Module Title

Principles of Programming

Weekly Assignment - Practical 8

Year

2025

Student Name: NIRVIK K.C.

UWE ID: 25024649

Assignment Submission Date: January 15, 2026

This assignment consists of the programming questions from practical exercises related to the topics from week 8.

Class design and more programming on Objects and Classes

Objective:

To provide students with the ability to design simple classes and write programs in Python and make use of the concepts of Object-Oriented Programming, public/private variables, Mutable, Immutable, and List of objects.

Questions

Exercise 1: Designing, implementing, and testing Python Classes

You have to design and implement TWO new classes (Exercise 2). You should choose them from the list below. One example considering the computer class is show in this exercise.

Circle Dog Shoe Book Song Bicycle Car Computer Movie

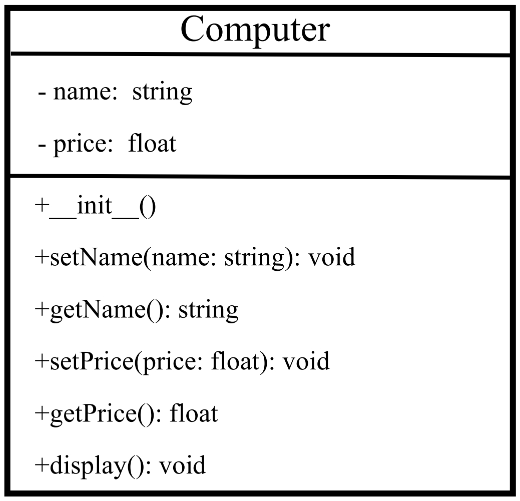
1.

Task 1.1: Study and practice the design and implementation of the Computer class.

Answer:

A UML diagram for the Computer class illustrates the structure of the class, which consists of a class name, attributes, and operations. All the attributes are private while all the methods are public.

Some constraints include the price as a float variable and may have a default value of 0.0, with minimum and maximum values of 99.99 and 999.99 respectively. The string variable name of the Computer may set a default empty string(‘’), and a minimum (say 2 characters) and maximum (say 10 characters) length of the string.



Note: price: default value is 0.0, min

value is 99.99 and max value is 999.99

name: default value ‘ ’, min length 2 and max length 10

2.

Task 1.2: (Public member variables) Study the following implementation of the Computer class designed above. Create a ComputerA.py file. Type, compile and run the code. Note that in this implementation all the member variables are declared as public (like your previous lab exercises). Therefore, they are accessible from outside the Class through an object of the class.

Given code:

```
  # parameterised constructor


  def __init__(self, name, price):

    # initilialising instance/member variable age

    self.name = name

    self.price = price


  #setters

  def setName(self, name):

   self.name = name

  def setPrice(self, price):

   self.price = price
```

```
    #getters
    def getName(self):
      return self.name


    def getPrice(self):
      return self.price


    # An instance/member method
    def display(self):
        print("Computer's name is: " +str(self.getName())+ " and its price is:
 "+str(self.getPrice()))


# User provides a valid computer name
isValidName = False
while not isValidName:
    computername = str(input("Enter computer's name: "))
    if 2 <= len(computername) <= 10:
            isValidName = True
```

```
else:
        print("You must enter a valid name between 2 and 10 chars length")

# User provides a valid computer price
isValidPrice = False
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))

  if 99.99 <= computerprice <= 999.99:
        isValidPrice = True
  else:
        print("You must enter a valid price between 99.99 and 999.99")

#creating the comp object based on user's input
comp = ComputerA(computername, computerprice)

# displaying computer’s information
comp.display()
```

```
#setting a different computer name using the setName method
comp.setName(“Dell”)

#setting new price using the setPrice method
comp.setPrice(890.90)

# displaying computer's (new) information
comp.display()
```

If you can successfully run your program, you should see output like this (e.g., while providing

name/price as HP/199.99):

```
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: Dell and its price is: 890.9
```

Answer:

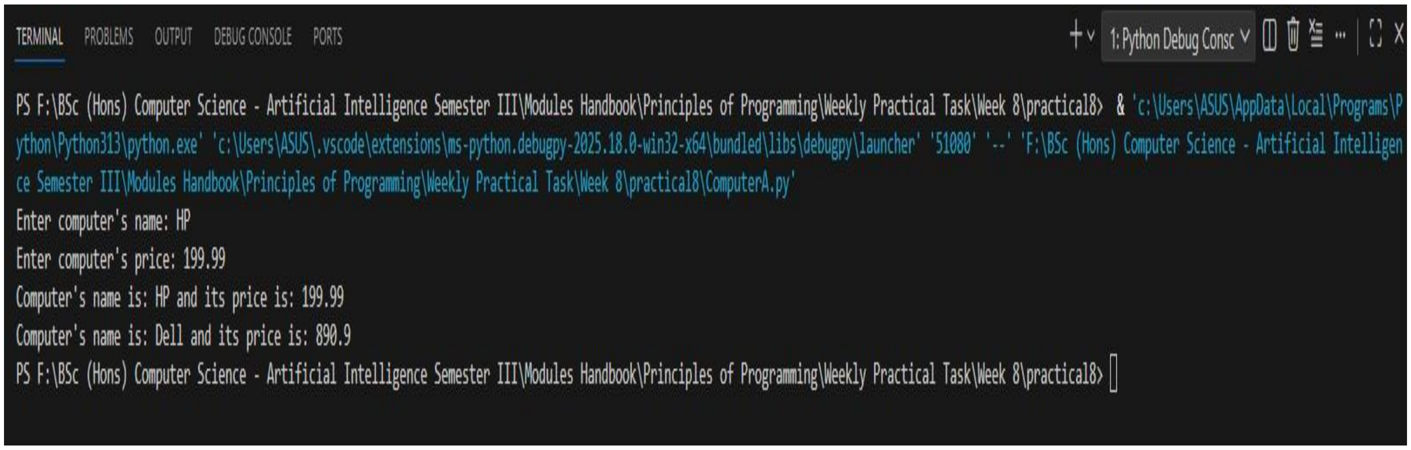
Following code for input:

```
# Implementation of the Computer class
# Practical 8 - Exercise 1 (Task 1.2)
# Nirvik K.C.
class ComputerA:
    # Parameterized constructor
    def __init__(self, name, price):
        # Public instance variables which can be accessed/modified directly from
outside)
        self.name = name
        self.price = price

    # Using setters method
    def setName(self, name):
        self.name = name
    def setPrice(self, price):
        self.price = price
    # Using getters
    def getName(self):
        return self.name


    def getPrice(self):
        return self.price


    # Display the current name and price of the Computer
    # An instance/member method
    def display(self):
        print(f"Computer's name is: {self.getName()} and its price is:
{self.getPrice()}")


# User provides a valid computer name
isValidName = False
while not isValidName:
    computername = input("Enter computer's name: ")
```

```
if 2 <= len(computername) <= 10:
        isValidName = True
    else:
        print("You must enter a value name between 2 and 10 characters long")


# User provides a valid computer price
isValidPrice = False
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))
    if 99.99 <= computerprice <= 999.99:
        isValidPrice = True
    else:
        print("You must enter a valid price between 99.99 and 999.99")


# Creating the comp object based on user's input
comp = ComputerA(computername, computerprice)
```

```
# Displaying computer's information
comp.display()


# Setting a different computer name using the setName method
comp.setName("Dell")


# Setting new price using the setPrice method
comp.setPrice(890.90)


# Displaying computer's (new) information
comp.display()
# Output:
# Enter computer's name: HP
# Enter computer's price: 199.99
# Computer's name is: HP and its price is: 199.99
# Computer's name is: Dell and its price is: 890.9
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>  & 'c:\Users\ASUS\AppData\Local\Programs\P
ython\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51080' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligen
ce Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\ComputerA.py'
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: Dell and its price is: 890.9
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8> 
```

