CS339 IoT Assignment 1

Name: - Priyansh Vaishnav **Student ID: - 202151120**

Link for Colaboratory File : - [202151120_Assignement1_IoT.ipynb]

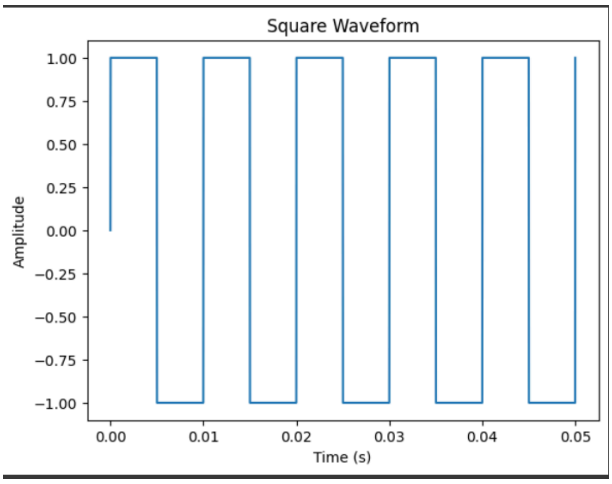
Solution 1: -

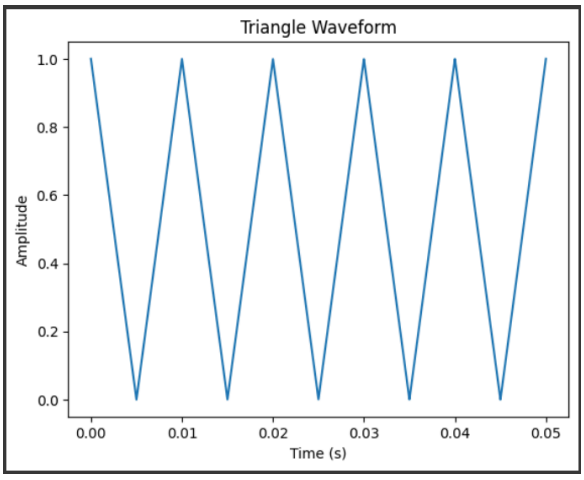
```
import numpy as np
import matplotlib.pyplot as plt

def sqWave(T, n, amp):
    t = np.linspace(0, T * n, int(1000 * n))
    sq = amp * np.sign(np.sin(2 * np.pi / T * t))
    return t, sq

def triWave(T, n, amp):
    t = np.linspace(0, T * n, int(1000 * n))
    tri = amp * (2 / T) * (t % T) - amp
    tri = np.where(tri < 0, -tri, tri)
    return t, tri

def sawWave(T, n, amp):
    t = np.linspace(0, T * n, int(1000 * n))
    saw = amp * (2 / T * (t % T) - 1)
    return t, saw

T = 0.01  # Time period
n = 5  # Number of n
amp = 1.0  # Amplitude


# Generate Square Waveform
t_square, sq = sqWave(T, n, amp)
plt.figure()
```

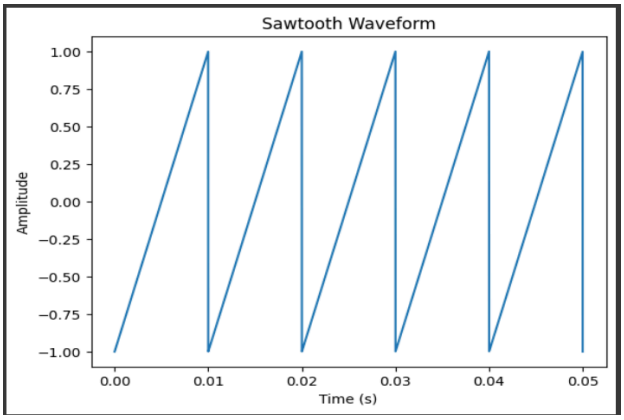
```
plt.plot(t_square, sq)
plt.title('Square Waveform')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()

# Generate Triangle Waveform
t_triangle, tri = triWave(T, n, amp)
plt.figure()
plt.plot(t_triangle, tri)
plt.title('Triangle Waveform')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()

# Generate Sawtooth Waveform
t_sawtooth, saw = sawWave(T, n, amp)
plt.figure()
plt.plot(t_sawtooth, saw)
plt.title('Sawtooth Waveform')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.show()
#Fourier Function
def fourier(t, waveform, n):
    result = np.zeros_like(t)
    for i in range(1, n + 1):
        k = 2 * i - 1
        term = (4 / (np.pi * k)) * np.sin(2 * np.pi * k * t / T)
        result += term
    return waveform - result
#Square Waveform using Fourier Series
for i in range(1, 16, 5):
    sq2 = fourier(t_square, sq, i)
    plt.plot(t_square, sq2, label=f'Reconstructed (n={i})')

plt.title('Square Waveform and Fourier Series Reconstruction')
plt.legend()
plt.show()
#Triangular Waveform using Fourier Series
```

