1)
plt.plot(clean_noisy_crosscorrelation)
plt.title('Cross-Correlation of Clean and Noisy Audio')

plt.subplot(1, 2, 2)
plt.plot(clean_periodic_crosscorrelation)
plt.title('Cross-Correlation of Clean and Periodic Audio')

plt.tight_layout()
plt.show()
```

Output:

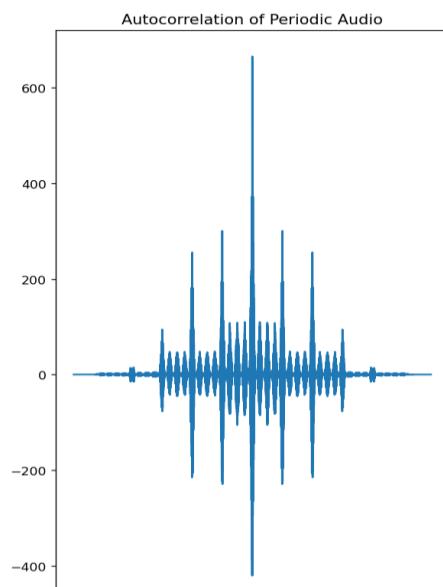
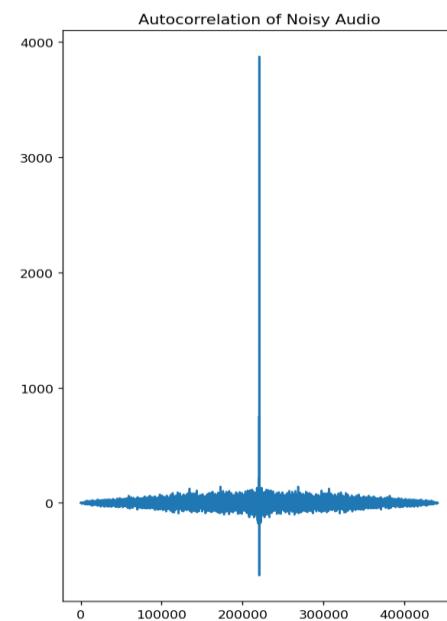
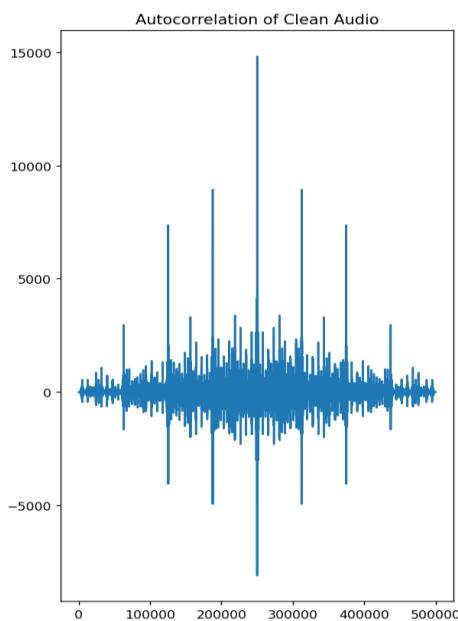
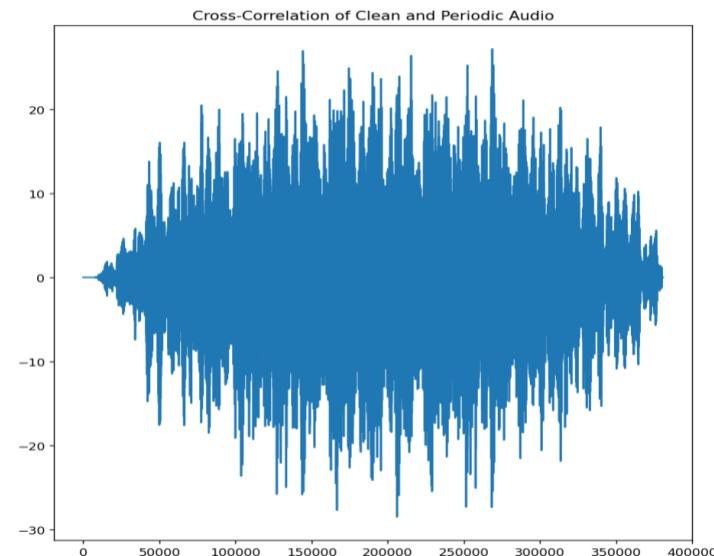
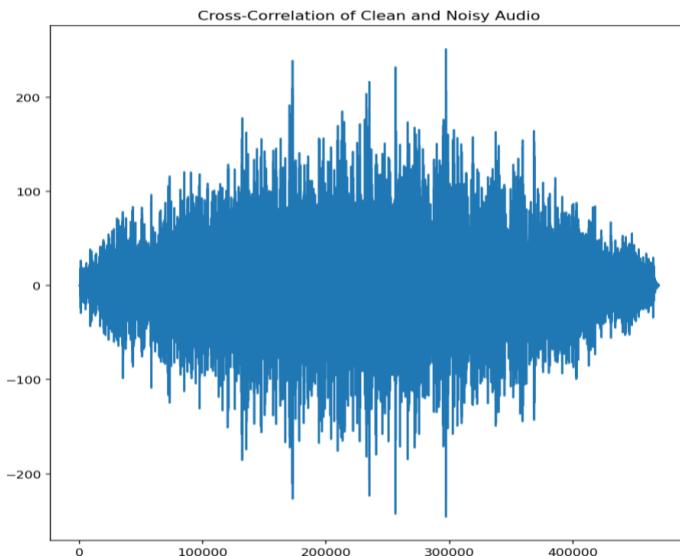
**Subject: Programming With Python (01CT1309)**

**Aim:** Exploring Convolution and Correlation in Discrete-Time Signals

**Experiment No: 18**

**Date:**

**Enrollment No: 92510133011**



Github link:

<https://github.com/trupalijasani05/trupali-jasani>