```python
class Book:
    Library_name = "National Library"
    def __init__(self, bookID, title, price, author, pages, publication_date,
year):
        self.__bookID = bookID
        self.title = title
        self.price = price
        self.author = author
        self.pages = pages
        self.publication_date = publication_date
        self.year = year

    def get_bookID(self):
        return self.__bookID

    def set_bookID(self, new_id):
        self.__bookID = new_id

    def __repr__(self):
        return f"library_name:{self.Library_name}, bookID:{self.__bookID},
title:{self.title}, price:{self.price}, author:{self.author}, pages:{self.pages},
publication_date:{self.publication_date}, year:{self.year}"


    def add_book(self):

        pass

    def book_details(self):

        print(f"Book ID: {self.__bookID}")
        print(f"Title: {self.title}")
        print(f"Author: {self.author}")
        print(f"Price: ${self.price}")
        print(f"Pages: {self.pages}")
        print(f"Publication Date: {self.publication_date}")
        print(f"Year: {self.year}")

    def change_book_id(self, new_id):
        self.__bookID = new_id
        print(f"Book ID updated to: {new_id}")
```

```
class Ebook(Book):
    def __init__(self, bookID, title, price, author, pages, publication_date,
year, format, size):
        super().__init__(bookID, title, price, author, pages, publication_date,
year)
        self.format = format
        self.size = size

    def __repr__(self):
        return f"library_name:{self.Library_name}, bookID:{self.get_bookID()},
title:{self.title}, price:{self.price}, author:{self.author}, pages:{self.pages},
publication_date:{self.publication_date}, year:{self.year}, format:{self.format},
size:{self.size}"


    def add_book(self):
        super().add_book()
        self.file_size = float(input("Enter the file size of the book: "))
        self.number_of_copies = int(input("Enter the number of copies: "))
        self.format = input("Enter the format of the ebook (PDF, EPUB, etc.): ")


    def book_details(self):
        print(f"File Size: {self.file_size} MB")
        print(f"Format: {self.format}")
        print(f"Total Size: {self.compute_file_size()} MB")


    def compute_file_size(self):
        self.total_size = self.file_size * self.number_of_copies
        return self.total_size


    def change_book_id(self, new_id):
        super().change_book_id(new_id)

class National_Library(Ebook):
    def __init__(self):

        super().__init__(0, "Library Catalog", 0, "System", 0, "01/01", 2023,
"N/A", 0)
        self.books = []

    def add_book(self):
        bookID = int(input("Enter Book ID: "))
```

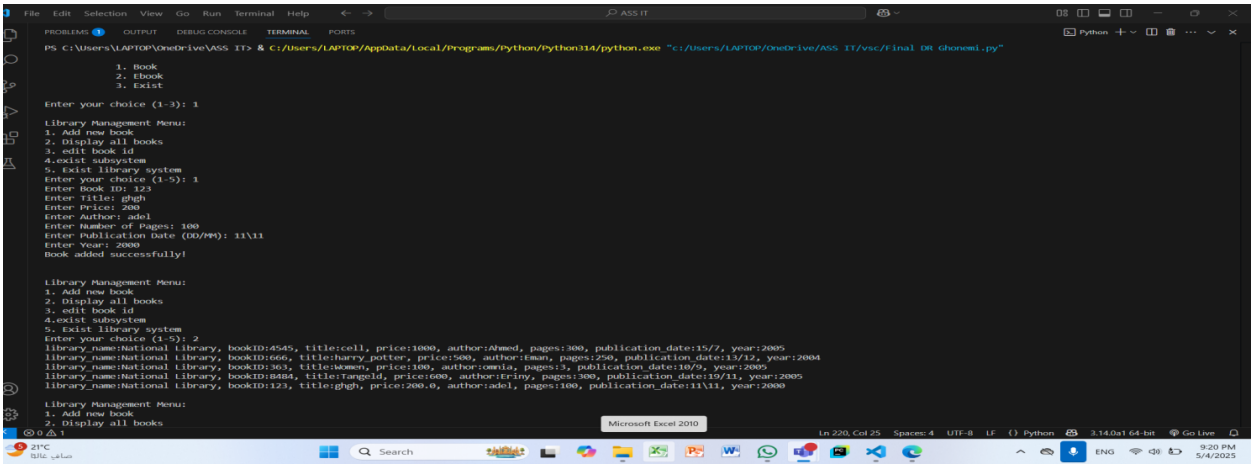
```
        title = input("Enter Title: ")
        price = float(input("Enter Price: "))
        author = input("Enter Author: ")
        pages = int(input("Enter Number of Pages: "))
        publication_date = input("Enter Publication Date (DD/MM): ")
        year = int(input("Enter Year: "))

        new_book = Book(bookID, title, price, author, pages, publication_date,
year)
        self.books.append(new_book)
        print("Book added successfully!\n")

    def display_books(self):
        if not self.books:
            print("No books in the library.\n")
        else:
            for book in self.books:
                print(book)

    def edit_bookid(self):

        new_id = int(input("Enter new book ID: "))
        self.set_bookID(new_id)
        print(f"Library catalog book ID updated to: {new_id}")
        return new_id

    def menu(self):
        while True:
            print("\nLibrary Management Menu:")
            print("1. Add new book")
            print("2. Display all books")
            print("3. edit book id")
            print("4.exist subsystem")
            print("5. Exist library system")

            option = int(input("Enter your choice (1-5): "))

            if option == 1:
                self.add_book()
            elif option == 2:
                self.display_books()
            elif option == 3:
                self.edit_bookid()
            elif option == 4:
                print("exist subsystem")
```