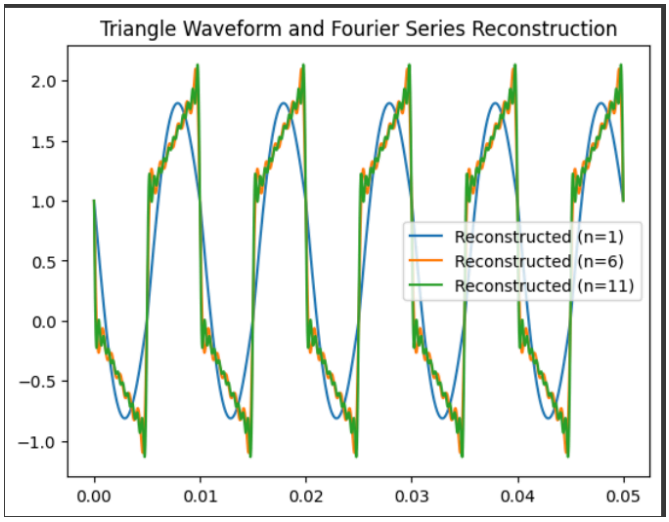
```
for i in range(1, 16, 5):
    tri2 = fourier(t_triangle, tri, i)
    plt.plot(t_triangle, tri2, label=f'Reconstructed (n={i})')

plt.title('Triangle Waveform and Fourier Series Reconstruction')
plt.legend()
plt.show()
##Sawtooth Waveform using Fourier Series
for i in range(1, 16, 5):
    saw2 = fourier(t_sawtooth, saw, i)
    plt.plot(t_sawtooth, saw2, label=f'Reconstructed (n={i})')

plt.title('Sawtooth Waveform and Fourier Series Reconstruction')
plt.legend()
plt.show()
```

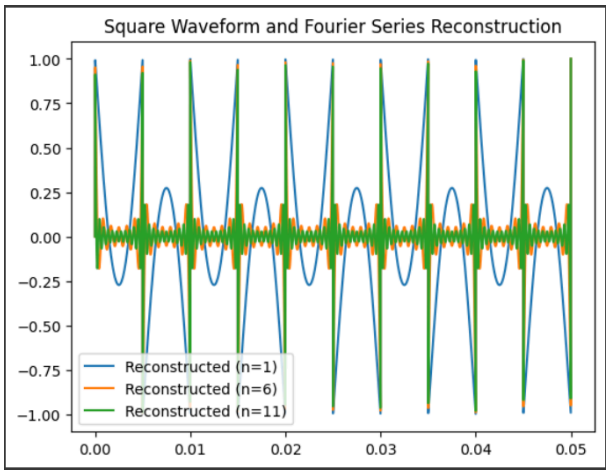
Plots:-

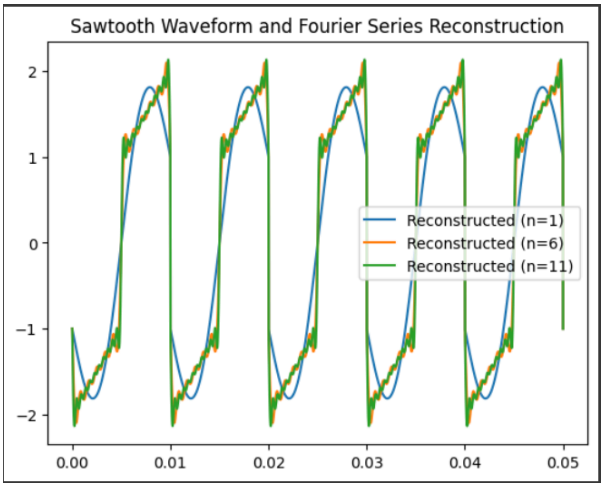

Square Waveform
Amplitude
Time (s)


Triangle Waveform
Amplitude
Time (s)


Sawtooth Waveform
Amplitude
Time (s)


Triangle Waveform and Fourier Series Reconstruction
Reconstructed (n=1)
Reconstructed (n=6)
Reconstructed (n=11)


Square Waveform and Fourier Series Reconstruction
Reconstructed (n=1)
Reconstructed (n=6)
Reconstructed (n=11)


