SET 2:

1. A program that models a bank account, with classes for the account, the customer,
and the bank.

ANS :

```
class Customer:
    def __init__(self, customer_id, name):
        self.customer_id = customer_id
        self.name = name

class BankAccount:
    def __init__(self, account_number, customer, balance=0.0):
        self.account_number = account_number
        self.customer = customer
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount}. New balance: ${self.balance}")
        else:
            print("Invalid deposit amount. Please enter a positive value.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew ${amount}. New balance: ${self.balance}")
        else:
  print("Invalid withdrawal amount or insufficient funds.")

    def get_balance(self):
        return self.balance

class Bank:
    def __init__(self, bank_name):
        self.bank_name = bank_name
        self.accounts = []

    def create_account(self, customer, initial_balance=0.0):
        account_number = len(self.accounts) + 1
        new_account = BankAccount(account_number, customer, initial_balance)
        self.accounts.append(new_account)
        print(f"Account created for {customer.name} with account number {account_number}.")
        return new_account

    def get_account_balance(self, account_number):
        account = self.find_account(account_number)
        if account:
            return account.get_balance()
        else:
            return None

    def find_account(self, account_number):
        for account in self.accounts:
            if account.account_number == account_number:
                return account
        print(f"Account with account number {account_number} not found.")
        return None

# Create a bank
my_bank = Bank("MyBank")

# Create customers
customer1 = Customer(1, "Alice")
customer2 = Customer(2, "Bob")

# Create accounts for customers
account1 = my_bank.create_account(customer1, 1000.0)
account2 = my_bank.create_account(customer2, 500.0)

# Perform transactions
account1.deposit(500)
account2.withdraw(200)

# Check account balances
balance1 = my_bank.get_account_balance(account1.account_number)
balance2 = my_bank.get_account_balance(account2.account_number)

print("Account", account1.account_number, "balance:", balance1)
print("Account", account2.account_number, "balance:", balance2)
```

OUTPUT :

```
Account created for Alice with account number 1.
Account created for Bob with account number 2.
Deposited $500. New balance: $1500.0
Withdrew $200. New balance: $300.0
Account 1 balance: 1500.0
Account 2 balance: 300.0
```

2. A program that simulates a school management system, with classes for the students, the teachers, and the courses

ANS :

```
class Student:
    def __init__(self, student_id, name):
        self.student_id = student_id
        self.name = name

class Teacher:
    def __init__(self, teacher_id, name):
        self.teacher_id = teacher_id
        self.name = name

class Course:
    def __init__(self, course_code, course_name, teacher):
        self.course_code = course_code
        self.course_name = course_name
        self.teacher = teacher

# Create students
student1 = Student(1, "Alice")
student2 = Student(2, "Bob")

# Create teachers
teacher1 = Teacher(101, "Mr. Smith")
teacher2 = Teacher(102, "Mrs. Johnson")

# Create courses
course1 = Course("MATH101", "Mathematics", teacher1)
course2 = Course("ENG101", "English", teacher2)

# Display information
print(f"Student: {student1.name}, ID: {student1.student_id}")
print(f"Teacher: {teacher1.name}, ID: {teacher1.teacher_id}")
print(f"Course: {course1.course_name}, Code: {course1.course_code}, Teacher:
{course1.teacher.name}")
```

OUTPUT :

```
Student: Alice, ID: 1
Teacher: Mr. Smith, ID: 101
Course: Mathematics, Code: MATH101, Teacher: Mr. Smith
```

3. A program that reads a text file and counts the number of words in it.

ANS :

```
def count_words(file_path):
    try:
        with open(file_path, 'r') as file:
            content = file.read()
            words = content.split()
            return len(words)
    except FileNotFoundError:
        print("File not found.")
        return None

# Example: Count words in a text file
file_path = input("Enter the path to the text file: ")

word_count = count_words(file_path)

if word_count is not None:
    print(f"The file '{file_path}' contains {word_count} words.")
```

OUTPUT :

```
Enter the path to the text file:main.py
 The file 'main.py' contains 55 words.

Enter the path to the text file: script1.py
The file 'script1.py' contains 17 words.

Enter the path to the text file: happy.py
The file 'happy.py' contains 4 words.

Enter the path to the text file:abc.py
 The file 'abc.py' contains 26 words.
```

4 . A program that reads a CSV file and calculates the average of the values in a specified column.

