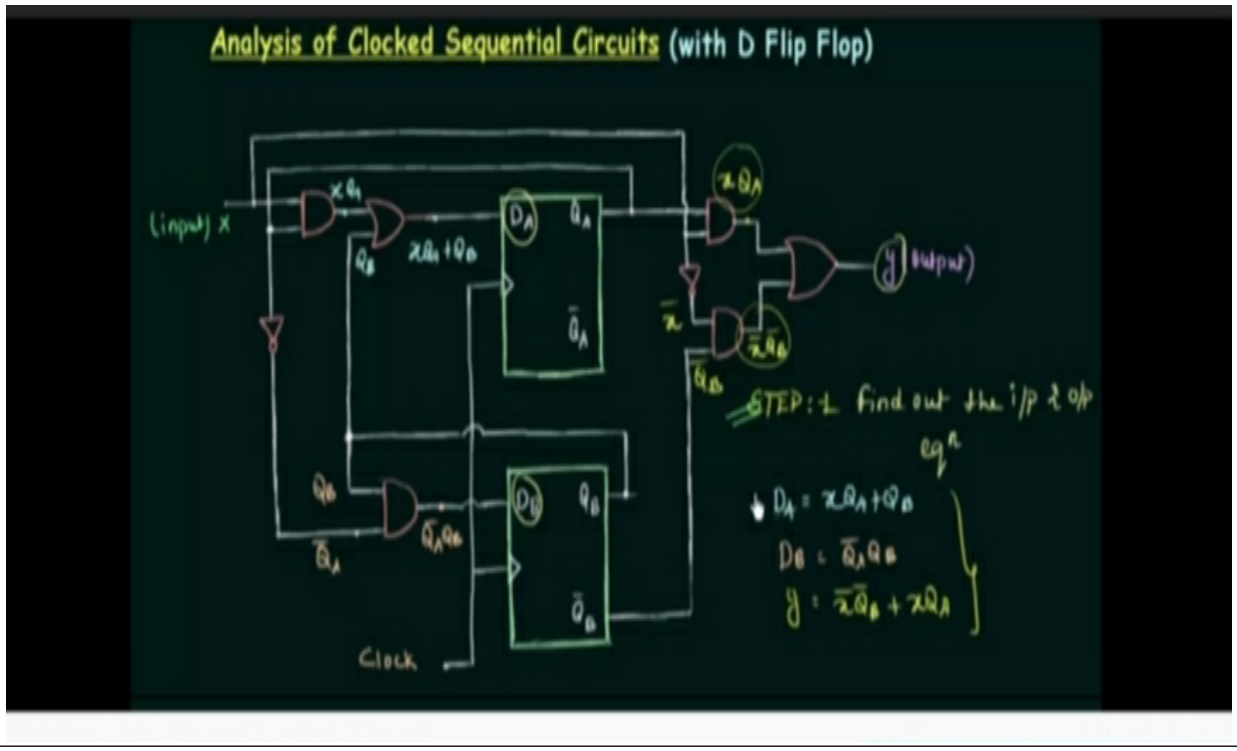
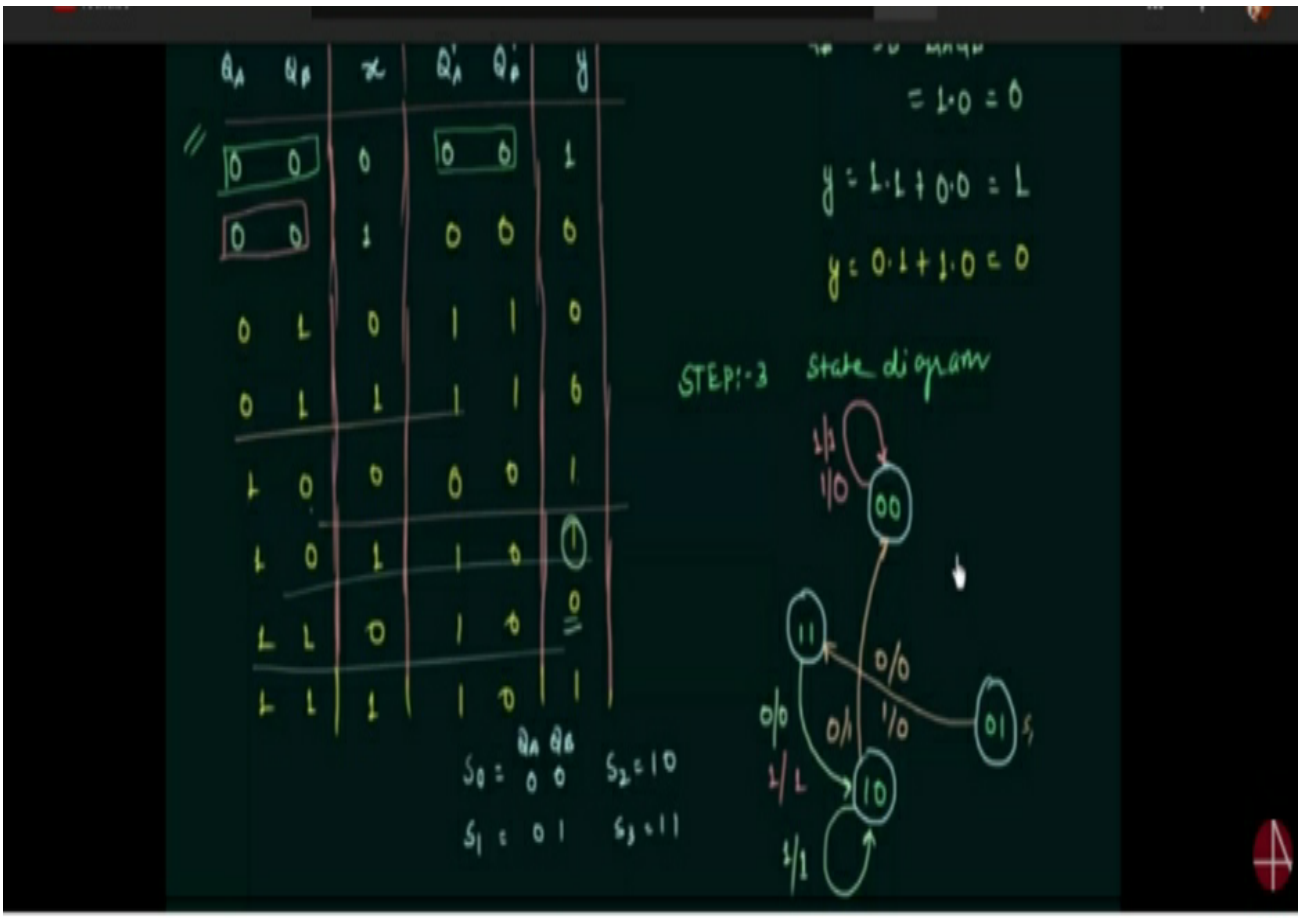
DAILY ASSESSMENT REPORT