Sawtooth Waveform and Fourier Series Reconstruction
Reconstructed (n=1)
Reconstructed (n=6)
Reconstructed (n=11)

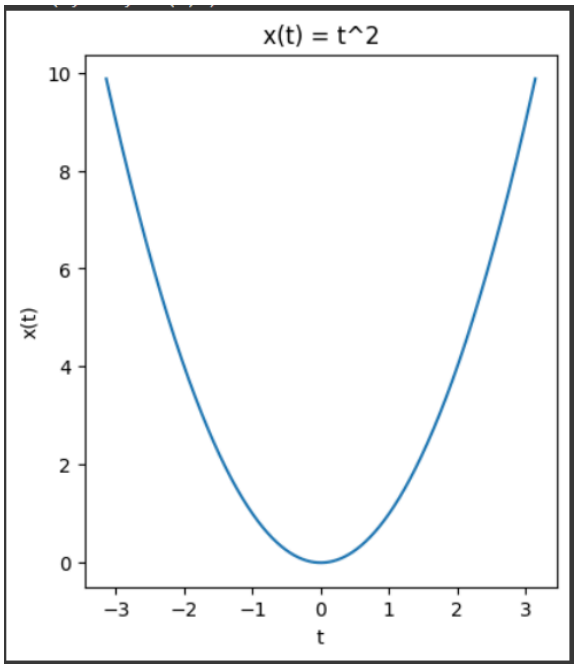
Solution 2: -

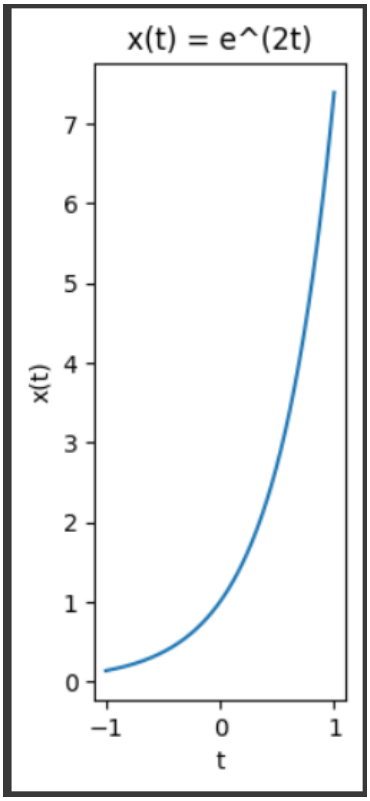
```
def a(t):
    return t**2

def b(t):
    return np.exp(2 * t)

def c(t):
    x = np.zeros_like(t)
    x[t < 0] = 1 + 2 * t[t < 0] / np.pi
    x[t >= 0] = 1 - 2 * t[t >= 0] / np.pi
    return x

ta = np.linspace(-np.pi, np.pi, 1000)
tb = np.linspace(-1, 1, 1000)
tc_neg = np.linspace(-np.pi, 0, 500)
tc_pos = np.linspace(0, np.pi, 500)


xa = a(ta)
xb = b(tb)
xc = c(tc_neg)
xc = np.concatenate([xc, c(tc_pos)])

# Plotting
plt.figure(figsize=(15, 5))
# For x(t) = t^2
plt.subplot(1, 3, 1)
plt.plot(ta, xa)
plt.title('x(t) = t^2')
plt.xlabel('t')
plt.ylabel('x(t)')

# For x(t) = e^(2t)
plt.subplot(1, 3, 2)
plt.plot(tb, xb)
plt.title('x(t) = e^(2t)')
plt.xlabel('t')
```

