# 1. Library Management System

#    - Create a system to manage books in a library.

#      - Tasks:

#      1. Store book details like title, author, and availability using appropriate data types.

#      2. Allow the user to borrow or return books. Use conditional statements to check availability.

#      3. Implement loops to display all available books.

#      4. Handle exceptions for invalid inputs (e.g., trying to borrow a non-existent book).

#      5. Use a module to manage library operations (e.g., a separate module for adding/removing books).

import Remove

import Add

#      1. Store book details like title, author, and availability using appropriate data types

books = {

    "The Great Gatsby": {"author": "F. Scott Fitzgerald", "availability": True},

    "1984": {"author": "George Orwell", "availability": False},

    "To Kill a Mockingbird": {"author": "Harper Lee", "availability": True},

    "Pride and Prejudice": {"author": "Jane Austen", "availability": True},

    "Moby-Dick": {"author": "Herman Melville", "availability": False},

    "The Catcher in the Rye": {"author": "J.D. Salinger", "availability": True},

    "The Odyssey": {"author": "Homer", "availability": True},

    "War and Peace": {"author": "Leo Tolstoy", "availability": False},

    "The Hobbit": {"author": "J.R.R. Tolkien", "availability": True},

    "Crime and Punishment": {"author": "Fyodor Dostoevsky", "availability": False}

}

#      3. Implement loops to display all available books.

def Display():

     global books

     for k,v in books.items():

        if v['availability']==True:

            print(k,"Author:",v['author'])

print("1.Borrow Book")

print("2.return Book")

print("3.Display Book List")

print("4.Exit")

n=int(input("enter your choice "))

while(n!=4):

    if n==1:

        Remove.borrow(books)

    elif n==2:

        Add.Return(books)

    elif n==3:

      Display()

    if n==4:

        break

    n=int(input("if you want to continue press Choice "))

    print("1.Borrow Book")

    print("2.return Book")

    print("3.Exit")

#      2. Allow the user to borrow or return books. Use conditional statements to check availability.

Remove.py

def borrow(d):

    b=input("Enter book name ")

    if b in d.keys():

        if d[b]['availability']==True:

            print("Please take your book")

            d.pop(b)

        else:

            print("You can't borrow this Book is currently not available")

    else:

        print("The book is not in book list")

Add.py

def Return(d):

  try:

    s=input("Enter book name ")

    author=input("Enter Author name ")

    d.update({s:{'author':author,'availability':True}})

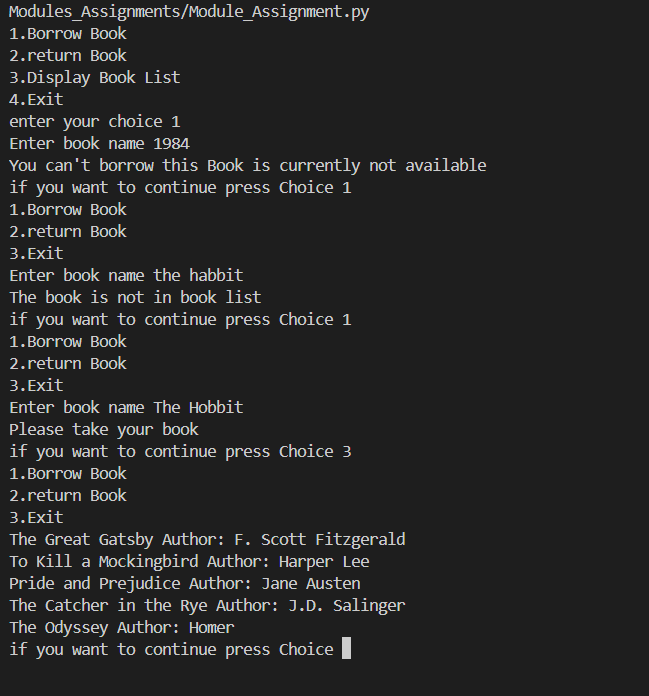
    print("Book Returned Succesfully")

  except:

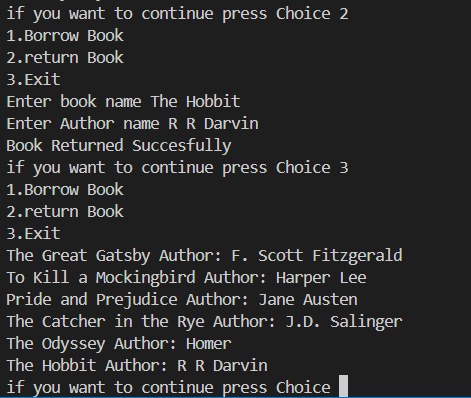
    print("Please Enter Correctly")

**output:**

**Borrow**

****

**Return :**

****

**Exit :**

****

#  2. Restaurant Billing System

#    - Develop a billing system for a restaurant.

#    - Tasks:

#      1. Create a menu using a dictionary where keys are item names and values are prices.

#      2. Use a loop to allow customers to order multiple items.

#      3. Calculate the total bill and apply a conditional discount if the bill exceeds a certain amount.

#      4. Handle exceptions for invalid item selection.

#      5. Organize the code into modules for the menu, order processing, and billing.

#      1. Create a menu using a dictionary where keys are item names and values are prices.

'''

import Menu

import Order\_procesing

import Billing

Menu.DiplayMenu()

Total\_Bill=Order\_procesing.Order(Menu.menu)

Bill=Billing.Bill(Total\_Bill)

print(Bill)

'''

Billing –Module-Billing.py

#      3. Calculate the total bill and apply a conditional discount if the bill exceeds a certain amount.

def Bill(Total\_Bill):

    if Total\_Bill>200 and Total\_Bill<1000:

        print("Congrasulations You Got Discount of 20% on above 200 Bill")

        return f'Total Bill: {Total\_Bill} Discounted Bill: {Total\_Bill-(20/100\*Total\_Bill)}'

    elif Total\_Bill>1000:

        print("Congrasulations You Got Discount of 50% on above 1000 Bill")

        return f'Total Bill: {Total\_Bill} Discounted Bill: {Total\_Bill-(50/100\*Total\_Bill)}'

    else:

        return f'Your Total Bill {Total\_Bill}'

Order-Processing:

def Order(menu):

  Total\_Bill=0

  print("Please Enter Your Items once done enter done")

  #      2. Use a loop to allow customers to order multiple items.

  while(True):

    s=input("please Enter name of Item ")

    if s=='Done':

       break

    else:

     if s in menu:

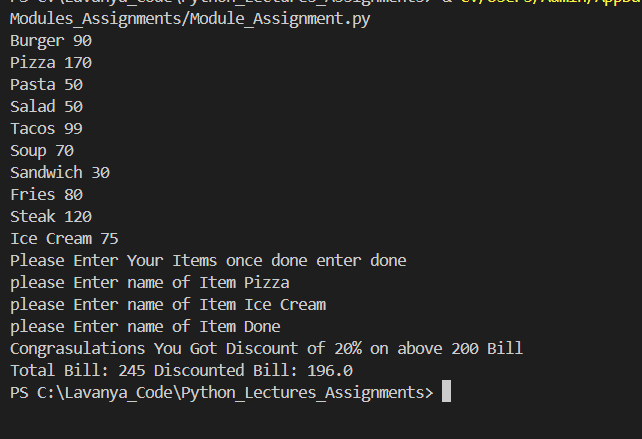
         Total\_Bill+=menu[s]

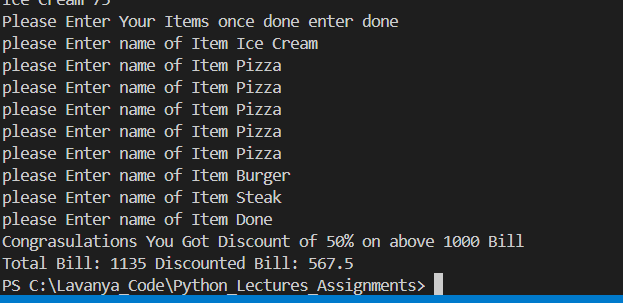
     else:

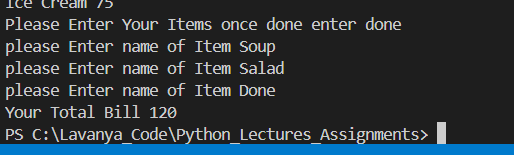
       print("Please Enter Items of Menu only")

  return Total\_Bill

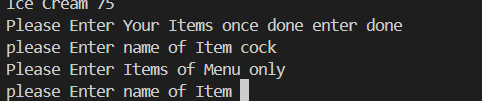
**Output:**







Items not in menu



# 3. Student Grading System

#    - Build a system to calculate grades for students.

#    - Tasks:

#      1. Accept student details and marks for subjects using appropriate data structures.

#      2. Use conditions to assign grades based on marks (e.g., A, B, C).

#      3. Use loops to calculate the average marks of multiple students.

#      4. Handle exceptions for invalid marks (e.g., marks outside the 0-100 range).

#      5. Create a module to format and display the student results.

import Result

students={

    'Alice': 76,

    'Bob': 45,

    'Charlie': 92,

    'David': 58,

    'Eva': 30,

    'Fay': 84,

    'George': 71,

    'Helen': 99,

    'Ivy': 500,

    'Jack': 64

}

name=input('Please Enter name of Student you want to see result ')

import Result

Result.result(name,students[name])

sum=0

l=0

for v in students.values():

    if v<=100:

        sum+=v

        l+=1

#      3. Use loops to calculate the average marks of multiple students.

print("Avrage Result Of Class is",sum/l)

**Result:**

def result(name,marks):

  if marks>100:

    print(name,"You Got Invalid Result ")

  else:

    print(f'-------------{name}-----------------')

    if marks>=75 and marks<=100:

        print("Total Marks:",marks,"Grade:",'A')

    elif marks>=55 and marks<=74:

        print("Total Marks:",marks,"Grade:",'B')

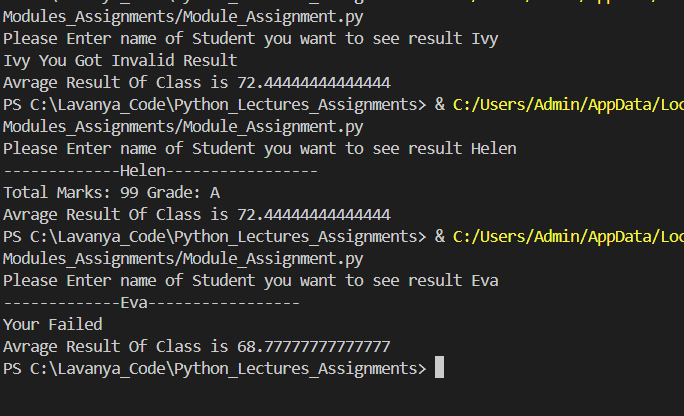
    elif marks>=35 and marks<=54:

        print("Total Marks:",marks,"Grade:",'C')

    elif marks<35:

        print("Your Failed ")

**Output:**

****

# 4. ATM Simulation System

#    - Simulate the operations of an ATM.

#    - Tasks:

#      1. Store account details (e.g., account number, balance) using a dictionary.

#      2. Implement withdrawal, deposit, and balance check functionalities.

#      3. Use conditional statements to check for sufficient balance before withdrawal.

#      4. Handle exceptions for invalid transactions (e.g., entering a non-numeric value for withdrawal).

#      5. Create separate modules for authentication and transaction processing.

import authentication

import Withdraw

#      1. Store account details (e.g., account number, balance) using a dictionary.

accounts = {

    1001000: 1500.75,

    1011123: 3200.50,

    1089456: 450.30,

    1004: 8900.00,

    1090: 1234.67,

    1006: 345.10,

    10075656: 7600.99,

    1088: 234.55,

    100967676: 1750.80,

    10105656: 2895.25

}

def Deposit(acc):

    global accounts

    balance=int(input("Enter amount "))

    accounts[acc]=accounts[acc]+balance

    print('Deposit succefull')

def CheckBalance(acc):

    global accounts

    print("Your Balance:",accounts[acc])

acc=authentication.isauthor(input("Enter Account number"),accounts)

print(acc)

#      2. Implement withdrawal, deposit, and balance check functionalities.

if acc!='No':

    print("Please choose you Choice")

    print("1.Deposit")

    print("2.Withdraw")

    print("3.CheckBalance")

    print("4.Exist")

    n=int(input("Enter Your Chioce "))

    while(n!=4):

         if n==1:

                Deposit(acc)

         elif n==2:

           Withdraw.withdraw(acc,accounts)

         elif n==3:

              CheckBalance(acc)

         print('if you want continue Enter choice else enter 4')

         print("1.Deposit")

         print("2.Withdraw")

         print("3.CheckBalance")

         n=int(input("enter choice "))

         if n==4:

              break

else:

    print("Try After 24 Hr ")

**Authentication-Model .py**

#      5. Create separate modules for authentication and transaction processing.

def isauthor(acc,d):

 i=0

 while(i<2):

    try:

        if int(acc) in d:

            print("Loggined Succefully")

            return int(acc)

        else:

           print("Account number is Invalid")

           i+=1

           acc=input("Enter Account Number ")

           if i==2:

              if int(acc) in d:

                 print("Loggined Succefully")

                 return int(acc)

    except:

       acc=input("Please enter Account number in positive integer")

 return "No"

**Withdrawal Model .py**

#      3. Use conditional statements to check for sufficient balance before withdrawal.

#      4. Handle exceptions for invalid transactions (e.g., entering a non-numeric value for withdrawal).

def withdraw(acc,d):

    while(True):

     try:

       amount=int(input("enter amount "))

       if d[acc]>=amount:

         d[acc]=d[acc]-amount

         print("Withdrawal Sussefully")

         return

       else:

         print("Insuffiensint Balance ")

         return

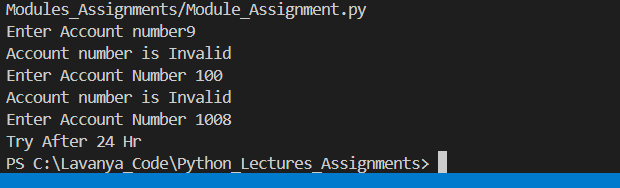
     except:

       print("Enter amount in positive integer ")

