DAILY ASSESSMENT

Date:	**29-5-2020**	**Name:**	**Kavyashree m**
Course:	**Logic design**	**USN:**	**4al15ec036**
Topic:	**Analysis of clocked sequential circuits, Digital clock design**	**Semester & Section:**	**8th A**
Github Repository:	**Kavya**		

FORENOON SESSION DETAILS

Image of session

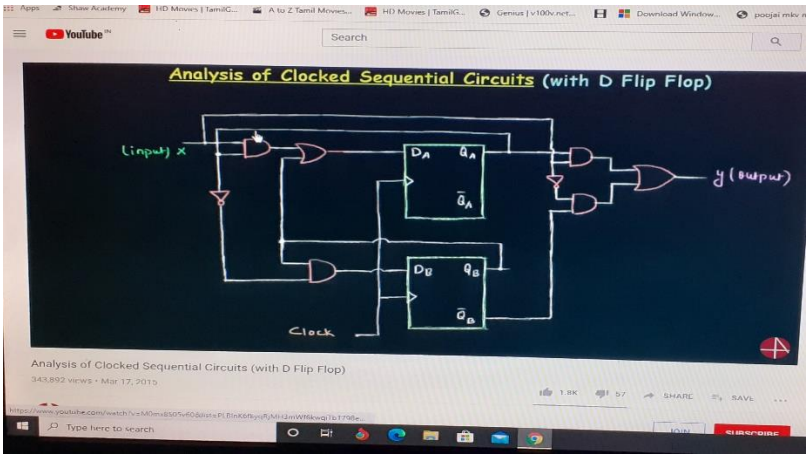


Fig 1 : Analysis of clocked sequential circuits

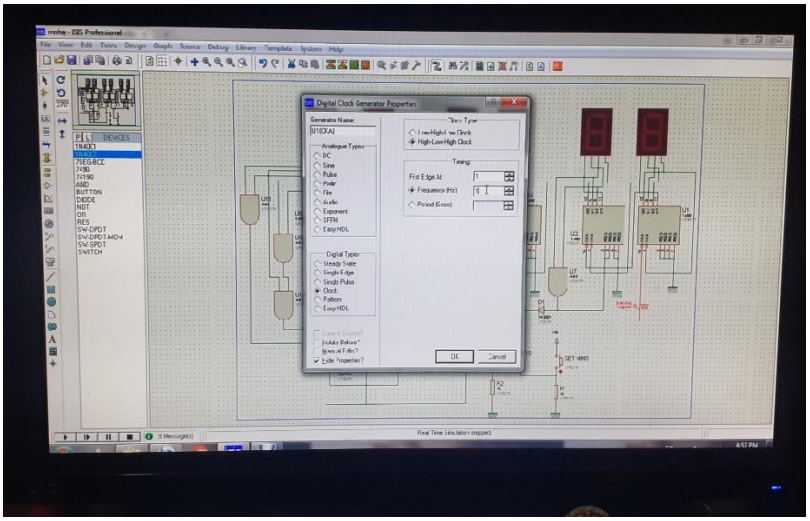


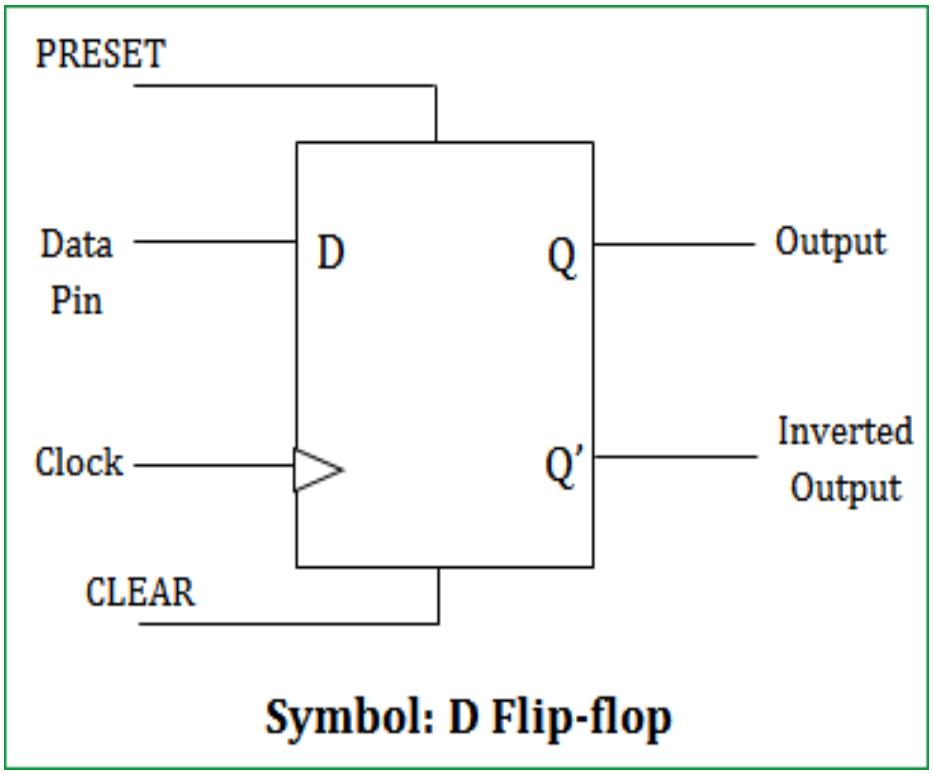
Fig 2 : Digital clock design

Analysis sequential circuits of clocked

D Flip-flop

D Flip-flops are used as a part of memory storage elements and data processors as well. D flip-flop can be built using NAND gate or with NOR gate. Due to its versatility they are available as IC packages. The major applications of D flip-flop are to introduce delay in timing circuit, as a buffer, sampling data at specific intervals. D flip-flop is simpler in terms of wiring connection compared to JK flip-flop. Here we are using NAND gates for demonstrating the D flip flop.

Whenever the clock signal is LOW, the input is never going to affect the output state. The clock has to be high for the inputs to get active. Thus, D flip-flop is a controlled Bi-stable latch where the clock signal is the control signal. Again, this gets divided into positive edge triggered D flip flop and negative edge triggered D flip-flop. Thus, the output has two stable states based on the inputs which have been discussed below.