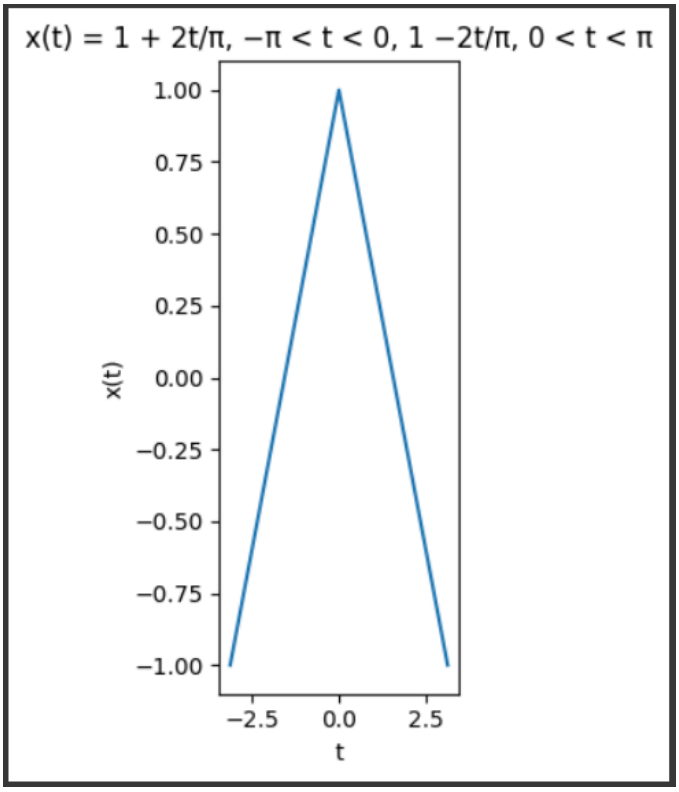
```
plt.ylabel('x(t)')

# For x(t) = 1 + 2t/π, -π < t < 0     = 1 -2t/π, 0 < t < π
plt.subplot(1, 3, 3)
plt.plot(np.concatenate([tc_neg, tc_pos]), xc)
plt.title('x(t) = 1 + 2t/π, -π < t < 0, 1 -2t/π, 0 < t < π')
plt.xlabel('t')
plt.ylabel('x(t)')

plt.tight_layout()
plt.show()

def fourier2(t, waveform, n):
    result = np.zeros_like(t)
    for i in range(1, n + 1):
        k = 2 * i - 1
        term = (4 / (np.pi * k)**2) * np.sin((np.pi * k * t) / np.max(t))
        result += term
    return waveform - result


# Fourier series reconstruction
values = [5, 10, 15]
plt.figure(figsize=(15, 15))

# Fourier series reconstruction for x(t) = t^2
plt.subplot(3, 3, 1)
plt.plot(ta, xa, label='Original')
for n in values:
    reconstructed_a = fourier2(ta, xa, n)
    plt.plot(ta, reconstructed_a, label=f'Reconstructed (n={n})')
plt.title('x(t) = t^2')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.legend()

# Fourier series reconstruction for x(t) = e^(2t)
plt.subplot(6, 6, 2)
plt.plot(tb, xb, label='Original')
for n in values:
```

