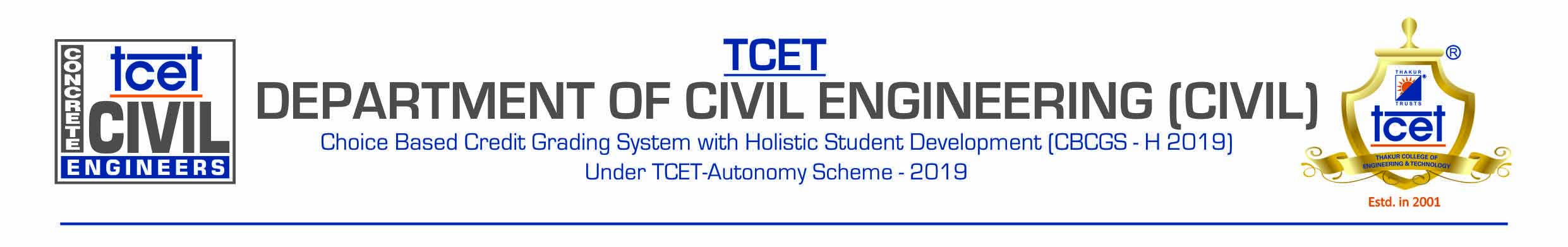
****

**Topic :-** **IoT-based Smart Street Light System**

**Concepts Used: Conditional Statements, Loops, OOP**

**Detect daylight levels and turn lights on/off automatically.**

**Optimize electricity usage based on real-time data.**

**1. Introduction**

With increasing focus on smart cities and energy conservation, IoT-based solutions offer efficient ways to automate public infrastructure. Street lights consume a major portion of city electricity. Many times, they stay ON during daylight or when not needed, leading to energy wastage.

This project implements a **smart street light control system** using Python. It simulates a sensor-based street light system that turns lights ON/OFF based on real-time ambient light levels, optimizing electricity usage and saving power.

**2. Project Description**

The system uses a simulated **light sensor** to measure ambient light levels. Based on a defined **threshold**, the program turns multiple **street lights** ON or OFF using conditional logic. The system also tracks electricity usage per cycle and generates a graph to visualize the energy consumed over time.

The solution is written in **Python** using Object-Oriented Programming (OOP) principles. It runs in cycles and logs the electricity units used during each cycle, helping analyze and improve energy efficiency.

**3. Code**

IoT-based Smart Street Light System

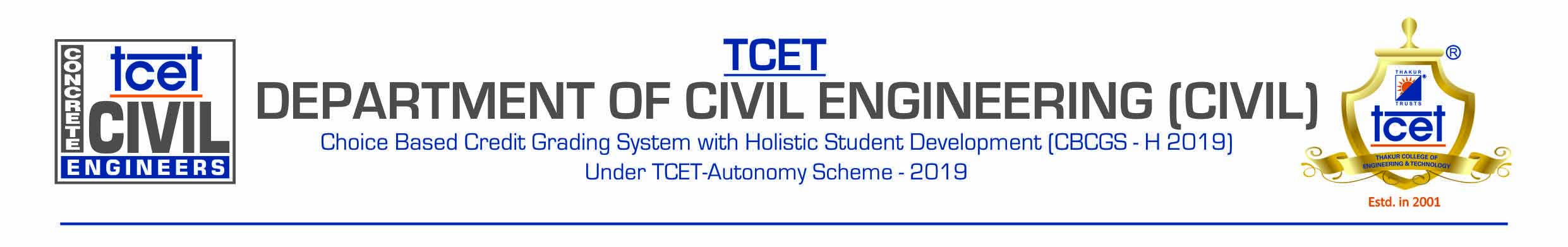
#This project simulates an IoT-based smart street light system that:

#- Uses light sensors to detect ambient light

#- Automatically turns street lights ON/OFF

#- Tracks electricity usage and optimizes power consumption

#- Generates a usage graph for analysis



# Configuration

LIGHT\_THRESHOLD = 50 # Light level below which lights turn on

class StreetLight:

def \_\_init\_\_(self, id):

self.id = id

self.state = False

def turn\_on(self):

if not self.state:

self.state = True

print(f"Street Light {self.id} is ON")

def turn\_off(self):

if self.state:

self.state = False

print(f"Street Light {self.id} is OFF")

import random

class LightSensor:

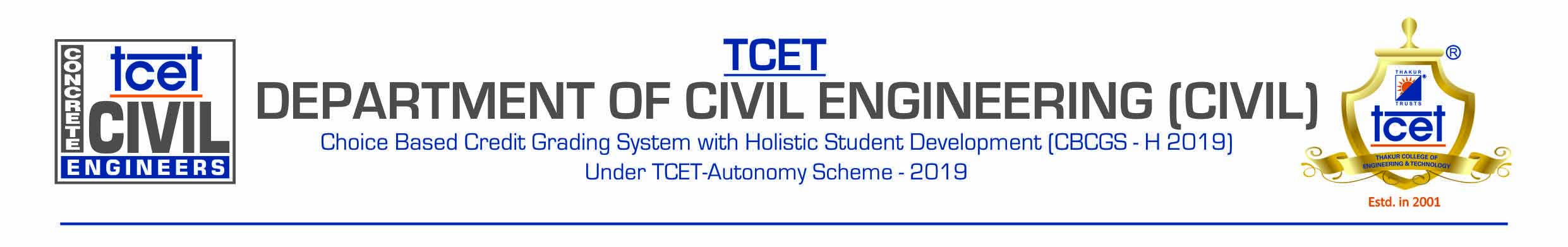
def get\_light\_level(self):

return random.randint(0, 100)

import matplotlib.pyplot as plt

energy\_log = []

def log\_energy\_usage(current, total):



energy\_log.append(current)

print(f"Cycle energy used: {current}, Total so far: {total}")

def generate\_energy\_report():

plt.plot(range(1, len(energy\_log)+1), energy\_log, marker='o', color='green')

plt.title("Electricity Usage per Cycle")

plt.xlabel("Cycle")

plt.ylabel("Units of Electricity")

plt.grid(True)

plt.show()

import time

class StreetLightController:

def \_\_init\_\_(self, num\_lights=5):

self.lights = [StreetLight(i+1) for i in range(num\_lights)]

self.sensor = LightSensor()

self.total\_energy\_used = 0

def update\_lights(self):

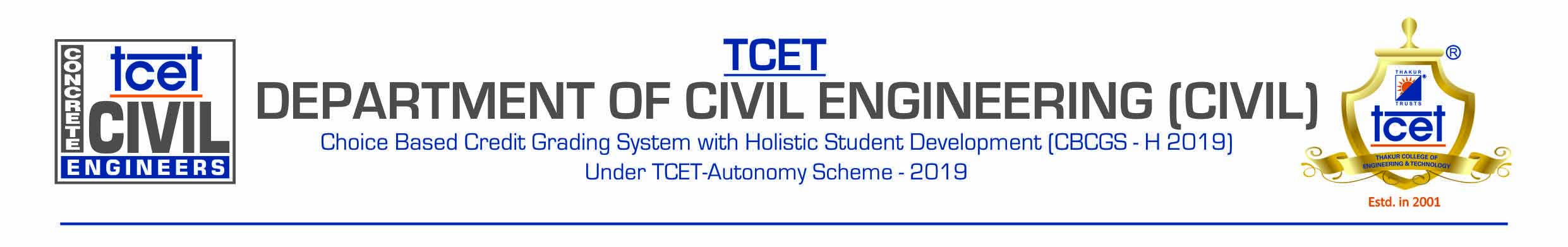
light\_level = self.sensor.get\_light\_level()

print(f"\nCurrent ambient light level: {light\_level}")

energy\_this\_cycle = 0

if light\_level < LIGHT\_THRESHOLD:

for light in self.lights:



light.turn\_on()

energy\_this\_cycle += 1

else:

for light in self.lights:

light.turn\_off()

self.total\_energy\_used += energy\_this\_cycle

log\_energy\_usage(energy\_this\_cycle, self.total\_energy\_used)

def print\_summary(self):

print(f"\nTotal electricity units used during simulation: {self.total\_energy\_used}")

controller = StreetLightController()

try:

for \_ in range(10): # simulate 10 cycles

controller.update\_lights()

time.sleep(1)

