==========slip 1
==========

Write a Java Program to implement I/O Decorator for converting uppercase letters to
lower case letters.
Write a Python program to prepare Scatter Plot for Iris Dataset
Create an HTML form that contain the Student Registration details and write a
JavaScript to validate Student first and last name as it should not contain other than
alphabets and age should be between 18 to 50.

LowerCaseInputStream.java

```
import java.io.*;
import java.util.*;
//extend FilterInputStream which is abstract decorator for all InputStreams
class LowerCaseInputStream extends FilterInputStream
{
      public LowerCaseInputStream(InputStream in)
      {
            super(in);
      }
      public int read() throws IOException
      {
            int c=super.read();
            return (c==-1?c:Character.toLowerCase((char)c));
      }
      public int read(byte[] b,int offset,int len) throws IOException
      {
            int result =super.read(b,offset,len);
            for (int i=offset;i<offset+result;i++)
            {
                  b[i]=(byte)Character.toLowerCase((char)b[i]);
            }
            return result;
      }
}
```

InputTest.java

```
class InputTest
{
      public static void main(String[] args) throws IOException
      {
      int c;
      try
      {
            //set up the FileInputStream and decorate it,first with a BufferedInputStream and then our
      //new LowerCaseInputStream filter
```

```
            InputStream in =new LowerCaseInputStream(new
BufferedInputStream(new FileInputStream("a.txt")));
            // Read the charater until end of file and print it
            while((c = in.read()) >= 0)
            {
                  System.out.print((char)c);
            }

            in.close();
      }
      catch(IOException e)
      {
            e.printStackTrace();
      }
      }
}
--------------------------------------------------------------------------
---------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("iris.csv")
print(data)
x=data["SepalLengthCm"]
y=data["SepalWidthCm"]
print(x)
print(y)
plt.scatter(x, y,c="green")
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.legend()
plt.show()
--------------------------------------------------------------------------
-------------------------------------------------
<html lang = 'en'>
<head>


    <script>
        function validateForm()
        {
            var firstname = document.getElementById("firstname").value;
            var lastname = document.getElementById("lastname").value;
            var age = document.getElementById("age").value;

            var nameRegex = /^[a-zA-Z]+$/
            var ageRegex = /^([1-4][8-9]|50)$/;

            if(!nameRegex.test(firstname) || !nameRegex.test(lastname))
            {
                alert("must be only alphabets");
                return false;
            }
```

```
            if(!ageRegex.test(age))
            {
                alert("must be between 18 and 50");
                return false;
            }
            alert("submitted");
        }
    </script>


</head>
<title>
    student registration form
</title>

<body>
    <h1>Student registration</h1>
    <form onsubmit="return validateForm()">
        <label for = "firstname"> First Name</label>
        <input type="text" id="firstname" name="firstname" required>

    <label for = "lastname">Last name</label>
    <input type="text" id="lastname" name="lastname" required>

    <label for="age">Age</label>
    <input type="text" id="age" name="age" required>

    <input type="submit" value="submit">
    </form>


</body>
</html>
```

==========slip 2==========

Write a Java Program to implement Singleton pattern for multithreading

Write a python program to find all null values in a given dataset and remove them.

Create an HTML form that contain the Employee Registration details and write a JavaScript to validate DOB, Joining Date, and Salary.

Singleton.java

```
public class Singleton{
      private static Singleton uniqueInstance;

      private Singleton() {

            System.out.println("Instance has been Created");
      }

      public static Singleton getInstance() {
```

```
            if(uniqueInstance== null) {
                  synchronized (Singleton.class) {

                        if(uniqueInstance == null) {
                              uniqueInstance=new Singleton();
                        }
                  }

            }
            return uniqueInstance;
      }

      public static void main(String[] args) {

            Thread t1= new Thread(new Runnable() {
                  public void run() {
                        Singleton obj=Singleton.getInstance();
                  }
            });

            Thread t2=new Thread(new Runnable() {
                  public void run() {
                        Singleton obj=Singleton.getInstance();
                  }
            });

            t1.start();
            t2.start();
      }
}
-------------------------------------------------------------------------------
-----------------------------------
import pandas as pd
import numpy as np
data=pd.read_csv("ass2.csv")
print(data)
print(data.isnull())
print(data.notnull())

data1=data.dropna(axis=1,how="all")
print(data1)
data["name"]=data["name"].replace(np.nan,"xyz")
data["m1"]=data["m1"].replace(np.nan,99)
data["m2"]=data["m2"].replace(np.nan, data["m2"].mean())
print(data)
-------------------------------------------------------------------------------
-------------------------------------------
<html>

<head>

    <script>
        function validateForm() {
```

```
            var salary = document.getElementById("salary").value;
            var joiningdate =
document.getElementById("joiningdate").value;
            var age = document.getElementById("age").value;
            var name = document.getElementById("name").value;


            var regex = /^[0-9]+$/;
            var nameregex = /^[a-zA-Z]+$/;



            if (salary < 2000) {
                alert("salary must be greater than 2000")
            }

            if (!regex.test(joiningdate)) {
                alert("joiningdate should be only numbers")
            }

            if (age < 18) {
                alert("age should be greater than 18")
            }

            if (!nameregex.test(name)) {
                alert("should contain only letters")
            }

        }
    </script>

</head>
<title>employee registration</title>

<body>
    <form onsubmit="return validateForm()">

        <label>name</label>
        <input type="text" id="name" name="name" required>
        <br>
        <label>salary</label>
        <input type="text" id="salary" name="salary" required>
        <br>
        <label>joiningdate</label>
        <input type="text" id="joiningdate" name="joiningdate" required>
        <br>
        <label>age</label>
        <input type="age" id="age" name="age" required>
        <br>
        <input type="submit" value="submit">
    </form>

</body>
</body>
```

```
</html>


==========slip
3==========
Write a JAVA Program to implement built-in support (java.util.Observable)
Weather
station with members temperature, humidity, pressure and methods
mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(),
getPressure()
Write a python program to make Categorical values in numeric format for a
given
dataset
Create an HTML form for Login and write a JavaScript to validate email ID
using Regular Expression.
--------------------------------------------------------------------------
----------------------------------------------
Observer.java
public interface Observer {
 public void update(float temp, float humidity, float pressure);
}

Subject.java
public interface Subject {
 public void registerObserver(Observer o);
 public void removeObserver(Observer o);
public void notifyObservers();
}

WeatherData.java
public class WeatherData implements Subject {
 private ArrayList<Observer> observers;
 private float temperature;
 private float humidity;
 private float pressure;

 public WeatherData() {
  observers = new ArrayList<>();
 }

 public void registerObserver(Observer o) {
  observers.add(o);
 }

 public void removeObserver(Observer o) {
  int i = observers.indexOf(o);
  if (i >= 0) {
   observers.remove(i);
  }
 }

 public void notifyObservers() {
  for (int i = 0; i < observers.size(); i++) {
```

```
   Observer observer = (Observer)observers.get(i);
   observer.update(temperature, humidity, pressure);
  }
 }

 public void measurementsChanged() {
  notifyObservers();
 }

 public void setMeasurements(float temperature, float humidity, float
pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  this.pressure = pressure;
  measurementsChanged();
 }

 public float getTemperature() {
  return temperature;
 }

 public float getHumidity() {
  return humidity;
 }

 public float getPressure() {
  return pressure;
 }
}

ForecastDisplay.java
public class ForecastDisplay implements Observer, DisplayElement {
private float currentPressure = 29.92f;
private float lastPressure;
private WeatherData weatherData;

 public ForecastDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temp, float humidity, float pressure) {
                lastPressure = currentPressure;
  currentPressure = pressure;

  display();
 }

 public void display() {
  System.out.print("Forecast: ");
  if (currentPressure > lastPressure) {
   System.out.println("Improving weather on the way!");
  } else if (currentPressure == lastPressure) {
   System.out.println("More of the same");
```

```
  } else if (currentPressure < lastPressure) {
   System.out.println("Watch out for cooler, rainy weather");
  }
 }
}
```

HeatIndexDisplay.java

```
public class HeatIndexDisplay implements Observer, DisplayElement {
 float heatIndex = 0.0f;
 private WeatherData weatherData;

 public HeatIndexDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float t, float rh, float pressure) {
  heatIndex = computeHeatIndex(t, rh);
  display();
 }

 private float computeHeatIndex(float t, float rh) {
  float index = (float)((16.923 + (0.185212 * t) + (5.37941 * rh) -
(0.100254 * t * rh)
   + (0.00941695 * (t * t)) + (0.00728898 * (rh * rh))
   + (0.000345372 * (t * t * rh)) - (0.000814971 * (t * rh * rh)) +
   (0.0000102102 * (t * t * rh * rh)) - (0.000038646 * (t * t * t)) +
(0.0000291583 *
   (rh * rh * rh)) + (0.00000142721 * (t * t * t * rh)) +
   (0.000000197483 * (t * rh * rh * rh)) - (0.0000000218429 * (t * t * t
* rh * rh)) +
   0.000000000843296 * (t * t * rh * rh * rh)) -
   (0.0000000000481975 * (t * t * t * rh * rh * rh)));
  return index;
 }

 public void display() {
  System.out.println("Heat index is " + heatIndex);
 }
}
```

StatisticsDisplay.java

```
public class StatisticsDisplay implements Observer, DisplayElement {
 private float maxTemp = 0.0f;
 private float minTemp = 200;
 private float tempSum= 0.0f;
 private int numReadings;
 private WeatherData weatherData;

 public StatisticsDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }
```

```
 public void update(float temp, float humidity, float pressure) {
  tempSum += temp;
  numReadings++;

  if (temp > maxTemp) {
   maxTemp = temp;
  }

  if (temp < minTemp) {
   minTemp = temp;
  }

  display();
 }

 public void display() {
  System.out.println("Avg/Max/Min temperature = " + (tempSum /
numReadings)
   + "/" + maxTemp + "/" + minTemp);
 }
}
```

CurrentConditionsDisplay.java

```
public class CurrentConditionsDisplay implements Observer, DisplayElement
{
 private float temperature;
 private float humidity;
 private Subject weatherData;

 public CurrentConditionsDisplay(Subject weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temperature, float humidity, float pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  display();
 }

 public void display() {
  System.out.println("Current conditions: " + temperature
   + "F degrees and " + humidity + "% humidity");
 }
}
```

WeatherStation.java

```
public class WeatherStation {

 public static void main(String[] args) {
  WeatherData weatherData = new WeatherData();

  CurrentConditionsDisplay currentDisplay =
   new CurrentConditionsDisplay(weatherData);
```

```
  StatisticsDisplay statisticsDisplay = new
StatisticsDisplay(weatherData);
  ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

  weatherData.setMeasurements(80, 65, 30.4f);
  weatherData.setMeasurements(82, 70, 29.2f);
  weatherData.setMeasurements(78, 90, 29.2f);
 }
}
--------------------------------------------------------------------------
----------------------------------------
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

data=pd.read_csv("ass3.csv")
print(data)
x=data.iloc[:,0:1].values
print(x)

le=LabelEncoder()
x1=le.fit_transform(x)
print(x1)

oe=OneHotEncoder()
x2=oe.fit_transform(x).toarray()
print(x2)
--------------------------------------------------------------------------
----------------------------------------------------
<!DOCTYPE html>
<html>
<head>
    <title>Login Form</title>
    <script>
        function validateForm() {
            var x = document.forms["loginForm"]["email"].value;
            var atpos = x.indexOf("@");
            var dotpos = x.lastIndexOf(".");
            if (atpos<1 || dotpos<atpos+2 || dotpos+2>=x.length) {
                alert("Enter a valid e-mail address");
                return false;
            }
            alert("submitted")

        }
    </script>
</head>
<body>
    <form name="loginForm" action="/login" onsubmit="return
validateForm()" method="post">
        <label for="email">Email:</label><br>
        <input type="text" id="email" name="email"><br>
        <label for="pwd">Password:</label><br>
```

```
        <input type="password" id="pwd" name="pwd"><br>
        <input type="submit" value="Submit">
    </form>
</body>
</html>

document.getElementById('loginForm').addEventListener('submit',
function(event) {
    event.preventDefault();

    // Get the email input value
    var email = document.getElementById('email').value;

    // Regular expression for email validation
    var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    // Check if the email matches the regular expression
    if (!emailRegex.test(email)) {
        alert('Please enter a valid email address.');
        return;
    }

    // If the email is valid, you can proceed with form submission or
further actions.
    alert('Login successful!');
});

==========================================================================sli
p 4==============================================
Write a Java Program to implement Factory method for Pizza Store with
createPizza(),
orederPizza(), prepare(), Bake(), cut(), box(). Use this to create
variety of pizza’s
like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.
Write a python program to Implement Simple Linear Regression for
predicting house
price.
Create a Node.js file that will convert the output "Hello World!" into
upper-case
letters.
--------------------------------------------------------------------------
---------------------------------
PizzaStore.java
public abstract class PizzaStore{
abstract Pizza createPizza(String item);
public Pizza orderPizza(String type){
    Pizza pizza=createPizza(type);
System.out.println("---Making a"+pizza.getName()+"---");
pizza.prepare();
pizza.bake();
pizza.cut();
pizza.box();
return pizza;
}
```

```
}

ChicagoPizzaStore.java
public class ChicagoPizzaStore extends PizzaStore {

    Pizza createPizza(String item) {
            if (item.equals("cheese")) {
                    return new ChicagoStyleCheesePizza();
            } else if (item.equals("veggie")) {
                    return new ChicagoStyleVeggiePizza();
            } else if (item.equals("clam")) {
                    return new ChicagoStyleClamPizza();
            } else if (item.equals("pepperoni")) {
                    return new ChicagoStylePepperoniPizza();
            } else return null;
    }
}

NYPizzaStore.java
public class NYPizzaStore extends PizzaStore{
Pizza createPizza(String item){
if(item.equals("cheese")){
return new;
NYStyleCheesePizza();
}else if(item.equals("veggie")){
return new;
NYStyleCheesePizza();
}else if(item.equals("clam")){
return new;
NYStyleCheesePizza();
}else if(item.equals("pepperoni")){
return new;
NYStyleCheesePizza();
}else return null;
}
}


Pizza.java
import java.util.ArrayList;

public abstract class Pizza {
    String name;
    String dough;
    String sauce;
    ArrayList<String> toppings = new ArrayList<String>();

    void prepare() {
        System.out.println("Prepare " + name);
        System.out.println("Tossing dough...");
        System.out.println("Adding sauce...");
        System.out.println("Adding toppings: ");
        for (String topping : toppings) {
            System.out.println("   " + topping);
```

```
        }
    }

    void bake() {
        System.out.println("Bake for 25 minutes at 350");
    }

    void cut() {
        System.out.println("Cut the pizza into diagonal slices");
    }

    void box() {
        System.out.println("Place pizza in official PizzaStore box");
    }

    public String getName() {
        return name;
    }

    public String toString() {
        StringBuffer display = new StringBuffer();
        display.append("---- " + name + " ----\n");
        display.append(dough + "\n");
        display.append(sauce + "\n");
        for (String topping : toppings) {
            display.append(topping + "\n");
        }
        return display.toString();
    }
}
```

NYStyleCheesePizza.java

```
public class NYStyleCheesePizza extends Pizza {

    public NYStyleCheesePizza() {
        name = "NY Style Sauce and Cheese Pizza";
        dough = "Thin Crust Dough";
        sauce = "Marinara Sauce";

        toppings.add("Grated Reggiano Cheese");
    }
}
```

NYStyleClamPizza.java

```
public class NYStyleClamPizza extends Pizza {

    public NYStyleClamPizza() {
        name = "NY Style Clam Pizza";
        dough = "Thin Crust Dough";
        sauce = "Marinara Sauce";

        toppings.add("Grated Reggiano Cheese");
        toppings.add("Fresh Clams from Long Island Sound");
    }
```

```
}

ChicagoStylePepperoniPizza.java
public class ChicagoStylePepperoniPizza extends Pizza {
    public ChicagoStylePepperoniPizza() {
        name = "Chicago Style Pepperoni Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Spinach");
        toppings.add("Eggplant");
        toppings.add("Sliced Pepperoni");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}

ChicagoStyleVeggiePizza.java
public class ChicagoStyleVeggiePizza extends Pizza {
    public ChicagoStyleVeggiePizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Spinach");
        toppings.add("Eggplant");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}

ChicagoStyleClamPizza.java
public class ChicagoStyleClamPizza extends Pizza {
    public ChicagoStyleClamPizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Clams");
        toppings.add("Jalapeons");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
```

```
    }
}

ChicagoStyleCheesePizza.java
public class ChicagoStyleCheesePizza extends Pizza {
    public ChicagoStyleCheesePizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Mayo");
        toppings.add("Cheddar");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}

PizzaTestDrive.java
public class PizzaTestDrive {

    public static void main(String[] args) {
        PizzaStore nyStore = new NYPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        Pizza pizza = nyStore.orderPizza("cheese");
        System.out.println("First order was a " + pizza.getName() +
"\n");

        pizza = nyStore.orderPizza("cheese");
        System.out.println("Second order was a " + pizza.getName() +
"\n");
    }
}
--------------------------------------------------------------------------
---------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data=pd.read_csv("house.csv")
print(data)

x=data[["bedrooms"]]
y=data.price
print(x)
print(y)
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print(xtrain)
print(xtest)
print(ytrain)
print(ytest)

le=LinearRegression()
le.fit(xtrain,ytrain)
print(le.intercept_)
print(le.coef_)
print(le.predict([[3.5]]))

ypred=le.predict(xtest)
me=mean_squared_error(ypred,ytest)
print(me)

plt.scatter(x,y,c="red")
plt.plot(xtest,le.predict(xtest),c="green")
plt.show()
```

```
function uppercase(string)
{
    return string.toUpperCase();
}

var a ="hello world"
var b= uppercase(a);
console.log(b)
```

==========slip 5==========

Write a Java Program to implement Adapter pattern for Enumeration iterator

Write a python program to implement Multiple Linear Regression for given dataset.