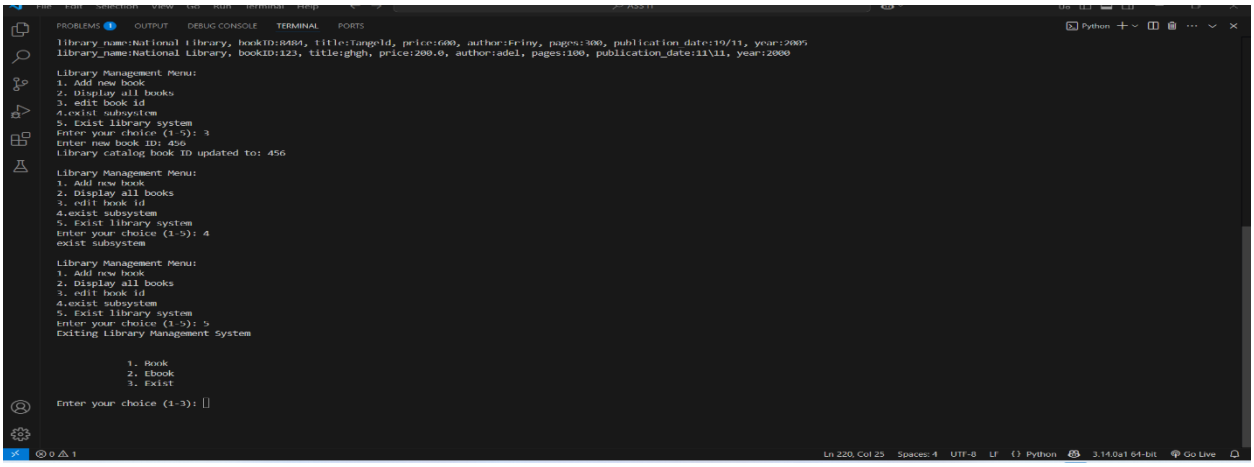
```
            elif option == 5:
                print("Exiting Library Management System\n")
                break
            else:
                print("Invalid option. Please try again.\n")

    def ebook_subsystem(self):

        if not hasattr(self, 'file_size'):
            self.file_size = 0
        if not hasattr(self, 'number_of_copies'):
            self.number_of_copies = 0

        while True:
            print("""EBOOK SUBSYSTEM MENU:
            1. Add new EBook
            2. Display EBook Details
            3. Change/Edit file_size
            4. Check file_size and format
            5. Compute total file size
            6. Exit the sub-system
            7. Exit the Library Management System
            """)

            option = int(input("Enter your choice (1-7): "))
            if option == 1:
                self.add_book()
                self.file_size = float(input("Enter the file size of the book:
"))
                self.number_of_copies = int(input("Enter the number of copies:
"))
                self.format = input("Enter the format of the ebook (PDF, EPUB,
etc.): ")
                print("EBook added successfully!")

            elif option == 2:
                self.book_details()

            elif option == 3:
                new_size = float(input("Enter new file size (MB): "))
                self.file_size = new_size
                print(f"File size updated to {new_size} MB")

            elif option == 4:
                print(f"File Size: {self.file_size} MB")
```

```
                print(f"Format: {self.format}")
                self.book_details()

            elif option == 5:
                copies = int(input("Enter number of copies: "))
                self.number_of_copies = copies
                total = self.compute_file_size()
                print(f"Total file size for {copies} copies: {total} MB")

            elif option == 6:
                print("Exiting EBook subsystem...")
                return

            elif option == 7:
                print("Exiting Library Management System...")
                break
            else:
                print("Invalid option! Please enter a number between 1-7.")

    def subsystem_menu(self):
        while True:
            print("""
            1. Book
            2. Ebook
            3. Exist
                """)

            option = int(input("Enter your choice (1-3): "))

            if option == 1:
                self.menu()
            elif option == 2:
                self.ebook_subsystem()

            elif option == 3:
                print("Exist")
            else:
                print("Invalid option! Please enter a number between 1-
3")


book1=Book(4545,'cell',1000,'Ahmed',300,"15/7",2005)
book2=Book(666,'harry_potter',500,'Eman',250,"13/12",2004)
book3=Book(363,'Women',100,'omnia',3,"10/9",2005)
book4=Book(8484,'Tangeld',600,'Eriny',300,"19/11",2005)
```

```python
book5=Ebook(7777,'Digital World',299,'Daniel',400,"25/5",2022,"PDF",15.5)

library = National_Library()

library.books.append(book1)
library.books.append(book2)
library.books.append(book3)
library.books.append(book4)

library.subsystem_menu()
```