ANS :

```
def read_csv_and_calculate_averages(filename):
  """Reads a CSV file, calculates averages for each row, and prints results."""

  try:
      with open(filename, "r") as file:
 lines = file.readlines()

          # Process each line as a list of integers
          values = [list(map(int, line.strip().split(","))) for line in lines]

          # Calculate and print averages for each row
          for i, row in enumerate(values):
              average = sum(row) / len(row)
              print(f"Average of row {i+1}: {average:.2f}")

  except FileNotFoundError:
      print(f"Error: File '{filename}' not found.")
  except PermissionError:
      print(f"Error: Permission denied for file '{filename}'.")
  except ValueError:
      print(f"Error: Invalid data format in file '{filename}'.")

# Example usage:
filename = "S2_Q4.csv"  # Replace with your actual file name
read_csv_and_calculate_averages(filename)

# S2_Q4.csv file consists :
```

```
1,2,3,4,5

1,3,4,5

1,5,9

10,20,40

30,40,60,78
```

OUTPUT :

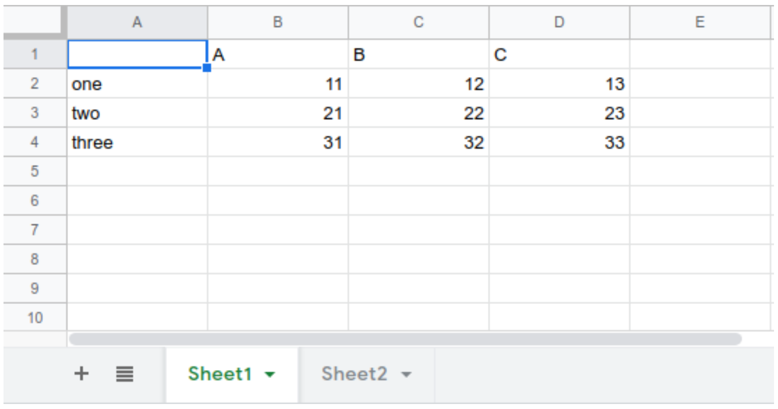
```
Average of row 1: 3.00
Average of row 2: 3.25
Average of row 3: 5.00
Average of row 4: 23.33
Average of row 5: 52.00
```

5. A program that reads an Excel file and prints the data in a tabular format.

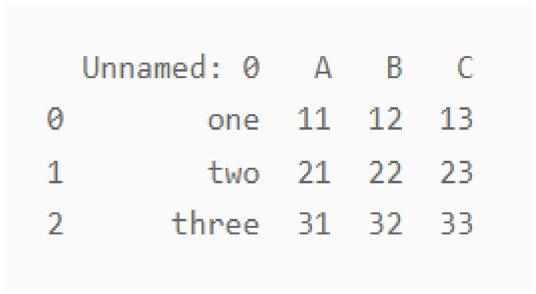
ANS :

```
import pandas as pd

def print_excel_data(excel_file):
    try:
        df = pd.read_excel(excel_file)
        print("Data in tabular format:")
        print(df)
    except FileNotFoundError:
        print("File not found.")
    except pd.errors.EmptyDataError:
        print("The Excel file is empty.")
    except Exception as e:
        print(f"An error occurred: {e}")

excel_file_path = input("Enter the path to the Excel file (.xlsx): ")
print_excel_data(excel_file_path)
```

OUTPUT :



	A	B	C	D	E
1		A	B	C	
2	one	11	12	13	
3	two	21	22	23	
4	three	31	32	33	
5					
6					
7					
8					
9					
10					



```
  Unnamed: 0   A   B   C
0        one  11  12  13
1        two  21  22  23
2      three  31  32  33
```