Using nodejs create a web page to read two file names from user and append contents of first file into second file.

EnumerationIterator.java

```
import java.util.*;
// Enumeration interface is used to get elements from vector
class EnumerationIterator implements Iterator {
      Enumeration enumeration;

      public EnumerationIterator(Enumeration enumeration) {
            this.enumeration = enumeration;
      }
```

```
	public boolean hasNext() {
		return enumeration.hasMoreElements();
	}

	public Object next() {
		return enumeration.nextElement();
	}

	public void remove() {
		throw new UnsupportedOperationException();
	}
}


class EnumIterator {
	public static void main (String args[]) {
		Vector v = new
Vector(Arrays.asList("JAVA","CPP","SQL","HTML"));
		Iterator iterator = new EnumerationIterator(v.elements());
		while (iterator.hasNext()) {
			System.out.println(iterator.next());
		}
	}
}

IterEnum.java
import java.util.*;
public class IterEnum
{

public static void main(String[] args)
{
    String[] wordArr = {"Book", "Number", "Place", "Lemon", "Apple",
"Tree"};
    Vector<String> wordList = new Vector<>(Arrays.asList (wordArr));
//directly convert array to vector
    System.out.println("\nThe word list: \n"+wordList + "\n");

    Enumeration<String> vectorEnum = wordList.elements();
    //Enumeration iterates through vector show elements one by one
    while(vectorEnum.hasMoreElements())
    { //when vector Enum has more element to get
System.out.println(vectorEnum.nextElement());
        System.out.println(vectorEnum.nextElement());
    }

    LinkedList<String> wordLinkedList = new LinkedList<>();
    wordLinkedList.addAll(wordList); //add elements from vector to linked
list add some additional items
    wordLinkedList.add("Ball");
    wordLinkedList.add("Mango");
    wordLinkedList.remove("Book");
    System.out.println("\nThe word list (LinkedList): \n" +
wordLinkedList + "\n");
```

```
    Iterator<String> it =wordLinkedList.iterator(); //the iterator it
will point elements of the linked list
    while(it.hasNext())
    { //when vector Enum has more element to get
        System.out.println(it.next());
    }
}
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data=pd.read_csv("house.csv")
print(data)

x=data[["bedrooms","sqft_living"]]
y=data.price
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print(xtrain)
print(xtest)
print(ytrain)
print(ytest)

le=LinearRegression()
le.fit(xtrain,ytrain)
print(le.intercept_)
print(le.coef_)
print(le.predict([[3.5,1000]]))

ypred=le.predict(xtest)
me=mean_squared_error(ypred,ytest)
print(me)
```

---

index-html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>File Appender</title>
</head>
<body>
```

```
    <h1>File Appender</h1>
    <form action="/append" method="post">
        <label for="sourceFile">Source File:</label>
        <input type="text" id="sourceFile" name="sourceFile"
required><br>

        <label for="targetFile">Target File:</label>
        <input type="text" id="targetFile" name="targetFile"
required><br>

        <input type="submit" value="Append Files">
    </form>
</body>
</html>

2nd file server-js
const fs = require('fs');
const express = require('express');
const bodyParser = require('body-parser');

const app = express();

app.use(bodyParser.urlencoded({ extended: true }));

app.get('/', (req, res) => {
    res.sendFile(__dirname + '/index.html');
});

app.post('/append', (req, res) => {
    const sourceFile = req.body.sourceFile;
    const targetFile = req.body.targetFile;

    fs.readFile(sourceFile, 'utf8', (err, data) => {
        if (err) {
            return res.send('Error reading source file: ' + err.message);
        }

        fs.appendFile(targetFile, data, (err) => {
            if (err) {
                return res.send('Error appending to target file: ' +
err.message);
            }

            res.send('Contents appended successfully!');
        });
    });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
==================================================slip
6=================================================================
```

Write a Java Program to implement command pattern to test Remote Control
Write a python program to implement Polynomial Linear Regression for given dataset
Create a Node.js file that opens the requested file and returns the content to the client.
If anything goes wrong, throw a 404 error.

---

remote control

Command.java

```
package javaprograms;
interface Command
{
public void execute();
}
```

2)Create Class

Light.java

```
package javaprograms;
public class Light
{
public void on()
{
System.out.println("Light is on");
}
public void off()
{
System.out.println("Light is off");
}
}
```

LightOnCommand.java

```
package javaprograms;
class LightOnCommand implements Command
{
Light light;
// The constructor is passed the light it
// is going to control.
public LightOnCommand(Light light)
{
this.light = light;
}
public void execute()
{
light.on();
}
}
```

LightOffCommand.java

```
package javaprograms;
class LightOffCommand implements Command
{
Light light;
public LightOffCommand(Light light)
{
this.light = light;
```

```
}
public void execute()
{
light.off();
}
}
Stereo.java
package javaprograms;
public class Stereo
{
public void on()
{
System.out.println("Stereo
}
public void off()
{
System.out.println("Stereo
}
public void setCD()
{
System.out.println("Stereo
"for CD
}
public void setDVD()
{
is on");
is off");
is set " +
input");System.out.println("Stereo is set"+
" for DVD input");
}
public void setRadio()
{
System.out.println("Stereo is set" +
" for Radio");
}
public void setVolume(int volume)
{
// code to set the volume
System.out.println("Stereo volume set"
+ " to " + volume);
}
}
StereoOffCommand.java
package javaprograms;
class StereoOffCommand implements Command
{
Stereo stereo;
public StereoOffCommand(Stereo stereo)
{
this.stereo = stereo;
}
public void execute()
{
```

```
stereo.off();
}
}
```
StereoOnWithCDCommand.java
```
package javaprograms;
class StereoOnWithCDCommand implements Command
{
Stereo stereo;
public StereoOnWithCDCommand(Stereo stereo)
{
this.stereo = stereo;
}
public void execute()
{
stereo.on();
stereo.setCD();
stereo.setVolume(11);
}
}
```
SimpleRemoteControl.java
```
package javaprograms;
class SimpleRemoteControl
{Command slot;
// only one button
public SimpleRemoteControl()
{
}
public void setCommand(Command command)
{
// set the command the remote will
// execute
slot = command;
}
public void buttonWasPressed()
{
slot.execute();
}
}
```
RemoteControlTest.java
```
package javaprograms;
class RemoteControlTest
{
public static void main(String[] args)
{
SimpleRemoteControl remote =
new SimpleRemoteControl();
Light light = new Light();
Stereo stereo = new Stereo();
// we can change command dynamically
remote.setCommand(new
LightOnCommand(light));
remote.buttonWasPressed();
remote.setCommand(new
StereoOnWithCDCommand(stereo));
```

```
remote.buttonWasPressed();
remote.setCommand(new
StereoOffCommand(stereo));
remote.buttonWasPressed();
}
}
-------------------------------------------------------------------------------
-------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix

data=pd.read_csv("position_salaries.csv")
x=data.iloc[:,1:2].values
y=data.iloc[:,2].values
print(x)
print(y)
le=LinearRegression()
le.fit(x,y)

po=PolynomialFeatures(degree=4)
xp=po.fit_transform(x)
pe=LinearRegression()
pe.fit(xp,y)

plt.scatter(x,y,c="red")
plt.plot(x,le.predict(x),c="green")
plt.show()

plt.scatter(x,y,c="red")
plt.plot(x,pe.predict(po.fit_transform(x)),c="green")
plt.show()

print(le.predict([[3.5]]))
print(pe.predict(po.fit_transform([[3.5]])))


-------------------------------------------------------------------------------
----------------------------------------
const http = require('http');
const fs = require('fs');

const server = http.createServer((req, res) => {
    const filePath = `.${req.url}`;
    fs.readFile(filePath, (err, data) => {
        if (err) {
            res.writeHead(404, {'Content-Type': 'text/plain'});
            res.end('404 Not Found');
        } else {
            res.writeHead(200, {'Content-Type': 'text/plain'});
            res.end(data);
```

```
        }
    });
});

const PORT = process.env.PORT || 3000;
server.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

==========slip 7==========

Write a Java Program to implement undo command to test Ceiling fan.
Write a python program to implement Naive Bayes. [20 M]
Create a Node.js file that writes an HTML form, with an upload field.

Command.java

```
interface Command {
        public void execute();
        public void undo();
}
```

NoCommand.java

```
class NoCommand implements Command {
        public void execute() { }
        public void undo() { }
}
```

CeilingFan.java

```
class CeilingFan {
        public static final int HIGH = 3;
        public static final int MEDIUM = 2;
        public static final int LOW = 1;
        public static final int OFF = 0;
        String location;
        int speed;

        public CeilingFan(String location) {
                this.location = location;
                speed = OFF;
        }

        public void high() {
                speed = HIGH;
                System.out.println(location + " ceiling fan is on high");
        }

        public void medium() {
                speed = MEDIUM;
                System.out.println(location + " ceiling fan is on
medium");
        }
```

```
        public void low() {
                speed = LOW;
                System.out.println(location + " ceiling fan is on low");
        }

        public void off() {
                speed = OFF;
                System.out.println(location + " ceiling fan is off");
        }

        public int getSpeed() {
                return speed;
        }
}
```

CeilingFanHighCommand.java

```
class CeilingFanHighCommand implements Command {
        CeilingFan ceilingFan;
        int prevSpeed;

        public CeilingFanHighCommand(CeilingFan ceilingFan) {
                this.ceilingFan = ceilingFan;
        }

        public void execute() {
                prevSpeed = ceilingFan.getSpeed();
                ceilingFan.high();
        }

        public void undo() {
                if (prevSpeed == CeilingFan.HIGH) {
                        ceilingFan.high();
                } else if (prevSpeed == CeilingFan.MEDIUM) {
                        ceilingFan.medium();
                } else if (prevSpeed == CeilingFan.LOW) {
                        ceilingFan.low();
                } else if (prevSpeed == CeilingFan.OFF) {
                        ceilingFan.off();
                }
        }
}
```

CeilingFanLowCommand.java

```
class CeilingFanLowCommand implements Command {
        CeilingFan ceilingFan;
        int prevSpeed;

        public CeilingFanLowCommand(CeilingFan ceilingFan) {
                this.ceilingFan = ceilingFan;
        }

        public void execute() {
                prevSpeed = ceilingFan.getSpeed();
```

```
                ceilingFan.low();
        }

        public void undo() {
                if (prevSpeed == CeilingFan.HIGH) {
                        ceilingFan.high();
                } else if (prevSpeed == CeilingFan.MEDIUM) {
                        ceilingFan.medium();
                } else if (prevSpeed == CeilingFan.LOW) {
                        ceilingFan.low();
                } else if (prevSpeed == CeilingFan.OFF) {
                        ceilingFan.off();
                }
        }
}

CeilingFanMediumCommand.java
class CeilingFanMediumCommand implements Command {
        CeilingFan ceilingFan;
        int prevSpeed;

        public CeilingFanMediumCommand(CeilingFan ceilingFan) {
                this.ceilingFan = ceilingFan;
        }

        public void execute() {
                prevSpeed = ceilingFan.getSpeed();
                ceilingFan.medium();
        }

        public void undo() {
                if (prevSpeed == CeilingFan.HIGH) {
                        ceilingFan.high();
                } else if (prevSpeed == CeilingFan.MEDIUM) {
                        ceilingFan.medium();
                } else if (prevSpeed == CeilingFan.LOW) {
                        ceilingFan.low();
                } else if (prevSpeed == CeilingFan.OFF) {
                        ceilingFan.off();
                }
        }
}

CeilingFanOffCommand.java
class CeilingFanOffCommand implements Command {
        CeilingFan ceilingFan;
        int prevSpeed;

        public CeilingFanOffCommand(CeilingFan ceilingFan) {
                this.ceilingFan = ceilingFan;
        }

        public void execute() {
                prevSpeed = ceilingFan.getSpeed();
```

```
                ceilingFan.off();
        }

        public void undo() {
                if (prevSpeed == CeilingFan.HIGH) {
                        ceilingFan.high();
                } else if (prevSpeed == CeilingFan.MEDIUM) {
                        ceilingFan.medium();
                } else if (prevSpeed == CeilingFan.LOW) {
                        ceilingFan.low();
                } else if (prevSpeed == CeilingFan.OFF) {
                        ceilingFan.off();
                }
        }
}

RemoteControlWithUndo.java
class RemoteControlWithUndo {
        Command[] onCommands;
        Command[] offCommands;
        Command undoCommand;

        public RemoteControlWithUndo() {
                onCommands = new Command[7];
                offCommands = new Command[7];

                Command noCommand = new NoCommand();
                for(int i=0;i<7;i++) {
                        onCommands[i] = noCommand;
                        offCommands[i] = noCommand;
                }
                undoCommand = noCommand;
        }

        public void setCommand(int slot, Command onCommand, Command
offCommand) {
                onCommands[slot] = onCommand;
                offCommands[slot] = offCommand;
        }

        public void onButtonWasPushed(int slot) {
                onCommands[slot].execute();
                undoCommand = onCommands[slot];
        }

        public void offButtonWasPushed(int slot) {
                offCommands[slot].execute();
                undoCommand = offCommands[slot];
        }

        public void undoButtonWasPushed() {
                undoCommand.undo();
        }
```

```
        public String toString() {
                StringBuffer stringBuff = new StringBuffer();
                stringBuff.append("\n------ Remote Control -------\n");
                for (int i = 0; i < onCommands.length; i++) {
                        stringBuff.append("[slot " + i + "] " +
onCommands[i].getClass().getName()
                                + "    " +
offCommands[i].getClass().getName() + "\n");
                }
                stringBuff.append("[undo] " +
undoCommand.getClass().getName() + "\n");
                return stringBuff.toString();
        }
}


RemoteLoader.java
class RemoteLoader {

        public static void main(String[] args) {
                RemoteControlWithUndo remoteControl = new
RemoteControlWithUndo();

                CeilingFan ceilingFan = new CeilingFan("Living Room");

                CeilingFanMediumCommand ceilingFanMedium = new
CeilingFanMediumCommand(ceilingFan);
                CeilingFanHighCommand ceilingFanHigh = new
CeilingFanHighCommand(ceilingFan);
                CeilingFanOffCommand ceilingFanOff = new
CeilingFanOffCommand(ceilingFan);

                remoteControl.setCommand(0, ceilingFanMedium,
ceilingFanOff);
                remoteControl.setCommand(1, ceilingFanHigh,
ceilingFanOff);

                remoteControl.onButtonWasPushed(0);
                remoteControl.offButtonWasPushed(0);
                System.out.println(remoteControl);
                remoteControl.undoButtonWasPushed();

                remoteControl.onButtonWasPushed(1);
                System.out.println(remoteControl);
                remoteControl.undoButtonWasPushed();
        }
}
---------------------------------------------------------------------------
---------
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

```
data=pd.read_csv("user_data1.csv")
x=data.iloc[:,[2,4]].values
y=data.iloc[:,4].values
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
gd=GaussianNB()
gd.fit(xtrain,ytrain)

ypred=gd.predict(xtest)
cm=confusion_matrix(ypred,ytest)
print(cm)
```

```
const fs = require('fs');

const htmlContent = `
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>File Upload Form</title>
</head>
<body>
    <h1>File Upload Form</h1>
    <form action="/upload" method="post" enctype="multipart/form-data">
        <label for="file">Select a file:</label>
        <input type="file" id="file" name="file" required><br><br>
        <input type="submit" value="Upload">
    </form>
</body>
</html>
`;

fs.writeFile('upload_form.html', htmlContent, (err) => {
    if (err) throw err;
    console.log('HTML file created successfully.');
});
```

==========slip 8==========

Write a Java Program to implement State Pattern for Gumball Machine.
Create instance variable that holds current state from there, we just need to handle all
actions, behaviors and state transition that can happen
Write a python program to implement Decision Tree whether or not to play Tennis.
Create a Node.js file that demonstrates create database and table in MySQL.