3.

Task 1.3: The code below is slightly modified. Instead of calling the setName() and setPrice() methods, the new name and price are set directly by accessing the member variables. Modify your code accordingly, compile and run it. Your program should produce the same output (given the same input as before). Explain to your tutors why this is the case.

Given code:

class ComputerA:

    # parameterised constructor

```
def __init__(self, name, price):
        # initilialising instance/member variable age
        self.name = name
        self.price = price

    #setters
    def setName(self, name):
      self.name = name

    def setPrice(self, price):
      self.price = price

    #getters
    def getName(self):
      return self.name

   def getPrice(self):
      return self.price
```

```
 # an instance/member method
  def display(self):
      print("Computer's name is: " +str(self.getName())+ " and its price is:
"+str(self.getPrice()))


# User provides a valid computer name
isValidName = False


while not isValidName:
  computername = str(input("Enter computer's name: "))


  if 2 <= len(computername) <= 10:
        isValidName = True
  else:
        print("You must enter a valid name between 2 and 10 chars length")


# User provides a valid computer price
isValidPrice = False
```

```
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))


    if 99.99 <= computerprice <= 999.99:
        isValidPrice = True
    else:
        print("You must enter a valid price between 99.99 and 999.99")


#creating the comp object based on user's input
comp = ComputerA(computername,computerprice)


#displaying computer's information
comp.display()


#setting a different computer name using the setName method
#comp.setName("Dell")


#setting a different computer name by accessing the name variable which is public
comp.name = 'Dell'
```

```
# setting new price using the setPrice method
# comp.setPrice(890.90)

# setting new price by accessing the price variable which is public
comp.price = 890.90

# displaying computer's (new) information
comp.display()
```

If you can successfully run your program, you should see output like this (e.g., while providing name/price as HP/199.99):

```
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: Dell and its price is: 890.9
```

Answer:

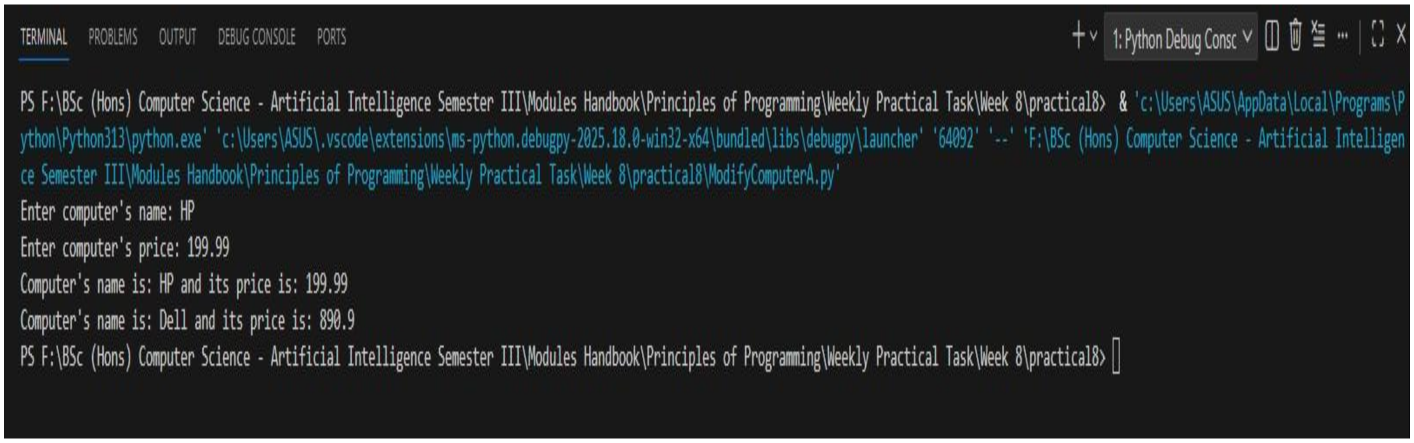
Following code for input:

```
# Public member variables (direct access/modification)
# Practical 8 - Exercise 1 (Task 1.3)
# Nirvik K.C.
class ComputerA:
    # Parameterized constructor
    def __init__(self, name, price):
        self.name = name
        self.price = price

    # Using setters method
    def setName(self, name):
        self.name = name

    # Using getters
    def getName(self):
        return self.name

    def getPrice(self):
        return self.price
    # Display the current name and price of the Computer
    # An instance/member method
    def display(self):
        print(f"Computer's name is: {self.getName()} and its price is:
{self.getPrice()}")

# User provides a valid computer name
isValidName = False
while not isValidName:
    computername = input("Enter computer's name: ")

    if 2 <= len(computername) <= 10:
        isValidName = True
    else:
        print("You must enter a value name between 2 and 10 characters long")

# User provides a valid computer price
isValidPrice = False
```

```
while not isValidPrice:

    computerprice = float(input("Enter computer's price: "))

    if 99.99 <= computerprice <= 999.99:

        isValidPrice = True

    else:

        print("You must enter a valid price between 99.99 and 999.99")


# Creating the comp object based on user's input

comp = ComputerA(computername, computerprice)


# displaying computer's information

comp.display()


# setting a different computer name using the setName method

# comp.setName("Dell")


# setting a different computer name by accessing the name variable which is public

comp.name = 'Dell'
```

```
# setting new price using the setPrice method

# comp.setPrice(890.90)


# Displaying computer's (new) information

comp.display()

# Output:

# Enter computer's name: HP

# Enter computer's price: 199.99

# Computer's name is: HP and its price is: 199.99

# Computer's name is: Dell and its price is: 890.9
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>  & 'c:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64092' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\ModifyComputerA.py'
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: Dell and its price is: 890.9
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>
```