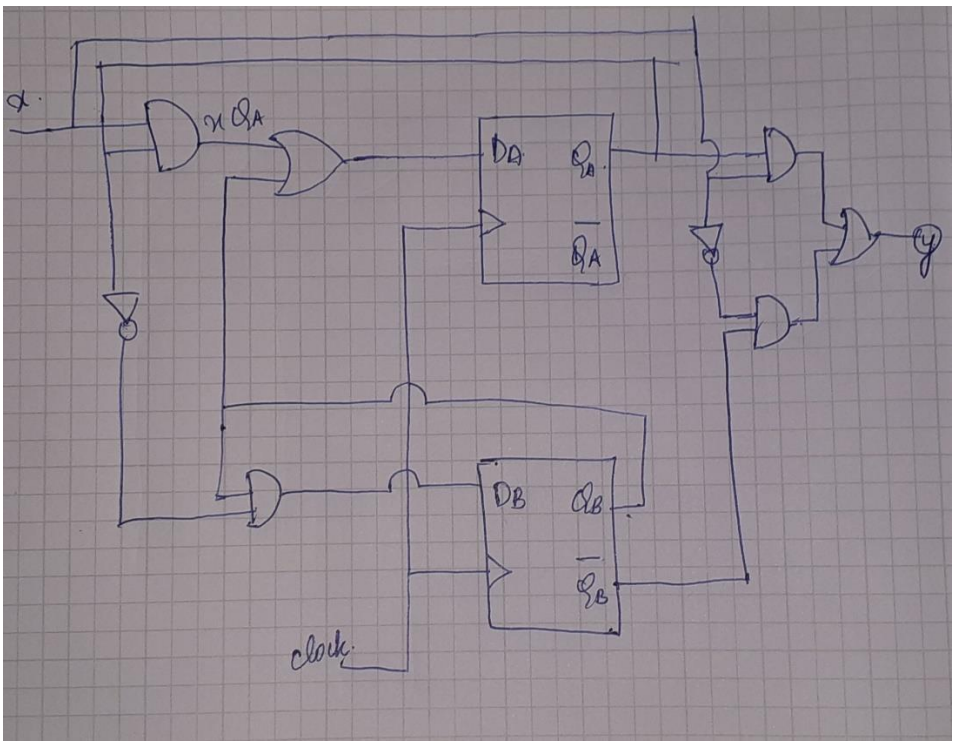


Symbol: D Flip-flop

Truth table of D Flip-Flop

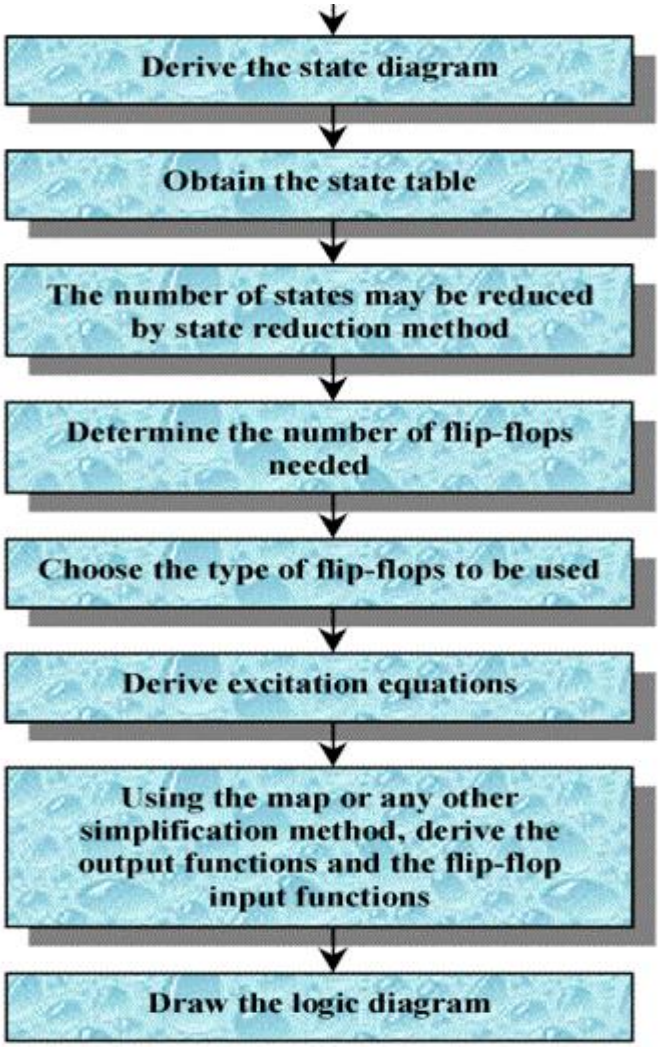
Clock	INPUT	OUTPUT	
	D	Q	Q’
LOW	x	0	1
HIGH	0	0	1