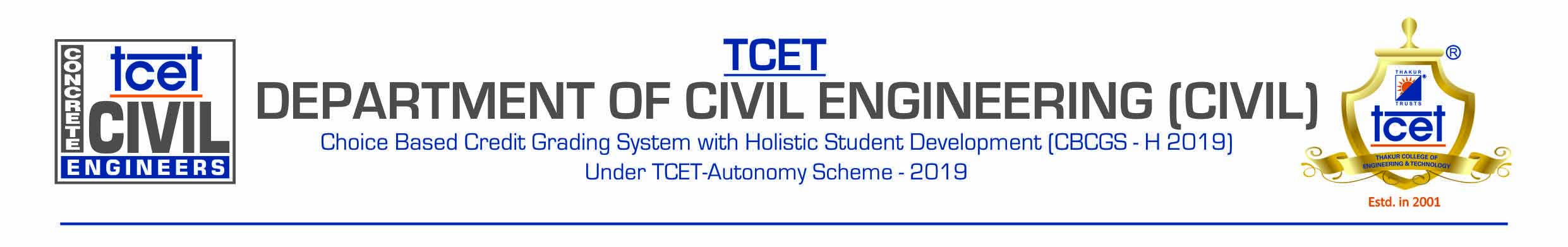
except KeyboardInterrupt:

print("Simulation interrupted.")

controller.print\_summary()

**4. Code Explanation**

a. Conditional Statements



Used to check if the light level is below the threshold:

Python:

if light\_level < LIGHT\_THRESHOLD:

# Turn lights ON

else:

# Turn lights OFF

b. Loops

Used to simulate multiple street lights and multiple update cycles:

Python:

for light in self.lights:

light.turn\_on()

Python:

for \_ in range(10):

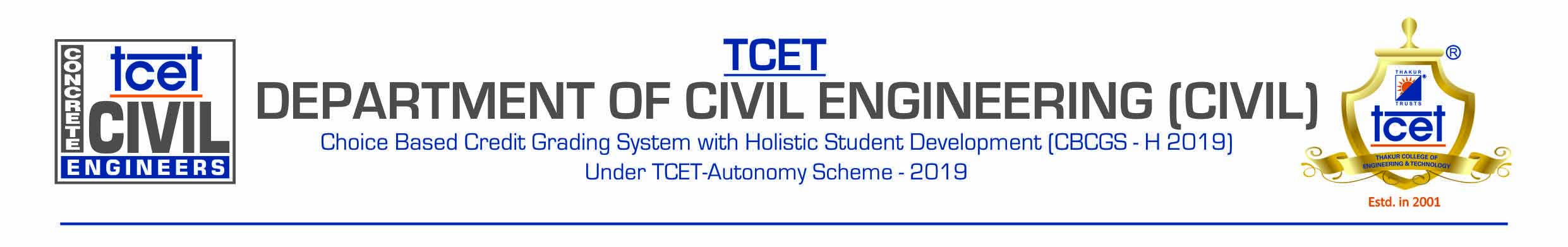
controller.update\_lights()

c. Object-Oriented Programming (OOP)

Classes like StreetLight, LightSensor, and StreetLightController organize the logic into reusable modules.

Objects represent real-world components like individual lights or a sensor.

d. Functions



Functions like log\_energy\_usage() and generate\_energy\_report() are used to perform specific tasks such as logging and graph generation.

e. Random Module

Used to simulate random daylight levels between 0 and 100.

Python:

random.randint(0, 100)

f. Matplotlib Library

Used to create a graph showing energy usage per cycle.