Q) Why does the program produce the same output even when don’t use setName() and setPrice()?

Ans: In this version of the ComputerA class, the member variables self.name and self.price are public. When they are public, we can read them directly using print(comp.name) or comp.getName(). We can also change them directly using comp.name = “Dell” or comp.price = 890.00.

We are modifying the same variables that were initialized in the constructor and read by getName() and getPrice(). The display method uses the getters, which return the current values of self.name and self.price. Those values were changed directly by us, the getters now return the new values (“Dell” and 890.9) and the output is exactly the same as if we had setName() and setPrice().

4.

Task 1.4: (Private member variables) Private members are only accessible from within the class. No outside access is allowed. We now make all the member variables as private. The addition of prefix __(double underscore) results in a member variable or function becomes private. Copy your ComputerA.py program and save it as ComputerB.py. In your ComputerB.py program make both the member variables as private. Save, compile and run your program. Explain to your tutors the output that you see as a result of this modification. Given code:

```
class ComputerB:
    # parameterised constructor
    def __init__(self, name, price):
        # initilialising instance/member variable age
        self.__name = name  #private variable
        self.__price = price  #private variable

    #setters
    def setName(self, name):
      self.__name = name

    def setPrice(self, price):
      self.__price = price

    #getters
    def getName(self):
      return self.__name
```

```
 def getPrice(self):
  return self.__price


 # an instance/member method
 def display(self):
    print("Computer's name is: " +str(self.getName())+ " and it's price is
: "+str(self.getPrice()))


# User provides a valid computer name
isValidName = False
while not isValidName:
 computername = str(input("Enter computer's name: "))


 if 2 <= len(computername) <= 10:
        isValidName = True
 else:
        print("You must enter a valid name between 2 and 10 chars length")
```

```
# User provides a valid computer price
isValidPrice = False
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))

    if 99.99 <= computerprice <= 999.99:
            isValidPrice = True
    else:
            print("You must enter a valid price between 99.99 and 999.99")

# creating the comp object based on user's input
comp = ComputerB(computername,computerprice)

# displaying computer's information
comp.display()

# setting a different computer name using the setName method
# comp.setName("Dell")
# setting a different computer name by accessing the name variable which is public
comp.__name = 'Dell'

# setting new price using the setPrice method
#comp.setPrice(890.90)

# setting new price by accessing the price variable which is public
comp.__price = 890.90

#displaying computer's (new) information
comp.display()
```

If you can successfully run your program, you should see output like this (e.g., while providing

name/price as HP/199.99):

```
Enter computer's name: HP

Enter computer's price: 199.99

Computer's name is: HP and it's price is: 199.99

Computer's name is: HP and it's price is: 199.99
```

Answer:

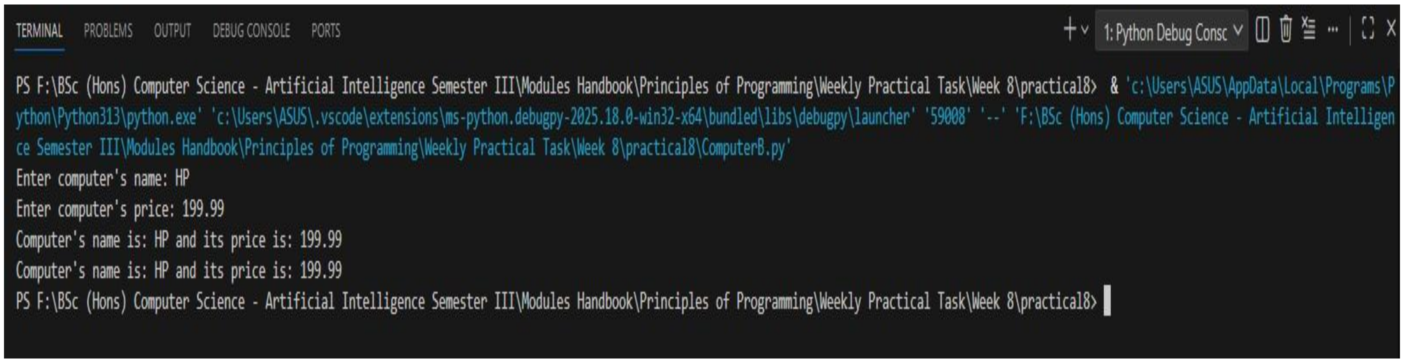
Following code for input:

```
# Accessing private member variables

# Practical 8 - Exercise 1 (Task 1.4)

# Nirvik K.C.

class ComputerB:

    # Parameterized constructor

    def __init__(self, name, price):

        # Private instance variables (Using double underscore)

        self.__name = name

        self.__price = price


    # Using setters method

    def setName(self, name):

        self.__name = name


    def setPrice(self, price):

        self.__price = price
    # Using getters method
    def getName(self):
        return self.__name


    def getPrice(self):
        return self.__price


    # Display method
    def display(self):
        print(f"Computer's name is: {self.getName()} and its price is:
{self.getPrice()}")


# User provides a valid computer name
isValidName = False
while not isValidName:
    computername = input("Enter computer's name: ")
    if 2 <= len(computername) <= 10:
        isValidName = True
    else:
        print("You must enter a value name between 2 and 10 characters long")


# User provides a valid computer price
isValidPrice = False
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))
    if 99.99 <= computerprice <= 999.99:
        isValidPrice = True
    else:
        print("You must enter a valid price between 99.99 and 999.99")


 # Creating the comp object based on user's input
comp = ComputerB(computername,computerprice)
```

```
# Displaying computer's information

comp.display()


# Setting a different computer name using the setName method

#comp.setName("Dell")


# Setting a different computer name by accessing the name variable which is
public

comp.__name = 'Dell'


# Setting new price using the setPrice method

#comp.setPrice(890.90)


# Setting new price by accessing the price variable which is public

comp.__price = 890.90


# Displaying computer's (new) information

comp.display()
```

```
# Output:

# Enter computer's name: HP

# Enter computer's price: 199.99

# Computer's name is: HP and its price is: 199.99

# Computer's name is: HP and its price is: 199.99
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>  & 'c:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59008' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\ComputerB.py'
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: HP and its price is: 199.99
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>
```

