Submitted To:

Engineer Sheharyar Khan

Name: Iqra Fatima

Reg. Number: 23-CP-62

Semester: 3rd

Department: CPED

Data Structures and Algorithms

(DSA)  
Lab Report 2

Marks Obtained: 8

Total Marks: 8

**Marks Distribution:**

Total Lab Report Marks: 04 Total Lab Activity Marks: 04

Table of Contents

[1](#_Toc189169279)

[Examples: 4](#_Toc189169280)

[Task 1: Python Arrays 4](#_Toc189169281)

[Code: 4](#_Toc189169282)

[Output: 4](#_Toc189169283)

[Task 2: Advanced Python Arrays 4](#_Toc189169284)

[Code: 4](#_Toc189169285)

[Output: 5](#_Toc189169286)

[Task 3: Solving Real-Life Problems with Arrays 5](#_Toc189169287)

[Code: 5](#_Toc189169288)

[Output: 5](#_Toc189169289)

[Task 4: Python Classes and Objects 5](#_Toc189169290)

[Code: 5](#_Toc189169291)

[Output: 5](#_Toc189169292)

[Task 5: Encapsulation Using Private Members 6](#_Toc189169293)

[Code: 6](#_Toc189169294)

[Output: 6](#_Toc189169295)

[Task 6: Python Inheritance 6](#_Toc189169296)

[Code: 6](#_Toc189169297)

[Output: 7](#_Toc189169298)

[Task 7: Polymorphism (Overloading and Overriding) 7](#_Toc189169299)

[Code a: 7](#_Toc189169300)

[Output: 7](#_Toc189169301)

[Code b: 7](#_Toc189169302)

[Output: 8](#_Toc189169303)

[Task 8: Abstraction Using Abstract Classes 8](#_Toc189169304)

[Code: 8](#_Toc189169305)

[Output: 9](#_Toc189169306)

[Task 9: Python Modules 9](#_Toc189169307)

[Code: 9](#_Toc189169308)

[Output: 9](#_Toc189169309)

[Task 10: Python File handling 9](#_Toc189169310)

[Code: 9](#_Toc189169311)

[Output: 9](#_Toc189169312)

[Task 11: Python JSON 10](#_Toc189169313)

[Code: 10](#_Toc189169314)

[Output: 10](#_Toc189169315)

[Task 12: Python Regular Expressions 10](#_Toc189169316)

[Code: 10](#_Toc189169317)

[Output: 10](#_Toc189169318)

[PROBLEMS 10](#_Toc189169319)

[Beginner Problems 10](#_Toc189169320)

[Question 1: Array Rotation 10](#_Toc189169321)

[Question 2: Encapsulation with Getters and Setters 11](#_Toc189169322)

[Question 3: Date Difference 13](#_Toc189169323)

[Question 4: Basic JSON Handling 13](#_Toc189169324)

[Question 5: Using Python Math Library 14](#_Toc189169325)

[Intermediate Problems 15](#_Toc189169326)

[Question 6: File Handling: Find Longest Word 15](#_Toc189169327)

[Question 7: Inheritance: Employee and Manager Classes 16](#_Toc189169328)

[Question 8: Regex Validation for Email 17](#_Toc189169329)

[Question 9: Python String Formatting 18](#_Toc189169330)

[Question 10: User Input and PIP Library Usage 19](#_Toc189169331)

[Advanced Problems 20](#_Toc189169332)

[Question 11: Polymorphism: Shapes Area Calculation 20](#_Toc189169333)

[Question 12: Abstract Class for Payment Processing 21](#_Toc189169334)

[Question 13: JSON Data Analysis 22](#_Toc189169335)

[Question 14: Regex Search in Log File 23](#_Toc189169336)

[Question 15: File Handling: Merge and Sort Files 24](#_Toc189169337)