The output:






1. Book
2. Ebook
3. Exist
Enter your choice (1-3): 2
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System
Enter your choice (1-7): 1
Enter Book ID: 908
Enter Title: lklk
Enter Price: 300
Enter Author: dr
Enter Number of Pages: 90
Enter Publication Date (DD/MM): 12\12
Enter Year: 2001
Book added successfully!
Enter the file size of the book: 20
Enter the number of copies: 10
Enter the format of the ebook (PDF, EPUB, etc.): pdf
EBook added successfully!
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System
Enter your choice (1-7): 2
File Size: 20.0 MB
Format: pdf
Enter your choice (1-7): 2
File Size: 20.0 MB
Format: pdf
Total Size: 200.0 MB
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System
Enter your choice (1-7): 3
Enter new file size (MB): 30
File size updated to 30.0 MB
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System
Enter your choice (1-7): 4
File Size: 30.0 MB
Format: pdf
File Size: 30.0 MB
Format: pdf
Total Size: 300.0 MB
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System

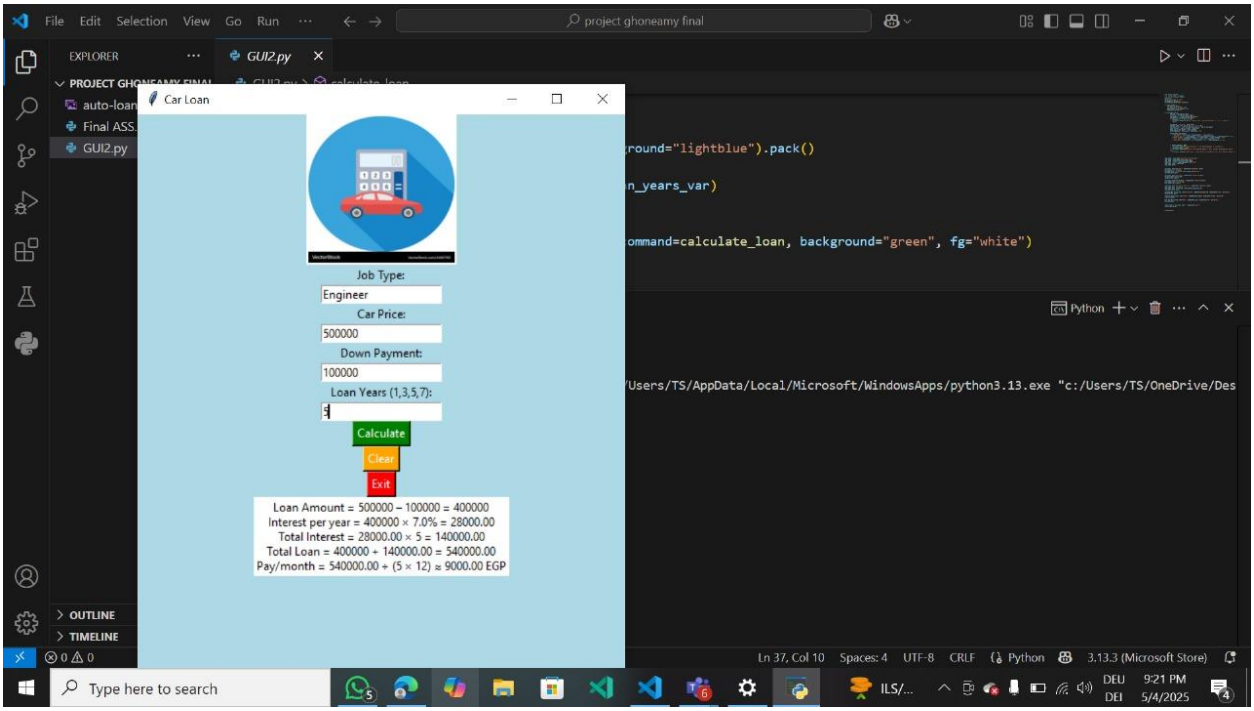

Enter your choice (1-7): 4
File Size: 30.0 MB
Format: pdf
File Size: 30.0 MB
Format: pdf
Total Size: 300.0 MB
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System
Enter your choice (1-7): 5
Enter number of copies: 40
Total file size for 40 copies: 1200.0 MB
EBOOK SUBSYSTEM MENU:
1. Add new EBook
2. Display EBook Details
3. Change/Edit file_size
4. Check file_size and format
5. Compute total file size
6. Exit the sub-system
7. Exit the Library Management System
Enter your choice (1-7): 6
Exiting EBook subsystem...
1. Book
2. Ebook
4. Exist
Enter your choice (1-3): 3
Exist
1. Book
2. Ebook
3. Exist