Why the output stayed the same - Computer's name is: HP and its price is: 199.99 ?

Ans: When we used name and price as private variables by using double underscore (__name and __price), the real names internal names become _ComputerB__name and _ComputerB__price. This make them private member variables which cannot be accessed or changed directly from ouside the class using comp.__name.

It creates two new public variables called __name and __price on the object (with double underscore but they are not the same as the private variables) as python does not modify the private variables. So, the private variables (_ComputerB__name and _ComputerB__price) stay unchanged. The getters (getName() and getPrice()) still return the original values which is displayed by the display() method.

5.

Task 1.5: (Private member variables are accessed using public methods). Instead of accessing the variables from outside of the class, call the setName() and getName() methods in your ComputerB.py program (like you did in Task 1.2). Save, compile and run your code and explain the output to your tutors.

Answer:

Following code for input:

```
# Private member variables accessed via public methods

# Practical 8 - Exercise 1 (Task 1.5)

# Nirvik K.C.
```

```
class ComputerB:
    # Parameterized constructor
    def __init__(self, name, price):
        # Private instance variables (Using double underscore)
        self.__name = name
        self.__price = price

    # Using Setters - public methods to change private variables
    def setName(self, name):
        self.__name = name

    def setPrice(self, price):
        self.__price = price
    # Using Getters - public methods to read private variables
    def getName(self):
        return self.__name

    def getPrice(self):
        return self.__price

    # Display method - using getters to access the variables
    def display(self):
        print(f"Computer's name is: {self.getName()} and its price is:
{self.getPrice()}")

# User provides a valid computer name
isValidName = False
while not isValidName:
    computername = input("Enter computer's name: ")
    if 2 <= len(computername) <= 10:
        isValidName = True
    else:
        print("You must enter a value name between 2 and 10 characters long")

# User provides a valid computer price
isValidPrice = False
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))
    if 99.99 <= computerprice <= 999.99:
        isValidPrice = True
    else:
        print("You must enter a valid price between 99.99 and 999.99")

# Creating the comp object based on user's input
comp = ComputerB(computername, computerprice)
```

```
# Displaying computer's information
comp.display()

# Setting a different computer name using the setName method
comp.setName("Dell")

# Setting a different computer name by accessing the name variable which is public
# comp.__name = ‘Dell’

# Setting new price using the setPrice method
comp.setPrice(890.90)

# Setting new price by accessing the price variable which is public
# comp.__price = 890.90

# Displaying computer’s (new) information
comp.display()
```