# Examples:

## Task 1: Python Arrays

### Code:

A screenshot of a computer code

Description automatically generated

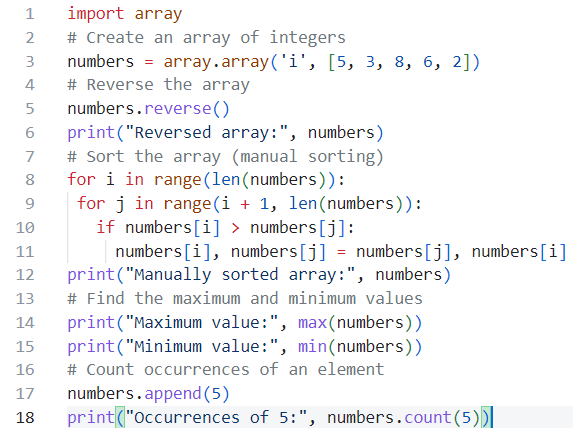
### Output:

A group of black text

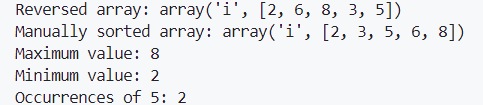
Description automatically generated

## Task 2: Advanced Python Arrays

### Code:

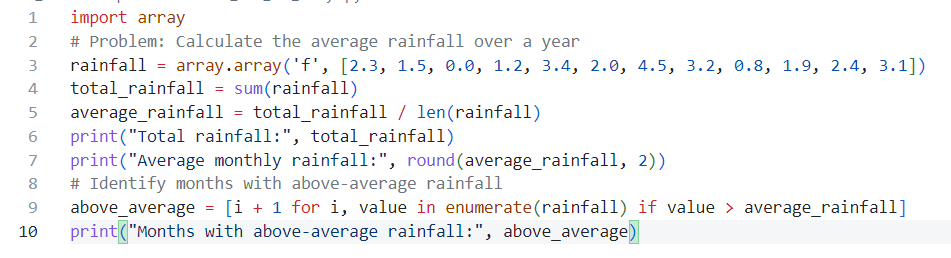


### Output:

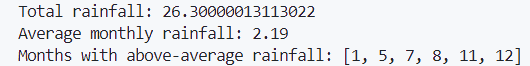


## Task 3: Solving Real-Life Problems with Arrays

### Code:

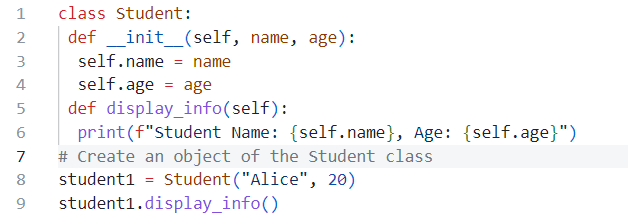


### Output:



## Task 4: Python Classes and Objects

### Code:

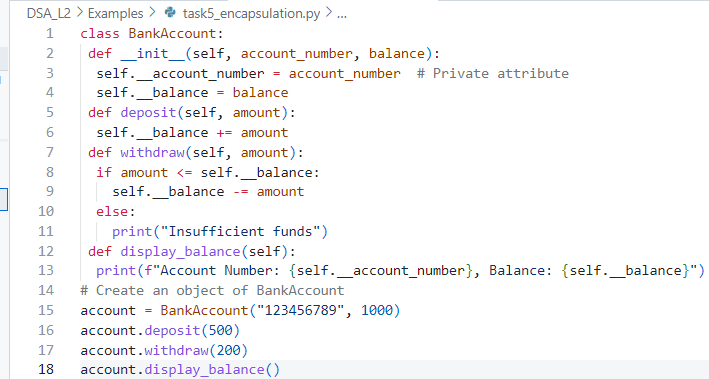


### Output:



## Task 5: Encapsulation Using Private Members

### Code:

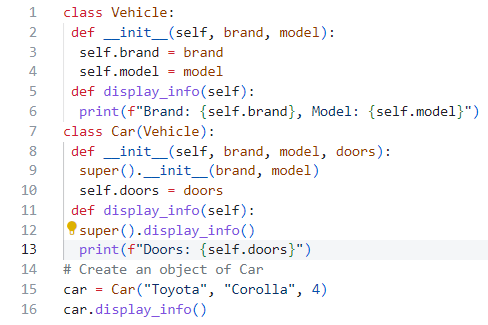


### Output:



## Task 6: Python Inheritance

### Code:



### Output:



## Task 7: Polymorphism (Overloading and Overriding)

1. Overloading

### Code a:

A screenshot of a computer

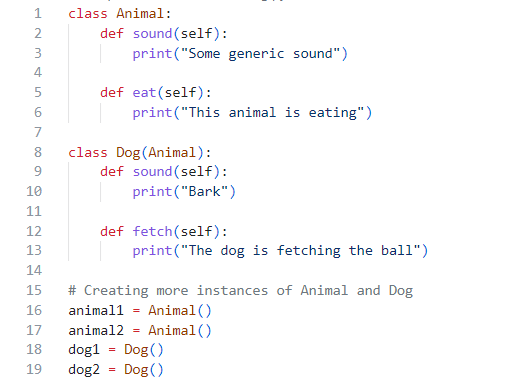
Description automatically generated

### Output:



1. Overriding

### Code b:



A screenshot of a computer program

Description automatically generated

### Output:

A close up of words

Description automatically generated

## Task 8: Abstraction Using Abstract Classes

### Code:

A screenshot of a computer program

Description automatically generated

### Output:



## Task 9: Python Modules

### Code:

A computer screen shot of a code

Description automatically generated

### Output:



## Task 10: Python File handling

### Code:

A screen shot of a computer program

Description automatically generated

### Output:



## Task 11: Python JSON

### Code:

A screenshot of a computer code

Description automatically generated

### Output:



## Task 12: Python Regular Expressions

### Code:

A screenshot of a computer code

Description automatically generated

### Output:



# PROBLEMS

## Beginner Problems

### Question 1: Array Rotation

Write a Python program to rotate an array by a given number of

steps.

o Input: [1, 2, 3, 4, 5], Steps: 2

o Output: [4, 5, 1, 2, 3]

#### Code:

n=int(input("Enter the number of elements in the array: "))

arr=[]

for i in range(n):