HIGH	1	1	0

Clocked sequential circuits in D-flip flop



clocked sequential circuits in D-flip flop

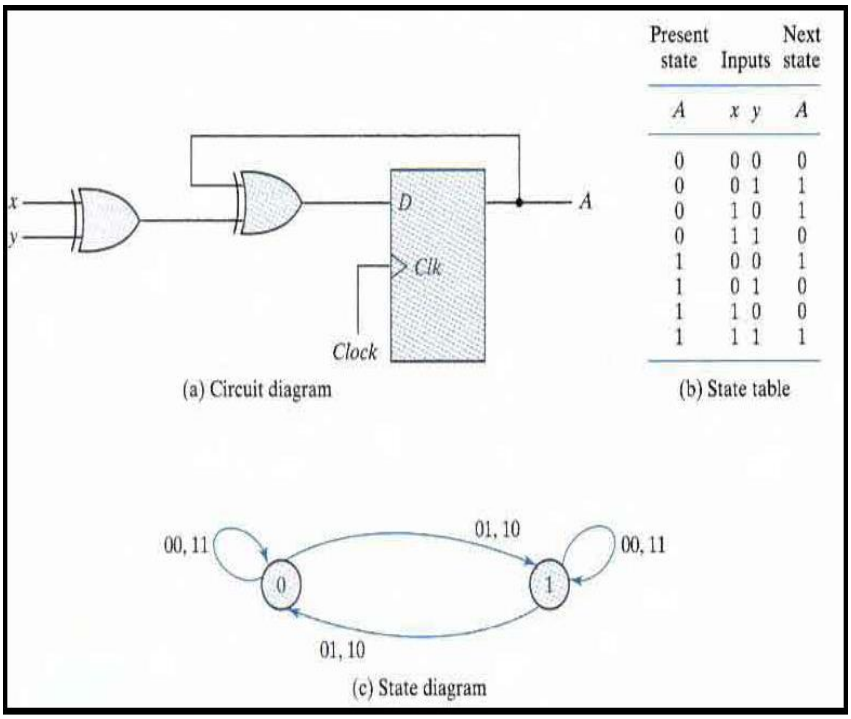
Steps followed in clocked sequential circuits