# Output:

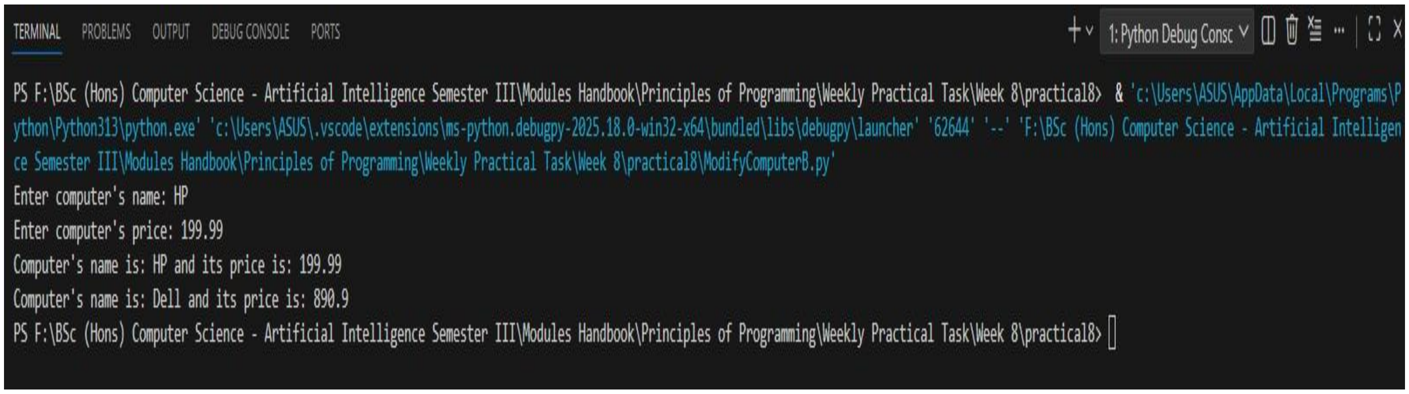
# Enter computer's name: HP

# Enter computer's price: 199.99

# Computer's name is: HP and its price is: 199.99

# Computer's name is: Dell and its price is: 890.9

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8> & 'c:\Users\ASUS\AppData\Local\Programs\P
ython\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62644' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligen
ce Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\ModifyComputerB.py'
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: Dell and its price is: 890.9
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8> 
```

Q) Why calling the setName() and getName() methods changed the output correctly instead of accessing the variables from outside of the class,?

Ans: The variables __name and __price are private to _ComputerB__name and _ComputerB__price). Direct assignment like comp.__name = ‘Dell’ does not reach private variables and creates new unrelated public attributes.

The setter methods (setName and setPrice) are inside the class, so they can access the private variables directly. When we call comp.setName("Dell", it changes the real private __name variables. The getters (getName and getPrice) and display() method then read these updated private values. Therefore, the second comp.display() shows the new values "Dell" and "890.0".

6.

Task 1.6 (Mutable and Immutable objects)

• Mutable object – the states and fields can be changed after the object is created

• Immutable object – nothing can be changed after the object is created

Consider you ComputerB.py program, copy and save it as ComputerC.py. You have made in your program all the member variables as private. This means that they are only accessible from within the class. No outside access is allowed. You have realised this in Task 1.4. However, you can still access them from outside via the public member methods. You have observed this in your Task 1.5. Now in your ComputerC.py program if you delete all the setters (just comment out setName()/setPrice() methods), then there is no way you can change the values of those member variables after the objects are created. For example, after creating an object with values HP/199.99, you cannot change them with Dell/890.90. Try this and explain your observation to your tutors.

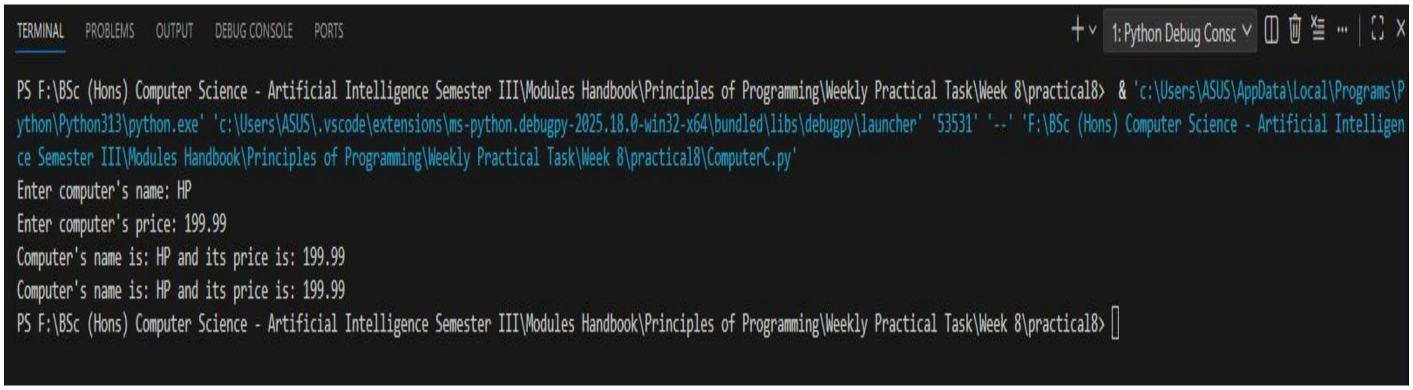
```
# Showing that private variables without setters become effectively immutable

# Practical 8 - Exercise 1 (Task 1.6)

# Nirvik K.C.


class ComputerC:

    # Parameterized constructor

    def __init__(self, name, price):

        # Private instance variables

        self.__name = name

        self.__price = price


    # Getters

    def getName(self):

        return self.__name


    def getPrice(self):

        return self.__price
    # Display method - using getters to access the variables
    def display(self):
        print(f"Computer's name is: {self.getName()} and its price is: 
{self.getPrice()}")


# User provides a valid computer name
isValidName = False
while not isValidName:
    computername = input("Enter computer's name: ")


    if 2 <= len(computername) <= 10:
        isValidName = True
    else:
        print("You must enter a value name between 2 and 10 characters long")
```

```
# User provides a valid computer price
isValidPrice = False
while not isValidPrice:
    computerprice = float(input("Enter computer's price: "))
    if 99.99 <= computerprice <= 999.99:
        isValidPrice = True
    else:
        print("You must enter a valid price between 99.99 and 999.99")


# Creating the comp object based on user's input
comp = ComputerC(computername, computerprice)


# Displaying computer's information
comp.display()


# Comment out setName()/setPrice() methods which means you cannot change the
values of those member variables after the objects are created
```

```
# Setting a different computer name using the setName method

# comp.setName("Dell")


# Setting a different computer name by accessing the name variable which is
public

# comp.__name = 'Dell'


# Setting new price using the setPrice method

# comp.setPrice(890.90)


# Setting new price by accessing the price variable which is public

# comp.__price = 890.90


# Displaying computer's (new) information

comp.display()
```

```
# Output:

# Enter computer's name: HP

# Enter computer's price: 199.99

# Computer's name is: HP and its price is: 199.99

# Computer's name is: HP and its price is: 199.99
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>  & 'c:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53531' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\ComputerC.py'
Enter computer's name: HP
Enter computer's price: 199.99
Computer's name is: HP and its price is: 199.99
Computer's name is: HP and its price is: 199.99
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>
```

Q) Why both lines show the same result - Computer's name is: HP and its price is: 199.99 ?

Ans: In this program, we removed all setter methods (setName and setPrice) are now not present in the class. It is done because the variables are private (__name and __price) name changed to _ComputerC__name and _ComputerC__price. There are no setter methods which means no public way to change the private variables. Direct assignment like comp.__name = 'Dell' does not work. So, it creates a new public attribute called __name on the object but it does not access the real private variable (_ComputerC__name). The getName() method still reads from the private variable so it keeps displaying "HP" and "199.99".

After creation of private variables, the values cannot be changed from outside the class which means the object is immutable for name and price. This makes the ComputerC object immutable with respect to name and price. Once created, its state cannot be changed from outside.

7.

Task 1.7 (List of objects) Consider you ComputerB.py program, copy and save it as ComputerD.py. Extend your ComputerD.py program to create an array of 5 objects of the ComputerD class to represent information about 5 computers. Your program should take input from the keyboard and display information of the five computers.

Given code and instructions:

You need to create a list:

```
objectList = []

# then run a for loop

for i in range (5):

  # User provides a valid computer name as before


  # User provides a valid computer price as before


# appending class instances to list
objectList.append(ComputerD(computername,computerprice))


# end for loop


# displaying computer's information

for obj in objectList:

    obj.display()
```

Output of your program will be like this:

```
Enter computer's name: HP

Enter computer's price: 378.89

Enter computer's name: Dell

Enter computer's price: 490.90

Enter computer's name: Lenovo

Enter computer's price: 308.78

Enter computer's name: Mac

Enter computer's price: 986.78

Enter computer's name: Asus

Enter computer's price: 234.56

Computer's name is: HP and its price is: 378.89

Computer's name is: Dell and its price is: 490.9

Computer's name is: Lenovo and its price is: 308.78

Computer's name is: Mac and its price is: 986.78

Computer's name is: Asus and its price is: 234.56
```

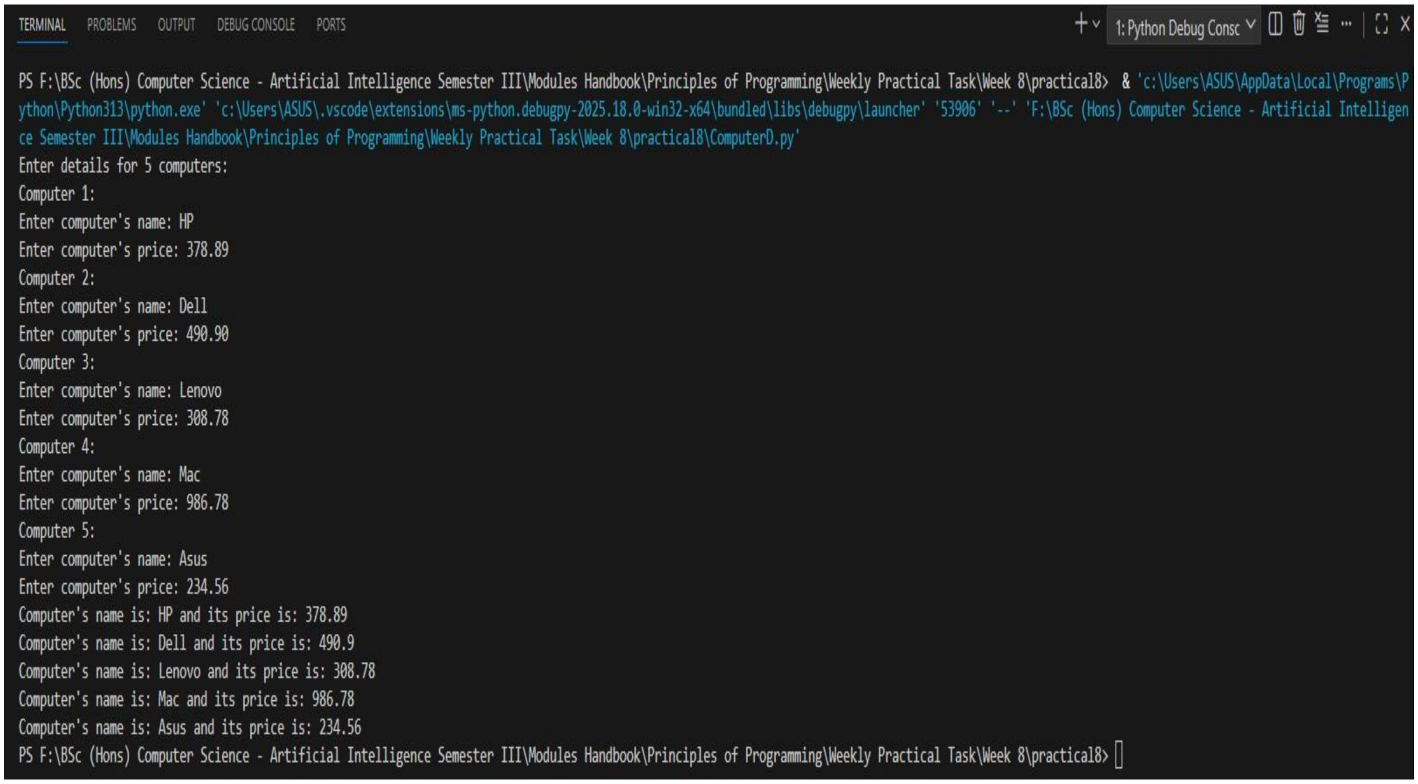
Answer:

Following code for input:

```
# Create an array of objects to represent information about the objects on the list
# Practical 8 - Exercise 1 (Task 1.7)
# Nirvik K.C
class ComputerD:
    # Parameterized constructor
    def __init__(self, name, price):
        self.__name = name
        self.__price = price

    # Getters
    def getName(self):
        return self.__name

    def getPrice(self):
        return self.__price
    # Display method - using getters to access the variables
    def display(self):
        print(f"Computer's name is: {self.getName()} and its price is: 
{self.getPrice()}")

# Create an empty list to store 5 computers
object_list = []

print("Enter details for 5 computers:")

# Use loop 5 times to get input for each computer
for i in range(5):
    print(f"Computer {i+1}:")

    # User provides a valid computer name
    isValidName = False
    while not isValidName:
        computername = input("Enter computer's name: ")
        if 2 <= len(computername) <= 10:
            isValidName = True
        else:
            print("You must enter a valid name between 2 and 10 characters long")

    # User provides a valid computer price
    isValidPrice = False
    while not isValidPrice:
        computerprice = float(input("Enter computer's price: "))
        if 99.99 <= computerprice <= 999.99:
            isValidPrice = True
        else:
            print("You must enter a valid price between 99.99 and 999.99")

    # appending class instances to list
    object_list.append(ComputerD(computername,computerprice))
```

```
# Displaying computer's information
for obj in object_list:
    obj.display()


# Output:
# Enter details for 5 computers:
# Computer 1:
#Enter computer's name: HP
# Enter computer's price: 378.89
# Computer 2:
# Enter computer's name: Dell
# Enter computer's price: 490.90
# Computer 3:
# Enter computer's name: Lenovo
# Enter computer's price: 308.78
# Computer 4:
# Enter computer's name: Mac
# Enter computer's price: 986.78
# Computer 5:

# Enter computer's name: Asus

# Enter computer's price: 234.56

# Computer's name is: HP and its price is: 378.89

# Computer's name is: Dell and its price is: 490.9

# Computer's name is: Lenovo and its price is: 308.78

# Computer's name is: Mac and its price is: 986.78

# Computer's name is: Asus and its price is: 234.56
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>  & 'c:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53906' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\ComputerD.py'
Enter details for 5 computers:
Computer 1:
Enter computer's name: HP
Enter computer's price: 378.89
Computer 2:
Enter computer's name: Dell
Enter computer's price: 490.90
Computer 3:
Enter computer's name: Lenovo
Enter computer's price: 308.78
Computer 4:
Enter computer's name: Mac
Enter computer's price: 986.78
Computer 5:
Enter computer's name: Asus
Enter computer's price: 234.56
Computer's name is: HP and its price is: 378.89
Computer's name is: Dell and its price is: 490.9
Computer's name is: Lenovo and its price is: 308.78
Computer's name is: Mac and its price is: 986.78
Computer's name is: Asus and its price is: 234.56
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>
```

Exercise 2

8. Create TWO UML diagrams for two classes from the list in Exercise 1 using pen and paper (or you can use MS Word/ PowerPoint editor but you don't need to use any formal tool for this). For each of these classes, come up with 2 most important data members that describe the state of the object you want to build and design appropriate constructor, getter and setter methods. You also need to design a display() method to print the information to the console and any other methods you may find appropriate.

Please use different data types for any class's member variables. That is, do not make them all ' int ' or all ' string '. Make all the data members as private and all the methods as public. Use the appropriate symbol to indicate whether a member (instance variable/method) of a class is public (+) or private (-). For each numeric variable, decide on a default value, minimum value and a maximum value. Implement and test your classes.

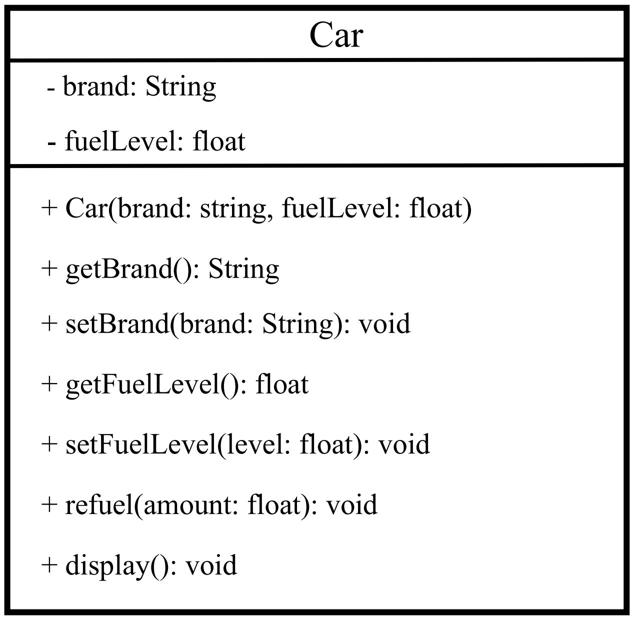
From Exercise 1:

You have to design and implement TWO new classes (Exercise 2). You should choose them from the list below. One example considering the Computer class is show in this exercise.

Circle Dog Shoe Book Song Bicycle Car Computer Movie

Answer:

a) UML for Class : Car


Car
- brand: String
- fuelLevel: float
+ Car(brand: string, fuelLevel: float)
+ getBrand(): String
+ setBrand(brand: String): void
+ getFuelLevel(): float
+ setFuelLevel(level: float): void
+ refuel(amount: float): void
+ display(): void

Note:

brand: default value ‘ ’, min length 2 and max length 20

fuelLevel: default value 0.0, min value 0.0 and max value 100.0

All attributes private (-)

All methods public (+)

Following code for input:

```
# UML to Code : Class Car

# Practical 8 - Exercise 2

# Nirvik K.C

class Car:

    def __init__(self, brand, fuelLevel):

        self.__brand = brand

        self.__fuelLevel = 0.0

        self.setFuelLevel(fuelLevel)


    # Getters

    def getBrand(self):

        return self.__brand


    def setBrand(self, brand):

        return self.__fuelLevel
```

```
def getFuelLevel(self):
    return self.__fuelLevel


# Setters
def setBrand(self, brand):
    self.__brand = brand


# Fuel level of the Car
def setFuelLevel(self, level):
    level = float(level)


    if level < 0.0:
        self.__fuelLevel = 0.0
    elif level > 100.0:
        self.__fuelLevel = 100.0
    else:
        self.__fuelLevel = level
    # Refuel the car
    def refuel(self, amount):
        print(f"Adding {amount} litre of fuel to {self.__brand}")
        self.setFuelLevel(self.__fuelLevel + amount)

    def display(self):
        print(f"Car info: Brand: {self.__brand} and Fuel Level: 
{self.__fuelLevel}%")

