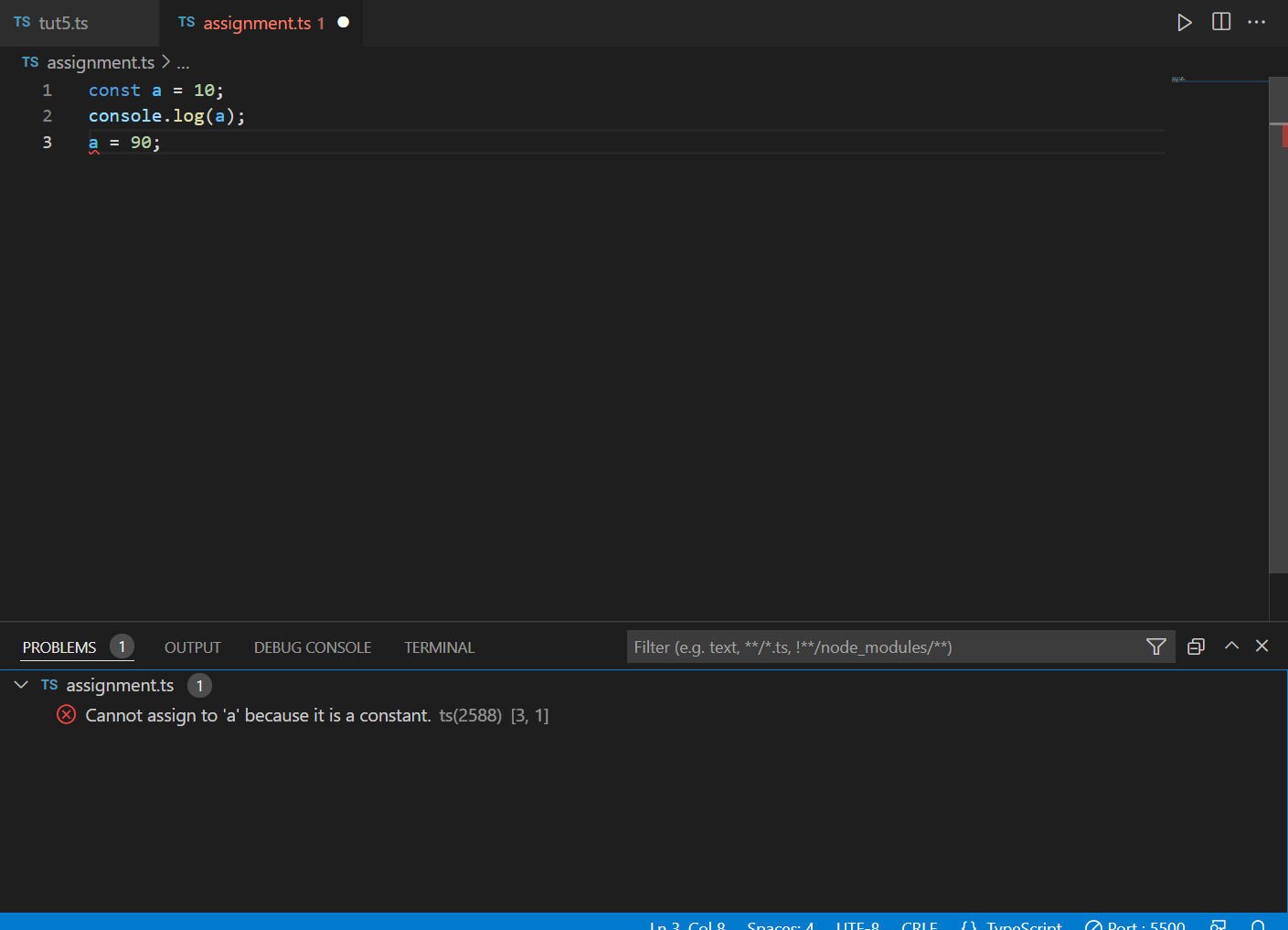