Analysis with D Flip-Flops

- The input equation of a D Flip-flop is given by $D_A = A \oplus x \oplus y$. D_A means a D Flip-flop with output A.
- The x and y variables are the inputs to the circuit. No output equations are given, which implies that the output comes from the output of the flip-flop.
- The state table has one column for the present state of flip-flop 'A' two columns for the two inputs, and one column for the next state of A.

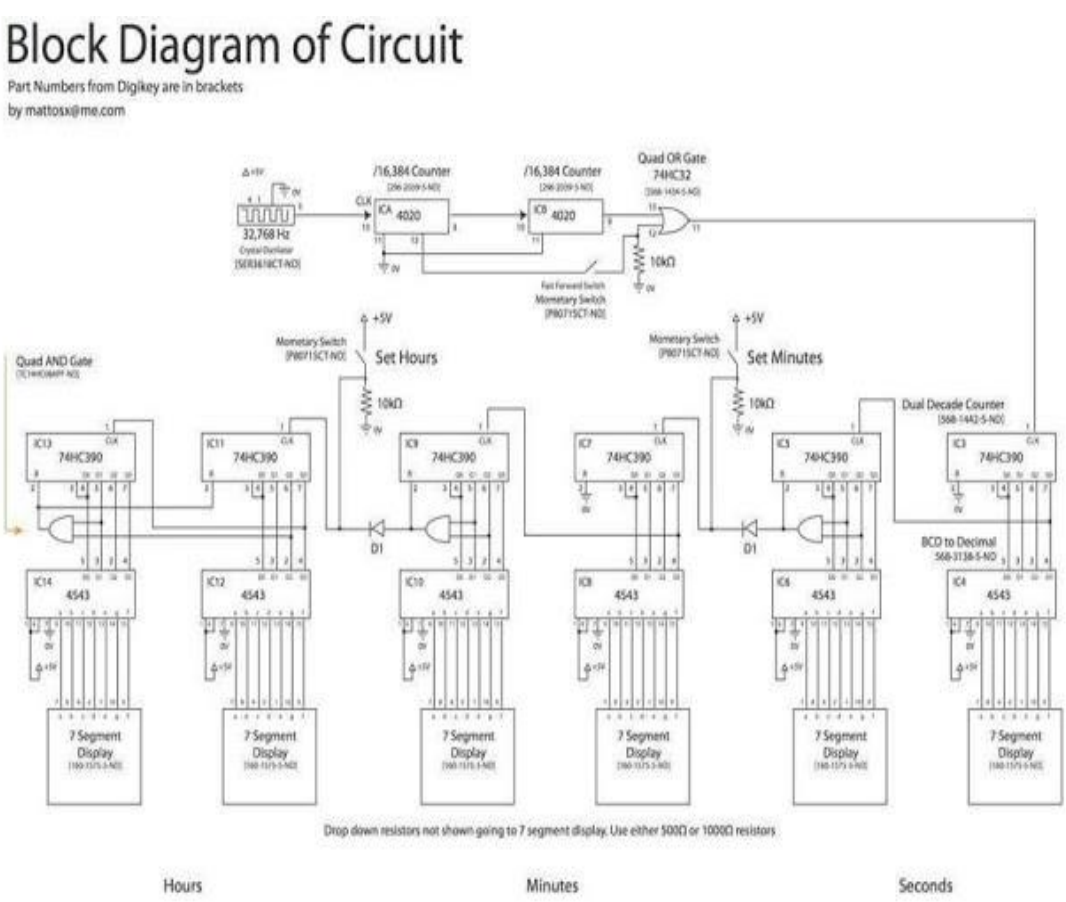
- The next-state values are obtained from the state equation A(t + 1) = A ⊕ x ⊕ y.
- The expression specifies an odd function and is equal to 1 when only one variable is 1 or when all three variables are 1.