Date:	29/05/2020	Name:	Abhishek
Subject:	Logic Design	USN:	4AL17EC001
Topic:	1] Analysis of clocked sequential circuits 2] Digital clock design 3] Webinar	Semester & Section:	6th 'A'
Github Repository:	Abhishek-online-courses		

FORENOON SESSION DETAILS
Image of session




Q_A Q_B x Q_A' Q_B' y
STEP:-3 State diagram
00
01
11
10
$S_0 = 0\,0$ $S_1 = 0\,1$ $S_2 = 1\,0$ $S_3 = 1\,1$

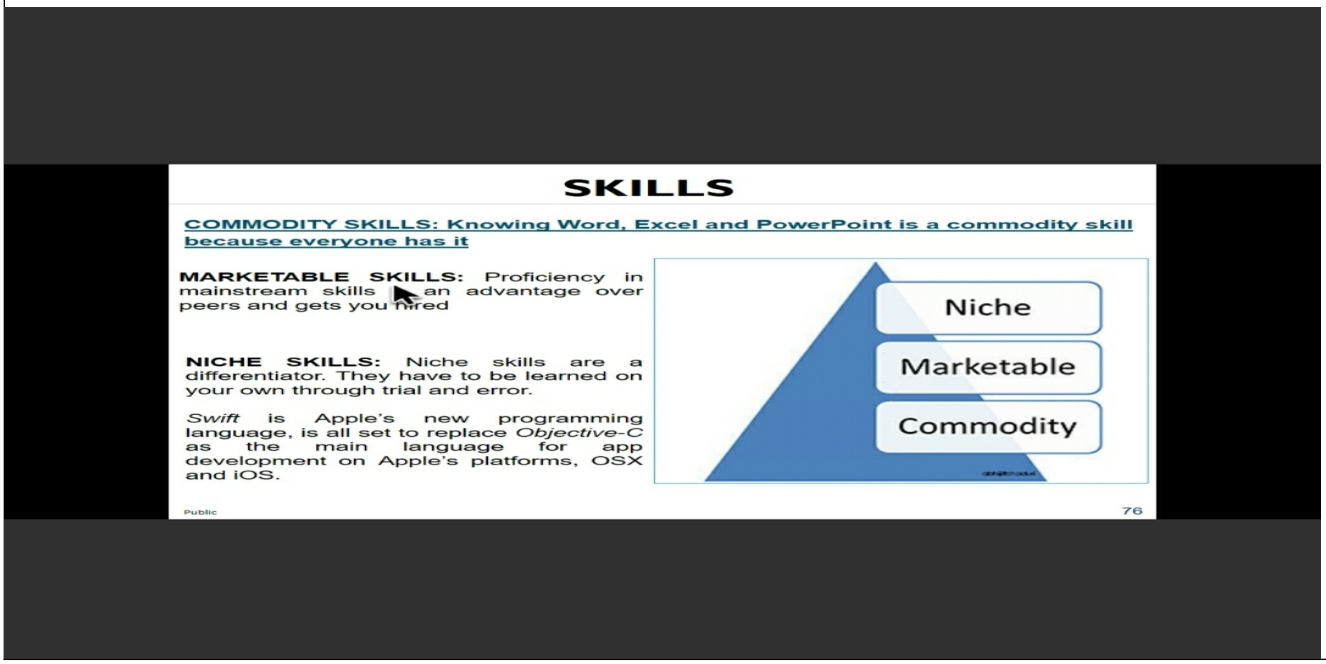
Report

Analysis of clocked sequential circuits :

- The behavior of a clocked sequential circuit is determined from its inputs, outputs and state of the flip-flops (i.e., the output of the flip-flops).
