**Task 2:**

Create an ABC class of Bank and add some abstract method AccountName, rate of interest, deposit, withdraw,

Now add some classes in which you will implement Bank abstract class and its methods.

# CODE

from abc import ABCMeta,abstractmethod

class Bank(metaclass=ABCMeta):

@abstractmethod

#def Account\_name(self):

#pass

#def Rate\_of\_interest(self):

#pass

def deposite(self,amount):

pass

def withdraw(self,amount):

pass

class Bank\_AlHabib(Bank):

def \_\_init\_\_(self):

self.balance=0

def deposite(self,amount):

self.balance=self.balance+amount

return("You deposit an amount of Rs {}".format(self.balance))

def withdraw(self,amount):

if (self.balance>=amount):

self.balance=self.balance-amount

return("You withdraw an amount of Rs {}".format(self.balance))

else:

print("Insufficient funds")

def print\_balance(self):

return(self.balance)

class Bank\_HabibMetro(Bank):

def \_\_init\_\_(self):

self.balance=0

def deposite(self,amount):

self.balance=self.balance+amount

return("You deposit an amount of Rs {}".format(self.balance))

def withdraw(self,amount):

if (self.balance>=amount):

self.balance=self.balance-amount

return("You withdraw an amount of Rs {}".format(self.balance))

else:

print("Insufficient funds")

def print\_balance(self):

return(self.balance)

class Bank\_Mezan(Bank):

def \_\_init\_\_(self):

self.balance=0

def deposite(self,amount):

self.balance=self.balance+amount

return("You deposit an amount of Rs {}".format(self.balance))

def withdraw(self,amount):

if (self.balance>=amount):

self.balance=self.balance-amount

return("You withdraw an amount of Rs {}".format(self.balance))

else:

print("Insufficient funds")

def print\_balance(self):

return(self.balance)

class Bank\_AlFalah(Bank):

def \_\_init\_\_(self):

self.balance=0

def deposite(self,amount):

self.balance=self.balance+amount

return("You deposit an amount of Rs {}".format(self.balance))

def withdraw(self,amount):

if (self.balance>=amount):

self.balance=self.balance-amount

return("You withdraw an amount of Rs {}".format(self.balance))

else:

print("Insufficient funds")

def print\_balance(self):

return(self.balance)

account1=Bank\_AlHabib()

account2=Bank\_HabibMetro()

account3=Bank\_Mezan()

account4=Bank\_AlFalah()

while(True):

print("In which of the following you have an account")

print("1) Bank-AlHabib")

print("2) Bank-HabibMetro")

print("3) Bank-Mezan")

print("4) Bank-AlFalah")

user\_choice=input()

if user\_choice not in ["1","2","3","4"]:

print("Please enter a Valid Option")

continue

else:

user\_choice=int(user\_choice)

if user\_choice ==1:

#lib.displaybook()

print("Do you want to deposit the ammount in Bank-AlHabib")

print("or")

print("Do you want to withdraw the amount in Bank-AlHabib")

a=input("If you want to deposit than put YES in this form or If you want to withdraw than put yes in this form")

if a=="YES":

print("How many ammount you want to deposit..?")

amount=int(input("Enter the amount you want to deposit"))

print(account1.deposite(amount))

print(account1.print\_balance())

elif a=="yes":

print("How many amount you want to withdraw...?")

amount=int(input("Enter the amount you want to withdraw"))

print(account1.withdraw(amount))

print(account1.print\_balance())

else:

print("Not a valid option")

elif user\_choice ==2:

print("Do you want to deposit the ammount in Bank-HabibMetro")

print("or")

print("Do you want to withdraw the amount in Bank-HabibMetro")

a=input("If you want to deposit than put YES in this form or If you want to withdraw than put yes in this form")

if a=="YES":

print("How many ammount you want to deposit..?")

amount=int(input("Enter the amount you want to deposit"))

print(account1.deposite(amount))

print(account1.print\_balance())

elif a=="yes":

print("How many amount you want to withdraw...?")

amount=int(input("Enter the amount you want to withdraw"))

print(account1.withdraw(amount))

print(account1.print\_balance())

else:

print("Not a valid option")

elif user\_choice ==3:

print("Do you want to deposit the ammount in Bank-Mezan")

print("or")

print("Do you want to withdraw the amount in Bank-Mezan")

a=input("If you want to deposit than put YES in this form or If you want to withdraw than put yes in this form")

if a=="YES":

print("How many ammount you want to deposit..?")

amount=int(input("Enter the amount you want to deposit"))

print(account1.deposite(amount))

print(account1.print\_balance())

elif a=="yes":

print("How many amount you want to withdraw...?")

amount=int(input("Enter the amount you want to withdraw"))

print(account1.withdraw(amount))

print(account1.print\_balance())

else:

print("Not a valid option")

elif user\_choice ==4:

print("Do you want to deposit the ammount in Bank-AlFalah")

print("or")

print("Do you want to withdraw the amount in Bank-AlFalah")

a=input("If you want to deposit than put YES in this form or If you want to withdraw than put yes in this form")

if a=="YES":

print("How many ammount you want to deposit..?")