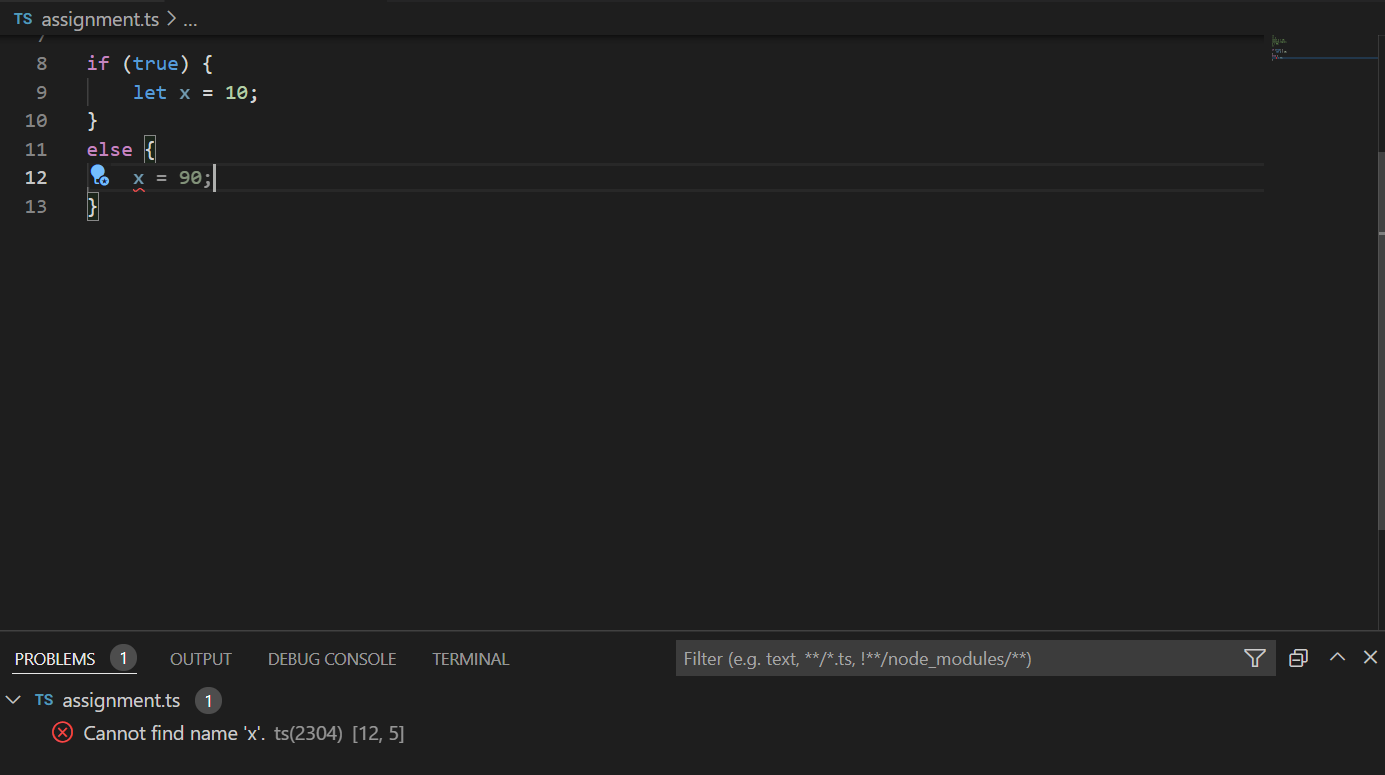
Assignment 1.