    arr.append(int(input("Enter the element: ")))

steps=int(input("Enter the number of steps to rotate the array: "))

print("Original array:",arr)

for i in range(steps):

    arr.insert(0,arr.pop())

print("Rotated array:",arr)

#### Output:

A screenshot of a computer code

Description automatically generated

### Question 2: Encapsulation with Getters and Setters

Create a class BankAccount with private attributes account\_number and balance. Use getters and setters to update and retrieve

these values.

o Input: account\_number=12345, balance=10000

o Output: Account Number: 12345, Balance: 10000

#### Code:

class BankAccount:

    def \_\_init\_\_(self, account\_number, balance):

        # Private attributes

        self.\_\_account\_number = account\_number

        self.\_\_balance = balance

    # Getter for account\_number

    def get\_account\_number(self):

        return self.\_\_account\_number

    # Setter for account\_number

    def set\_account\_number(self, account\_number):

        self.\_\_account\_number = account\_number

    # Getter for balance

    def get\_balance(self):

        return self.\_\_balance

    # Setter for balance

    def set\_balance(self, balance):

        if balance >= 0:  # Ensure balance is not negative

            self.\_\_balance = balance

        else:

            print("Balance cannot be negative!")

    # Display account details

    def display\_details(self):

        print(f"Account Number: {self.\_\_account\_number}")

        print(f"Balance: {self.\_\_balance}")

# Main block for testing

if \_\_name\_\_ == "\_\_main\_\_":

    # Take input from the user

    account\_number = int(input("Enter account number: "))

    balance = float(input("Enter initial balance: "))

    # Create BankAccount object

    account = BankAccount(account\_number, balance)

    # Display initial details

    print("\nInitial Account Details:")

    account.display\_details()

**# Update values using user input**

    new\_account\_number = int(input("\nEnter new account number: "))

    account.set\_account\_number(new\_account\_number)

    new\_balance = float(input("Enter new balance: "))