- The analysis of a clocked sequential circuit consists of obtaining a table of a diagram of the time sequences of inputs, outputs and states.
- The basic procedure for analyzing a clocked sequential circuit:
  - ✓ Write down the equations for the outputs and the flip-flop inputs.
  - ✓ Using these equations, derive a state table which describes the next state.
  - ✓ Obtain a state diagram from the state table.
- It is the state table and/or state diagram that specifies the behavior of the circuit.

Webinar :

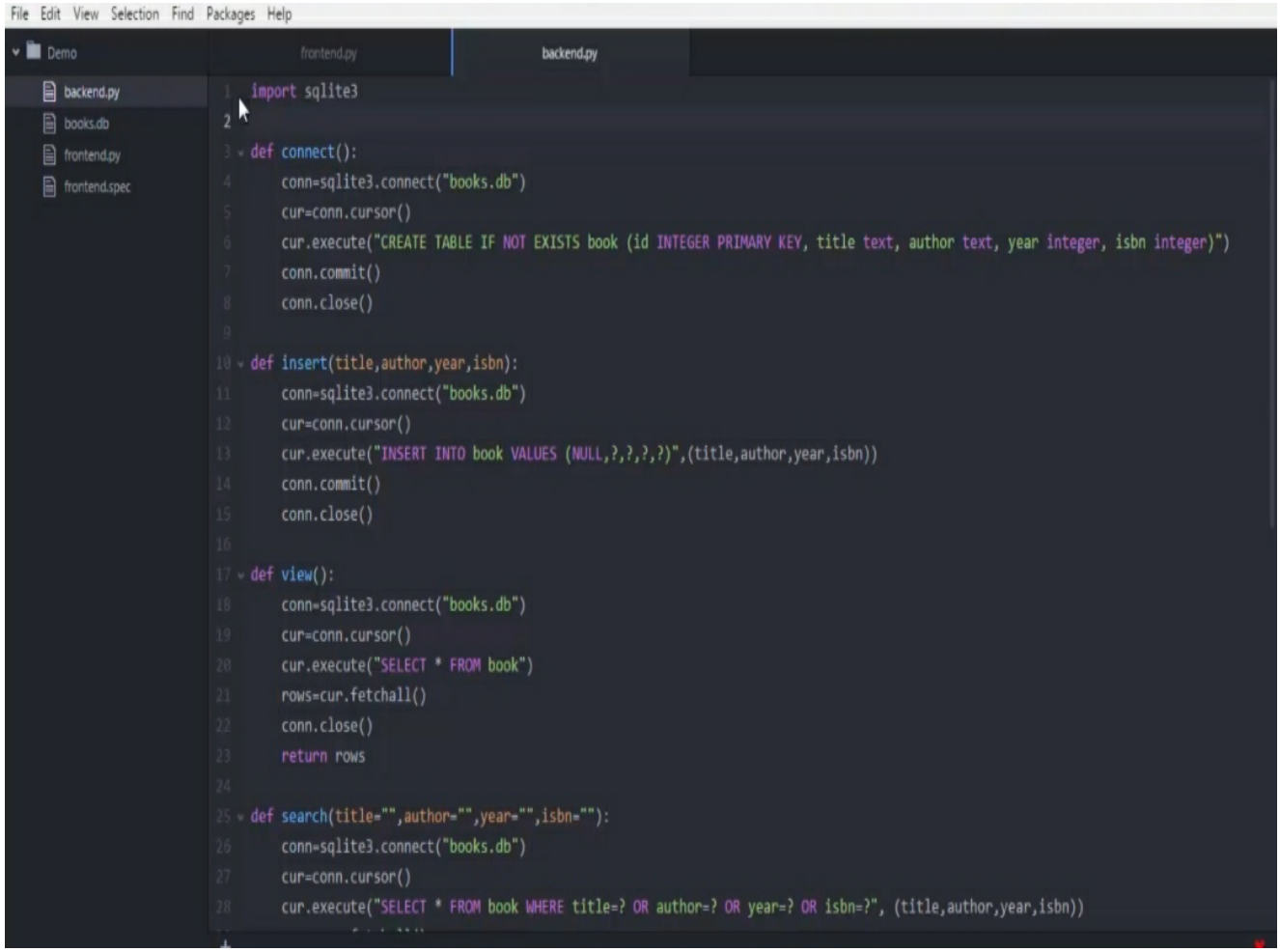
Attended webinar on '**PREPERATION FOR THE NEXT NORMAL**' by **Mr.Mohan Kumar** from wipro