Digital clock design

The main parts of the circuit are as follows:

- Timer 555: Responsible for generating the clock pulses for the counters, the frequency of the output shoul be 1 hz which means 1 second for each pulse.
- Counters: Responsible for generating the time in BCD (Binary Coded decimal).
- Decoders : Takes the BCD of the counter as input and produces 7 segment output.
- 7 segments : Displays the time.



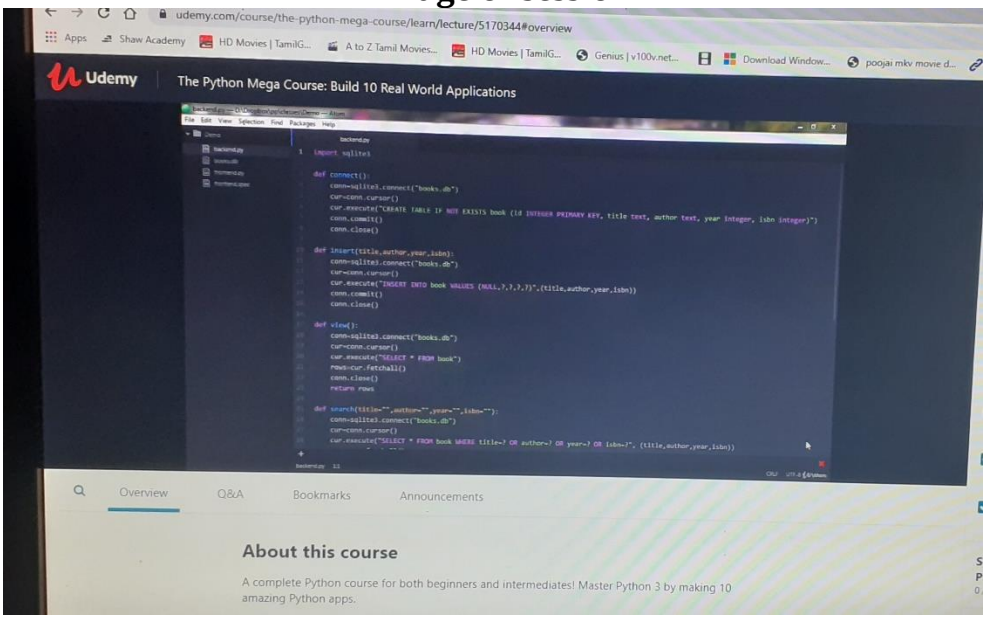
The circuit works as follows :

- 555 timer produces 1 seconde pulses to the clock input of the first counter which is responsible the first column of seconds, so its output will change every second.
- The counter produces numbers from 0 to 9 in BCD form and automatically resets to 0 after that.
- so the output of the first counter will count from 0 to 9 every second and that's exactly what we want from it, so we are done here. let's move to the next one.

AFTERNOON SESSION DETAILS

Date:	**29-5-2020**	**Name:**	**Kavyashreem**
Course:	**Python programming**	**USN:**	**4al15ec036**
Topic:	**Object oriented programming**	**Semester & Section:**	**8th A**
Github Repository:	**Kavya**		

Image of session



Object oriented programming

Object-oriented programming (OOP) refers to a type of computer programming in which programmers define the data type of a data structure, and also the types of operations that can be applied to the data structure.

Class

A class is a blueprint for the object.

We can think of class as an sketch of a parrot with labels. It contains all the details about the name, colors, size etc. Based on these descriptions, we can study about the parrot. Here, parrot is an object.

The example for class of parrot can be :

```
class Parrot:
    pass
```

Here, we use class keyword to define an empty class Parrot. From class, we construct instances. An instance is a specific object created from a particular class.

Object

An object 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.

The example for object of parrot class can be:

```
obj = Parrot()
```

Here, obj is object of class Parrot.

Suppose we have details of parrot. Now, we are going to show how to build the class and objects of parrot.

Methods

Methods are functions defined inside the body of a class. They are used to define the behaviors of an object.

