**Task 01:**

&gt;&gt;Abstraction in Python is the process of hiding the real implementation of an

application from the user and emphasizing only on usage of it. For example, consider

you have bought a new electronic gadget. Along with the gadget, you get a user guide,

instructing how to use the application, but this user guide has no info regarding the

internal working of the gadget.

Another example is, when you use TV remote, you do not know how pressing a key in

the remote changes the channel internally on the TV. You just know that pressing +

volume key will increase the volume.

&gt;&gt;Abstraction in Python is achieved by using abstract classes and interfaces.An

abstract class is a class that generally provides incomplete functionality and contains

one or more abstract methods. Abstract methods are the methods that generally don’t

have any implementation, it is left to the sub classes to provide implementation for the

abstract methods.

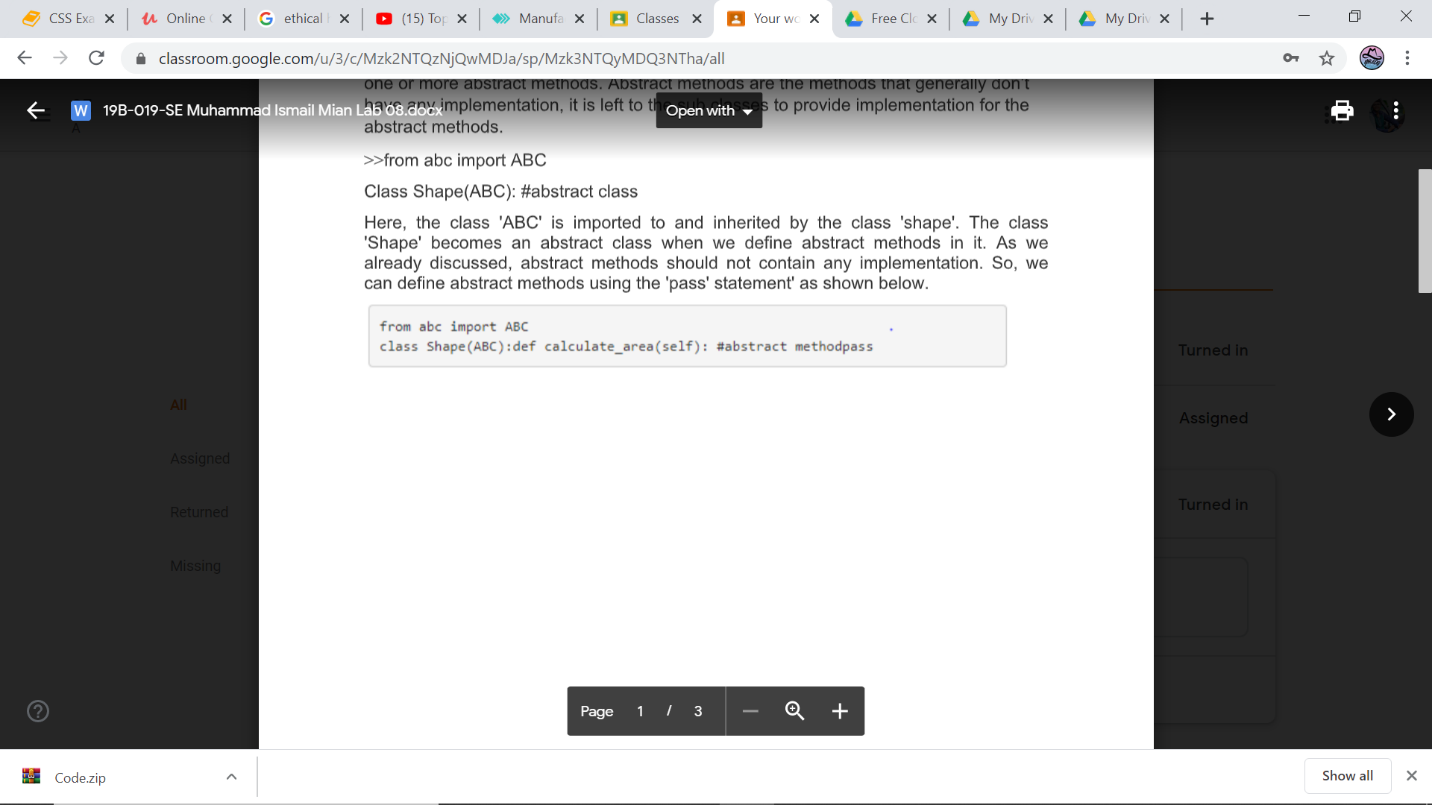
&gt;&gt;from abc import ABC

Class Shape(ABC): #abstract class

Here, the class &#39;ABC&#39; is imported to and inherited by the class &#39;shape&#39;. The class

&#39;Shape&#39; becomes an abstract class when we define abstract methods in it. As we

already discussed, abstract methods should not contain any implementation. So, we

can define abstract methods using the &#39;pass&#39; statement&#39; as shown below.

**Task 02**

from abc import ABC, abstractmethod

class Bank(ABC):

def AccountName(self):

pass

def RateOfInterest(self):

pass

def Deposit(self):

pass

def Withdraw(self):

pass

class Habib(Bank):

def Name(self):

print("Habib Bank Limited")

def AccountName(self):

print("Account Name: 7883-98765432")

def RateOfInterest(self):

print("Interest Rate: 25%")

def Deposit(self):

print("Deposit: 27000")

def Withdraw(self):

print("Withdraw: 3000")

class Metro(Bank):

def Name(self):

print("Metro Bank Limited")

def AccountName(self):

print("Account Name: 8772-09876373")

def RateOfInterest(self):

print("Interest Rate: 15%")

def Deposit(self):

print("Deposit: 13000")

def Withdraw(self):

print("Withdraw: 3500")

class National(Bank):

def Name(self):

print("National Bank")

def AccountName(self):

print("Account Name: 8877-091122445")

def RateOfInterest(self):

print("Interest Rate: 30%")

def Deposit(self):

print("Deposit: 220000")

def Withdraw(self):

print("Withdraw: 45000")

m1= Habib()

m1.Name()

m1.AccountName()

m1.RateOfInterest()

m1.Deposit()

m1.Withdraw()

print(36\*'\_')

a1= Metro()

a1.Name()

a1.AccountName()

a1.RateOfInterest()

a1.Deposit()

a1.Withdraw()

"\n""\n"

a2= National()

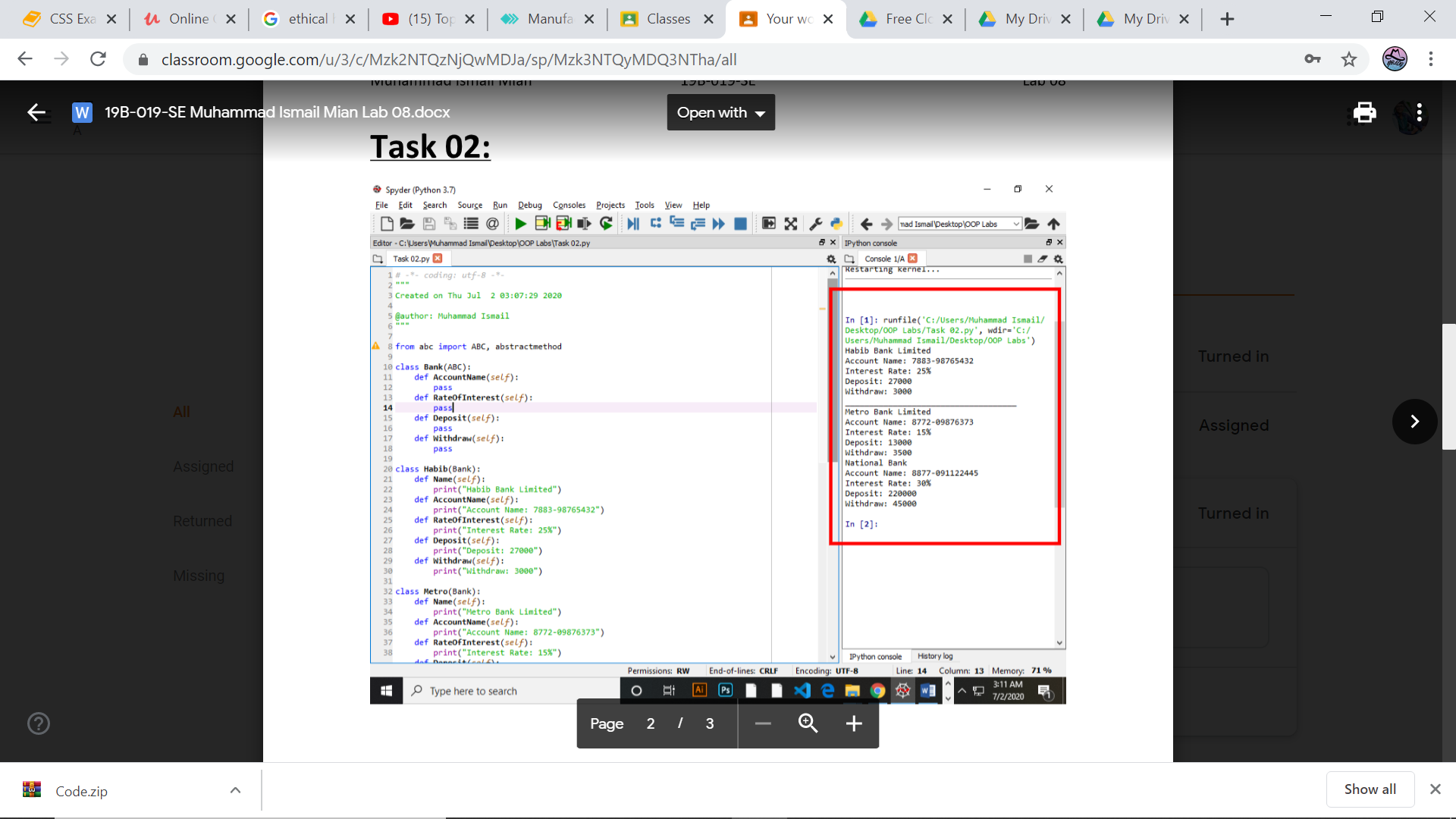
a2.Name()

a2.AccountName()

a2.RateOfInterest()

a2.Deposit()

a2.Withdraw()



**Task 03**

from abc import ABC, abstractmethod

class SmartPhone(ABC):

def Price(self):

pass

def YearOfInvention(self):

pass

def RefreshRate(self):

pass

class Samsung(SmartPhone):

def Name(self):

print("Samsung S7")

def Price(self):

print("Price: 63000")

def YearOfInvention(self):

print("Year Of Invention: 2016")

def RefreshRate(self):

print("Refresh Rate: 120 Hz")

class Iphone(SmartPhone):

def Name(self):

print("Iphone 7 Plus")

def Price(self):

print("Price: 30000")

def YearOfInvention(self):

print("Year Of Invention: 2015")

def RefreshRate(self):

print("Refresh Rate: 60 Hz up to 240 Hz")

o1= Samsung()

o1.Name()

o1.Price()

o1.YearOfInvention()

o1.RefreshRate()

print(36\*'\_')

l1= Iphone()

l1.Name()

l1.Price()

l1.YearOfInvention()

l1.RefreshRate()