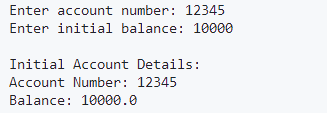
    account.set\_balance(new\_balance)

**# Display updated details**

    print("\nUpdated Account Details:")

    account.display\_details()

#### Output:



### Question 3: Date Difference

Write a program to calculate the number of days between two given dates.

o Input: Date 1: 2025-01-01, Date 2: 2025-01-15

o Output: Difference: 14 days

#### Code:

from datetime import datetime

def date\_difference(date1\_str, date2\_str):

    # Convert the string dates to datetime objects

    date\_format = "%Y-%m-%d"

    date1 = datetime.strptime(date1\_str, date\_format)

    date2 = datetime.strptime(date2\_str, date\_format)

    # Calculate the difference in days

    difference = (date2 - date1).days

    return difference

# Input dates

date1 = "2025-01-01"

date2 = "2025-01-15"

# Calculate and print the difference

days\_difference = date\_difference(date1, date2)

print("Difference:", days\_difference, "days")

#### Output:



### Question 4: Basic JSON Handling

Write a program to create a JSON object with keys name, age,

and grade. Then read and print the values.

o Input: { "name": "Alice", "age": 20, "grade": "A" }

o Output:

Name: Alice

Age: 20

Grade: A

#### Code:

import json

json\_data = '{"name": "Alice", "age": 20, "grade": "A"}'

# Convert JSON data to Python dictionary

data = json.loads(json\_data)

# Print the values

print("Name:", data["name"])  # Output: Name: Alice

print("Age:", data["age"])    # Output: Age: 20

print("Grade:", data["grade"]) # Output: Grade: A

#### Output:



### Question 5: Using Python Math Library

Create a program to find the greatest common divisor (GCD) and least common multiple (LCM) of two numbers using the math module.

o Input: 12, 15

o Output:

GCD: 3

LCM: 60

#### Code:

import math

def find\_gcd\_lcm(num1, num2):

    # Calculate GCD using math.gcd()

    gcd = math.gcd(num1, num2)

    # Calculate LCM using the formula: LCM = (num1 \* num2) // GCD

    lcm = (num1 \* num2) // gcd

    return gcd, lcm

# Input numbers

num1 = 12

num2 = 15

# Find GCD and LCM

gcd, lcm = find\_gcd\_lcm(num1, num2)

# Print the results

print("GCD:", gcd)  # Output: GCD: 3

print("LCM:", lcm)  # Output: LCM: 60

#### Output:

****

## Intermediate Problems

### Question 6: File Handling: Find Longest Word

Write a program to read a text file and find the longest word in the file.

o Input File Content: "The quick brown fox jumps over the lazy dog"

o Output: Longest Word: jumps

#### Code:

def find\_longest\_word(file\_path):

    with open(file\_path, 'r') as file:

        content = file.read()

    words = content.split()

    longest\_word = max(words, key=len)

    return longest\_word

file\_path = 'D:\\4thSemester\\DSA(Python)\\DSA\_Lab\\DSA\_Lab\_Tasks\_CodeFiles\\DSA\_L2\\Exercises\\B\_Intermediate Problems\\input.txt'

# Find and print the longest word

longest\_word = find\_longest\_word(file\_path)

print("Longest Word:", longest\_word)

#### Output:



Where text file was as follows:

A screenshot of a computer

Description automatically generated

### Question 7: Inheritance: Employee and Manager Classes

Create a base class Employee with

attributes name and salary. Create a derived class Manager with an additional

attribute department. Write methods to display their details.

o Input: name=John, salary=50000, department=HR

o Output:

Name: John

Salary: 50000

Department: HR

#### Code:

class Employee:

    def \_\_init\_\_(self, name, salary):

        self.name = name

        self.salary = salary

    def display\_details(self):

        print(f"Name: {self.name}")

        print(f"Salary: {self.salary}")

class Manager(Employee):

    def \_\_init\_\_(self, name, salary, department):

        super().\_\_init\_\_(name, salary)

        self.department = department

    def display\_details(self):

        super().display\_details()

        print(f"Department: {self.department}")