1. **Constants:** Declare a constant & confirm its value cannot be changed.



**2. Scoping:** Declare a variable inside if condition & make sure that it is not accessible outside if condition.



3. **Enhanced object properties:** Create an ‘Order’ object having data members ‘id’, ‘title’, ‘price’. Add the methods printOrder() & getPrice(). Now, copy the order object using Object.assign().

var order = {

    id: 101,

    title: "This is super!",

    price: 200,

    printOrder() {

        console.log("heloo");

    },

    getPrice() {

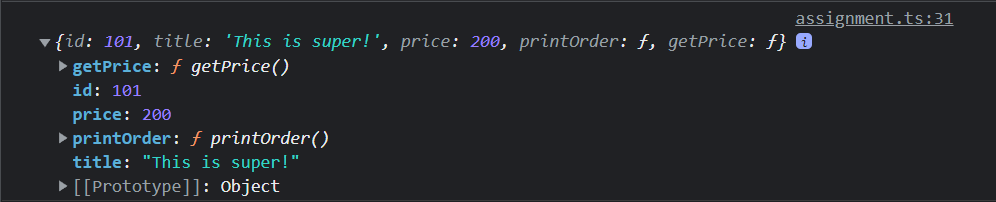
        console.log("Rs. 200");

    }

}

const newObj = Object.assign({}, order)

console.log(newObj);



**4. Arrow functions:** Take an array of strings & convert it into another array of object which has two properties {string, string\_length}. For example:

let names = [‘Tom’, ‘Ivan’, ‘Jerry’]

Output: [ {name: ’Tom’, length: 3}, {name: ’Ivan’, length: 4 }, {name: ’Jerry’, length: 5} ]

var arrow = (names: string[]) => {

    let out = [];

    for (let color of names) {

        var row: any = {

        };

        row.name = color;

        row.length = color.length;

        out.push(row);

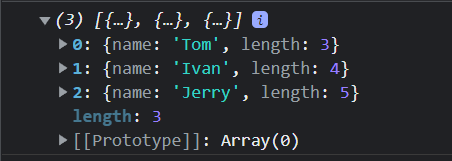
    }

    return out;

};

let names = ['Tom', 'Ivan', 'Jerry'];

console.log(arrow(names));



**5.Extended parameter handling:**

a. Write a add() with default values.

b. Write a function userFriends() that takes 2 arguments username & array of user friends. The function should print username & his list of friends. (Use rest parameters)

c. Write a function printCapitalNames() that takes five names as argument & prints them in capital letters. Use spread operator in order to call printCapitalNames() function.

let addval = function (a = 10) {

    console.log(a);

};

addval(3);

addval();

let friends = ['A', 'B', 'C', 'D', 'E'];

let userFriends = function (userName, ...friends) {

    console.log(userName);

    for (const friend of friends) {

        console.log(friend);

    }

}

let userName = "Pradumnya";

userFriends(userName, friends);

let names = ['a', 'b', 'c', 'd', 'e'];

let printCapitalNames = function (...names) {

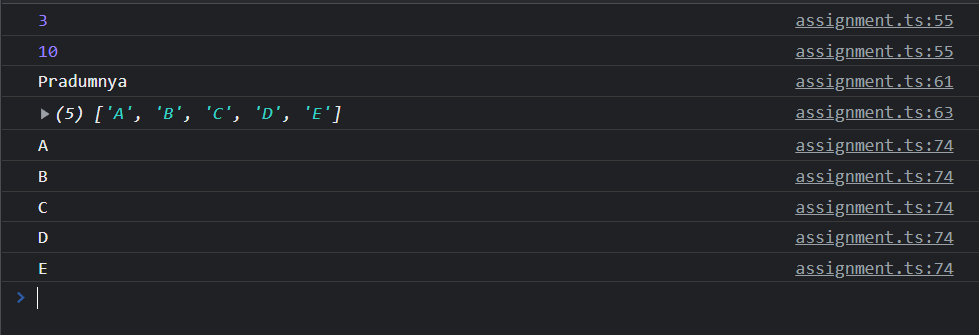
    for (const name of names) {

        console.log(name.toUpperCase());

    }

}

printCapitalNames(...names);



**6.Template literals:** Draft a ticket to Sysnet that describes problem with your laptop. Use ‘template literals’ to add value of laptop model, your desk no, your name etc.