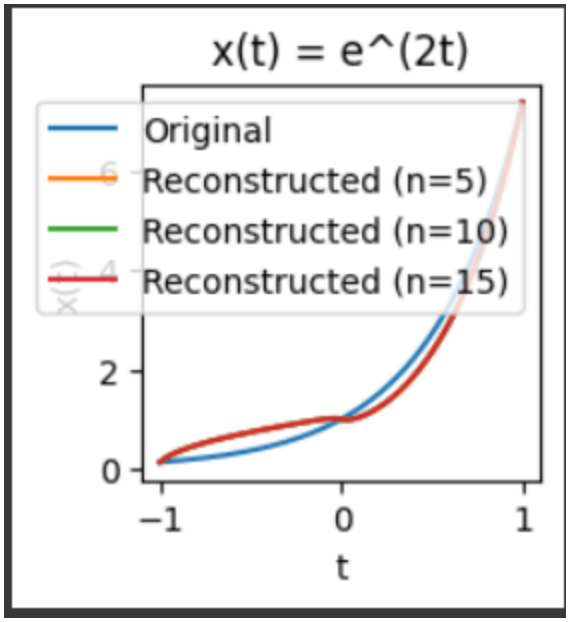
```
    reconstructed_b = fourier2(tb, xb, n)
    plt.plot(tb, reconstructed_b, label=f'Reconstructed (n={n})')
plt.title('x(t) = e^(2t)')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.legend()


 #Fourier series reconstruction for x(t) = 1 + 2t/π, -π < t < 0     = 1
-2t/π, 0 < t < π
plt.subplot(3, 3, 3)
plt.plot(np.concatenate([tc_neg, tc_pos]), xc, label='Original')
for n in values:
    reconstructed_c = fourier2(np.concatenate([tc_neg, tc_pos]), xc, n)
    plt.plot(np.concatenate([tc_neg, tc_pos]), reconstructed_c,
label=f'Reconstructed (n={n})')
plt.title('x(t) = 1 + 2t/π, -π < t < 0, 1 -2t/π, 0 < t < π')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.legend()

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

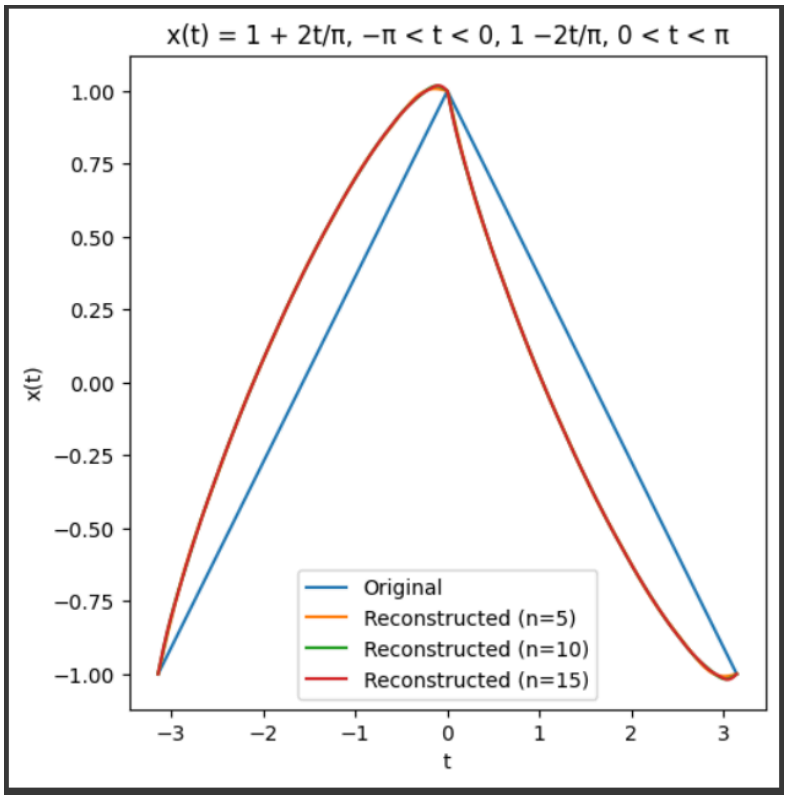
Plots: -

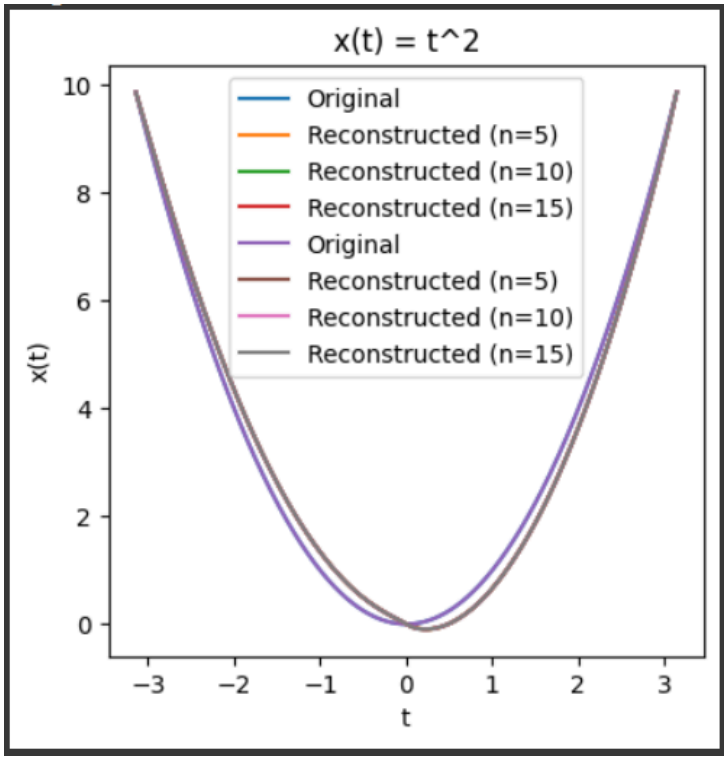

x(t) = t^2
10
8
6
4
2
0
x(t)
−3
−2
−1
0
1
2
3
t


x(t) = e^(2t)
7
6
5
4
3
2
1
0
x(t)
−1
0
1
t


x(t) = 1 + 2t/π, −π < t < 0, 1 −2t/π, 0 < t < π
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
x(t)
−2.5
0.0
2.5
t


x(t) = e^(2t)
Original
Reconstructed (n=5)
Reconstructed (n=10)
Reconstructed (n=15)
x(t)
t


x(t) = 1 + 2t/π, −π < t < 0, 1 −2t/π, 0 < t < π
Original
Reconstructed (n=5)
Reconstructed (n=10)
Reconstructed (n=15)
x(t)
t


x(t) = t^2
Original
Reconstructed (n=5)
Reconstructed (n=10)
Reconstructed (n=15)
Original
Reconstructed (n=5)
Reconstructed (n=10)
Reconstructed (n=15)
x(t)
t