```
-------------------------------------------------------------------------
-----------------------------------------------------
State.java
public interface State {

      public void insertQuarter();
      public void ejectQuarter();
      public void turnCrank();
      public void dispense();

      public void refill();
}

NoQuarterState.java
public class NoQuarterState implements State {
    GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("You inserted a quarter");
            gumballMachine.setState(gumballMachine.getHasQuarterState());
      }

      public void ejectQuarter() {
            System.out.println("You haven't inserted a quarter");
      }

      public void turnCrank() {
            System.out.println("You turned, but there's no quarter");
       }

      public void dispense() {
            System.out.println("You need to pay first");
      }

      public void refill() { }

      public String toString() {
            return "waiting for quarter";
      }
}

HasQuarterState.java
public class HasQuarterState implements State {
      GumballMachine gumballMachine;

      public HasQuarterState(GumballMachine gumballMachine) {
            this.gumballMachine = gumballMachine;
      }
```

```
      public void insertQuarter() {
            System.out.println("You can't insert another quarter");
      }

      public void ejectQuarter() {
            System.out.println("Quarter returned");
            gumballMachine.setState(gumballMachine.getNoQuarterState());
      }

      public void turnCrank() {
            System.out.println("You turned...");
            gumballMachine.setState(gumballMachine.getSoldState());
      }

    public void dispense() {
        System.out.println("No gumball dispensed");
    }

    public void refill() { }

      public String toString() {
            return "waiting for turn of crank";
      }
}

SoldOutState.java
public class SoldOutState implements State {
    GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("You can't insert a quarter, the machine
is sold out");
      }

      public void ejectQuarter() {
            System.out.println("You can't eject, you haven't inserted a
quarter yet");
      }

      public void turnCrank() {
            System.out.println("You turned, but there are no gumballs");
      }

      public void dispense() {
            System.out.println("No gumball dispensed");
      }

      public void refill() {
            gumballMachine.setState(gumballMachine.getNoQuarterState());
      }
```

```
      public String toString() {
            return "sold out";
      }
}

SoldState.java
public class SoldState implements State {

    GumballMachine gumballMachine;

    public SoldState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("Please wait, we're already giving you a
gumball");
      }

      public void ejectQuarter() {
            System.out.println("Sorry, you already turned the crank");
      }

      public void turnCrank() {
            System.out.println("Turning twice doesn't get you another
gumball!");
      }

      public void dispense() {
            gumballMachine.releaseBall();
            if (gumballMachine.getCount() > 0) {

      gumballMachine.setState(gumballMachine.getNoQuarterState());
            } else {
                  System.out.println("Oops, out of gumballs!");

      gumballMachine.setState(gumballMachine.getSoldOutState());
            }
      }

      public void refill() { }

      public String toString() {
            return "dispensing a gumball";
      }
}

GumballMachine.java
public class GumballMachine {

      State soldOutState;
      State noQuarterState;
      State hasQuarterState;
```

```
	State soldState;

	State state;
	int count = 0;

	public GumballMachine(int numberGumballs) {
		soldOutState = new SoldOutState(this);
		noQuarterState = new NoQuarterState(this);
		hasQuarterState = new HasQuarterState(this);
		soldState = new SoldState(this);

		this.count = numberGumballs;
		if (numberGumballs > 0) {
			state = noQuarterState;
		} else {
			state = soldOutState;
		}
	}

	public void insertQuarter() {
		state.insertQuarter();
	}

	public void ejectQuarter() {
		state.ejectQuarter();
	}

	public void turnCrank() {
		state.turnCrank();
		state.dispense();
	}

	void releaseBall() {
		System.out.println("A gumball comes rolling out the
slot...");
		if (count != 0) {
			count = count - 1;
		}
	}

	int getCount() {
		return count;
	}

	void refill(int count) {
		this.count += count;
		System.out.println("The gumball machine was just refilled;
it's new count is: " + this.count);
		state.refill();
	}

	void setState(State state) {
		this.state = state;
	}
```

```
    public State getState() {
        return state;
    }

    public State getSoldOutState() {
        return soldOutState;
    }

    public State getNoQuarterState() {
        return noQuarterState;
    }

    public State getHasQuarterState() {
        return hasQuarterState;
    }

    public State getSoldState() {
        return soldState;
    }

      public String toString() {
            StringBuffer result = new StringBuffer();
            result.append("\nMighty Gumball, Inc.");
            result.append("\nJava-enabled Standing Gumball Model #2004");
            result.append("\nInventory: " + count + " gumball");
            if (count != 1) {
                  result.append("s");
            }
            result.append("\n");
            result.append("Machine is " + state + "\n");
            return result.toString();
      }
}

GumballMachineTestDrive.java
public class GumballMachineTestDrive {

      public static void main(String[] args) {
            GumballMachine gumballMachine = new GumballMachine(2);

            System.out.println(gumballMachine);

            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            System.out.println(gumballMachine);

            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();
            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            gumballMachine.refill(5);
            gumballMachine.insertQuarter();
```

```
            gumballMachine.turnCrank();

            System.out.println(gumballMachine);
      }
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
data=pd.read_csv("tennis.csv")
print(data)

le=LabelEncoder()

data["Outlook"]=le.fit_transform(data["Outlook"])
data["Temprature"]=le.fit_transform(data["Temprature"])
data["Humidity"]=le.fit_transform(data["Humidity"])
data["Wind"]=le.fit_transform(data["Wind"])
data["Play_Tennis"]=le.fit_transform(data["Play_Tennis"])
print(data)


x=data.iloc[:,1:5].values
y=data["Play_Tennis"]

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

dc=DecisionTreeClassifier(criterion="entropy")
dc.fit(xtrain,ytrain)

from sklearn.tree import export_graphviz
export_graphviz(dc,out_file="a1.txt")

ypred=dc.predict(xtest)
cm=confusion_matrix(ypred,ytest)
print(cm)
```

```
const mysql = require("mysql");
const mycon = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "test",
});
mycon.connect((err) => {
  if (err) {
    console.log(err);
  } else {
```

```
    console.log("success");
    mycon.query("create database newdb", function (err, result) {
      if (err) console.log(err);
      console.log("Database Created");
    });

    mycon.query("use newdb", function (err) {
      if (err) console.log(err);
      else console.log("Database { " + "newdb } selected!");
    });

    var sql = "create table customer(name varchar(10), address
varchar(10))";
    mycon.query(sql, function (err, result) {
      if (err) console.log(err);
      else console.log("Table Created");
    });
  }
});
```

==========Slip 9==========

Design simple HR Application using Spring Framework
Write a python program to implement Linear SVM.
Create a node.js file that Select all records from the "customers" table, and display the
result object on console.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

data=pd.read_csv("user_data1.csv")
print(data)

x=data.iloc[:,2:3].values
y=data.iloc[:,3].values
print(x)
print(y)
sr=SVR()
sr.fit(x,y)

print(sr.predict([[150]]))

plt.scatter(x,y,c="red")
plt.plot(x,sr.predict(x),c="green")
plt.show()
```

```
const mysql = require("mysql");
const con = mysql.createConnection({
```

```
  host: "localhost",
  user: "root",
  password: "",
  database: "newdb",
});
con.connect((err) => {
  if (err) {
    console.log(err);
  } else {
    console.log("success");
    con.query("select * from customer", function (err, rows) {
      if (err) throw err;
      else console.log(rows);
      con.query("select * from customer", function (err, rows) {
        if (err) throw err;
        else console.log(rows);
      });
    });
  }
});
```

==========Slip 10==========

Write a Java Program to implement Strategy Pattern for Duck Behavior. Create instance variable that holds current state of Duck from there, we just need to handle all Flying Behaviors and Quack Behavior

Write a Python program to prepare Scatter Plot for Iris Dataset.

Create a node.js file that Insert Multiple Records in "student" table, and display the result object on console.

QuackBehaviour.java

```
package javaprograms;
public interface QuackBehaviour {
public default void quack() {
System.out.println("Quack");
}
}
```

FlyBehaviour.java

```
package javaprograms;
public interface FlyBehaviour {
public void fly();
}
```

2)Create Class - FlyWithWings.java

```
package javaprograms;public class FlyWithWings implements FlyBehaviour {
public void fly() {
System.out.println("I'm flying!!");
}
```

```
}
```
Quack.java
```
package javaprograms;
public class Quack implements QuackBehaviour {
public void quack() {
System.out.println("Quack");
}
}
```
ModolDuck.java
```
package javaprograms;
public class ModolDuck extends Duck {
public ModolDuck() {
flyBehaviour = new FlyNoWay();
quackBehaviour = new Quack();
}
public void display() {
System.out.println("I'm a model duck");
}
}
```
MallardDuck.java
```
package javaprograms;
public class MallardDuck extends Duck {
public MallardDuck() {
quackBehaviour = new Quack();
flyBehaviour = new FlyWithWings();
}
public void display() {
System.out.println("I'm a real Mallard duck");
}
}
```
Duck.java
```
package javaprograms;
public class MallardDuck extends Duck {
public MallardDuck() {
quackBehaviour = new Quack();
flyBehaviour = new FlyWithWings();
}
public void display() {
System.out.println("I'm a real Mallard duck");}
}
```
FlyRocketPowered.java
```
package javaprograms;
public class FlyRocketPowered implements FlyBehaviour {
public void fly() {
System.out.println("I'm flying with a rocket!");
}
}
```
FlyNoWay.java
```
package javaprograms;
public class FlyNoWay implements FlyBehaviour {
public void fly() {
System.out.println("I can't fly");
}
}
```

MiniDuckSimulator.java

```
package javaprograms;
public class MiniDuckSimulator {
public static void main(String[] args) {
Duck mallard = new MallardDuck();
mallard.performQuack();
mallard.performFly();
Duck model = new ModolDuck();
model.performFly();
model.setFlyBehaviour(new FlyRocketPowered());
model.performFly();
}
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("iris.csv")
print(data)
x=data["SepalLengthCm"]
y=data["SepalWidthCm"]
print(x)
print(y)
plt.scatter(x, y,c="green")
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.legend()
plt.show()
```

```
const mysql = require("mysql");
const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "newdb",
});
con.connect((err) => {
  if (err) {
    console.log(err);
  } else {
    console.log("success");
    con.query("create table students(roll int primary key, name
text,address text)",function (err) {
        if (err) throw err;
        else {
          con.query(
            "insert into students
values(111,'sam','kondhwa'),(222,'vaish','kirkitwadi'),(333,'saarth','wan
awadi')",
            function (err) {
              if (err) throw err;
              else {
```

```
                console.log("data inserted!");
                con.query("select * from students", function (err,
result) {
                  if (err) throw err;
                  else console.log(result);
                });
              }
            }
          );
        }
      }
    );
  }
});
```

==========Slip 11==========

Write a java program to implement Adapter pattern to design Heart Model to Beat Model

Write a python program to find all null values in a given dataset and remove them.

Create a node.js file that Select all records from the "customers" table, and delete the specified record.

```
// Target Interface
interface Beat {
    void makeSound();
}

// Adaptee Class (Existing class with incompatible interface)
class Heart {
    void beat() {
        System.out.println("Heart is beating...");
    }
}

// Adapter Class
class HeartAdapter implements Beat {
    private Heart heart;

    public HeartAdapter(Heart heart) {
        this.heart = heart;
    }

    @Override
    public void makeSound() {
        heart.beat();
    }
}

// Client Code
```

```
public class AdapterPatternExample {
    public static void main(String[] args) {
        // Existing Heart model
        Heart heart = new Heart();

        // Adapter to make Heart compatible with Beat interface
        Beat heartAdapter = new HeartAdapter(heart);

        // Using the Beat interface to make the Heart model "beat"
        heartAdapter.makeSound();
    }
}
```

```
import pandas as pd
import numpy as np
data=pd.read_csv("ass2.csv")
print(data)
print(data.isnull())
print(data.notnull())

data1=data.dropna(axis=1,how="all")
print(data1)
data["name"]=data["name"].replace(np.nan,"xyz")
data["m1"]=data["m1"].replace(np.nan,99)
data["m2"]=data["m2"].replace(np.nan, data["m2"].mean())
print(data)
```

```
const mysql = require("mysql");
const con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "newdb",
});
con.connect((err) => {
  if (err) {
    console.log(err);
  } else {
    console.log("success");
    con.query("select * from customer", function (err, result) {
      if (err) throw err;
      else {
        console.log(result);
        con.query(
          "delete from customer where name = 'teja'",
          function (err, result) {
            if (err) throw err;
            else {
              console.log("Deleted Record : " + result.affectedRows);
              con.query("select * from customer", function (err, result)
{
```

```
                if (err) throw err;
                else
                {
                    console.log("Data After delete!");
                    console.log(result);
                }
              });
            }
          }
        );
      }
    });
  }
});
```

========== Slip 12==========
=
Write a Java Program to implement Decorator Pattern for interface Car to define the
assemble() method and then decorate it to Sports car and Luxury Car
Write a python program to make Categorical values in numeric format for a given
dataset
Create a Simple Web Server using node js.

Car.java

```
package com.journaldev.design.decorator;
public interface Car {
      public void assemble();
}
```

BasicCar.java

```
package com.journaldev.design.decorator;
public class BasicCar implements Car {
      @Override
      public void assemble() {
            System.out.print("Basic Car.");
      }
}
```

CarDecorator.java

```
package com.journaldev.design.decorator;
public class CarDecorator implements Car {
      protected Car car;
      public CarDecorator(Car c){
            this.car=c;
      }

      @Override
      public void assemble() {
            this.car.assemble();
      }
}
```

SportsCar.java

```
package com.journaldev.design.decorator;
public class SportsCar extends CarDecorator {
      public SportsCar(Car c) {
            super(c);
      }
      @Override
      public void assemble(){
            super.assemble();
            System.out.print(" Adding features of Sports Car.");
      }
}
LuxuryCar.java
package com.journaldev.design.decorator;
public class LuxuryCar extends CarDecorator {

      public LuxuryCar(Car c) {
            super(c);
      }
      @Override
      public void assemble(){
            super.assemble();
            System.out.print(" Adding features of Luxury Car.");
      }
}
package com.journaldev.design.test;
import com.journaldev.design.decorator.BasicCar;
import com.journaldev.design.decorator.Car;
import com.journaldev.design.decorator.LuxuryCar;
import com.journaldev.design.decorator.SportsCar;
DecoratorPatternTest.java
public class DecoratorPatternTest {

      public static void main(String[] args) {
            Car sportsCar = new SportsCar(new BasicCar());
            sportsCar.assemble();
            System.out.println("\n*****");

            Car sportsLuxuryCar = new SportsCar(new LuxuryCar(new
BasicCar()));
            sportsLuxuryCar.assemble();
      }

}
--------------------------------------------------------------------------
--------------------------------
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

data=pd.read_csv("ass3.csv")
print(data)
x=data.iloc[:,0:1].values
print(x)
```

```
le=LabelEncoder()
x1=le.fit_transform(x)
print(x1)

oe=OneHotEncoder()
x2=oe.fit_transform(x).toarray()
print(x2)
```

```
const express = require('express');
const app = express();
const port = 3000;

// Define a route
app.get('/', (req, res) => {
    res.send('Hello World!');
});

// Start the server
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
```

==========Slip 13==========

Write a Java Program to implement an Adapter design pattern in mobile charger.
Define two classes – Volt (to measure volts) and Socket (producing constant volts of
120V). Build an adapter that can produce 3 volts, 12 volts and default 120 volts.
Implements Adapter pattern using Class Adapter
Write a Python program to prepare Scatter Plot for Iris Dataset [20 M]
Using node js create a User Login System.

Voltage.java

```
public class Voltage
{
    private int voltage;
    public Voltage(int v)
    {
        this.voltage = v;
    }
    public int getVolts()
    {
        return voltage;
    }
```

```
    public void setVolts(int voltage)
    {
        this.voltage = voltage;
    }
}
```

SocketAdapter.java

```
public interface SocketAdapter
{
     public Voltage get120Voltage();
     public Voltage get12Voltage();
     public Voltage get3VVoltage();
}
```

Socket.java

```
public class Socket
{
    public Voltage getVoltage()
    {
        return new Voltage(120); //In India 240 is the default voltage
    }
}
```

SocketAdapterImpl.java

```
public class SocketAdapterImpl extends Socket implements SocketAdapter
{

    //Using Composition for adapter pattern
    private Socket sock = new Socket();
    private Voltage convertVolt(Voltage v, int i)
    {
        return new Voltage(v.getVolts() / i);
    }
    @Override
    public Voltage get120Voltage()
    {
        return sock.getVoltage();
    }
    @Override
    public Voltage get12Voltage()
    {
        Voltage v = sock.getVoltage();
        return convertVolt(v, 20);
    }
    @Override
    public Voltage get3VVoltage()
    {
        Voltage v = sock.getVoltage();
        return convertVolt(v, 80);
    }
}
```

AdapterEx.java

```
public class AdapterEx

{

    public static void main(String[] args)
```

```
    {
        SocketAdapter socketAdapter = new SocketAdapterImpl();
        Voltage voltage12 = socketAdapter.get12Voltage();
        System.out.println(voltage12.getVolts());

        Voltage voltage3 = socketAdapter.get3VVoltage();
        System.out.println(voltage3.getVolts());

        Voltage voltage120 = socketAdapter.get120Voltage();
        System.out.println(voltage120.getVolts());
    }
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("iris.csv")
print(data)
x=data["SepalLengthCm"]
y=data["SepalWidthCm"]
print(x)
print(y)
plt.scatter(x, y,c="green")
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.legend()
plt.show()
```

```
const express = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');

const app = express();
const port = 8000;

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
    secret: 'your-secret-key',
    resave: false,
    saveUninitialized: true
}));

// In-memory database (for demo purposes, use a real database in
production)
const users = [
    { username: 'user1', password: 'password1' },
    { username: 'user2', password: 'password2' }
];
```

```
// Routes
app.get('/', (req, res) => {
    res.send('Welcome to the login page');
});

app.get('/login', (req, res) => {
    res.sendFile(__dirname + '/login.html');
});

app.post('/login', (req, res) => {
    const { username, password } = req.body;

    const user = users.find(u => u.username === username && u.password
=== password);

    if (user) {
        req.session.user = user;
        res.redirect('/dashboard');
    } else {
        res.send('Invalid username or password');
    }
});

app.get('/dashboard', (req, res) => {
    if (req.session.user) {
        res.send(`Welcome, ${req.session.user.username}!`);
    } else {
        res.redirect('/login');
    }
});

app.get('/logout', (req, res) => {
    req.session.destroy(err => {
        if (err) {
            return res.send('Error logging out');
        }
        res.redirect('/');
    });
});

app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});

login html file
<form action="/login" method="post">
    <div>
        <label for="username">Username:</label>
        <input type="text" id="username" name="username">
    </div>
    <div>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password">
    </div>
```

```
    <div>
        <input type="submit" value="Submit">
    </div>
</form>
```

==========Slip 14==========

Write a Java Program to implement Command Design Pattern for Command Interface with execute() . Use this to create variety of commands for LightOnCommand, LightOffCommand, GarageDoorUpCommand, StereoOnWithCDComman. [20 M]

Write a python program to find all null values in a given dataset and remove them.