# Input

name = input("Enter name: ")

salary = input("Enter salary: ")

department = input("Enter department: ")

# Create a Manager object and display details

manager = Manager(name, salary, department)

manager.display\_details()

#### Output:

A screen shot of a computer

Description automatically generated

### Question 8: Regex Validation for Email

Write a program to validate whether an input string is a valid email address using re (Regex).

o Input: [test@example.com](mailto:test@example.com)

o Output: Valid Email

#### Code:

import re

def validate\_email(email):

    pattern = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+**\.**[a-zA-Z]{2,}$'

    if re.match(pattern, email):

        return "Valid Email"

    else:

        return "Invalid Email"

# Input

email = input("Enter email: ")

# Validate email

result = validate\_email(email)

print(result)

#### Output:



### Question 9: Python String Formatting

Write a program to generate a formatted invoice using string formatting.

o Input: Item=Pen, Price=5, Quantity=10

o Output:

Item: Pen

Price: Rs. 5

Quantity: 10

Total: Rs. 50

#### Code:

def generate\_invoice(item, price, quantity):

    total = price \* quantity

    invoice = (

        f"Item: {item}\n"

        f"Price: Rs. {price}\n"

        f"Quantity: {quantity}\n"

        f"Total: Rs. {total}"

    )

    return invoice

# Input

item = input("Enter item name: ")

price = float(input("Enter price: "))

quantity = float(input("Enter quantity: "))

# Generate and print invoice

invoice = generate\_invoice(item, price, quantity)

print(invoice)

#### Output:

A screen shot of a computer

Description automatically generated

### Question 10: User Input and PIP Library Usage

Write a program to install a Python package using pip and ask the user for the package name at runtime.

o Input: Package name: numpy

o Output:

Installing numpy...

Installation successful.

#### Code:

import subprocess

def install\_package(package\_name):

    try:

        subprocess.check\_call(["pip", "install", package\_name])

        print(f"Installing {package\_name}...")

        print("Installation successful.")

    except subprocess.CalledProcessError:

        print("Installation failed.")

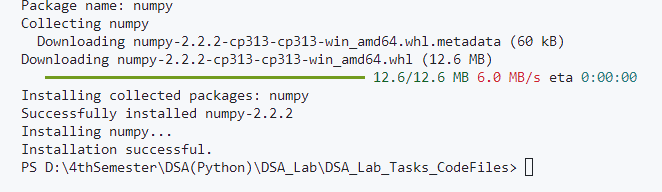
# Input

package\_name = input("Package name: ")

# Install the package

install\_package(package\_name)

#### Output:



## Advanced Problems

### Question 11: Polymorphism: Shapes Area Calculation

Create a base class Shape with a method calculate\_area().

Implement two child classes Rectangle and Circle with overridden methods to calculate the area of their respective shapes.

o Input: Rectangle: length=5, width=3; Circle: radius=4

o Output:

Rectangle Area: 15

Circle Area: 50.24

#### Code:

import math

class Shape:

    def calculate\_area(self):

        pass

class Rectangle(Shape):

    def \_\_init\_\_(self, length, width):

        self.length = length

        self.width = width

    def calculate\_area(self):

        return self.length \* self.width

class Circle(Shape):

    def \_\_init\_\_(self, radius):

        self.radius = radius

    def calculate\_area(self):

        return math.pi \* self.radius \*\* 2

# Input

length = float(input("Enter length of rectangle: "))

width = float(input("Enter width of rectangle: "))

radius = float(input("Enter radius of circle: "))

rect = Rectangle(length, width)

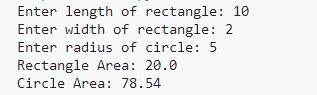
circle = Circle(radius)

# Output

print("Rectangle Area:", rect.calculate\_area())  # Output: 15

print("Circle Area:", round(circle.calculate\_area(), 2))  # Output: 50.24

#### Output:



### Question 12: Abstract Class for Payment Processing

Create an abstract class Payment with an abstract method process\_payment(). Implement subclasses CreditCardPayment and