Example : Creating Methods in Python

```
class Parrot:


    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

```
    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)

# instantiate the object
blu = Parrot("Blu", 10)

# call our instance methods
print(blu.sing("'Happy'"))
print(blu.dance())
```

Run Code

When we run program, the output will be:

```
Blu sings 'Happy'
Blu is now dancing
```

In the above program, we define two methods i.e sing() and dance(). These are called instance method because they are called on an instance object i.e blu.

Inheritance

Inheritance is a way of creating new class for using details of existing class without modifying it. The newly formed class is a derived class (or child class). Similarly, the existing class is a base class (or parent class).

Example : Use of Inheritance in Python

```
# parent class
```

```
class Bird:

    def __init__(self):
        print("Bird is ready")

    def whoisThis(self):
        print("Bird")

    def swim(self):
        print("Swim faster")

# child class
class Penguin(Bird):

    def __init__(self):
        # call super() function
        super().__init__()
        print("Penguin is ready")

    def whoisThis(self):
        print("Penguin")

    def run(self):
        print("Run faster")

peggy = Penguin()
peggy.whoisThis()
```

peggy.swim()

peggy.run()

Run Code

When we run this program, the output will be:

Bird is ready

Penguin is ready

Penguin

Swim faster

Run faster

In the above program, we created two classes i.e. Bird (parent class) and Penguin (child class). The child class inherits the functions of parent class. We can see this from swim() method. Again, the child class modified the behavior of parent class. We can see this from whoisThis() method. Furthermore, we extend the functions of parent class, by creating a new run() method.

Additionally, we use super() function before __init__() method. This is because we want to pull the content of __init__() method from the parent class into the child class.

Encapsulation

Using OOP in Python, we can restrict access to methods and variables. This prevent data from direct modification which is called encapsulation. In Python, we denote private attribute using underscore as prefix i.e single " _ " or double " __".

Example 4: Data Encapsulation in Python

```
class Computer:


    def __init__(self):
        self.__maxprice = 900
```

```
    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()
```

Run Code

When we run this program, the output will be:

```
Selling Price: 900
Selling Price: 900
Selling Price: 1000
```

In the above program, we defined a class Computer. We use __init__() method to store the maximum selling price of computer. We tried to modify the price. However, we can't change it because Python treats the __maxprice as private attributes. To change the value, we used a setter function i.e setMaxPrice() which takes price as parameter.

Polymorphism

Polymorphism is an ability to use common interface for multiple form .

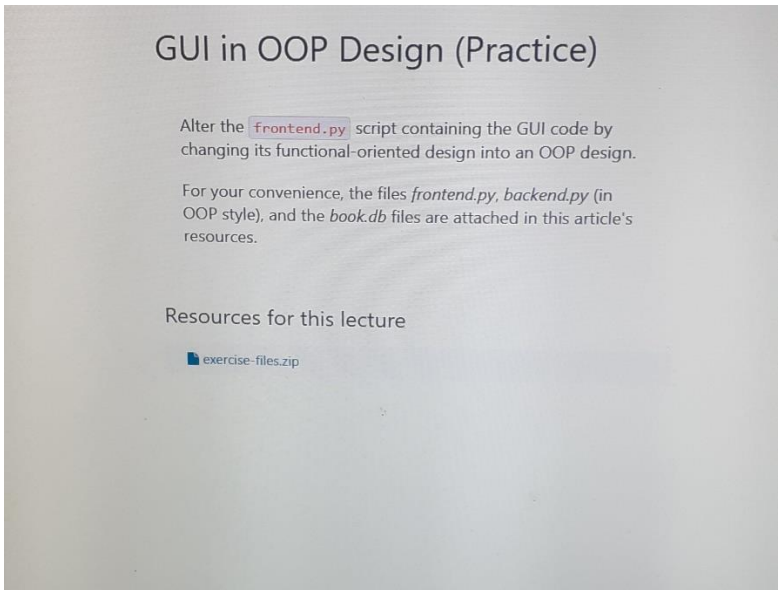
Suppose, we need to color a shape, there are multiple shape option (rectangle, square, circle). However we could use same method to color any shape. This concept is called Polymorphism.

Example 5: Using Polymorphism in Python

```
class Parrot:

    def fly(self):
        print("Parrot can fly")

    def swim(self):
        print("Parrot can't swim")

class Penguin:

    def fly(self):
        print("Penguin can't fly")

    def swim(self):
        print("Penguin can swim")

# common interface
def flying_test(bird):
    bird.fly()

#instantiate objects
blu = Parrot()
```

```
peggy = Penguin()

# passing the object
flying_test(blu)
flying_test(peggy)
```

GUI OOP in practice



Frontend code

```
from tkinter import *
from backend import Database

database=Database("books.db")

def get_selected_row(event):
    global selected_tuple
    index=list1.curselection()[0]
    selected_tuple=list1.get(index)
    e1.delete(0,END)
```