Write node js script to interact with the filesystem, and serve a web page from a file

---

Command.java

```
public interface Command {
      public void execute();
}
```

LightOnCommand.java

```
public class LightOnCommand implements Command {
      Light light;

      public LightOnCommand(Light light) {
            this.light = light;
      }

      public void execute() {
            light.on();
      }
}
```

LightOffCommand.java

```
public class LightOffCommand implements Command {
      Light light;

      public LightOffCommand(Light light) {
            this.light = light;
      }

      public void execute() {
            light.off();
      }
}
```

Light.java

```
public class Light {
      String location = "";

      public Light(String location) {
            this.location = location;
      }
```

```
	public void on() {
		System.out.println(location + " light is on");
	}
	public void off() {
		System.out.println(location + " light is off");
	}
}
StereoOnWithCDCommand.java
public class StereoOnWithCDCommand implements Command {
	Stereo stereo;

	public StereoOnWithCDCommand(Stereo stereo) {
		this.stereo = stereo;
	}
	public void execute() {
		stereo.on();
		stereo.setCD();
		stereo.setVolume(11);
	}
}
StereoOffCommand.java
public class StereoOffCommand implements Command {
	Stereo stereo;

	public StereoOffCommand(Stereo stereo) {
		this.stereo = stereo;
	}
	public void execute() {
		stereo.off();
	}
}
RemoteControlWithUndo.java

public class RemoteControlWithUndo {
	Command[] onCommands;
	Command[] offCommands;
	Command undoCommand;

	public RemoteControlWithUndo() {
		onCommands = new Command[7];
		offCommands = new Command[7];

		Command noCommand = new NoCommand();
		for(int i=0;i<7;i++) {
			onCommands[i] = noCommand;
			offCommands[i] = noCommand;
		}
		undoCommand = noCommand;
	}
	public void setCommand(int slot, Command onCommand, Command
offCommand) {
		onCommands[slot] = onCommand;
		offCommands[slot] = offCommand;
	}
```

```
	public void onButtonWasPushed(int slot) {
		onCommands[slot].execute();
		undoCommand = onCommands[slot];
	}
	public void offButtonWasPushed(int slot) {
		offCommands[slot].execute();
		undoCommand = offCommands[slot];
	}
	public void undoButtonWasPushed() {
		undoCommand.undo();
	}

	public String toString() {
		StringBuffer stringBuff = new StringBuffer();
		stringBuff.append("\n------ Remote Control -------\n");
		for (int i = 0; i < onCommands.length; i++) {
			stringBuff.append("[slot " + i + "] " +
onCommands[i].getClass().getName()
					+ "    " + offCommands[i].getClass().getName() +
"\n");
		}
		stringBuff.append("[undo] " +
undoCommand.getClass().getName() + "\n");
		return stringBuff.toString();
	}
}
NoCommand.java
public class NoCommand implements Command {
	public void execute() { }
	public void undo() { }
}
RemoteLoader.java
public class RemoteLoader {

	public static void main(String[] args) {
		RemoteControlWithUndo remoteControl = new
RemoteControlWithUndo();

		Light livingRoomLight = new Light("Living Room");
		Light kitchenLight = new Light("Kitchen");
		CeilingFan ceilingFan= new CeilingFan("Living Room");
		GarageDoor garageDoor = new GarageDoor("");
		Stereo stereo = new Stereo("Living Room");

		LightOnCommand livingRoomLightOn =
				new LightOnCommand(livingRoomLight);
		LightOffCommand livingRoomLightOff =
				new LightOffCommand(livingRoomLight);
		LightOnCommand kitchenLightOn =
				new LightOnCommand(kitchenLight);
		LightOffCommand kitchenLightOff =
				new LightOffCommand(kitchenLight);

		CeilingFanOnCommand ceilingFanOn =
```

```
                    new CeilingFanOnCommand(ceilingFan);
            CeilingFanOffCommand ceilingFanOff =
                    new CeilingFanOffCommand(ceilingFan);

            GarageDoorUpCommand garageDoorUp =
                    new GarageDoorUpCommand(garageDoor);
            GarageDoorDownCommand garageDoorDown =
                    new GarageDoorDownCommand(garageDoor);

            StereoOnWithCDCommand stereoOnWithCD =
                    new StereoOnWithCDCommand(stereo);
            StereoOffCommand  stereoOff =
                    new StereoOffCommand(stereo);

            remoteControl.setCommand(0, livingRoomLightOn,
livingRoomLightOff);
            remoteControl.setCommand(1, kitchenLightOn, kitchenLightOff);
            remoteControl.setCommand(2, ceilingFanOn, ceilingFanOff);
            remoteControl.setCommand(3, stereoOnWithCD, stereoOff);

            System.out.println(remoteControl);

            remoteControl.onButtonWasPushed(0);
            remoteControl.offButtonWasPushed(0);
            remoteControl.onButtonWasPushed(1);
            remoteControl.offButtonWasPushed(1);
            remoteControl.onButtonWasPushed(2);
            remoteControl.offButtonWasPushed(2);
            remoteControl.onButtonWasPushed(3);
            remoteControl.offButtonWasPushed(3);
      }
}
-------------------------------------------------------------------------
-----------------------------------------------------------
import pandas as pd
import numpy as np
data=pd.read_csv("ass2.csv")
print(data)
print(data.isnull())
print(data.notnull())

data1=data.dropna(axis=1,how="all")
print(data1)
data["name"]=data["name"].replace(np.nan,"xyz")
data["m1"]=data["m1"].replace(np.nan,99)
data["m2"]=data["m2"].replace(np.nan, data["m2"].mean())
print(data)

-------------------------------------------------------------------------
-----------------------------------------------
const fs = require('fs');
const http = require('http');

const port = 8000;
```

```
const server = http.createServer((req, res) => {
    if (req.url === '/') {
        fs.readFile(__dirname + '/index.html', 'utf8', (err, data) => {
            if (err) {
                res.writeHead(500);
                res.end('Error reading the file');
                return;
            }

            res.writeHead(200, { 'Content-Type': 'text/html' });
            res.end(data);
        });
    } else {
        res.writeHead(404);
        res.end('Page not found');
    }
});

server.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});

index html file
<!DOCTYPE html>
<html>
<head>
    <title>Simple Web Page</title>
</head>
<body>
    <h1>Hello from Node.js!</h1>
    <p>This is a simple web page served by a Node.js script.</p>
</body>
</html>
```

==========

slip 15==========

Write a Java Program to implement Facade Design Pattern for HomeTheater

Write a python program to make Categorical values in numeric format for a given dataset

Write node js script to build Your Own Node.js Module. Use require ('http') module is a built-in Node module that invokes the functionality of the HTTP library to create a local server. Also use the export statement to make functions in your module available externally. Create a new text file to contain the functions in your module called, "modules.js" and add this function to return today's date and time.

```
-------------------------------------------------------------------------
-----------------------------------------------------
public class Amplifier {
 String description;
 Tuner tuner;
 DvdPlayer dvd;
 CdPlayer cd;

 public Amplifier(String description) {
  this.description = description;
 }
 public void on() {
  System.out.println(description + " on");
 }
 public void off() {
  System.out.println(description + " off");
 }
 public void setStereoSound() {
  System.out.println(description + " stereo mode on");
 }
 public void setSurroundSound() {
  System.out.println(description + " surround sound on (5 speakers, 1
subwoofer)");
 }
 public void setVolume(int level) {
  System.out.println(description + " setting volume to " + level);
 }
 public void setTuner(Tuner tuner) {
  System.out.println(description + " setting tuner to " + dvd);
  this.tuner = tuner;
 }
 public void setDvd(DvdPlayer dvd) {
  System.out.println(description + " setting DVD player to " + dvd);
  this.dvd = dvd;
 }
 public void setCd(CdPlayer cd) {
  System.out.println(description + " setting CD player to " + cd);
  this.cd = cd;
 }
public String toString() {
  return description;
 }
}

CdPlayer.java

public class CdPlayer {
 String description;
 int currentTrack;
 Amplifier amplifier;
 String title;

 public CdPlayer(String description, Amplifier amplifier) {
  this.description = description;
```

```
  this.amplifier = amplifier;
 }
 public void on() {
  System.out.println(description + " on");
 }
 public void off() {
  System.out.println(description + " off");
 }
 public void eject() {
  title = null;
  System.out.println(description + " eject");
 }
 public void play(String title) {
  this.title = title;
  currentTrack = 0;
  System.out.println(description + " playing \"" + title + "\"");
 }
 public void play(int track) {
  if (title == null) {
   System.out.println(description + " can't play track " + currentTrack +
     ", no cd inserted");
  } else {
   currentTrack = track;
   System.out.println(description + " playing track " + currentTrack);
  }
 }

 public void stop() {
  currentTrack = 0;
  System.out.println(description + " stopped");
 }

 public void pause() {
  System.out.println(description + " paused \"" + title + "\"");
 }

 public String toString() {
  return description;
 }
}
```

DvdPlayer.java

```
public class DvdPlayer {
 String description;
 int currentTrack;
 Amplifier amplifier;
 String movie;

 public DvdPlayer(String description, Amplifier amplifier) {
  this.description = description;
  this.amplifier = amplifier;
 }
```

```
 public void on() {
  System.out.println(description + " on");
 }

 public void off() {
  System.out.println(description + " off");
 }

        public void eject() {
  movie = null;
                System.out.println(description + " eject");
        }

 public void play(String movie) {
  this.movie = movie;
  currentTrack = 0;
  System.out.println(description + " playing \"" + movie + "\"");
 }

 public void play(int track) {
  if (movie == null) {
   System.out.println(description + " can't play track " + track + " no
dvd inserted");
  } else {
   currentTrack = track;
   System.out.println(description + " playing track " + currentTrack + "
of \"" + movie + "\"");
  }
 }

 public void stop() {
  currentTrack = 0;
  System.out.println(description + " stopped \"" + movie + "\"");
 }

 public void pause() {
  System.out.println(description + " paused \"" + movie + "\"");
 }

 public void setTwoChannelAudio() {
  System.out.println(description + " set two channel audio");
 }

 public void setSurroundAudio() {
  System.out.println(description + " set surround audio");
 }

 public String toString() {
  return description;
 }
}
```

Projector.java

```
public class Projector {
 String description;
 DvdPlayer dvdPlayer;

 public Projector(String description, DvdPlayer dvdPlayer) {
  this.description = description;
  this.dvdPlayer = dvdPlayer;
 }

 public void on() {
  System.out.println(description + " on");
 }

 public void off() {
  System.out.println(description + " off");
 }

 public void wideScreenMode() {
  System.out.println(description + " in widescreen mode (16x9 aspect
ratio)");
 }

 public void tvMode() {
  System.out.println(description + " in tv mode (4x3 aspect ratio)");
 }

        public String toString() {
                return description;
        }
}
```

TheaterLights.java

```
public class TheaterLights {
 String description;

 public TheaterLights(String description) {
  this.description = description;
 }

 public void on() {
  System.out.println(description + " on");
 }

 public void off() {
  System.out.println(description + " off");
 }

 public void dim(int level) {
  System.out.println(description + " dimming to " + level  + "%");
 }

        public String toString() {
                return description;
```

```
        }
}
```

Screen.java

```
public class Screen {
 String description;

 public Screen(String description) {
  this.description = description;
 }

 public void up() {
  System.out.println(description + " going up");
 }

 public void down() {
  System.out.println(description + " going down");
 }
        public String toString() {
                return description;
        }
}
```

PopcornPopper.java

```
public class PopcornPopper {
 String description;

 public PopcornPopper(String description) {
  this.description = description;
 }

 public void on() {
  System.out.println(description + " on");
 }

 public void off() {
  System.out.println(description + " off");
 }

 public void pop() {
  System.out.println(description + " popping popcorn!");
 }
        public String toString() {
                return description;
        }
}
```

HomeTheaterFacade.java

```
public class HomeTheaterFacade {
 Amplifier amp;
 Tuner tuner;
 DvdPlayer dvd;
```

```
CdPlayer cd;
Projector projector;
TheaterLights lights;
Screen screen;
PopcornPopper popper;

public HomeTheaterFacade(Amplifier amp,
    Tuner tuner,
    DvdPlayer dvd,
    CdPlayer cd,
    Projector projector,
    Screen screen,
    TheaterLights lights,
    PopcornPopper popper) {

 this.amp = amp;
 this.tuner = tuner;
 this.dvd = dvd;
 this.cd = cd;
 this.projector = projector;
 this.screen = screen;
 this.lights = lights;
 this.popper = popper;
}

public void watchMovie(String movie) {
 System.out.println("Get ready to watch a movie...");
 popper.on();
 popper.pop();
 lights.dim(10);
 screen.down();
 projector.on();
 projector.wideScreenMode();
 amp.on();
 amp.setDvd(dvd);
 amp.setSurroundSound();
 amp.setVolume(5);
 dvd.on();
 dvd.play(movie);
}


public void endMovie() {
 System.out.println("Shutting movie theater down...");
 popper.off();
 lights.on();
 screen.up();
 projector.off();
 amp.off();
 dvd.stop();
 dvd.eject();
 dvd.off();
}
```

```
 public void listenToCd(String cdTitle) {
  System.out.println("Get ready for an audiopile experence...");
  lights.on();
  amp.on();
  amp.setVolume(5);
  amp.setCd(cd);
  amp.setStereoSound();
  cd.on();
  cd.play(cdTitle);
 }
 public void endCd() {
  System.out.println("Shutting down CD...");
  amp.off();
  amp.setCd(cd);
  cd.eject();
  cd.off();
 }
 public void listenToRadio(double frequency) {
  System.out.println("Tuning in the airwaves...");
  tuner.on();
  tuner.setFrequency(frequency);
  amp.on();
  amp.setVolume(5);
  amp.setTuner(tuner);
 }
 public void endRadio() {
  System.out.println("Shutting down the tuner...");
  tuner.off();
  amp.off();
 }
}
```

HomeTheaterTestDrive.java

```
public class HomeTheaterTestDrive {
 public static void main(String[] args) {
  Amplifier amp = new Amplifier("Top-O-Line Amplifier");
  Tuner tuner = new Tuner("Top-O-Line AM/FM Tuner", amp);
  DvdPlayer dvd = new DvdPlayer("Top-O-Line DVD Player", amp);
  CdPlayer cd = new CdPlayer("Top-O-Line CD Player", amp);
  Projector projector = new Projector("Top-O-Line Projector", dvd);
  TheaterLights lights = new TheaterLights("Theater Ceiling Lights");
  Screen screen = new Screen("Theater Screen");
  PopcornPopper popper = new PopcornPopper("Popcorn Popper");

  HomeTheaterFacade homeTheater =
    new HomeTheaterFacade(amp, tuner, dvd, cd,
      projector, screen, lights, popper);

  homeTheater.watchMovie("Raiders of the Lost Ark");
  homeTheater.endMovie();
 }
}
```

Tuner.java

```
public class Tuner {
	String description;
	Amplifier amplifier;
	double frequency;
	public Tuner(String description, Amplifier amplifier) {
		this.description = description;
	}
	public void on() {
		System.out.println(description + " on");
	}
	public void off() {
		System.out.println(description + " off");
	}
	public void setFrequency(double frequency) {
		System.out.println(description + " setting frequency to " +
frequency);
		this.frequency = frequency;
	}
	public void setAm() {
		System.out.println(description + " setting AM mode");
	}

	public void setFm() {
		System.out.println(description + " setting FM mode");
	}
	public String toString() {
		return description;
	}
}
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

data=pd.read_csv("ass3.csv")
print(data)
x=data.iloc[:,0:1].values
print(x)

le=LabelEncoder()
x1=le.fit_transform(x)
print(x1)

oe=OneHotEncoder()
x2=oe.fit_transform(x).toarray()
print(x2)
```

---

app js file

```
const myModule = require('./modules.js');
```

```
myModule.startServer();

const dateTime = myModule.getDateTime();
console.log(`Current date and time: ${dateTime}`);

module js file
const http = require('http');

function startServer() {
  const server = http.createServer((req, res) => {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello, World!\n');
  });

  const port = 8000;
  server.listen(port, () => {
    console.log(`Server running at http://localhost:${port}/`);
  });
}

function getDateTime() {
  const now = new Date();
  return now.toLocaleString();
}

module.exports = {
  startServer,
  getDateTime
};
```

==========slip 16==========

Write a Java Program to implement Observer Design Pattern for number conversion.
Accept a number in Decimal form and represent it in Hexadecimal, Octal and Binary.
Change the Number and it reflects in other forms also
Write a python program to Implement Simple Linear Regression for predicting house
price.
Create a js file named main.js for event-driven application. There should be a main
loop that listens for events, and then triggers a callback function when one of those
events is detected.

