PseudoCode & Annotations

Pseudo:

Variables

customer_name = “ “

Balance= 0,0

Count=0.0

#Creating class BankAccount

class BankAccount

customer_name

balance

count

__init__(customer_name)

#inislize customer_name

#Generating account_num using generateAccountNumber() method

account_num = generateAccountNumber()

balance =0

#Calling display method

display()

#increment the count

count+1

/#This method creates random account Number and return the value

generateAccountNumber()

return 1+random.randint(0,1000)*1000+random.randint(0,1000)

addAmount(amount)

```
#add amount to balance
balance += amount
withdrawAmount(amount)
if(check for balance-amount should not be negative)
subtract amount from balance
else
print(Not sufficient balance)
display()
print(customer_name,account_num,balance)


#Create Empty Bank account Collection
BankAccountCollection = []

Variable to count of entered account
count = 1

#method to create new Back Account
opennewAccount()
name = str(input("..."))
#Creating new BankAccount object with name
obj = BankAccounts(name)
Adding BankAccount to Collection
increment count Variable

#method to close account
closeAccount()
account_number = int(input(....))
```

```
try:
for i in range(count):
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
#remove back account Collection by calling remove method
BankAccountCollection.remove(BankAccountCollection[i])
decrease value of count
break the loop


except ValueError as ve:
print(Enter valid account number)
except IndexError as e
print(Account does not exist)

displayAccountDetails()
try:
account_number = int(input(...))
for i in range(count)
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
break the loop

except ValueError as ve:
print(Enter valid account number)
except IndexError as e
```

```
print(Account does not exist)

depositeAmount()
try
account_number = int(input(...))
for i in range(count)
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
amount = int(input(....))
check amount is greatter than zero and add amount to account
break the loop


except ValueError as ve:
print(Enter valid account number)
except IndexError as e
print(Account does not exist)

withdrawAmount()
try
account_number = int(input(...))
for i in range(count)
if BankAccountCollection[i].getAccountNumber() is equal to account_number
display bank account
BankAccountCollection[i].display()
amount = int(input(....))
check amount is greatter than zero and withdraw amount to account
```

break the loop

except ValueError as ve:

print(Enter valid account number)

except IndexError as e

print(Account does not exist)

Annotations:

Our code has 5 main steps with annotations

1: To create a class Bank_Acccount. Then define a function using __init__ with default argument self. This keyword is used in Python to initialize attributes of the class when an object of that class is created. This step is followed by initializing the balance as 0.

```
class Bank_Account:
def __init__(self):
self.balance=0
print("Welcome to Deposit & Withdrawal Machine!")
```

2: To create a function deposit such that the amount of money is taken by input using float and is then added to the balance. Then the amount deposited will be displayed using the print statement in the next line as shown in the code below:

```
def deposit(self):
amount=float(input("Enter amount to be deposited: "))
self.balance += amount
print("Amount Deposited: ",amount)
```

3: To create another function withdraw in which is going to take float input for the amount to get withdraw. Using an if condition here just to check if there's sufficient balance available to perform a withdrawal of any amount from the account. If the balance is not sufficient then our class will show "Insufficient balance".

```
def withdraw(self):
```

```
amount = float(input("Enter amount to withdraw: "))
if self.balance>=amount:
self.balance-=amount
print("You withdraw: ",amount)
else:
print("Insufficient balance ")
```

4: To create our final function which is `display` function. It will display the final balance of the account after withdrawal and deposit.

```
def display(self):
print("Net Available Balance=",self.balance)
```

5: Lastly, to create an object of our class so that we can call all the functions with that class to execute our code.

```
#creating an object of class
s = Bank_Account()
#calling functions with that class
s.deposit()
s.withdraw()
s.display()
```