Date:	29/05/2020	Name:	Abhishek
Course:	The Python Mega Course: Build 10 Real World Applications	USN:	4AL17EC001
Topic:	1] Object Oriented Programming	Semester & Section:	6th 'A'
Github Repository:	Abhishek-online-courses		

AFTERNOON SESSION DETAILS
Image of sess

ion


File Edit View Selection Find Packages Help
Demo
backend.py
books.db
frontend.py
frontend.spec
frontend.py
backend.py
import sqlite3
def connect():
conn=sqlite3.connect("books.db")
cur=conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER PRIMARY KEY, title text, author text, year integer, isbn integer)")
conn.commit()
conn.close()
def insert(title,author,year,isbn):
conn=sqlite3.connect("books.db")
cur=conn.cursor()
cur.execute("INSERT INTO book VALUES (NULL,?,?,?,?)",(title,author,year,isbn))
conn.commit()
conn.close()
def view():
conn=sqlite3.connect("books.db")
cur=conn.cursor()
cur.execute("SELECT * FROM book")
rows=cur.fetchall()
conn.close()
return rows
def search(title="",author="",year="",isbn=""):
conn=sqlite3.connect("books.db")
cur=conn.cursor()
cur.execute("SELECT * FROM book WHERE title=? OR author=? OR year=? OR isbn=?", (title,author,year,isbn))

Report

Object Oriented Programming

- In Python, the concept of OOP follows some basic principles:
    - ✓ Inheritance - A process of using details from a new class without modifying existing class.
    - ✓ Encapsulation - Hiding the private details of a class from other objects.
    - ✓ Polymorphism - A concept of using common operation in different ways for different data input.
- Some terminologies in OOP:
    - ✓ Class - A class is a blueprint for the object which contains all the details about the object.
    - ✓ Object - An object (instance) is an instantiation of a class. When class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.
    - ✓ Methods - Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.
    - ✓ Inheritance - Inheritance is a way of creating new class using the details of existing class without modifying it and extra functions can also be added to the derived class.
    - ✓ Data member - A class variable or instance variable that holds data associated with a class and its objects.
    - ✓ Function overloading - The assignment of more than one behavior to a particular function. The operation performed varies by the types of objects or arguments involved.

- ✓ Instantiation - The creation of an instance of a class.
- ✓ Operator overloading - The assignment of more than one function to a particular operator.

- Some of the built-in class attributes:
  - ✓ **"__init__"** is a reserved method in python classes. It is called as a constructor in object-oriented terminology. This method is called when an object is created from a class and it allows the class to initialize the attributes of the class.
  - ✓ The **__del__()** method is a known as a destructor method in Python. It is called when all references to the object have been deleted i.e. when an object is garbage collected.
  - ✓ Python objects have an attribute called **__doc__** that provides a documentation of the object.
- Class variables are defined within the class construction.
- Instance variables are owned by instances of the class.
- Client-server systems, Object-oriented database, Real-time system design, etc. are some of the applications of OOP.