PaypalPayment with their respective implementations.

o Input: Payment Type: Credit Card, Amount: 1000

o Output:

Processing Credit Card Payment of Rs. 1000

#### Code:

from abc import ABC, abstractmethod

**class Payment(ABC):**

    @abstractmethod

    def process\_payment(self, amount):

        pass

class CreditCardPayment(Payment):

    def process\_payment(self, amount):

        print(f"Processing Credit Card Payment of Rs. {amount}")

class PaypalPayment(Payment):

    def process\_payment(self, amount):

        print(f"Processing PayPal Payment of Rs. {amount}")

**# Input**

payment\_type = input("Enter payment type('Credit Card' or 'Pay Pal'): ")

amount = float(input("Enter amount: "))

if payment\_type == "Credit Card":

    payment = CreditCardPayment()

elif payment\_type == "PayPal":

    payment = PaypalPayment()

#### Output:



### Question 13: JSON Data Analysis

Write a program to load a JSON file containing student data (name, marks) and calculate the average marks of all students.

o Input JSON:

[

{"name": "Alice", "marks": 85},

{"name": "Bob", "marks": 90},

{"name": "Charlie", "marks": 78}

]

o Output: Average Marks: 84.33

#### Code:

import json

def calculate\_average\_marks(json\_data):

    students = json.loads(json\_data)

    total\_marks = sum(student['marks'] for student in students)

    average\_marks = total\_marks / len(students)

    return round(average\_marks, 2)

# Input JSON

json\_data = '''

[

    {"name": "Alice", "marks": 85},

    {"name": "Bob", "marks": 90},

    {"name": "Charlie", "marks": 78}

]

'''

average\_marks = calculate\_average\_marks(json\_data)

print("Average Marks:", average\_marks)

#### Output:



### Question 14: Regex Search in Log File

Write a program to extract all IP addresses from a given

server log file using regex.

o Input File Content:

192.168.1.1 - Accessed on 2025-01-19

10.0.0.2 - Accessed on 2025-01-20

o Output:

IP Addresses: 192.168.1.1, 10.0.0.2

#### Code:

import re

def extract\_ip\_addresses(file\_content):

    pattern = r'\b(?:\d{1,3}**\.**){3}\d{1,3}\b'

    ip\_addresses = re.findall(pattern, file\_content)

    return ip\_addresses

# Input File Content

log\_content = '''

192.168.1.1 - Accessed on 2025-01-19

10.0.0.2 - Accessed on 2025-01-20

'''

ip\_addresses = extract\_ip\_addresses(log\_content)

print("IP Addresses:", ", ".join(ip\_addresses))

#### Output:



### Question 15: File Handling: Merge and Sort Files

Write a program to merge two text files and sort the combined content alphabetically.

o Input File 1: "apple, banana, orange"

o Input File 2: "cherry, fig, grape"

o Output File: "apple, banana, cherry, fig, grape, orange"

#### Code:

def merge\_and\_sort\_files(file1\_path, file2\_path, output\_file\_path):

    with open(file1\_path, 'r') as file1, open(file2\_path, 'r') as file2:

        content1 = file1.read().split(', ')

        content2 = file2.read().split(', ')

    merged\_content = sorted(content1 + content2)

    with open(output\_file\_path, 'w') as output\_file:

        output\_file.write(', '.join(merged\_content))

# Input Files Content

file1\_path = 'file1.txt'

file2\_path = 'file2.txt'

output\_file\_path = 'output.txt'

with open(file1\_path, 'w') as file1:

    file1.write("apple, banana, orange")

with open(file2\_path, 'w') as file2:

    file2.write("cherry, fig, grape")

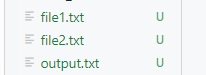
# Process and Output

merge\_and\_sort\_files(file1\_path, file2\_path, output\_file\_path)

with open(output\_file\_path, 'r') as output\_file:

    print(output\_file.read())  # Output: apple, banana, cherry, fig, grape, orange

**Files Created:**

****

#### Output:

