class BankAccount:

def \_\_init\_\_(self,account\_number:int,name:str,balance:float=0.0):

self.account\_number=account\_number

self.name=name

self.balance=balance

def check\_balance(self):

return self.balance

def display\_balance(self):

print("Current balance is {}".format(self.balance))

return "Current balance is {}".format(self.balance)

def deposit\_amt(self,amt):

print("\nInitial balance before deposit is {}".format(self.balance))

print("Deposit amount is {}".format(amt))

self.balance+=amt

self.display\_balance()

def withdraw\_amt(self,amt):

print("\nInitial balance before withdrawal is {}".format(self.balance))

print("Withdrawal amount is {}".format(amt))

if amt>self.balance:

print("Balance Low")

raise ValueError("Balance is Low!!")

self.balance-=amt

self.display\_balance()

import unittest

class TestBankAccount(unittest.TestCase):

def setUp(self):

self.acc1=BankAccount(12238,"Prsn1",5000.0)

self.acc2=BankAccount(73487,"Prsn2")

def test\_check\_balance(self):

self.assertEqual(self.acc1.check\_balance(),5000.0)

def test\_check\_default\_balance(self):

self.assertEqual(self.acc2.check\_balance(),0.0)

def test\_withdraw\_amt(self):

self.acc1.withdraw\_amt(3000)

self.assertEqual(self.acc1.check\_balance(),2000.0)

def test\_low\_balance(self):

with self.assertRaises(ValueError):

self.acc1.withdraw\_amt(9000)

def test\_deposit\_amt(self):

self.acc1.deposit\_amt(4000)

self.assertEqual(self.acc1.check\_balance(),9000.0)

def test\_check\_balance(self):

self.assertEqual(self.acc1.check\_balance(),5000.0)

def test\_display\_balance(self):

self.assertEqual(self.acc1.display\_balance(),"Current balance is 5000.0")

if \_\_name\_\_=='\_\_main\_\_':

unittest.main(argv=['first-arg-is-ignored'], exit=False)

obj1=BankAccount(6787931,"Person",1500)

obj1.deposit\_amt(500)

obj1.withdraw\_amt(200)

obj2=BankAccount(6787931,"Person")

obj2.withdraw\_amt(6000)