Subject.java

```
import java.util.ArrayList;
import java.util.List;

public class Subject {

   private List<Observer> observers = new ArrayList<Observer>();
```

```
   private int state;

   public int getState() {
      return state;
   }

   public void setState(int state) {
      this.state = state;
      notifyAllObservers();
   }

   public void attach(Observer observer){
      observers.add(observer);
   }

   public void notifyAllObservers(){
      for (Observer observer : observers) {
         observer.update();
      }
   }
}
```

Observer.java

```
public abstract class Observer {
   protected Subject subject;
   public abstract void update();
}
```

BinaryObserver.java

```
public class BinaryObserver extends Observer{

   public BinaryObserver(Subject subject){
      this.subject = subject;
      this.subject.attach(this);
   }

   @Override
   public void update() {
      System.out.println( "Binary String: " + Integer.toBinaryString(
subject.getState() ) );
   }
}
```

OctalObserver.java

```
public class OctalObserver extends Observer{

   public OctalObserver(Subject subject){
      this.subject = subject;
      this.subject.attach(this);
   }

   @Override
   public void update() {
     System.out.println( "Octal String: " + Integer.toOctalString(
subject.getState() ) );
   }
}
```

HexaObserver.java

```
public class HexaObserver extends Observer{

   public HexaObserver(Subject subject){
      this.subject = subject;
      this.subject.attach(this);
   }

   @Override
   public void update() {
      System.out.println( "Hex String: " + Integer.toHexString(
subject.getState() ).toUpperCase() );
   }
}
```

ObserverPatternDemo.java

```
public class ObserverPatternDemo {
   public static void main(String[] args) {
      Subject subject = new Subject();

      new HexaObserver(subject);
      new OctalObserver(subject);
      new BinaryObserver(subject);

      System.out.println("First state change: 15");
      subject.setState(15);
      System.out.println("Second state change: 10");
      subject.setState(10);
   }
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data=pd.read_csv("house.csv")
print(data)

x=data[["bedrooms"]]
y=data.price
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print(xtrain)
print(xtest)
print(ytrain)
print(ytest)

le=LinearRegression()
le.fit(xtrain,ytrain)
```

```
print(le.intercept_)
print(le.coef_)
print(le.predict([[3.5]]))

ypred=le.predict(xtest)
me=mean_squared_error(ypred,ytest)
print(me)

plt.scatter(x,y,c="red")
plt.plot(xtest,le.predict(xtest),c="green")
plt.show()
```

```
const EventEmitter = require('events');

// Create an instance of EventEmitter
const myEmitter = new EventEmitter();

// Define a callback function for the 'customEvent' event
function eventHandler() {
  console.log('Custom event detected!');
}

// Listen for the 'customEvent' event and attach the event handler
myEmitter.on('customEvent', eventHandler);

// Start the main loop
function mainLoop() {
  // Simulate some asynchronous tasks or events
  setTimeout(() => {
    // Trigger the 'customEvent' event
    myEmitter.emit('customEvent');

    // Continue with the main loop
    mainLoop();
  }, 1000); // This timeout represents some asynchronous operation
}

// Start the main loop
mainLoop();
```

==========slip 17==========

Write a Java Program to implement Abstract Factory Pattern for Shape interface.
Write a python program to implement Multiple Linear Regression for a given dataset.
Write node js application that transfer a file as an attachment on web and enables
browser to prompt the user to download file using express js.

```
--------------------------------------------------------------------------
-----------------
Shape.java
public interface Shape {
   void draw();
}
public class RoundedRectangle implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside RoundedRectangle::draw() method.");
   }
}
RoundedSquare.java

public class RoundedSquare implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside RoundedSquare::draw() method.");
   }
}
Rectangle.java

public class Rectangle implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
Step 3
Create an Abstract class to get factories for Normal and Rounded Shape
Objects.

AbstractFactory.java

public abstract class AbstractFactory {
   abstract Shape getShape(String shapeType) ;
}
Step 4
Create Factory classes extending AbstractFactory to generate object of
concrete class based on given information.

ShapeFactory.java

public class ShapeFactory extends AbstractFactory {
   @Override
   public Shape getShape(String shapeType){
      if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new Rectangle();
      }else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new Square();
      }
      return null;
   }
}
```

RoundedShapeFactory.java

```
public class RoundedShapeFactory extends AbstractFactory {
   @Override
   public Shape getShape(String shapeType){
      if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new RoundedRectangle();
      }else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new RoundedSquare();
      }
      return null;
   }
}
```

Step 5
Create a Factory generator/producer class to get factories by passing an information such as Shape

FactoryProducer.java

```
public class FactoryProducer {
   public static AbstractFactory getFactory(boolean rounded){
      if(rounded){
         return new RoundedShapeFactory();
      }else{
         return new ShapeFactory();
      }
   }
}
```

Step 6
Use the FactoryProducer to get AbstractFactory in order to get factories of concrete classes by passing an information such as type.

AbstractFactoryPatternDemo.java

```
public class AbstractFactoryPatternDemo {
   public static void main(String[] args) {
      //get shape factory
      AbstractFactory shapeFactory = FactoryProducer.getFactory(false);
      //get an object of Shape Rectangle
      Shape shape1 = shapeFactory.getShape("RECTANGLE");
      //call draw method of Shape Rectangle
      shape1.draw();
      //get an object of Shape Square
      Shape shape2 = shapeFactory.getShape("SQUARE");
      //call draw method of Shape Square
      shape2.draw();
      //get shape factory
      AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);
      //get an object of Shape Rectangle
      Shape shape3 = shapeFactory1.getShape("RECTANGLE");
      //call draw method of Shape Rectangle
      shape3.draw();
      //get an object of Shape Square
      Shape shape4 = shapeFactory1.getShape("SQUARE");
```

```
      //call draw method of Shape Square
      shape4.draw();

   }
}
--------------------------------------------------------------------------
-------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data=pd.read_csv("house.csv")
print(data)

x=data[["bedrooms","sqft_living"]]
y=data.price
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print(xtrain)
print(xtest)
print(ytrain)
print(ytest)

le=LinearRegression()
le.fit(xtrain,ytrain)
print(le.intercept_)
print(le.coef_)
print(le.predict([[3.5,1000]]))

ypred=le.predict(xtest)
me=mean_squared_error(ypred,ytest)
print(me)


--------------------------------------------------------------------------
----------------------------
const express = require('express');
const multer = require('multer');
const path = require('path');

const app = express();

// Set up multer for file uploads
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, 'uploads');
  },
  filename: (req, file, cb) => {
```

```
    cb(null, `${file.fieldname}-
${Date.now()}${path.extname(file.originalname)}`);
  }
});

const upload = multer({ storage });

// Serve static files
app.use(express.static('public'));

// Route for the file upload form
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

// Handle file uploads
app.post('/upload', upload.single('file'), (req, res) => {
  res.send(`<a href="/download/${req.file.filename}">Click here to
download the file</a>`);
});

// Route to handle downloads
app.get('/download/:filename', (req, res) => {
  const file = path.join(__dirname, 'uploads', req.params.filename);
  res.download(file);
});

// Start the server
const port = 8000;
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}`);
});

create a txt file and write any data into it
create a uploads folder then run the js file

==========================================slip
18==============================================
Write a JAVA Program to implement built-in support (java.util.Observable)
Weather
station with members temperature, humidity, pressure and methods
mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(),
getPressure()
Write a python program to implement Polynomial Linear Regression for
given dataset
Create your Django app in which after running the server, you should see
on the
browser, the text “Hello! I am learning Django”, which you defined in the
index view.

------------------------------------------------------------------------
-----------------------------------

Observer.java
```

```
public interface Observer {
 public void update(float temp, float humidity, float pressure);
}

Subject.java
public interface Subject {
 public void registerObserver(Observer o);
 public void removeObserver(Observer o);
public void notifyObservers();
}

WeatherData.java
public class WeatherData implements Subject {
 private ArrayList<Observer> observers;
 private float temperature;
 private float humidity;
 private float pressure;

 public WeatherData() {
  observers = new ArrayList<>();
 }

 public void registerObserver(Observer o) {
  observers.add(o);
 }

 public void removeObserver(Observer o) {
  int i = observers.indexOf(o);
  if (i >= 0) {
   observers.remove(i);
  }
 }

 public void notifyObservers() {
  for (int i = 0; i < observers.size(); i++) {
   Observer observer = (Observer)observers.get(i);
   observer.update(temperature, humidity, pressure);
  }
 }

 public void measurementsChanged() {
  notifyObservers();
 }

 public void setMeasurements(float temperature, float humidity, float
pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  this.pressure = pressure;
  measurementsChanged();
 }

 public float getTemperature() {
  return temperature;
```

```
 }

 public float getHumidity() {
  return humidity;
 }

 public float getPressure() {
  return pressure;
 }
}

ForecastDisplay.java
public class ForecastDisplay implements Observer, DisplayElement {
private float currentPressure = 29.92f;
private float lastPressure;
private WeatherData weatherData;

 public ForecastDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temp, float humidity, float pressure) {
                lastPressure = currentPressure;
  currentPressure = pressure;

  display();
 }

 public void display() {
  System.out.print("Forecast: ");
  if (currentPressure > lastPressure) {
   System.out.println("Improving weather on the way!");
  } else if (currentPressure == lastPressure) {
   System.out.println("More of the same");
  } else if (currentPressure < lastPressure) {
   System.out.println("Watch out for cooler, rainy weather");
  }
 }
}

HeatIndexDisplay.java
public class HeatIndexDisplay implements Observer, DisplayElement {
 float heatIndex = 0.0f;
 private WeatherData weatherData;

 public HeatIndexDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float t, float rh, float pressure) {
  heatIndex = computeHeatIndex(t, rh);
  display();
```

```
 }

 private float computeHeatIndex(float t, float rh) {
  float index = (float)((16.923 + (0.185212 * t) + (5.37941 * rh) -
(0.100254 * t * rh)
   + (0.00941695 * (t * t)) + (0.00728898 * (rh * rh))
   + (0.000345372 * (t * t * rh)) - (0.000814971 * (t * rh * rh)) +
   (0.0000102102 * (t * t * rh * rh)) - (0.000038646 * (t * t * t)) +
(0.0000291583 *
   (rh * rh * rh)) + (0.00000142721 * (t * t * t * rh)) +
   (0.000000197483 * (t * rh * rh * rh)) - (0.0000000218429 * (t * t * t
* rh * rh)) +
   0.000000000843296 * (t * t * rh * rh * rh)) -
   (0.0000000000481975 * (t * t * t * rh * rh * rh)));
  return index;
 }

 public void display() {
  System.out.println("Heat index is " + heatIndex);
 }
}

StatisticsDisplay.java
public class StatisticsDisplay implements Observer, DisplayElement {
 private float maxTemp = 0.0f;
 private float minTemp = 200;
 private float tempSum= 0.0f;
 private int numReadings;
 private WeatherData weatherData;

 public StatisticsDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temp, float humidity, float pressure) {
  tempSum += temp;
  numReadings++;

  if (temp > maxTemp) {
   maxTemp = temp;
  }

  if (temp < minTemp) {
   minTemp = temp;
  }

  display();
 }

 public void display() {
  System.out.println("Avg/Max/Min temperature = " + (tempSum /
numReadings)
   + "/" + maxTemp + "/" + minTemp);
```

```
 }
}

CurrentConditionsDisplay.java
public class CurrentConditionsDisplay implements Observer, DisplayElement
{
 private float temperature;
 private float humidity;
 private Subject weatherData;

 public CurrentConditionsDisplay(Subject weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temperature, float humidity, float pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  display();
 }

 public void display() {
  System.out.println("Current conditions: " + temperature
   + "F degrees and " + humidity + "% humidity");
 }
}

WeatherStation.java
public class WeatherStation {

 public static void main(String[] args) {
  WeatherData weatherData = new WeatherData();

  CurrentConditionsDisplay currentDisplay =
   new CurrentConditionsDisplay(weatherData);
  StatisticsDisplay statisticsDisplay = new
StatisticsDisplay(weatherData);
  ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

  weatherData.setMeasurements(80, 65, 30.4f);
  weatherData.setMeasurements(82, 70, 29.2f);
  weatherData.setMeasurements(78, 90, 29.2f);
 }
}

--------------------------------------------------------------------------
----------------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix
```

```
data=pd.read_csv("position_salaries.csv")
x=data.iloc[:,1:2].values
y=data.iloc[:,2].values
print(x)
print(y)
le=LinearRegression()
le.fit(x,y)

po=PolynomialFeatures(degree=4)
xp=po.fit_transform(x)
pe=LinearRegression()
pe.fit(xp,y)

plt.scatter(x,y,c="red")
plt.plot(x,le.predict(x),c="green")
plt.show()

plt.scatter(x,y,c="red")
plt.plot(x,pe.predict(po.fit_transform(x)),c="green")
plt.show()

print(le.predict([[3.5]]))
print(pe.predict(po.fit_transform([[3.5]])))
```

---

views.py

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def display(request):
    return HttpResponse("Hello")
```

urls.py

```
from django.contrib import admin
from django.urls import path
from myapp.views import display

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mypath/',display)
]
```

==========slip 19==========

Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza's like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.

Write a python program to implement Naive Bayes.
Design a Django application that adds web pages with views and templates.

PizzaStore.java

```
public abstract class PizzaStore{
abstract Pizza createPizza(String item);
public Pizza orderPizza(String type){
    Pizza pizza=createPizza(type);
System.out.println("---Making a"+pizza.getName()+"---");
pizza.prepare();
pizza.bake();
pizza.cut();
pizza.box();
return pizza;
}
}
```

ChicagoPizzaStore.java

```
public class ChicagoPizzaStore extends PizzaStore {

    Pizza createPizza(String item) {
            if (item.equals("cheese")) {
                    return new ChicagoStyleCheesePizza();
            } else if (item.equals("veggie")) {
                    return new ChicagoStyleVeggiePizza();
            } else if (item.equals("clam")) {
                    return new ChicagoStyleClamPizza();
            } else if (item.equals("pepperoni")) {
                    return new ChicagoStylePepperoniPizza();
            } else return null;
    }
}
```

NYPizzaStore.java

```
public class NYPizzaStore extends PizzaStore{
Pizza createPizza(String item){
if(item.equals("cheese")){
return new;
NYStyleCheesePizza();
}else if(item.equals("veggie")){
return new;
NYStyleCheesePizza();
}else if(item.equals("clam")){
return new;
NYStyleCheesePizza();
}else if(item.equals("pepperoni")){
return new;
NYStyleCheesePizza();
}else return null;
}
}
```

Pizza.java

```
import java.util.ArrayList;

public abstract class Pizza {
    String name;
    String dough;
    String sauce;
    ArrayList<String> toppings = new ArrayList<String>();

    void prepare() {
        System.out.println("Prepare " + name);
        System.out.println("Tossing dough...");
        System.out.println("Adding sauce...");
        System.out.println("Adding toppings: ");
        for (String topping : toppings) {
            System.out.println("   " + topping);
        }
    }

    void bake() {
        System.out.println("Bake for 25 minutes at 350");
    }

    void cut() {
        System.out.println("Cut the pizza into diagonal slices");
    }

    void box() {
        System.out.println("Place pizza in official PizzaStore box");
    }

    public String getName() {
        return name;
    }

    public String toString() {
        StringBuffer display = new StringBuffer();
        display.append("---- " + name + " ----\n");
        display.append(dough + "\n");
        display.append(sauce + "\n");
        for (String topping : toppings) {
            display.append(topping + "\n");
        }
        return display.toString();
    }
}
```

NYStyleCheesePizza.java

```
public class NYStyleCheesePizza extends Pizza {

    public NYStyleCheesePizza() {
        name = "NY Style Sauce and Cheese Pizza";
```

```
        dough = "Thin Crust Dough";
        sauce = "Marinara Sauce";

        toppings.add("Grated Reggiano Cheese");
    }
}
```

NYStyleClamPizza.java

```
public class NYStyleClamPizza extends Pizza {

    public NYStyleClamPizza() {
        name = "NY Style Clam Pizza";
        dough = "Thin Crust Dough";
        sauce = "Marinara Sauce";

        toppings.add("Grated Reggiano Cheese");
        toppings.add("Fresh Clams from Long Island Sound");
    }
}
```

ChicagoStylePepperoniPizza.java

```
public class ChicagoStylePepperoniPizza extends Pizza {
    public ChicagoStylePepperoniPizza() {
        name = "Chicago Style Pepperoni Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Spinach");
        toppings.add("Eggplant");
        toppings.add("Sliced Pepperoni");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}
```

ChicagoStyleVeggiePizza.java

```
public class ChicagoStyleVeggiePizza extends Pizza {
    public ChicagoStyleVeggiePizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Spinach");
        toppings.add("Eggplant");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
```

```
    }
}

ChicagoStyleClamPizza.java
public class ChicagoStyleClamPizza extends Pizza {
    public ChicagoStyleClamPizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Clams");
        toppings.add("Jalapeons");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}

ChicagoStyleCheesePizza.java
public class ChicagoStyleCheesePizza extends Pizza {
    public ChicagoStyleCheesePizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Mayo");
        toppings.add("Cheddar");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}

PizzaTestDrive.java
public class PizzaTestDrive {

    public static void main(String[] args) {
        PizzaStore nyStore = new NYPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        Pizza pizza = nyStore.orderPizza("cheese");
        System.out.println("First order was a " + pizza.getName() +
"\n");

        pizza = nyStore.orderPizza("cheese");
        System.out.println("Second order was a " + pizza.getName() +
"\n");
    }
```