Python:

import matplotlib.pyplot as plt

**5. Output**

Current ambient light level: 78

Cycle energy used: 0, Total so far: 0

Current ambient light level: 3

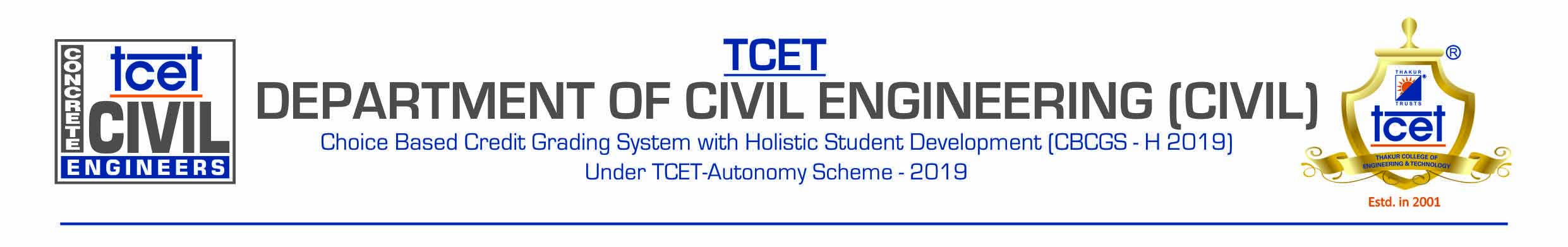
Street Light 1 is ON

Street Light 2 is ON

Street Light 3 is ON

Street Light 4 is ON

Street Light 5 is ON



Cycle energy used: 5, Total so far: 5

Current ambient light level: 39

Cycle energy used: 5, Total so far: 10

Current ambient light level: 96

Street Light 1 is OFF

Street Light 2 is OFF

Street Light 3 is OFF

Street Light 4 is OFF

Street Light 5 is OFF

Cycle energy used: 0, Total so far: 10

Current ambient light level: 49

Street Light 1 is ON

Street Light 2 is ON

Street Light 3 is ON

Street Light 4 is ON

Street Light 5 is ON

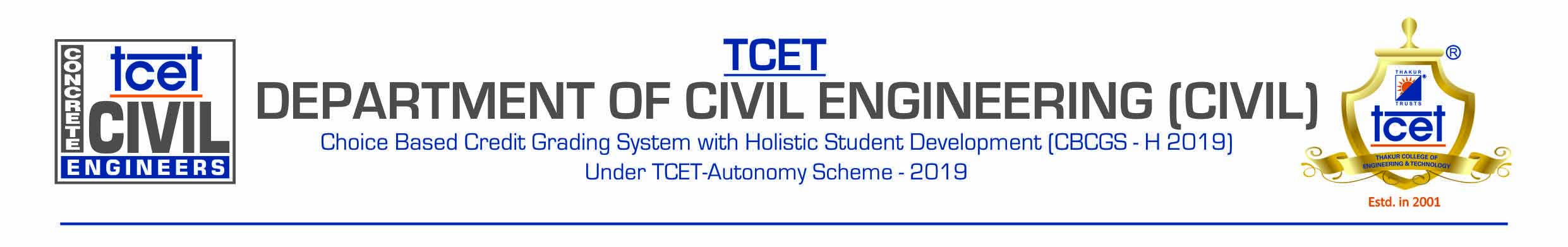
Cycle energy used: 5, Total so far: 15

Current ambient light level: 87

Street Light 1 is OFF

Street Light 2 is OFF

Street Light 3 is OFF



Street Light 4 is OFF

Street Light 5 is OFF

Cycle energy used: 0, Total so far: 15

Current ambient light level: 83

Cycle energy used: 0, Total so far: 15

Current ambient light level: 91

Cycle energy used: 0, Total so far: 15

Current ambient light level: 73

Cycle energy used: 0, Total so far: 15

Current ambient light level: 77

Cycle energy used: 0, Total so far: 15

Total electricity units used during simulation: 15