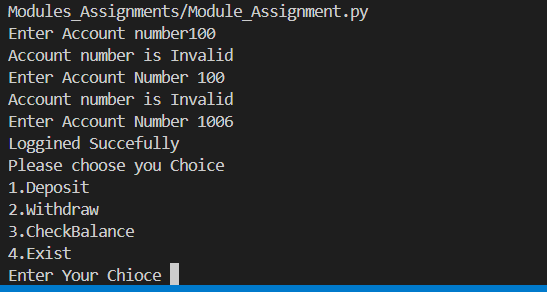
**Output:**

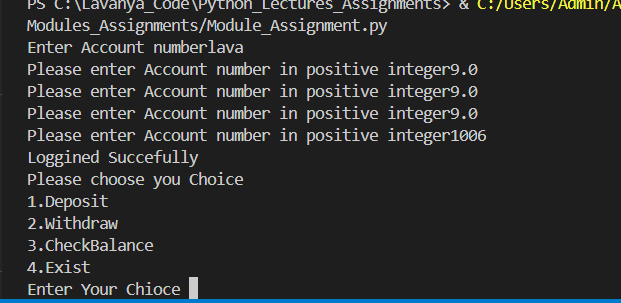
**Authentication test**

**Fails after 3 attempt**

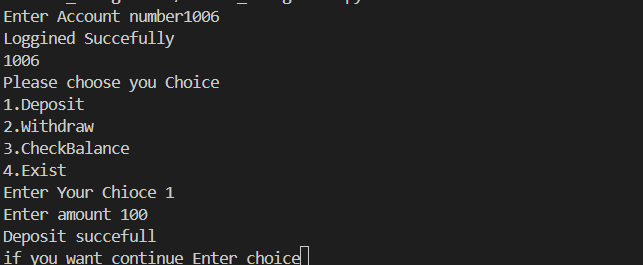
****

**If 3rd attempt is right**

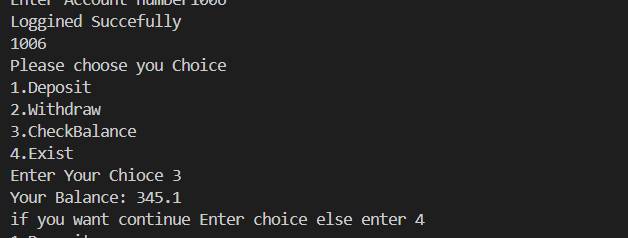
****

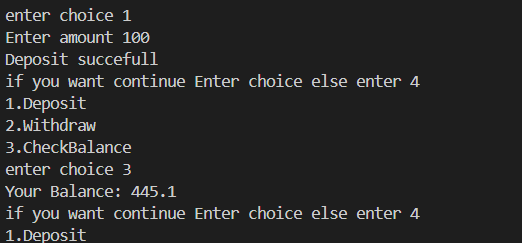
**If invalid input**

**Deposite Test:**

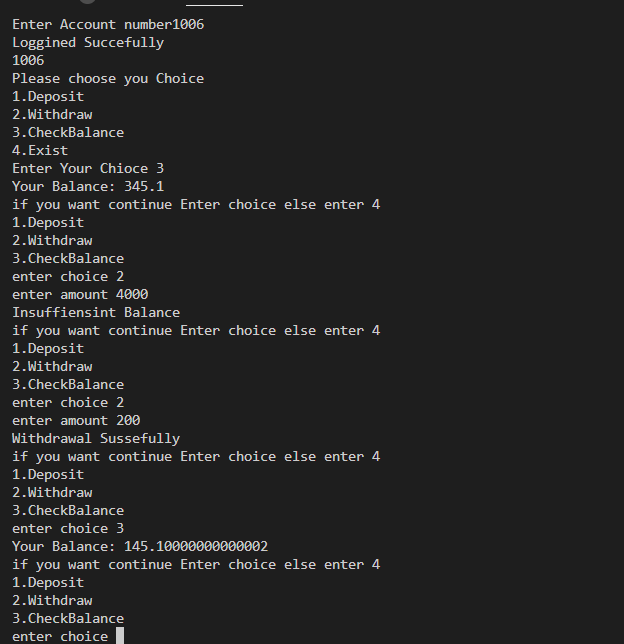
****

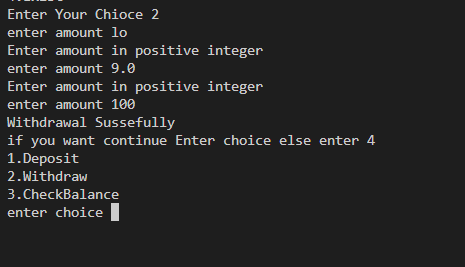
**Check balance**

****

****

**Withdrawal:**

****

**Withdrawal of exceptions:**

# 5. Inventory Management System

#    - Create a system to manage product inventory.

#    - Tasks:

#      1. Store product details like name, price, and stock quantity using a list of dictionaries.

#      2. Allow the user to add or remove products from the inventory using conditions.

#      3. Use loops to display all products and their details.

#      4. Handle exceptions for invalid inputs (e.g., negative stock quantity).

#      5. Organize the inventory operations in a module for better structure.

import AddItem

import RemoveItem

l=[{'name': 'Laptop', 'price': 1200.99, 'stock': 30},

{'name': 'Smartphone', 'price': 899.49, 'stock': 50},

{'name': 'Headphones', 'price': 199.99, 'stock': 75},

{'name': 'Smartwatch', 'price': 299.99, 'stock': 40},

{'name': 'Tablet', 'price': 350.50, 'stock': 60},

{'name': 'Keyboard', 'price': 45.99, 'stock': 100},

{'name': 'Mouse', 'price': 25.75, 'stock': 120},

{'name': 'Monitor', 'price': 350.00, 'stock': 25},

{'name': 'External Hard Drive', 'price': 85.49, 'stock': 40},

{'name': 'Wireless Charger', 'price': 35.00, 'stock': 150}]

print('''

    1.Add

    2.Remove

    3.Display

    4.Exit

''')

ch=int(input("Enter choice "))

while(ch!=4):

    if ch==1:

        AddItem.add(l)

    elif ch==2:

        RemoveItem.remove(l)

    elif ch==3:

        for i in l:

            for k,v in i.items():

                print(k,'->',v,"  ",end=" ")

            print()

    ch=int(input("Enter your choice "))

**AddItems Model- add(Item).py**

def add(Item):

    for i in range(int(input("Enter no of Items you want to add "))):

      name=input("Enter name of Item ")

      price=float(input("Enter Price "))

      while(True):

        try:

          QTY=int(input("Enter quantity "))

          if QTY<=0:

            print("enter QTY in positive number")

          else:

            Item.append({'name':name,'price':price,'stock':QTY})

            print("")

            break

        except:

          print("enter QTY in positive number")

    print("Items Succesully Added")

**RemoveItems Model- add(Item).py**

def remove(Items):

    print(Items)

    i=0

    n=(int(input("Enter number of Items you want to remove ")))

    while(i<n):

        name=input("Enter name ")

        f=1

        for d in Items:

           if name in d.values():

               f=0

               QTY=int(input("Enter Quntity "))

               i+=1

               if QTY>d['stock']:

                   print("Insuffient Quntity ")

                   break

               else:

                   d['stock']-=QTY

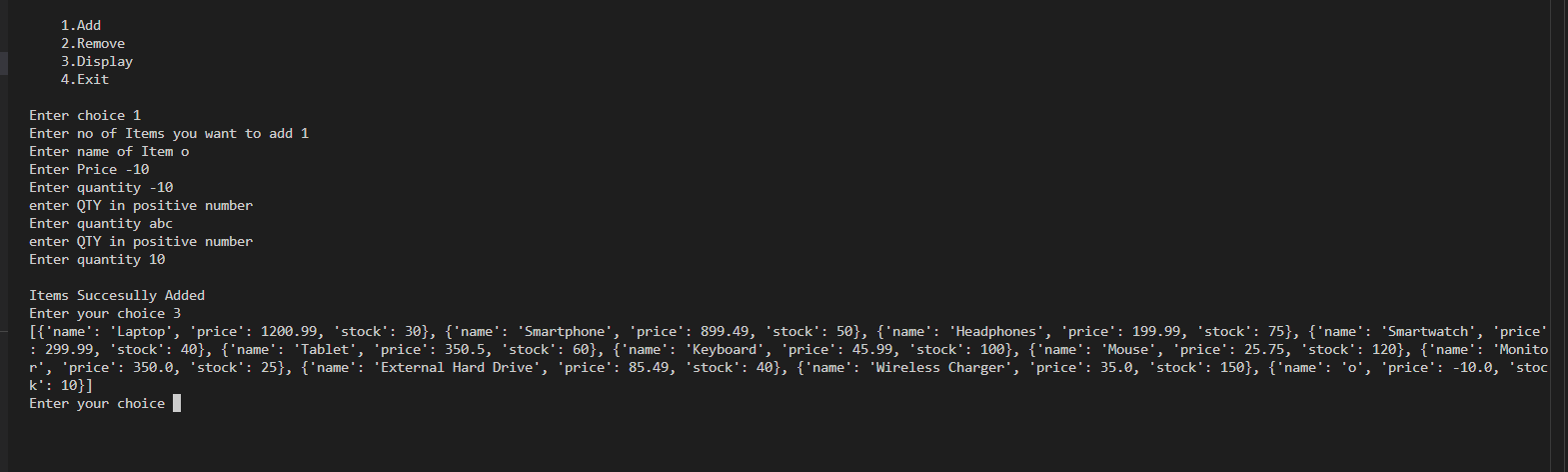
                   break

        if f==1:

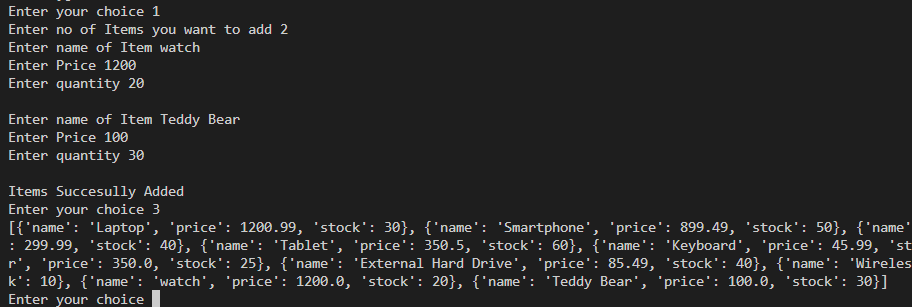
          print("Item Not in List")

**Output:**

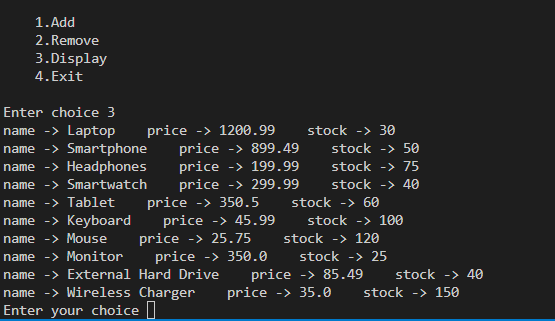
**Adding QTY as string or negative number**

****

**Adding in right way**

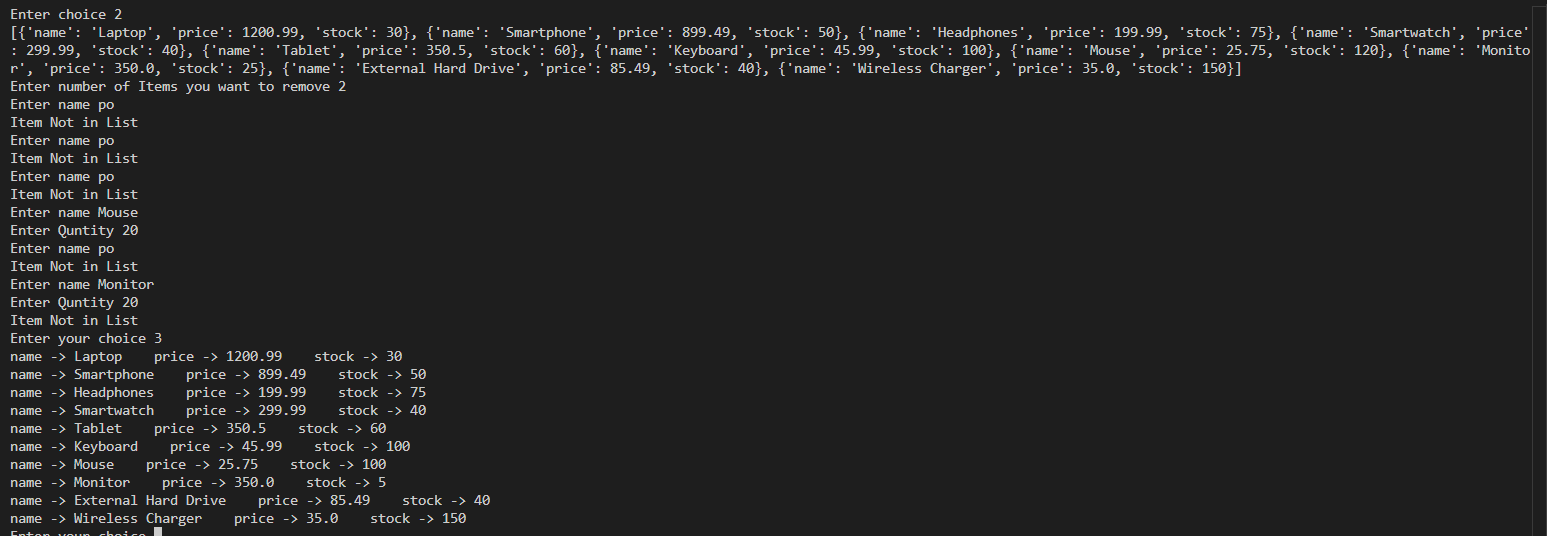
****

**Display Items**

****

**Remove Items:**

If Item not found



If QTY is greater