let model = "laptop"

let desknumber = 2022;

let names = "Pradumnya";

let laptop = {

    model,

    desknumber,

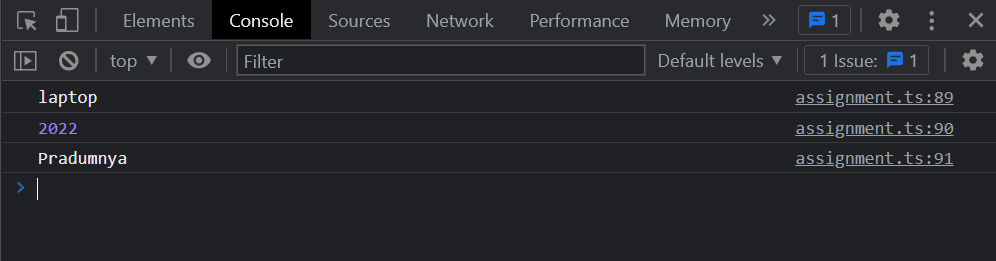
    names

};

console.log(laptop.model);

console.log(laptop.desknumber);

console.log(laptop.names);



**7. De-structuring assignment:**

a. Suppose there is a javascript array with 4 elements. Print the value of 3rd element using array matching.

b. Create an organization object having attributes name, address. Write a program to retrieve pin code of an address using object deep matching.

let arr = ['A', 'B', 'C', 'D'];

let [a, b, c, d] = arr;

console.log(c);

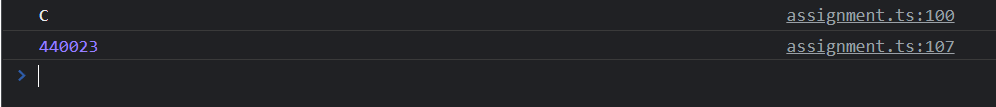
let organisation = {

    name: "pradumnya",

    address: { pin: 440023 }

};

console.log(organisation.address.pin);



**8. Classes & Modules:** Write a class Account with attributes id, name, balance. Add two sub classes SavingAccount & CurrentAccount having specific attribute interest & cash\_credit respectively. Create multiple saving & current account objects. Write a functionality to find out total balance in the bank.

class Account {

    id: any;

    name: any;

    balance: any;

    constructor(id, name, balance) {

        this.id = id;

        this.name = name;

        this.balance = balance

    }

    totalBalance() {

        return this.balance;

    }

};

class SavingAccount extends Account {

    constructor(id, name, balance, intrest) {

        super(id, name, balance = balance + (balance \* intrest) / 100);

    }

}

class CurrentAccount extends Account {

    constructor(id, name, balance, cash\_credit) {

        super(id, name, balance = balance + cash\_credit);

    }

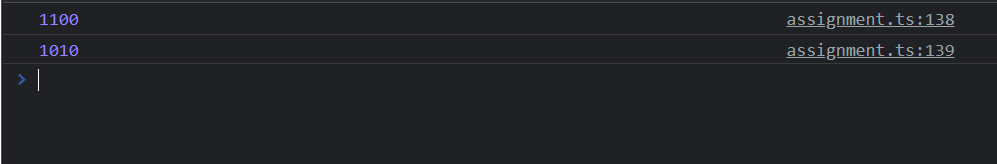
}

let savingAccount = new SavingAccount(1, "Pradumnya", 1000, 10);

let currentAccount = new CurrentAccount(2, "Rishikesh", 1000, 10);

console.log(savingAccount.totalBalance());

console.log(currentAccount.totalBalance());



Assignment 2.

**1. Symbols:** Write a class that defines next() method to return next number from Fibonacci series. The class will have a private attributes ‘previousNo’ & ‘currentNo’.

class MyClass {

    private previousNo;

    private currentNo;

    constructor() {

        this.previousNo = fibonacci(this.currentNo - 1);

        this.currentNo = this.currentNo;

    }

    next() {

        return fibonacci(this.currentNo + 1);

    }

}

var fibonacci = function (num) {

    var a = 0,

        b = 1,

        f = 1;

    for (var i = 2; i <= num; i++) {

        f = a + b;

        a = b;

        b = f;