amount=int(input("Enter the amount you want to deposit"))

print(account1.deposite(amount))

print(account1.print\_balance())

elif a=="yes":

print("How many amount you want to withdraw...?")

amount=int(input("Enter the amount you want to withdraw"))

print(account1.withdraw(amount))

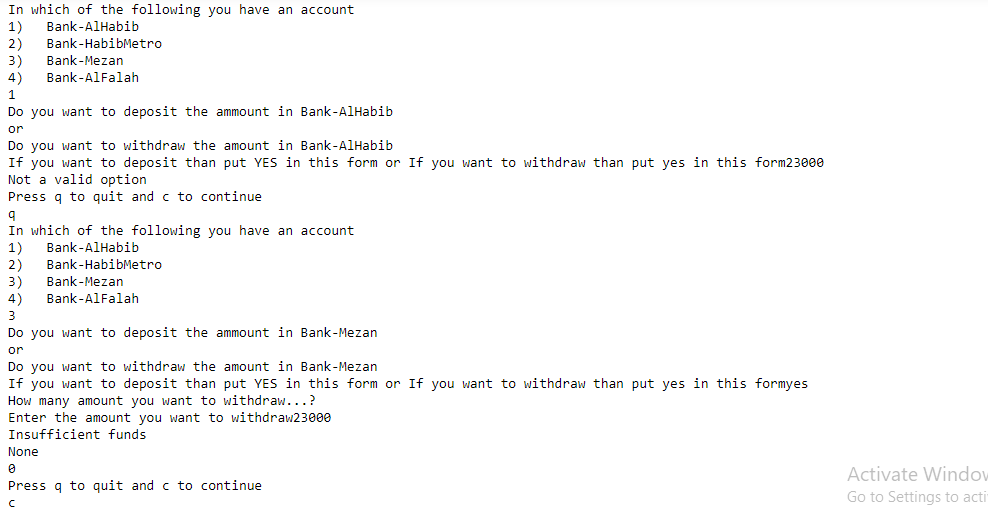
print(account1.print\_balance())

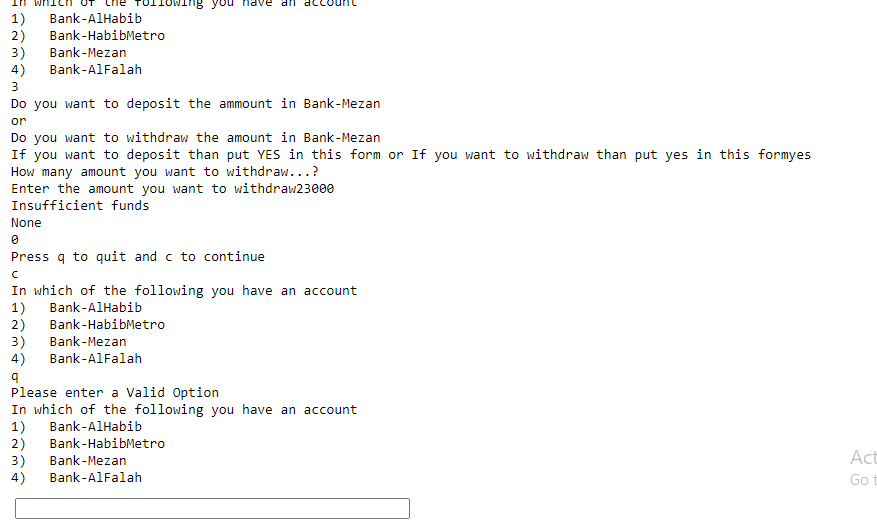
else:

print("Not a valid option")

else:

print("Not a Valid Option")





**Task 1:** Discuss in detail

 What is Abstraction in Python?

 How can we achieve Abstraction in Python?

 Mention the name of the module to be imported for an abstract class

* Abstract classes are classes that contain one or more abstract methods.
* An abstract method is a method that is declared, but contains no implementation.
* Abstract classes cannot be instantiated, and require subclasses to provide implementations for the abstract methods.
* Abstract classes may not be instantiated, and its abstract methods must be implemented by its subclasses.

from abc import ABCMeta,abstractmethod

**Task 3:**

Find out one real world example of abstract class and abstract method and implement it by using python code

# CODE

from abc import ABCMeta,abstractmethod

class Robot(metaclass=ABCMeta):

@abstractmethod

def orders(self):

pass

def cleaning(self):

pass

class cookerRobo(Robot):

def orders(self):

return("My Robots Follows All my orders")

def cleaning(self):

return("My Robots is a good cleaning master")

def cooking(self):

return("My Robot is a good cooker")

class GardnerRobo(Robot):

def orders(self):

return("My Robots Follows All my orders")

def cleaning(self):

return("My Robots is a good cleaning master")

def waterplants(self):

return("My robot watering the plants on time")

co=cookerRobo()

print(co.orders())

print(co.cleaning())

print(co.cooking())

Ga=GardnerRobo()

print(Ga.orders())

print(Ga.cleaning())

print(Ga.waterplants())