```
}

--------------------------------------------------------------------------
-------------------------------------------------------------

import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

data=pd.read_csv("user_data1.csv")
x=data.iloc[:,[2,4]].values
y=data.iloc[:,4].values
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
gd=GaussianNB()
gd.fit(xtrain,ytrain)

ypred=gd.predict(xtest)
cm=confusion_matrix(ypred,ytest)
print(cm)

--------------------------------------------------------------------------
---------------------------------

urls.py
from django.contrib import admin
from django.urls import path
from myapp.views import f
urlpatterns = [
    path('admin/', admin.site.urls),
    path('mypath/',f),
]

views.py
from django.shortcuts import render

# Create your views here.
def f(request):
    return render(request, './home.html')

create templates folder in myapp and in that folder create home.html and
paste below code<!DOCTYPE html>
<html>
<head>
    <title>Home Page</title>
</head>
<body>
    <h1>Welcome to the Home Page</h1>
</body>
</html>
```

now in setting.py under installed apps add this 'myapp',

==========slip 20==========

Write a Java Program to implement I/O Decorator for converting uppercase letters to
lower case letters.
Write a python program to implement Decision Tree whether or not to play Tennis.
Develop a basic poll application (app).It should consist of two parts:
a) A public site in which user can pick their favourite programming language and vote.
b) An admin site that lets you add, change and delete programming languages

---

LowerCaseInputStream.java

```
import java.io.*;
import java.util.*;
//extend FilterInputStream which is abstract decorator for all
InputStreams
class LowerCaseInputStream extends FilterInputStream
{
      public LowerCaseInputStream(InputStream in)
      {
            super(in);
      }
      public int read() throws IOException
      {
            int c=super.read();
            return (c==-1?c:Character.toLowerCase((char)c));
      }
      public int read(byte[] b,int offset,int len) throws IOException
      {
            int result =super.read(b,offset,len);
            for (int i=offset;i<offset+result;i++)
            {
                  b[i]=(byte)Character.toLowerCase((char)b[i]);
            }
            return result;
      }
}
```

InputTest.java

```
class InputTest
{
      public static void main(String[] args) throws IOException
      {
      int c;
      try
```

```
      {
            //set up the FileInputStream and decorate it,first with a
BufferedInputStream and then our
      //new LowerCaseInputStream filter
            InputStream in =new LowerCaseInputStream(new
BufferedInputStream(new FileInputStream("a.txt")));
            // Read the charater until end of file and print it
            while((c = in.read()) >= 0)
            {
                  System.out.print((char)c);
            }

            in.close();
      }
      catch(IOException e)
      {
            e.printStackTrace();
      }
      }
}
--------------------------------------------------------------------------
-----------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
data=pd.read_csv("tennis.csv")
print(data)

le=LabelEncoder()

data["Outlook"]=le.fit_transform(data["Outlook"])
data["Temprature"]=le.fit_transform(data["Temprature"])
data["Humidity"]=le.fit_transform(data["Humidity"])
data["Wind"]=le.fit_transform(data["Wind"])
data["Play_Tennis"]=le.fit_transform(data["Play_Tennis"])
print(data)


x=data.iloc[:,1:5].values
y=data["Play_Tennis"]

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

dc=DecisionTreeClassifier(criterion="entropy")
dc.fit(xtrain,ytrain)

from sklearn.tree import export_graphviz
export_graphviz(dc,out_file="a1.txt")

ypred=dc.predict(xtest)
```

```
cm=confusion_matrix(ypred,ytest)
print(cm)
--------------------------------------------------------------------------
-----------------------------------------------------
models.py
from django.db import models
class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.name

views.py
      from django.shortcuts import render, redirect
from .models import ProgrammingLanguage

def poll(request):
    languages = ProgrammingLanguage.objects.all()
    context = {'languages': languages}
    return render(request, 'poll_app/poll.html', context)

def vote(request, language_id):
language = ProgrammingLanguage.objects.get(pk=language_id)
    language.votes += 1
    language.save()
    return redirect('myapp:poll')

myapp/urls.py
      from django.urls import path, re_path
from . import views

app_name = 'myapp'

urlpatterns = [
    path('', views.poll, name='poll'),
    re_path(r'^vote/(?P<language_id>\d+)/$', views.vote, name='vote'),
]

myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('poll/', include('myapp.urls', namespace='myapp')),
]

admin.py
from django.contrib import admin
from .models import ProgrammingLanguage

admin.site.register(ProgrammingLanguage)
```

poll.html

```
      <!DOCTYPE html>
<html>
  <head>
    <title>Programming Language Poll</title>
  </head>
  <body>
    <h1>Vote for Your Favorite Programming Language</h1>
    <ul>
      {% for language in languages %}
      <form method="post" action="{% url 'myapp:vote' language.id %}">
        {% csrf_token %}
        <li>
          {{ language.name }}
          <button type="submit" name="vote" value="{{ language.id }}">
            Vote
          </button>
          ({{ language.votes }} votes)
        </li>
      </form>
      {% endfor %}
    </ul>
  </body>
</html>
```

==========slip 21==========

Write a Java Program to implement command pattern to test Remote Control
Write a python program to implement Linear SVM.
Design a Django application: A public site in which user can pick their favourite
programming language and vote.

---

remote control

Command.java

```
package javaprograms;
interface Command
{
public void execute();
}
```

2)Create Class
Light.java

```
package javaprograms;
public class Light
{
public void on()
{
System.out.println("Light is on");
}
public void off()
{
```

```
System.out.println("Light is off");
}
}LightOnCommand.java
package javaprograms;
class LightOnCommand implements Command
{
Light light;
// The constructor is passed the light it
// is going to control.
public LightOnCommand(Light light)
{
this.light = light;
}
public void execute()
{
light.on();
}
}
LightOffCommand.java
package javaprograms;
class LightOffCommand implements Command
{
Light light;
public LightOffCommand(Light light)
{
this.light = light;
}
public void execute()
{
light.off();
}
}
Stereo.java
package javaprograms;
public class Stereo
{
public void on()
{
System.out.println("Stereo
}
public void off()
{
System.out.println("Stereo
}
public void setCD()
{
System.out.println("Stereo
"for CD
}
public void setDVD()
{
is on");
is off");
is set " +
```

```
input");System.out.println("Stereo is set"+
" for DVD input");
}
public void setRadio()
{
System.out.println("Stereo is set" +
" for Radio");
}
public void setVolume(int volume)
{
// code to set the volume
System.out.println("Stereo volume set"
+ " to " + volume);
}
}
```

StereoOffCommand.java

```
package javaprograms;
class StereoOffCommand implements Command
{
Stereo stereo;
public StereoOffCommand(Stereo stereo)
{
this.stereo = stereo;
}
public void execute()
{
stereo.off();
}
}
```

StereoOnWithCDCommand.java

```
package javaprograms;
class StereoOnWithCDCommand implements Command
{
Stereo stereo;
public StereoOnWithCDCommand(Stereo stereo)
{
this.stereo = stereo;
}
public void execute()
{
stereo.on();
stereo.setCD();
stereo.setVolume(11);
}
}
```

SimpleRemoteControl.java

```
package javaprograms;
class SimpleRemoteControl
{Command slot;
// only one button
public SimpleRemoteControl()
{
}
public void setCommand(Command command)
```

```
{
// set the command the remote will
// execute
slot = command;
}
public void buttonWasPressed()
{
slot.execute();
}
}
RemoteControlTest.java
package javaprograms;
class RemoteControlTest
{
public static void main(String[] args)
{
SimpleRemoteControl remote =
new SimpleRemoteControl();
Light light = new Light();
Stereo stereo = new Stereo();
// we can change command dynamically
remote.setCommand(new
LightOnCommand(light));
remote.buttonWasPressed();
remote.setCommand(new
StereoOnWithCDCommand(stereo));
remote.buttonWasPressed();
remote.setCommand(new
StereoOffCommand(stereo));
remote.buttonWasPressed();
}
}

--------------------------------------------------------------------------------
---------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

data=pd.read_csv("user_data1.csv")
print(data)

x=data.iloc[:,2:3].values
y=data.iloc[:,3].values
print(x)
print(y)
sr=SVR()
sr.fit(x,y)

print(sr.predict([[150]]))

plt.scatter(x,y,c="red")
plt.plot(x,sr.predict(x),c="green")
```

```
plt.show()

--------------------------------------------------------------------------------
-------------------------------------

models.py
from django.db import models
class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)

    def __str__(self):
        return self.name

views.py
      from django.shortcuts import render, redirect
from .models import ProgrammingLanguage

def poll(request):
    languages = ProgrammingLanguage.objects.all()
    context = {'languages': languages}
    return render(request, 'poll_app/poll.html', context)

def vote(request, language_id):
language = ProgrammingLanguage.objects.get(pk=language_id)
    language.votes += 1
    language.save()
    return redirect('myapp:poll')

myapp/urls.py
      from django.urls import path, re_path
from . import views

app_name = 'myapp'

urlpatterns = [
    path('', views.poll, name='poll'),
    re_path(r'^vote/(?P<language_id>\d+)/$', views.vote, name='vote'),
]

myproject/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('poll/', include('myapp.urls', namespace='myapp')),
]
poll.html
      <!DOCTYPE html>
<html>
  <head>
    <title>Programming Language Poll</title>
  </head>
```

```
  <body>
    <h1>Vote for Your Favorite Programming Language</h1>
    <ul>
      {% for language in languages %}
      <form method="post" action="{% url 'myapp:vote' language.id %}">
        {% csrf_token %}
        <li>
          {{ language.name }}
          <button type="submit" name="vote" value="{{ language.id }}">
            Vote
          </button>
          ({{ language.votes }} votes)
        </li>
      </form>
      {% endfor %}
    </ul>
  </body>
</html>
```

==========slip 22==========

Design simple HR Application using Spring Framework

Write a Python program to prepare Scatter Plot for Iris Dataset

Design a Django application: An admin site that lets you add, change and delete
programming languages.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data=pd.read_csv("iris.csv")
print(data)
x=data["SepalLengthCm"]
y=data["SepalWidthCm"]
print(x)
print(y)
plt.scatter(x, y,c="green")
plt.xlabel("SepalLengthCm")
plt.ylabel("SepalWidthCm")
plt.legend()
plt.show()
```

```
models.py
from django.db import models
class ProgrammingLanguage(models.Model):
    name = models.CharField(max_length=200)
    votes = models.IntegerField(default=0)
```

```
    def __str__(self):
        return self.name
admin.py
      from django.contrib import admin
from .models import ProgrammingLanguage

admin.site.register(ProgrammingLanguage)
myproject/urls.py
      from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

==========slip 23==========

Write a Java Program to implement State Pattern for Gumball Machine. Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen

Write a python program to find all null values in a given dataset and remove them.

Create your own blog using Django.

```
State.java
public interface State {

      public void insertQuarter();
      public void ejectQuarter();
      public void turnCrank();
      public void dispense();

      public void refill();
}

NoQuarterState.java
public class NoQuarterState implements State {
    GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("You inserted a quarter");
            gumballMachine.setState(gumballMachine.getHasQuarterState());
      }
```

```
      public void ejectQuarter() {
            System.out.println("You haven't inserted a quarter");
      }

      public void turnCrank() {
            System.out.println("You turned, but there's no quarter");
       }

      public void dispense() {
            System.out.println("You need to pay first");
      }

      public void refill() { }

      public String toString() {
            return "waiting for quarter";
      }
}
```

HasQuarterState.java

```
public class HasQuarterState implements State {
      GumballMachine gumballMachine;

      public HasQuarterState(GumballMachine gumballMachine) {
            this.gumballMachine = gumballMachine;
      }

      public void insertQuarter() {
            System.out.println("You can't insert another quarter");
      }

      public void ejectQuarter() {
            System.out.println("Quarter returned");
            gumballMachine.setState(gumballMachine.getNoQuarterState());
      }

      public void turnCrank() {
            System.out.println("You turned...");
            gumballMachine.setState(gumballMachine.getSoldState());
      }

    public void dispense() {
        System.out.println("No gumball dispensed");
    }

    public void refill() { }

      public String toString() {
            return "waiting for turn of crank";
      }
}
```

SoldOutState.java

```
public class SoldOutState implements State {
```

```
    GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("You can't insert a quarter, the machine
is sold out");
      }

      public void ejectQuarter() {
            System.out.println("You can't eject, you haven't inserted a
quarter yet");
      }

      public void turnCrank() {
            System.out.println("You turned, but there are no gumballs");
      }

      public void dispense() {
            System.out.println("No gumball dispensed");
      }

      public void refill() {
            gumballMachine.setState(gumballMachine.getNoQuarterState());
      }

      public String toString() {
            return "sold out";
      }
}
```

SoldState.java

```
public class SoldState implements State {

    GumballMachine gumballMachine;

    public SoldState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("Please wait, we're already giving you a
gumball");
      }

      public void ejectQuarter() {
            System.out.println("Sorry, you already turned the crank");
      }

      public void turnCrank() {
            System.out.println("Turning twice doesn't get you another
gumball!");
```

```
	}

	public void dispense() {
		gumballMachine.releaseBall();
		if (gumballMachine.getCount() > 0) {

	gumballMachine.setState(gumballMachine.getNoQuarterState());
		} else {
			System.out.println("Oops, out of gumballs!");

	gumballMachine.setState(gumballMachine.getSoldOutState());
		}
	}

	public void refill() { }

	public String toString() {
		return "dispensing a gumball";
	}
}

GumballMachine.java
public class GumballMachine {

	State soldOutState;
	State noQuarterState;
	State hasQuarterState;
	State soldState;

	State state;
	int count = 0;

	public GumballMachine(int numberGumballs) {
		soldOutState = new SoldOutState(this);
		noQuarterState = new NoQuarterState(this);
		hasQuarterState = new HasQuarterState(this);
		soldState = new SoldState(this);

		this.count = numberGumballs;
		if (numberGumballs > 0) {
			state = noQuarterState;
		} else {
			state = soldOutState;
		}
	}

	public void insertQuarter() {
		state.insertQuarter();
	}

	public void ejectQuarter() {
		state.ejectQuarter();
	}
```

```
      public void turnCrank() {
            state.turnCrank();
            state.dispense();
      }

      void releaseBall() {
            System.out.println("A gumball comes rolling out the
slot...");
            if (count != 0) {
                  count = count - 1;
            }
      }

      int getCount() {
            return count;
      }

      void refill(int count) {
            this.count += count;
            System.out.println("The gumball machine was just refilled;
it's new count is: " + this.count);
            state.refill();
      }

      void setState(State state) {
            this.state = state;
      }
    public State getState() {
        return state;
    }

    public State getSoldOutState() {
        return soldOutState;
    }

    public State getNoQuarterState() {
        return noQuarterState;
    }

    public State getHasQuarterState() {
        return hasQuarterState;
    }

    public State getSoldState() {
        return soldState;
    }

      public String toString() {
            StringBuffer result = new StringBuffer();
            result.append("\nMighty Gumball, Inc.");
            result.append("\nJava-enabled Standing Gumball Model #2004");
            result.append("\nInventory: " + count + " gumball");
            if (count != 1) {
                  result.append("s");
```

```
            }
            result.append("\n");
            result.append("Machine is " + state + "\n");
            return result.toString();
      }
}
```

GumballMachineTestDrive.java

```
public class GumballMachineTestDrive {

      public static void main(String[] args) {
            GumballMachine gumballMachine = new GumballMachine(2);

            System.out.println(gumballMachine);

            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            System.out.println(gumballMachine);

            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();
            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            gumballMachine.refill(5);
            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            System.out.println(gumballMachine);
      }
}
```

```
import pandas as pd
import numpy as np
data=pd.read_csv("ass2.csv")
print(data)
print(data.isnull())
print(data.notnull())

data1=data.dropna(axis=1,how="all")
print(data1)
data["name"]=data["name"].replace(np.nan,"xyz")
data["m1"]=data["m1"].replace(np.nan,99)
data["m2"]=data["m2"].replace(np.nan, data["m2"].mean())
print(data)
```

---

models.py

```
      from django.db import models
class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
```

```
    pub_date = models.DateTimeField('date published')
```

views.py

```
      from django.shortcuts import render
from .models import Post
def post_list(request):
    posts = Post.objects.all()
    return render(request, 'blog/post_list.html', {'posts': posts})

def post_detail(request, pk):
    post = Post.objects.get(pk=pk)
    return render(request, 'blog/post_details.html', {'post': post})
```

post_details.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>{{ post.title }}</title>
  </head>
  <body>
    <h1>{{ post.title }}</h1>
    <p>{{ post.pub_date|date:"F d, Y" }}</p>
    <p>{{ post.content|linebreaks }}</p>
    <a href="{% url 'post_list' %}">Back to post list</a>
  </body>
</html>
```

post_list.html

```
      <!DOCTYPE html>
<html>
  <head>
    <title>Blog Posts</title>
  </head>
  <body>
    <h1>Blog Posts</h1>
    <ul>
      {% for post in posts %}
      <li>
        <a href="{% url 'post_detail' pk=post.pk %}">{{ post.title }}</a>
      </li>
      {% endfor %}
    </ul>
  </body>
</html>
```

blog/urls.py

```
      from django.urls import path
from . import views

urlpatterns = [
    path('', views.post_list, name='post_list'),
    path('post/<int:pk>/', views.post_detail, name='post_detail'),
]
```

myblog/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')) ]
admin.py
      from django.contrib import admin
from .models import Post
admin.site.register(Post)
```

==========slip 24==========

Write a Java Program to implement Iterator Pattern for Designing Menu like Breakfast, Lunch or Dinner Menu

Write a python program to make Categorical values in numeric format for a given dataset

Implement Login System using Django.