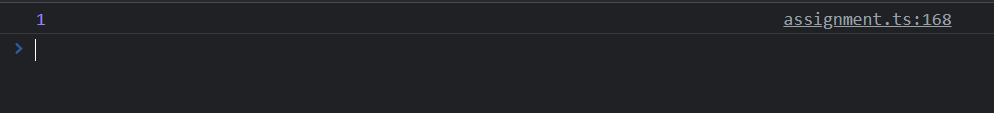
    }

    return f;

};

let myClass = new MyClass();

console.log(myClass.next());



**2. Iterators:** Write a program that returns next Armstrong number after calling getNextArmstrong() method.

let lowNumber = 10;

function armstrong() {

    for (let i = lowNumber; i > 0; i++) {

        let numberOfDigits = i.toString().length;

        let sum = 0;

        let temp = i.toString();

        for (let char of temp) {

            sum += Math.pow(parseInt(char), numberOfDigits);

        }

        if (sum == i) {

            console.log(i);

            lowNumber = i;

            break;

        }

    }

}

function getNextArmstrong() {

    lowNumber++;

    armstrong();

}

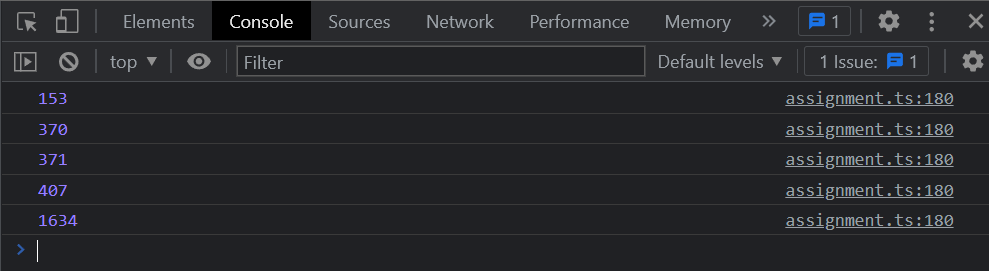
armstrong();

getNextArmstrong();

getNextArmstrong();

getNextArmstrong();

getNextArmstrong();



**3. Generators:** Write a program that returns next Armstrong number after calling getNextArmstrong() method. Add functionality to reset generating Armstrong number from zero. In case, Armstrong number goes above one thousand then throw an error.

let lowNumber2 = 9;

function armstrong2() {

    for (let i = lowNumber2; i > 0; i++) {

        let numberOfDigits = i.toString().length;

        let sum = 0;

        let temp = i.toString();

        for (let char of temp) {

            sum += Math.pow(parseInt(char), numberOfDigits);

        }

        if (sum == i) {

            console.log(i);

            lowNumber2 = i;

            break;

        }

    }

}

function getNextArmstrong2() {

    lowNumber2++;

    armstrong2();

}

function reset() {

    lowNumber2 = 9;

}

armstrong2();

getNextArmstrong2();

getNextArmstrong2();

getNextArmstrong2();

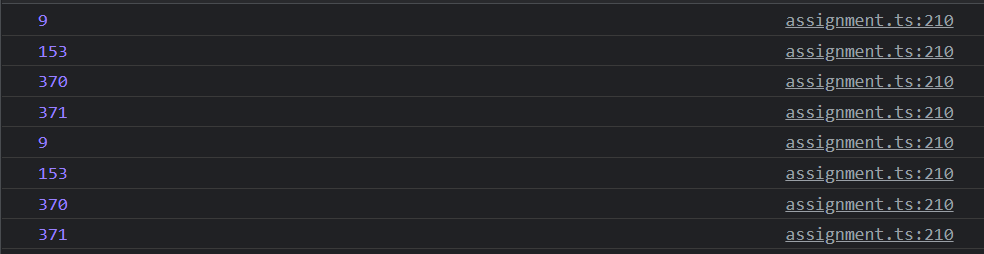
reset();

armstrong2();

getNextArmstrong2();

getNextArmstrong2();

getNextArmstrong2();