# Displaying Car's Info
my_car = Car("Toyota", 55.5)
my_car.display()

my_car.refuel(65.0)
my_car.display()
```

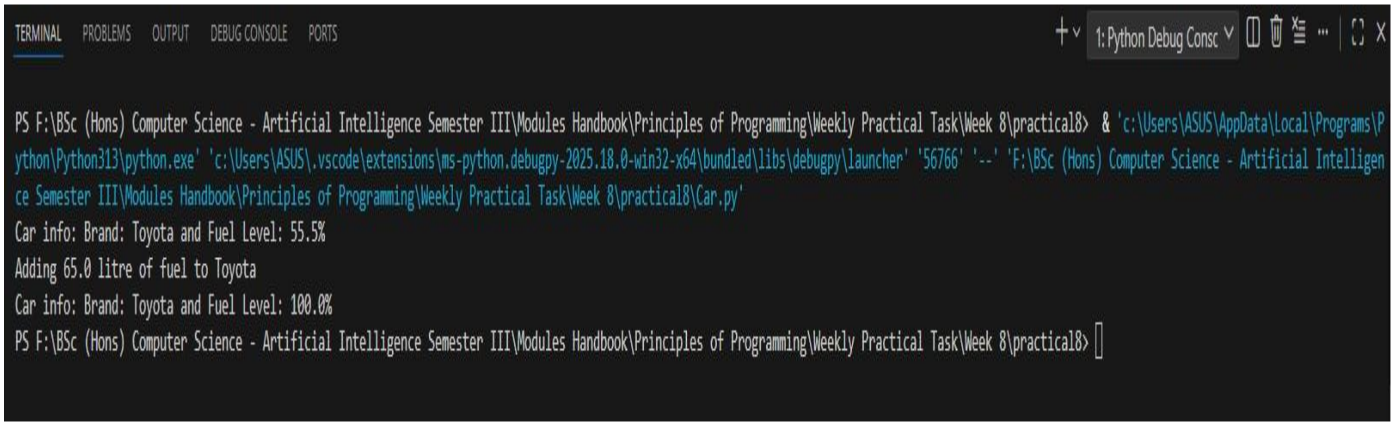
# Output:

# Car info: Brand: Toyota and Fuel Level: 55.5%

# Adding 65.0 litre of fuel to Toyota

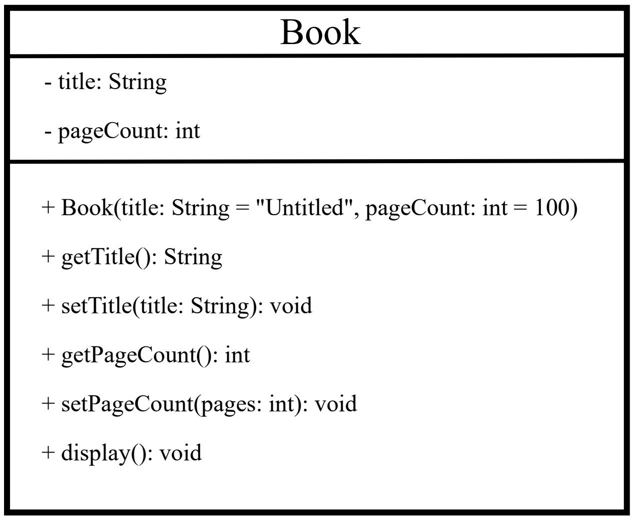
# Car info: Brand: Toyota and Fuel Level: 100.0%

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8> & 'c:\Users\ASUS\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56766' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\Car.py'
Car info: Brand: Toyota and Fuel Level: 55.5%
Adding 65.0 litre of fuel to Toyota
Car info: Brand: Toyota and Fuel Level: 100.0%
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>
```

b) UML for Class : Book


Book
- title: String
- pageCount: int
+ Book(title: String = "Untitled", pageCount: int = 100)
+ getTitle(): String
+ setTitle(title: String): void
+ getPageCount(): int
+ setPageCount(pages: int): void
+ display(): void

Note:

title constraints: default "Untitled"

pageCount constraints: default 100, min 100, max 3000 pages

All attributes private (-)

All methods public (+)

Following code for input:

# UML to Code : Class Book

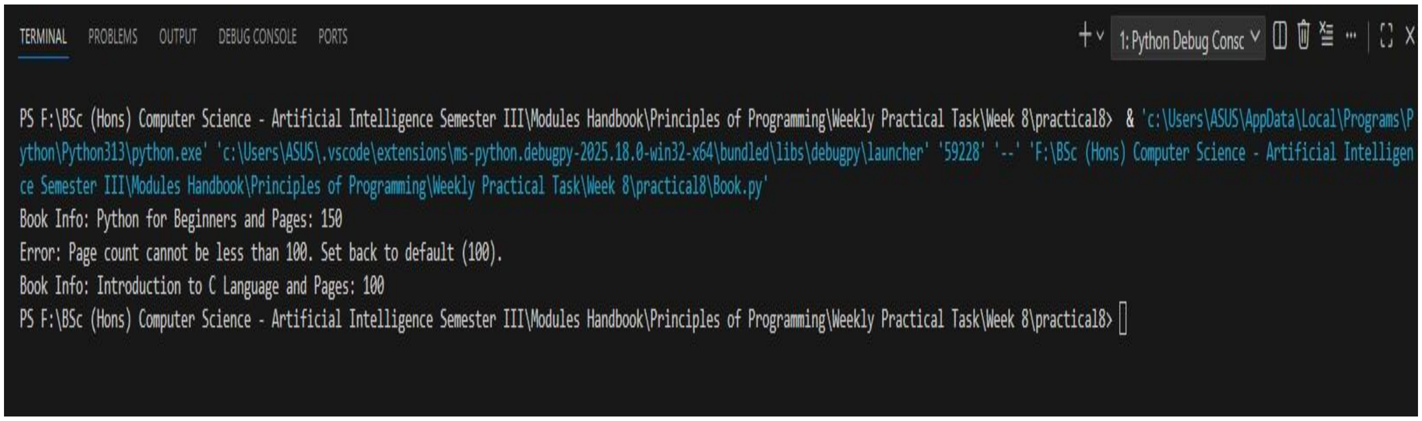
# Practical 8 - Exercise 2

# Nirvik K.C

```
class Book:

    def __init__(self, title="Untitled", pageCount=100):

        self.__title = title

        # Use setter

        self.setPageCount(pageCount)


    # Getters

    def getTitle(self):

        return self.__title


    def getPageCount(self):

        return self.__pageCount


    # Setters


    def setTitle(self, title):

        self.__title = title
    def setPageCount(self, pages):
        # Minimum pages : 100 and Maximum pages : 3000
        if pages < 100:
            print(f"Error: Page count cannot be less than 100. Set back to default
(100).")
            self.__pageCount = 100
        elif pages > 3000:
            print(f"Error: Page count exceeds the limit. Set to the maximum count of
3000.")
            self.__pageCount = 3000
        else:
            self.__pageCount = pages

    # instance/member method
    def display(self):
        print(f"Book Info: {self.__title} and Pages: {self.__pageCount}")
```

```
# Display Book's Info

book1 = Book("Python for Beginners", 150)

book1.display()


book2 = Book("Introduction to C Language", 50)  # Less than default number of
pages

book2.display()

# Output:

# Book Info: Python for Beginners and Pages: 150

# Error: Page count cannot be less than 100. Set back to default (100).

# Book Info: Introduction to C Language and Pages: 100
```

Output obtained in execution:



```
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>  & 'c:\Users\ASUS\AppData\Local\Programs\P
ython\Python313\python.exe' 'c:\Users\ASUS\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59228' '--' 'F:\BSc (Hons) Computer Science - Artificial Intelligen
ce Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8\Book.py'
Book Info: Python for Beginners and Pages: 150
Error: Page count cannot be less than 100. Set back to default (100).
Book Info: Introduction to C Language and Pages: 100
PS F:\BSc (Hons) Computer Science - Artificial Intelligence Semester III\Modules Handbook\Principles of Programming\Weekly Practical Task\Week 8\practical8>
```