```
import java.util.ArrayList;
import java.util.Iterator;

// Menu Item class
class MenuItem {
    private String name;
    private String description;

    public MenuItem(String name, String description) {
        this.name = name;
        this.description = description;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }
}

// Menu Interface
interface Menu {
    Iterator<MenuItem> createIterator();
}

// Breakfast Menu
class BreakfastMenu implements Menu {
    private ArrayList<MenuItem> menuItems;
```

```
    public BreakfastMenu() {
        menuItems = new ArrayList<>();
        addItem("Eggs and Toast", "Scrambled eggs with toast");
        addItem("Pancakes", "Fluffy pancakes with syrup");
    }

    public void addItem(String name, String description) {
        MenuItem menuItem = new MenuItem(name, description);
        menuItems.add(menuItem);
    }

    @Override
    public Iterator<MenuItem> createIterator() {
        return menuItems.iterator();
    }
}

// Lunch Menu
class LunchMenu implements Menu {
    private MenuItem[] menuItems;
    private int numberOfItems = 0;

    public LunchMenu() {
        menuItems = new MenuItem[5];
        addItem("Chicken Sandwich", "Grilled chicken sandwich");
        addItem("Caesar Salad", "Fresh Caesar salad with chicken");
    }

    public void addItem(String name, String description) {
        MenuItem menuItem = new MenuItem(name, description);
        if (numberOfItems < menuItems.length) {
            menuItems[numberOfItems] = menuItem;
            numberOfItems++;
        }
    }

    @Override
    public Iterator<MenuItem> createIterator() {
        return new LunchMenuIterator(menuItems);
    }
}

// Iterator for Lunch Menu
class LunchMenuIterator implements Iterator<MenuItem> {
    private MenuItem[] items;
    private int position = 0;

    public LunchMenuIterator(MenuItem[] items) {
        this.items = items;
    }

    @Override
    public boolean hasNext() {
        return position < items.length && items[position] != null;
```

```
    }

    @Override
    public MenuItem next() {
        MenuItem menuItem = items[position];
        position++;
        return menuItem;
    }
}

// Client Code
public class MenuClient {
    public static void printMenu(Menu menu) {
        Iterator<MenuItem> iterator = menu.createIterator();
        System.out.println("Menu:");
        while (iterator.hasNext()) {
            MenuItem menuItem = iterator.next();
            System.out.println(menuItem.getName() + ": " +
menuItem.getDescription());
        }
        System.out.println();
    }

    public static void main(String[] args) {
        BreakfastMenu breakfastMenu = new BreakfastMenu();
        LunchMenu lunchMenu = new LunchMenu();

        printMenu(breakfastMenu);
        printMenu(lunchMenu);
    }
}
```

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

data=pd.read_csv("ass3.csv")
print(data)
x=data.iloc[:,0:1].values
print(x)

le=LabelEncoder()
x1=le.fit_transform(x)
print(x1)

oe=OneHotEncoder()
x2=oe.fit_transform(x).toarray()
print(x2)
```

---

settings.py (add following):

```
      AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
)
LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = 'home'
views.py
      from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect

def user_login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username,
password=password)
        if user is not None:
            login(request, user)
            return redirect('welcome')
    return render(request, 'registration/login.html')

def welcome(request):
    return render(request, 'registration/home.html')

login.html
      <!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <h1>Login</h1>
    <form method="post">
      {% csrf_token %}
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" required />
      <br />
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required />
      <br />
      <button type="submit">Login</button>
    </form>
  </body>
</html>
home.html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Document</title>
  </head>
  <body>
    Welcome
```

```
  </body>
</html>

myapp/urls.py
      from django.urls import path
from . import views

urlpatterns = [
    path('login/', views.user_login, name='login'),
    path('welcome/', views.welcome, name='welcome')
]

myproject/urls.py
      from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]

==========slip
25==========
Write a Java Program to implement Singleton pattern for multithreading
Write a python program to Implement Simple Linear Regression for
predicting house
price.
Create a Simple Web Server using node js.
--------------------------------------------------------------------------
------------------------


Singleton.java
public class Singleton{
      private static Singleton uniqueInstance;

      private Singleton() {

            System.out.println("Instance has been Created");
      }

      public static Singleton getInstance() {
            if(uniqueInstance== null) {
                  synchronized (Singleton.class) {

                        if(uniqueInstance == null) {
                              uniqueInstance=new Singleton();
                        }
                  }

            }
            return uniqueInstance;
      }
```

```
	public static void main(String[] args) {

		Thread t1= new Thread(new Runnable() {
			public void run() {
				Singleton obj=Singleton.getInstance();
			}
		});

		Thread t2=new Thread(new Runnable() {
			public void run() {
				Singleton obj=Singleton.getInstance();
			}
		});

		t1.start();
		t2.start();
	}
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data=pd.read_csv("house.csv")
print(data)

x=data[["bedrooms"]]
y=data.price
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print(xtrain)
print(xtest)
print(ytrain)
print(ytest)

le=LinearRegression()
le.fit(xtrain,ytrain)
print(le.intercept_)
print(le.coef_)
print(le.predict([[3.5]]))

ypred=le.predict(xtest)
me=mean_squared_error(ypred,ytest)
print(me)

plt.scatter(x,y,c="red")
plt.plot(xtest,le.predict(xtest),c="green")
plt.show()
```

```
--------------------------------------------------------------------------------
----------------------------------
const express = require('express');
const app = express();
const port = 3000;

// Define a route
app.get('/', (req, res) => {
    res.send('Hello World!');
});

// Start the server
app.listen(port, () => {
    console.log(`Server is running on http://localhost:${port}`);
});
============================================================slip
26================================================================
Write a Java Program to implement Strategy Pattern for Duck Behavior.
Create
instance variable that holds current state of Duck from there, we just
need to handle all
Flying Behaviors and Quack Behavior.
Write a python program to implement Multiple Linear Regression for given
dataset.
Create a Node.js file that demonstrates create database and table in
MySQL.
--------------------------------------------------------------------------------
-------------------------------------------------

1) Create Interface
QuackBehaviour.java
package javaprograms;
public interface QuackBehaviour {
public default void quack() {
System.out.println("Quack");
}
}
FlyBehaviour.java
package javaprograms;
public interface FlyBehaviour {
public void fly();
}
2)Create Class -
FlyWithWings.java
package javaprograms;public class FlyWithWings implements FlyBehaviour {
public void fly() {
System.out.println("I'm flying!!");
}
}
Quack.java
package javaprograms;
public class Quack implements QuackBehaviour {
```

```
public void quack() {
System.out.println("Quack");
}
}
```
ModolDuck.java
```
package javaprograms;
public class ModolDuck extends Duck {
public ModolDuck() {
flyBehaviour = new FlyNoWay();
quackBehaviour = new Quack();
}
public void display() {
System.out.println("I'm a model duck");
}
}
```
MallardDuck.java
```
package javaprograms;
public class MallardDuck extends Duck {
public MallardDuck() {
quackBehaviour = new Quack();
flyBehaviour = new FlyWithWings();
}
public void display() {
System.out.println("I'm a real Mallard duck");
}
}
```
Duck.java
```
package javaprograms;
public class MallardDuck extends Duck {
public MallardDuck() {
quackBehaviour = new Quack();
flyBehaviour = new FlyWithWings();
}
public void display() {
System.out.println("I'm a real Mallard duck");}
}
```
FlyRocketPowered.java
```
package javaprograms;
public class FlyRocketPowered implements FlyBehaviour {
public void fly() {
System.out.println("I'm flying with a rocket!");
}
}
```
FlyNoWay.java
```
package javaprograms;
public class FlyNoWay implements FlyBehaviour {
public void fly() {
System.out.println("I can't fly");
}
}
```
MiniDuckSimulator.java
```
package javaprograms;
public class MiniDuckSimulator {
public static void main(String[] args) {
```

```
Duck mallard = new MallardDuck();
mallard.performQuack();
mallard.performFly();
Duck model = new ModolDuck();
model.performFly();
model.setFlyBehaviour(new FlyRocketPowered());
model.performFly();
}
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

data=pd.read_csv("house.csv")
print(data)

x=data[["bedrooms","sqft_living"]]
y=data.price
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
print(xtrain)
print(xtest)
print(ytrain)
print(ytest)

le=LinearRegression()
le.fit(xtrain,ytrain)
print(le.intercept_)
print(le.coef_)
print(le.predict([[3.5,1000]]))

ypred=le.predict(xtest)
me=mean_squared_error(ypred,ytest)
print(me)
```

```
const mysql = require("mysql");
const mycon = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "",
  database: "test",
});
mycon.connect((err) => {
  if (err) {
```

```
    console.log(err);
  } else {
    console.log("success");
    mycon.query("create database newdb", function (err, result) {
      if (err) console.log(err);
      console.log("Database Created");
    });

    mycon.query("use newdb", function (err) {
      if (err) console.log(err);
      else console.log("Database { " + "newdb } selected!");
    });

    var sql = "create table customer(name varchar(10), address
varchar(10))";
    mycon.query(sql, function (err, result) {
      if (err) console.log(err);
      else console.log("Table Created");
    });
  }
});
```

==========slip 27==========

Write a Java Program to implement Abstract Factory Pattern for Shape interface.
Write a python program to implement Polynomial Linear Regression for given dataset
Create your Django app in which after running the server, you should see on the
browser, the text “Hello! I am learning Django”, which you defined in the index view

---

Shape.java

```
public interface Shape {
   void draw();
}
public class RoundedRectangle implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside RoundedRectangle::draw() method.");
   }
}
```

RoundedSquare.java

```
public class RoundedSquare implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside RoundedSquare::draw() method.");
   }
}
```

Rectangle.java

```
public class Rectangle implements Shape {
   @Override
   public void draw() {
      System.out.println("Inside Rectangle::draw() method.");
   }
}
```

Step 3

Create an Abstract class to get factories for Normal and Rounded Shape Objects.

AbstractFactory.java

```
public abstract class AbstractFactory {
   abstract Shape getShape(String shapeType) ;
}
```

Step 4

Create Factory classes extending AbstractFactory to generate object of concrete class based on given information.

ShapeFactory.java

```
public class ShapeFactory extends AbstractFactory {
   @Override
   public Shape getShape(String shapeType){
      if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new Rectangle();
      }else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new Square();
      }
      return null;
   }
}
```

RoundedShapeFactory.java

```
public class RoundedShapeFactory extends AbstractFactory {
   @Override
   public Shape getShape(String shapeType){
      if(shapeType.equalsIgnoreCase("RECTANGLE")){
         return new RoundedRectangle();
      }else if(shapeType.equalsIgnoreCase("SQUARE")){
         return new RoundedSquare();
      }
      return null;
   }
}
```

Step 5

Create a Factory generator/producer class to get factories by passing an information such as Shape

FactoryProducer.java

```
public class FactoryProducer {
```

```
   public static AbstractFactory getFactory(boolean rounded){
      if(rounded){
         return new RoundedShapeFactory();
      }else{
         return new ShapeFactory();
      }
   }
}
```

Step 6

Use the FactoryProducer to get AbstractFactory in order to get factories of concrete classes by passing an information such as type.

AbstractFactoryPatternDemo.java

```
public class AbstractFactoryPatternDemo {
   public static void main(String[] args) {
      //get shape factory
      AbstractFactory shapeFactory = FactoryProducer.getFactory(false);
      //get an object of Shape Rectangle
      Shape shape1 = shapeFactory.getShape("RECTANGLE");
      //call draw method of Shape Rectangle
      shape1.draw();
      //get an object of Shape Square
      Shape shape2 = shapeFactory.getShape("SQUARE");
      //call draw method of Shape Square
      shape2.draw();
      //get shape factory
      AbstractFactory shapeFactory1 = FactoryProducer.getFactory(true);
      //get an object of Shape Rectangle
      Shape shape3 = shapeFactory1.getShape("RECTANGLE");
      //call draw method of Shape Rectangle
      shape3.draw();
      //get an object of Shape Square
      Shape shape4 = shapeFactory1.getShape("SQUARE");
      //call draw method of Shape Square
      shape4.draw();

   }
}
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import confusion_matrix

data=pd.read_csv("position_salaries.csv")
x=data.iloc[:,1:2].values
y=data.iloc[:,2].values
print(x)
print(y)
```

```
le=LinearRegression()
le.fit(x,y)

po=PolynomialFeatures(degree=4)
xp=po.fit_transform(x)
pe=LinearRegression()
pe.fit(xp,y)

plt.scatter(x,y,c="red")
plt.plot(x,le.predict(x),c="green")
plt.show()

plt.scatter(x,y,c="red")
plt.plot(x,pe.predict(po.fit_transform(x)),c="green")
plt.show()

print(le.predict([[3.5]]))
print(pe.predict(po.fit_transform([[3.5]])))
```

---

views.py

```
from django.shortcuts import render

# Create your views here.
from django.http import HttpResponse

def display(request):
    return HttpResponse("Hello")
```

urls.py

```
from django.contrib import admin
from django.urls import path
from myapp.views import display

urlpatterns = [
    path('admin/', admin.site.urls),
    path('mypath/',display)
]
```

==========slip 28==========

Write a JAVA Program to implement built-in support (java.util.Observable) Weather
station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(), getPressure()
Write a python program to implement Naive Bayes.
Create your own blog using Django

Observer.java

```
public interface Observer {
 public void update(float temp, float humidity, float pressure);
}
```

Subject.java

```
public interface Subject {
 public void registerObserver(Observer o);
 public void removeObserver(Observer o);
public void notifyObservers();
}
```

WeatherData.java

```
public class WeatherData implements Subject {
 private ArrayList<Observer> observers;
 private float temperature;
 private float humidity;
 private float pressure;

 public WeatherData() {
  observers = new ArrayList<>();
 }

 public void registerObserver(Observer o) {
  observers.add(o);
 }

 public void removeObserver(Observer o) {
  int i = observers.indexOf(o);
  if (i >= 0) {
   observers.remove(i);
  }
 }

 public void notifyObservers() {
  for (int i = 0; i < observers.size(); i++) {
   Observer observer = (Observer)observers.get(i);
   observer.update(temperature, humidity, pressure);
  }
 }

 public void measurementsChanged() {
  notifyObservers();
 }

 public void setMeasurements(float temperature, float humidity, float
pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  this.pressure = pressure;
  measurementsChanged();
 }
```

```
 public float getTemperature() {
  return temperature;
 }

 public float getHumidity() {
  return humidity;
 }

 public float getPressure() {
  return pressure;
 }
}
```

ForecastDisplay.java

```
public class ForecastDisplay implements Observer, DisplayElement {
private float currentPressure = 29.92f;
private float lastPressure;
private WeatherData weatherData;

 public ForecastDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temp, float humidity, float pressure) {
                lastPressure = currentPressure;
  currentPressure = pressure;

  display();
 }

 public void display() {
  System.out.print("Forecast: ");
  if (currentPressure > lastPressure) {
   System.out.println("Improving weather on the way!");
  } else if (currentPressure == lastPressure) {
   System.out.println("More of the same");
  } else if (currentPressure < lastPressure) {
   System.out.println("Watch out for cooler, rainy weather");
  }
 }
}
```