**4. Collections:** Using Set & Map, create a static data for chatting application. Here we have 2 chatrooms, every chatroom is having 3 users & every user has posted different messages in a chat room. Note that one user can belong to a single chat room only. Now you need to find out how you will hold this data using Set & Map data structures. Also add functionality to get list of all users from a specific chatroom & listing down all message from a chatroom.

class User {

    name: string;

    message: Map<string, string>;

    constructor(name: string, message: Map<string, string>) {

        this.name = name;

        this.message = message;

    }

}

let msgUser1 = new Map();

msgUser1.set("message1", "This is the message 1");

msgUser1.set("message2", "This is the message 2");

let user1 = new User("Alpha", msgUser1);

let msgUser2 = new Map();

msgUser2.set("message1", "This is the message 1");

msgUser2.set("message2", "This is the message 2");

let user2 = new User("Bravo", msgUser2);

let msgUser3 = new Map();

msgUser3.set("message1", "This is the message 1");

msgUser3.set("message2", "This is the message 2");

let user3 = new User("Chralie", msgUser3);

let msgUser4 = new Map();

msgUser4.set("message1", "This is the message 1");

msgUser4.set("message2", "This is the message 2");

let user4 = new User("Delta", msgUser4);

let msgUser5 = new Map();

msgUser5.set("message1", "This is the message 1");

msgUser5.set("message2", "This is the message 2");

let user5 = new User("Echo", msgUser5);

let msgUser6 = new Map();

msgUser6.set("message1", "This is the message 1");

msgUser6.set("message2", "This is the message 2");

let user6 = new User("Foxtrot", msgUser6);

let usersSet = new Set();

let usersSet2 = new Set();

usersSet.add(user1);

usersSet.add(user2);

usersSet.add(user3);

function addUserRoom2(user: User) {

    if (!usersSet.has(user) && usersSet2.size < 3) {

        usersSet2.add(user);

    }

}

addUserRoom2(user4);

addUserRoom2(user5);

addUserRoom2(user1);

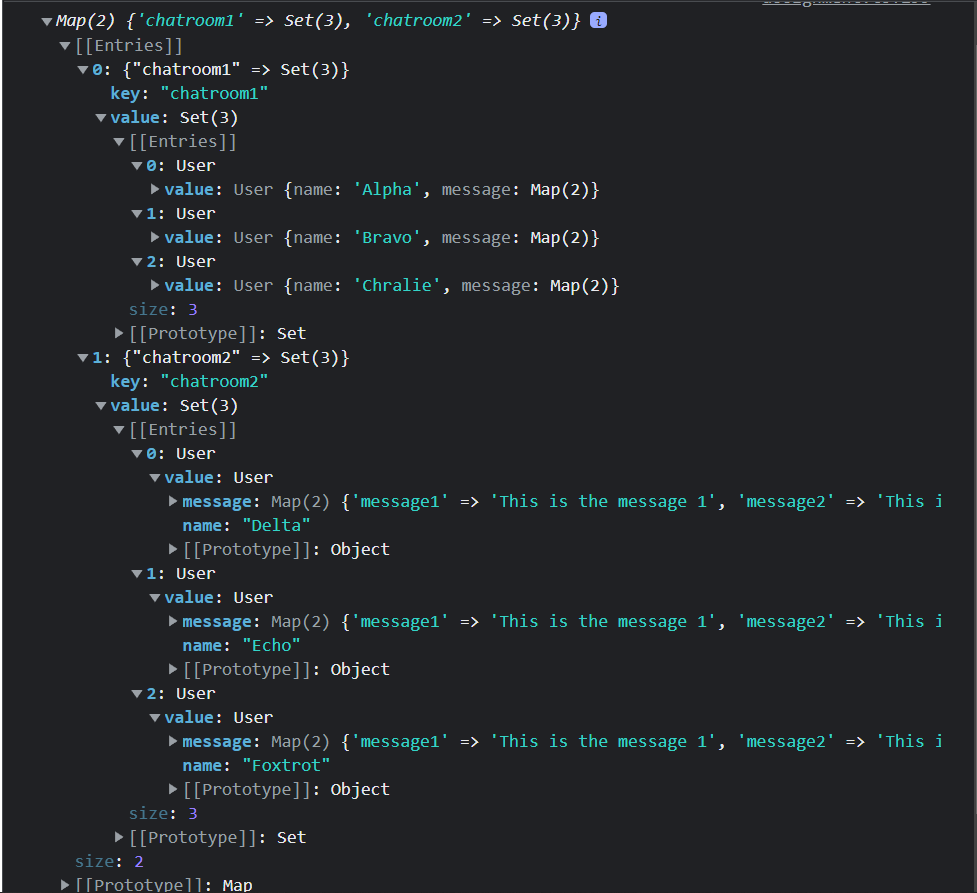
addUserRoom2(user6);