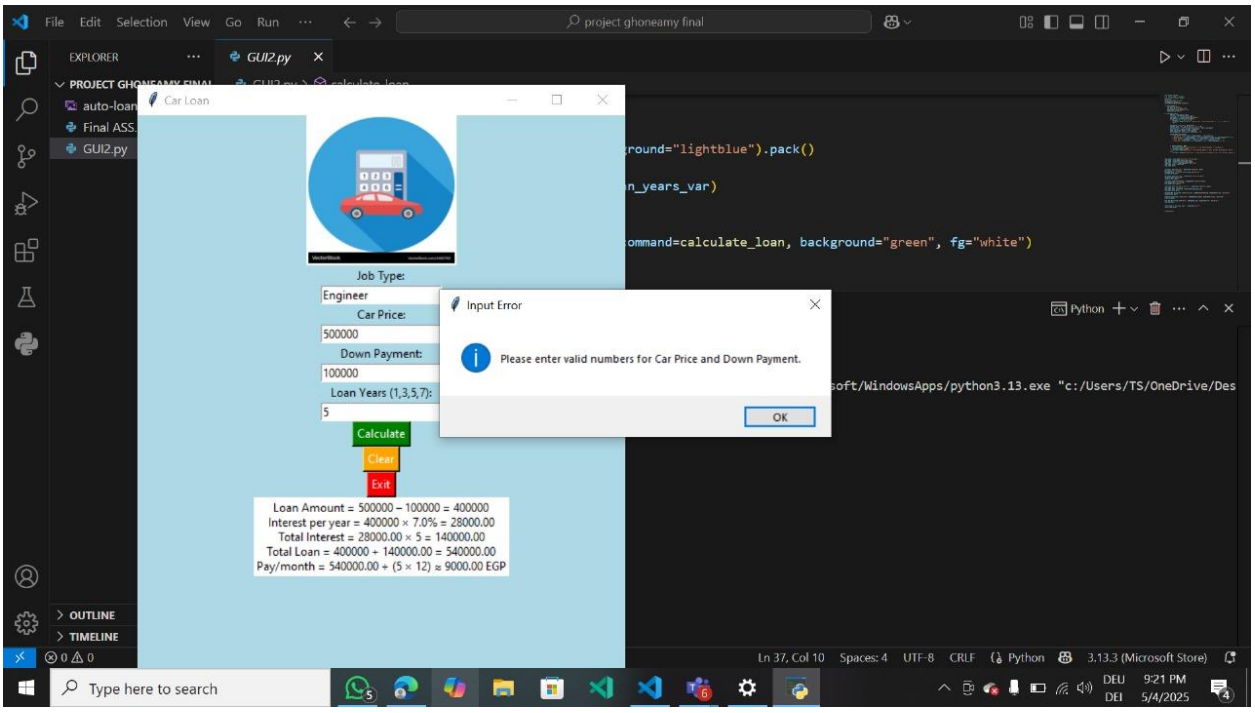
```
from tkinter import*
from tkinter import messagebox
from PIL import Image, ImageTk
top = Tk()
top.title(string='Car Loan')
top.geometry('500x600')
top.configure(background="lightblue")

def clear_inputs():
    job_type_var.set("")
    car_price_entry.delete(0, END)
    down_payment_entry.delete(0, END)
    loan_years_var.set("")

def calculate_loan():
        job_type = job_type_var.get()
        car_price = int(car_price_entry.get())
        down_payment = int(down_payment_entry.get())
        loan_years = int(loan_years_var.get())
        if loan_years not in [1, 3, 5, 7]:
            messagebox.showerror("Error", "Please enter a valid loan period: 1,
3, 5, or 7 years.")
            return

        loan_amount = car_price - down_payment
        interest_rates = {1: 5.5, 3: 6.2, 5: 7.0, 7: 7.5}
        yearly_interest = (interest_rates[loan_years] / 100) * loan_amount
        total_interest = yearly_interest * loan_years
        total_loan = loan_amount + total_interest
        monthly_payment = total_loan / (loan_years * 12)

        result_label.config(text=
            f"Loan Amount = {car_price} - {down_payment} = {loan_amount}\n"
            f"Interest per year = {loan_amount} × {interest_rates[loan_years]}% =
{yearly_interest:.2f}\n"
            f"Total Interest = {yearly_interest:.2f} × {loan_years} =
{total_interest:.2f}\n"
            f"Total Loan = {loan_amount} + {total_interest:.2f} =
{total_loan:.2f}\n"
            f"Pay/month = {total_loan:.2f} ÷ ({loan_years} × 12) ≈
{monthly_payment:.2f} EGP"
        )
```

```
        if monthly_payment < 5000:
            messagebox.showinfo("Congratulations!", "Your monthly payment is
affordable!")
        elif monthly_payment > 15000:
            messagebox.showinfo("Warning", "Your monthly payment is high,
consider adjusting your loan!")
        else:
            messagebox.showinfo("Input Error", "Please enter valid numbers for
Car Price and Down Payment.")




logo_image = Image.open("auto-loan-calculator.png")