```
    e1.insert(END,selected_tuple[1])
    e2.delete(0,END)
    e2.insert(END,selected_tuple[2])
    e3.delete(0,END)
    e3.insert(END,selected_tuple[3])
    e4.delete(0,END)
    e4.insert(END,selected_tuple[4])

def view_command():
    list1.delete(0,END)
    for row in database.view():
        list1.insert(END,row)

def search_command():
    list1.delete(0,END)
    for row in
database.search(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()):
        list1.insert(END,row)

def add_command():
    database.insert(title_text.get(),author_text.get(),year_text.get(),isbn_text.get())
    list1.delete(0,END)
    list1.insert(END,(title_text.get(),author_text.get(),year_text.get(),isbn_text.get()))

def delete_command():
    database.delete(selected_tuple[0])
```

```
def update_command():

database.update(selected_tuple[0],title_text.get(),author_text.get(),year_text.get(),isbn_
text.get())

window=Tk()

window.wm_title("BookStore")

l1=Label(window,text="Title")
l1.grid(row=0,column=0)

l2=Label(window,text="Author")
l2.grid(row=0,column=2)

l3=Label(window,text="Year")
l3.grid(row=1,column=0)

l4=Label(window,text="ISBN")
l4.grid(row=1,column=2)

title_text=StringVar()
e1=Entry(window,textvariable=title_text)
e1.grid(row=0,column=1)

author_text=StringVar()
e2=Entry(window,textvariable=author_text)
```

```
e2.grid(row=0,column=3)

year_text=StringVar()
e3=Entry(window,textvariable=year_text)
e3.grid(row=1,column=1)

isbn_text=StringVar()
e4=Entry(window,textvariable=isbn_text)
e4.grid(row=1,column=3)

list1=Listbox(window, height=6,width=35)
list1.grid(row=2,column=0,rowspan=6,columnspan=2)

sb1=Scrollbar(window)
sb1.grid(row=2,column=2,rowspan=6)

list1.configure(yscrollcommand=sb1.set)
sb1.configure(command=list1.yview)

list1.bind('<<ListboxSelect>>',get_selected_row)

b1=Button(window,text="View all", width=12,command=view_command)
b1.grid(row=2,column=3)

b2=Button(window,text="Search entry", width=12,command=search_command)
b2.grid(row=3,column=3)
```

```
b3=Button(window,text="Add entry", width=12,command=add_command)
b3.grid(row=4,column=3)

b4=Button(window,text="Update selected", width=12,command=update_command)
b4.grid(row=5,column=3)

b5=Button(window,text="Delete selected", width=12,command=delete_command)
b5.grid(row=6,column=3)

b6=Button(window,text="Close", width=12,command=window.destroy)
b6.grid(row=7,column=3)
window.mainloop()
```

Backend code

```
import sqlite3

class Database:

    def __init__(self, db):
        self.conn=sqlite3.connect(db)
        self.cur=self.conn.cursor()
        self.cur.execute("CREATE TABLE IF NOT EXISTS book (id INTEGER
PRIMARY KEY, title text, author text, year integer, isbn integer)")
        self.conn.commit()

    def insert(self,title,author,year,isbn):
```

```
        self.cur.execute("INSERT INTO book VALUES
(NULL,?,?,?,?)",(title,author,year,isbn))
        self.conn.commit()

    def view(self):
        self.cur.execute("SELECT * FROM book")
        rows=self.cur.fetchall()
        return rows

    def search(self,title="",author="",year="",isbn=""):
        self.cur.execute("SELECT * FROM book WHERE title=? OR author=? OR
year=? OR isbn=?", (title,author,year,isbn))
        rows=self.cur.fetchall()
        return rows

    def delete(self,id):
        self.cur.execute("DELETE FROM book WHERE id=?",(id,))
        self.conn.commit()

    def update(self,id,title,author,year,isbn):
        self.cur.execute("UPDATE book SET title=?, author=?, year=?, isbn=? WHERE
id=?",(title,author,year,isbn,id))
        self.conn.commit()

    def __del__(self):
        self.conn.close()
```

```
#insert("The Sun","John Smith",1918,913123132)
#delete(3)
#update(4,"The moon","John Smooth",1917,99999)
#print(view())
#print(search(author="John Smooth"))
```