let chatroom = new Map<String, Set<User>>();

chatroom.set("chatroom1", usersSet);

chatroom.set("chatroom2", usersSet2);

console.log(chatroom);



Assignment 3

**1. Promises:** Create 2 promises, one generates value of x & another generates value of you. Write a program to print sum of x & y. (Use Promise.all)

let x = new Promise((resolve, reject) => {

    setTimeout(resolve, 1000, 9);

})

let y = new Promise((resolve, reject) => {

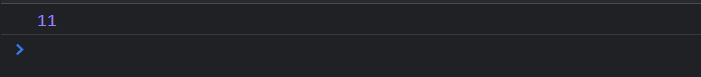
    setTimeout(resolve, 1000, 2);

})

Promise.all([x, y]).then((values) => {

    console.log(values[0] + values[1]);

});



**2. TypeScript classes & types:** Write a class Account with attributes id, name, balance. Add two sub classes SavingAccount & CurrentAccount having specific attribute interest & cash\_credit respectively. Create multiple saving & current account objects. Write a functionality to find out total balance in the bank.

class Account {

    id: any;

    name: any;

    balance: any;

    constructor(id, name, balance) {

        this.id = id;

        this.name = name;

        this.balance = balance

    }

    totalBalance() {

        return this.balance;

    }

};

class SavingAccount extends Account {

    constructor(id, name, balance, intrest) {

        super(id, name, balance = balance + (balance \* intrest) / 100);

    }

}

class CurrentAccount extends Account {

    constructor(id, name, balance, cash\_credit) {

        super(id, name, balance = balance + cash\_credit);

    }

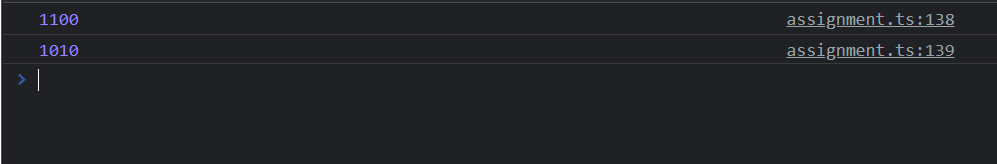
}

let savingAccount = new SavingAccount(1, "Pradumnya", 1000, 10);

let currentAccount = new CurrentAccount(2, "Rishikesh", 1000, 10);

console.log(savingAccount.totalBalance());

console.log(currentAccount.totalBalance());



**3. TypeScript Interfaces:** Write an interface Printable. Create 2 objects circle & employee those implement Printable interface. Write a function printAll() that takes all objects as argument & invoke print() method on every object.

interface Printable {

    print();

}

class Circle implements Printable {

    radius: any;

    area: any;

    constructor(radius) {

        this.radius = radius;

        this.area = 3.18 \* radius \* radius;

    }

    print() {

        console.log(this.radius);

        console.log(this.area);

    }

}

class Employee implements Printable {

    id: any;

    name: any;

    constructor(id, name) {

        this.id = id;

        this.name = name;

    }

    print() {

        console.log("Id: " + this.id + " \nName: " + this.name);

    }

}

let circle = new Circle(2);

let employee = new Employee(101, "Pradumnya");

function PrintAll(obj1, obj2) {

    obj1.print();

    obj2.print();

}

PrintAll(circle, employee);