HeatIndexDisplay.java

```
public class HeatIndexDisplay implements Observer, DisplayElement {
 float heatIndex = 0.0f;
 private WeatherData weatherData;

 public HeatIndexDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float t, float rh, float pressure) {
```

```
  heatIndex = computeHeatIndex(t, rh);
  display();
 }

 private float computeHeatIndex(float t, float rh) {
  float index = (float)((16.923 + (0.185212 * t) + (5.37941 * rh) -
(0.100254 * t * rh)
   + (0.00941695 * (t * t)) + (0.00728898 * (rh * rh))
   + (0.000345372 * (t * t * rh)) - (0.000814971 * (t * rh * rh)) +
   (0.0000102102 * (t * t * rh * rh)) - (0.000038646 * (t * t * t)) +
(0.0000291583 *
   (rh * rh * rh)) + (0.00000142721 * (t * t * t * rh)) +
   (0.000000197483 * (t * rh * rh * rh)) - (0.0000000218429 * (t * t * t
* rh * rh)) +
   0.000000000843296 * (t * t * rh * rh * rh)) -
   (0.0000000000481975 * (t * t * t * rh * rh * rh)));
  return index;
 }

 public void display() {
  System.out.println("Heat index is " + heatIndex);
 }
}

StatisticsDisplay.java
public class StatisticsDisplay implements Observer, DisplayElement {
 private float maxTemp = 0.0f;
 private float minTemp = 200;
 private float tempSum= 0.0f;
 private int numReadings;
 private WeatherData weatherData;

 public StatisticsDisplay(WeatherData weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temp, float humidity, float pressure) {
  tempSum += temp;
  numReadings++;

  if (temp > maxTemp) {
   maxTemp = temp;
  }

  if (temp < minTemp) {
   minTemp = temp;
  }

  display();
 }

 public void display() {
```

```
  System.out.println("Avg/Max/Min temperature = " + (tempSum /
numReadings)
   + "/" + maxTemp + "/" + minTemp);
 }
}

CurrentConditionsDisplay.java
public class CurrentConditionsDisplay implements Observer, DisplayElement
{
 private float temperature;
 private float humidity;
 private Subject weatherData;

 public CurrentConditionsDisplay(Subject weatherData) {
  this.weatherData = weatherData;
  weatherData.registerObserver(this);
 }

 public void update(float temperature, float humidity, float pressure) {
  this.temperature = temperature;
  this.humidity = humidity;
  display();
 }

 public void display() {
  System.out.println("Current conditions: " + temperature
   + "F degrees and " + humidity + "% humidity");
 }
}

WeatherStation.java
public class WeatherStation {

 public static void main(String[] args) {
  WeatherData weatherData = new WeatherData();

  CurrentConditionsDisplay currentDisplay =
   new CurrentConditionsDisplay(weatherData);
  StatisticsDisplay statisticsDisplay = new
StatisticsDisplay(weatherData);
  ForecastDisplay forecastDisplay = new ForecastDisplay(weatherData);

  weatherData.setMeasurements(80, 65, 30.4f);
  weatherData.setMeasurements(82, 70, 29.2f);
  weatherData.setMeasurements(78, 90, 29.2f);
 }
}

--------------------------------------------------------------------------
-----------------------------------------
import pandas as pd
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import confusion_matrix

data=pd.read_csv("user_data1.csv")
x=data.iloc[:,[2,4]].values
y=data.iloc[:,4].values
print(x)
print(y)

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)
gd=GaussianNB()
gd.fit(xtrain,ytrain)

ypred=gd.predict(xtest)
cm=confusion_matrix(ypred,ytest)
print(cm)
```

```
models.py
      from django.db import models
class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    pub_date = models.DateTimeField('date published')

views.py
      from django.shortcuts import render
from .models import Post
def post_list(request):
    posts = Post.objects.all()
    return render(request, 'blog/post_list.html', {'posts': posts})

def post_detail(request, pk):
    post = Post.objects.get(pk=pk)
    return render(request, 'blog/post_details.html', {'post': post})

post_details.html
<!DOCTYPE html>
<html>
  <head>
    <title>{{ post.title }}</title>
  </head>
  <body>
    <h1>{{ post.title }}</h1>
    <p>{{ post.pub_date|date:"F d, Y" }}</p>
    <p>{{ post.content|linebreaks }}</p>
    <a href="{% url 'post_list' %}">Back to post list</a>
  </body>
</html>

post_list.html
      <!DOCTYPE html>
<html>
  <head>
    <title>Blog Posts</title>
```

```
  </head>
  <body>
    <h1>Blog Posts</h1>
    <ul>
      {% for post in posts %}
      <li>
        <a href="{% url 'post_detail' pk=post.pk %}">{{ post.title }}</a>
      </li>
      {% endfor %}
    </ul>
  </body>
</html>
```

blog/urls.py

```
      from django.urls import path
from . import views

urlpatterns = [
    path('', views.post_list, name='post_list'),
    path('post/<int:pk>/', views.post_detail, name='post_detail'),
]
```

myblog/urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')) ]
```

admin.py

```
      from django.contrib import admin
from .models import Post
admin.site.register(Post)
```

==========slip 29==========

Write a Java Program to implement State Pattern for Gumball Machine.
Create instance variable that holds current state from there, we just need to handle all
actions, behaviors and state transition that can happen
Write a python program to implement Decision Tree whether or not to play Tennis.
Create a clone of the “Hacker News” website.

State.java

```
public interface State {

      public void insertQuarter();
      public void ejectQuarter();
      public void turnCrank();
      public void dispense();
```

```
      public void refill();
}

NoQuarterState.java
public class NoQuarterState implements State {
    GumballMachine gumballMachine;

    public NoQuarterState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("You inserted a quarter");
            gumballMachine.setState(gumballMachine.getHasQuarterState());
      }

      public void ejectQuarter() {
            System.out.println("You haven't inserted a quarter");
      }

      public void turnCrank() {
            System.out.println("You turned, but there's no quarter");
       }

      public void dispense() {
            System.out.println("You need to pay first");
      }

      public void refill() { }

      public String toString() {
            return "waiting for quarter";
      }
}

HasQuarterState.java
public class HasQuarterState implements State {
      GumballMachine gumballMachine;

      public HasQuarterState(GumballMachine gumballMachine) {
            this.gumballMachine = gumballMachine;
      }

      public void insertQuarter() {
            System.out.println("You can't insert another quarter");
      }

      public void ejectQuarter() {
            System.out.println("Quarter returned");
            gumballMachine.setState(gumballMachine.getNoQuarterState());
      }

      public void turnCrank() {
            System.out.println("You turned...");
```

```
            gumballMachine.setState(gumballMachine.getSoldState());
      }

    public void dispense() {
        System.out.println("No gumball dispensed");
    }

    public void refill() { }

      public String toString() {
            return "waiting for turn of crank";
      }
}
```

SoldOutState.java

```
public class SoldOutState implements State {
    GumballMachine gumballMachine;

    public SoldOutState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("You can't insert a quarter, the machine
is sold out");
      }

      public void ejectQuarter() {
            System.out.println("You can't eject, you haven't inserted a
quarter yet");
      }

      public void turnCrank() {
            System.out.println("You turned, but there are no gumballs");
      }

      public void dispense() {
            System.out.println("No gumball dispensed");
      }

      public void refill() {
            gumballMachine.setState(gumballMachine.getNoQuarterState());
      }

      public String toString() {
            return "sold out";
      }
}
```

SoldState.java

```
public class SoldState implements State {

    GumballMachine gumballMachine;
```

```
    public SoldState(GumballMachine gumballMachine) {
        this.gumballMachine = gumballMachine;
    }

      public void insertQuarter() {
            System.out.println("Please wait, we're already giving you a
gumball");
      }

      public void ejectQuarter() {
            System.out.println("Sorry, you already turned the crank");
      }

      public void turnCrank() {
            System.out.println("Turning twice doesn't get you another
gumball!");
      }

      public void dispense() {
            gumballMachine.releaseBall();
            if (gumballMachine.getCount() > 0) {

      gumballMachine.setState(gumballMachine.getNoQuarterState());
            } else {
                  System.out.println("Oops, out of gumballs!");

      gumballMachine.setState(gumballMachine.getSoldOutState());
            }
      }

      public void refill() { }

      public String toString() {
            return "dispensing a gumball";
      }
}

GumballMachine.java
public class GumballMachine {

      State soldOutState;
      State noQuarterState;
      State hasQuarterState;
      State soldState;

      State state;
      int count = 0;

      public GumballMachine(int numberGumballs) {
            soldOutState = new SoldOutState(this);
            noQuarterState = new NoQuarterState(this);
            hasQuarterState = new HasQuarterState(this);
            soldState = new SoldState(this);
```

```
            this.count = numberGumballs;
            if (numberGumballs > 0) {
                  state = noQuarterState;
            } else {
                  state = soldOutState;
            }
      }

      public void insertQuarter() {
            state.insertQuarter();
      }

      public void ejectQuarter() {
            state.ejectQuarter();
      }

      public void turnCrank() {
            state.turnCrank();
            state.dispense();
      }

      void releaseBall() {
            System.out.println("A gumball comes rolling out the
slot...");
            if (count != 0) {
                  count = count - 1;
            }
      }

      int getCount() {
            return count;
      }

      void refill(int count) {
            this.count += count;
            System.out.println("The gumball machine was just refilled;
it's new count is: " + this.count);
            state.refill();
      }

      void setState(State state) {
            this.state = state;
      }
    public State getState() {
        return state;
    }

    public State getSoldOutState() {
        return soldOutState;
    }

    public State getNoQuarterState() {
        return noQuarterState;
    }
```

```
    public State getHasQuarterState() {
        return hasQuarterState;
    }

    public State getSoldState() {
        return soldState;
    }

      public String toString() {
            StringBuffer result = new StringBuffer();
            result.append("\nMighty Gumball, Inc.");
            result.append("\nJava-enabled Standing Gumball Model #2004");
            result.append("\nInventory: " + count + " gumball");
            if (count != 1) {
                  result.append("s");
            }
            result.append("\n");
            result.append("Machine is " + state + "\n");
            return result.toString();
      }
}

GumballMachineTestDrive.java
public class GumballMachineTestDrive {

      public static void main(String[] args) {
            GumballMachine gumballMachine = new GumballMachine(2);

            System.out.println(gumballMachine);

            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            System.out.println(gumballMachine);

            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();
            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            gumballMachine.refill(5);
            gumballMachine.insertQuarter();
            gumballMachine.turnCrank();

            System.out.println(gumballMachine);
      }
}
------------------------------------------------------------------------
------------------------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
```

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
data=pd.read_csv("tennis.csv")
print(data)

le=LabelEncoder()

data["Outlook"]=le.fit_transform(data["Outlook"])
data["Temprature"]=le.fit_transform(data["Temprature"])
data["Humidity"]=le.fit_transform(data["Humidity"])
data["Wind"]=le.fit_transform(data["Wind"])
data["Play_Tennis"]=le.fit_transform(data["Play_Tennis"])
print(data)


x=data.iloc[:,1:5].values
y=data["Play_Tennis"]

xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2)

dc=DecisionTreeClassifier(criterion="entropy")
dc.fit(xtrain,ytrain)

from sklearn.tree import export_graphviz
export_graphviz(dc,out_file="a1.txt")

ypred=dc.predict(xtest)
cm=confusion_matrix(ypred,ytest)
print(cm)
--------------------------------------------------------------------------
--------------------------
```

```
# Create a new Django project
django-admin startproject hackernews

# Navigate to the project directory
cd hackernews

# Create a new Django app
python manage.py startapp news
```

models.py

```
# news/models.py
from django.db import models
from django.contrib.auth.models import User

class Article(models.Model):
    title = models.CharField(max_length=200)
    url = models.URLField()
    created_at = models.DateTimeField(auto_now_add=True)
    upvotes = models.IntegerField(default=0)
```

```
    author = models.ForeignKey(User, on_delete=models.CASCADE)

    def __str__(self):
        return self.title


view.py
# news/views.py
from django.shortcuts import render, get_object_or_404
from .models import Article

def article_list(request):
    articles = Article.objects.all()
    return render(request, 'news/article_list.html', {'articles':
articles})

def article_detail(request, article_id):
    article = get_object_or_404(Article, pk=article_id)
    return render(request, 'news/article_detail.html', {'article':
article})


Create HTML templates in the news/templates/news directory:

article_list.html:
<!-- news/templates/news/article_list.html -->
{% for article in articles %}
  <h2><a href="{% url 'article_detail' article.id %}">{{ article.title
}}</a></h2>
  <p>Author: {{ article.author.username }} | Upvotes: {{ article.upvotes
}} | Created at: {{ article.created_at }}</p>
  <hr>
{% endfor %}


article_detail.html:
<!-- news/templates/news/article_detail.html -->
<h2>{{ article.title }}</h2>
<p>Author: {{ article.author.username }} | Upvotes: {{ article.upvotes }}
| Created at: {{ article.created_at }}</p>
<p>URL: <a href="{{ article.url }}" target="_blank">{{ article.url
}}</a></p>


news/urls.py
# news/urls.py
from django.urls import path
from .views import article_list, article_detail

urlpatterns = [
    path('', article_list, name='article_list'),
    path('article/<int:article_id>/', article_detail,
name='article_detail'),
]
```

urls.py

```
# hackernews/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('news.urls')),
]
```

==========slip 30==========

Write a Java Program to implement Factory method for Pizza Store with createPizza(),
orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza's
like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.
Write a python program to implement Linear SVM. [20 M]
Implement Login System using Django.

PizzaStore.java

```
public abstract class PizzaStore{
abstract Pizza createPizza(String item);
public Pizza orderPizza(String type){
    Pizza pizza=createPizza(type);
System.out.println("---Making a"+pizza.getName()+"---");
pizza.prepare();
pizza.bake();
pizza.cut();
pizza.box();
return pizza;
}
}
```

ChicagoPizzaStore.java

```
public class ChicagoPizzaStore extends PizzaStore {

    Pizza createPizza(String item) {
            if (item.equals("cheese")) {
                    return new ChicagoStyleCheesePizza();
            } else if (item.equals("veggie")) {
                    return new ChicagoStyleVeggiePizza();
            } else if (item.equals("clam")) {
                    return new ChicagoStyleClamPizza();
            } else if (item.equals("pepperoni")) {
                    return new ChicagoStylePepperoniPizza();
            } else return null;
```

```
    }
}
```

NYPizzaStore.java

```
public class NYPizzaStore extends PizzaStore{
Pizza createPizza(String item){
if(item.equals("cheese")){
return new;
NYStyleCheesePizza();
}else if(item.equals("veggie")){
return new;
NYStyleCheesePizza();
}else if(item.equals("clam")){
return new;
NYStyleCheesePizza();
}else if(item.equals("pepperoni")){
return new;
NYStyleCheesePizza();
}else return null;
}
}
```

Pizza.java

```
import java.util.ArrayList;

public abstract class Pizza {
    String name;
    String dough;
    String sauce;
    ArrayList<String> toppings = new ArrayList<String>();

    void prepare() {
        System.out.println("Prepare " + name);
        System.out.println("Tossing dough...");
        System.out.println("Adding sauce...");
        System.out.println("Adding toppings: ");
        for (String topping : toppings) {
            System.out.println("   " + topping);
        }
    }

    void bake() {
        System.out.println("Bake for 25 minutes at 350");
    }

    void cut() {
        System.out.println("Cut the pizza into diagonal slices");
    }

    void box() {
        System.out.println("Place pizza in official PizzaStore box");
    }
```

```
    public String getName() {
        return name;
    }

    public String toString() {
        StringBuffer display = new StringBuffer();
        display.append("---- " + name + " ----\n");
        display.append(dough + "\n");
        display.append(sauce + "\n");
        for (String topping : toppings) {
            display.append(topping + "\n");
        }
        return display.toString();
    }
}
```

NYStyleCheesePizza.java

```
public class NYStyleCheesePizza extends Pizza {

    public NYStyleCheesePizza() {
        name = "NY Style Sauce and Cheese Pizza";
        dough = "Thin Crust Dough";
        sauce = "Marinara Sauce";

        toppings.add("Grated Reggiano Cheese");
    }
}
```

NYStyleClamPizza.java

```
public class NYStyleClamPizza extends Pizza {

    public NYStyleClamPizza() {
        name = "NY Style Clam Pizza";
        dough = "Thin Crust Dough";
        sauce = "Marinara Sauce";

        toppings.add("Grated Reggiano Cheese");
        toppings.add("Fresh Clams from Long Island Sound");
    }
}
```

ChicagoStylePepperoniPizza.java

```
public class ChicagoStylePepperoniPizza extends Pizza {
    public ChicagoStylePepperoniPizza() {
        name = "Chicago Style Pepperoni Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Spinach");
        toppings.add("Eggplant");
        toppings.add("Sliced Pepperoni");
    }
```

```
    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}
```

ChicagoStyleVeggiePizza.java

```
public class ChicagoStyleVeggiePizza extends Pizza {
    public ChicagoStyleVeggiePizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Spinach");
        toppings.add("Eggplant");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}
```

ChicagoStyleClamPizza.java

```
public class ChicagoStyleClamPizza extends Pizza {
    public ChicagoStyleClamPizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Clams");
        toppings.add("Jalapeons");
    }

    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}
```

ChicagoStyleCheesePizza.java

```
public class ChicagoStyleCheesePizza extends Pizza {
    public ChicagoStyleCheesePizza() {
        name = "Chicago Deep Dish Veggie Pizza";
        dough = "Extra Thick Crust Dough";
        sauce = "Plum Tomato Sauce";

        toppings.add("Shredded Mozzarella Cheese");
        toppings.add("Black Olives");
        toppings.add("Mayo");
        toppings.add("Cheddar");
    }
```

```
    void cut() {
        System.out.println("Cutting the pizza into square slices");
    }
}

PizzaTestDrive.java
public class PizzaTestDrive {

    public static void main(String[] args) {
        PizzaStore nyStore = new NYPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        Pizza pizza = nyStore.orderPizza("cheese");
        System.out.println("First order was a " + pizza.getName() +
"\n");

        pizza = nyStore.orderPizza("cheese");
        System.out.println("Second order was a " + pizza.getName() +
"\n");
    }
}
--------------------------------------------------------------------------
---------------------------------------
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVR

data=pd.read_csv("user_data1.csv")
print(data)

x=data.iloc[:,2:3].values
y=data.iloc[:,3].values
print(x)
print(y)
sr=SVR()
sr.fit(x,y)

print(sr.predict([[150]]))

plt.scatter(x,y,c="red")
plt.plot(x,sr.predict(x),c="green")
plt.show()

--------------------------------------------------------------------------
----------------------------------
settings.py (add following):
      AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
)
LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = 'home'
views.py
```

```
      from django.contrib.auth import authenticate, login
from django.shortcuts import render, redirect

def user_login(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username,
password=password)
        if user is not None:
            login(request, user)
            return redirect('welcome')
    return render(request, 'registration/login.html')

def welcome(request):
    return render(request, 'registration/home.html')
```

login.html

```
      <!DOCTYPE html>
<html>
  <head>
    <title>Login</title>
  </head>
  <body>
    <h1>Login</h1>
    <form method="post">
      {% csrf_token %}
      <label for="username">Username:</label>
      <input type="text" id="username" name="username" required />
      <br />
      <label for="password">Password:</label>
      <input type="password" id="password" name="password" required />
      <br />
      <button type="submit">Login</button>
    </form>
  </body>
</html>
```

home.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Document</title>
  </head>
  <body>
    Welcome
  </body>
</html>
```

myapp/urls.py

```
      from django.urls import path
from . import views
```

```
urlpatterns = [
    path('login/', views.user_login, name='login'),
    path('welcome/', views.welcome, name='welcome')
]

myproject/urls.py
      from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('myapp.urls')),
]
```

==========

==========