class CashRegister:

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

self.reset\_register()

def add\_item(self, item\_name, price, count):

if item\_name in self.total\_items.keys():

raise ValueError("Item already exist")

self.total\_items[item\_name] = {"count": count, "price": price}

self.total\_price += price

def remove\_item(self, item\_name):

if item\_name not in self.total\_items.keys():

raise ValueError("Item doesnt exist")

price = self.total\_items[item\_name]["price"]

del self.total\_items[item\_name]

self.total\_price -= price

def apply\_discount(self, discount):

if(discount > 100):

raise ValueError("Discount must be in %")

self.discount = discount

return self.get\_total()

def get\_total(self):

total = 0

for item in self.total\_items.keys():

total += self.total\_items[item]["price"] \* self.total\_items[item]["count"]

if self.discount > 0:

total -= total \* self.discount/100

return total

def show\_items(self):

for item in self.total\_items.keys():

print("Item Name {} : Price {} : Count {}".format(item, self.total\_items[item]["price"], self.total\_items[item]["count"]))

print("Total Price: {}".format(self.total\_price))

def reset\_register(self):

self.total\_items = {}

self.total\_price = 0

self.discount = 0

cashRegister = CashRegister()

cashRegister.show\_items()

print("=====================================")

cashRegister.add\_item("Socks", 10, 5)

cashRegister.add\_item("Jacket", 40, 1)

cashRegister.show\_items()

print("Total Amount %d" % cashRegister.get\_total())

print("Total Amount After Discount of 50%% %d" % cashRegister.apply\_discount(50))

print("=====================================")

cashRegister.remove\_item("Socks")

cashRegister.show\_items()

cashRegister.apply\_discount(0)

print("Total Amount %d" % cashRegister.get\_total())

print("Total Amount After Discount of 50%% %d" % cashRegister.apply\_discount(50))

print("=====================================")

cashRegister.reset\_register()

cashRegister.show\_items()

class Student:

def \_\_init\_\_(self, name, age, id, specialization):

self.name = name

self.age = age

self.id = id

self.subjects = {}

self.specialization = specialization

def get\_name(self):

return self.name;

def get\_age(self):

return self.age;

def get\_id(self):

return self.id;

def add\_subject(self, subject\_name, grade):

self.subjects[subject\_name] = float(grade)

def remove\_subject(self, subject\_name):

if subject\_name in self.subjects:

del self.subjects[subject\_name]

else:

raise ValueError("Subject doesnt exit")

def get\_subjects(self):

return self.subjects;

def overall(self):

if len(self.subjects.keys()) == 0:

return 0

total = 0

for subject in self.subjects.keys():

total += self.subjects[subject]

return total/len(self.subjects.keys());

def get\_specialization(self):

return self.specialization

class DataScienceStudent(Student):

def \_\_init\_\_(self, name, age, id):

super().\_\_init\_\_(name, age, id, "DataScience")

class SoftwareStudent(Student):

def \_\_init\_\_(self, name, age, id):

super().\_\_init\_\_(name, age, id, "Software")

dataScienceStudent1 = DataScienceStudent("Ada",25,"11111")

dataScienceStudent1.add\_subject("maths", 80)

dataScienceStudent1.add\_subject("statistics", 85)

dataScienceStudent1.add\_subject("python", 100)

print(dataScienceStudent1.overall())

print(dataScienceStudent1.get\_specialization())

dataScienceStudent1.remove\_subject("python")

print(dataScienceStudent1.overall())

softwareStudent = SoftwareStudent("Lara",25,"11112")

print(softwareStudent.overall())