logo_image = logo_image.resize((150, 150))
logo_photo = ImageTk.PhotoImage(logo_image)
logo_label = Label( image=logo_photo, bg="white")
logo_label.image = logo_photo
logo_label.pack()


Label(top, text="Job Type:", background="lightblue").pack()
job_type_var = StringVar()
job_type_entry = Entry(top, textvariable=job_type_var)
job_type_entry.pack()

Label(top, text="Car Price:", background="lightblue").pack()
car_price_entry =Entry(top)
car_price_entry.pack()

Label(top, text="Down Payment:", background="lightblue").pack()
down_payment_entry =Entry(top)
down_payment_entry.pack()

Label(top, text="Loan Years (1,3,5,7):", background="lightblue").pack()
loan_years_var = StringVar()
loan_years_entry = Entry(top, textvariable=loan_years_var)
loan_years_entry.pack()

calculate_btn =Button(top, text="Calculate", command=calculate_loan,
background="green", fg="white")
calculate_btn.pack()

clear_btn =Button(top, text="Clear", command=clear_inputs, background="orange",
fg="white")
```

```
clear_btn.pack()

exit_btn =Button(top, text="Exit", command=top.quit, background="red", fg="white"
)
exit_btn.pack()


result_label = Label(top, text="", background="white")
result_label.pack()


top.mainloop()
```

The output:




File Edit Selection View Go Run
project ghoneamy final
EXPLORER
GUI2.py
PROJECT GHONEAMY FINAL
auto-loan
Final ASS...
GUI2.py
round="lightblue").pack()
n_years_var)
ommand=calculate_loan, background="green", fg="white")
Car Loan
Job Type:
Engineer
Car Price:
500000
Down Payment:
100000
Loan Years (1,3,5,7):
5
Calculate
Clear
Exit
Loan Amount = 500000 – 100000 = 400000
Interest per year = 400000 × 7.0% = 28000.00
Total Interest = 28000.00 × 5 = 140000.00
Total Loan = 400000 + 140000.00 = 540000.00
Pay/month = 540000.00 ÷ (5 × 12) ≈ 9000.00 EGP
Input Error
Please enter valid numbers for Car Price and Down Payment.
OK
Python
soft/WindowsApps/python3.13.exe "c:/Users/TS/OneDrive/Des
OUTLINE
TIMELINE
Ln 37, Col 10 Spaces: 4 UTF-8 CRLF Python 3.13.3 (Microsoft Store)
Type here to search
DEU
DEI
9:21 PM
5/4/2025