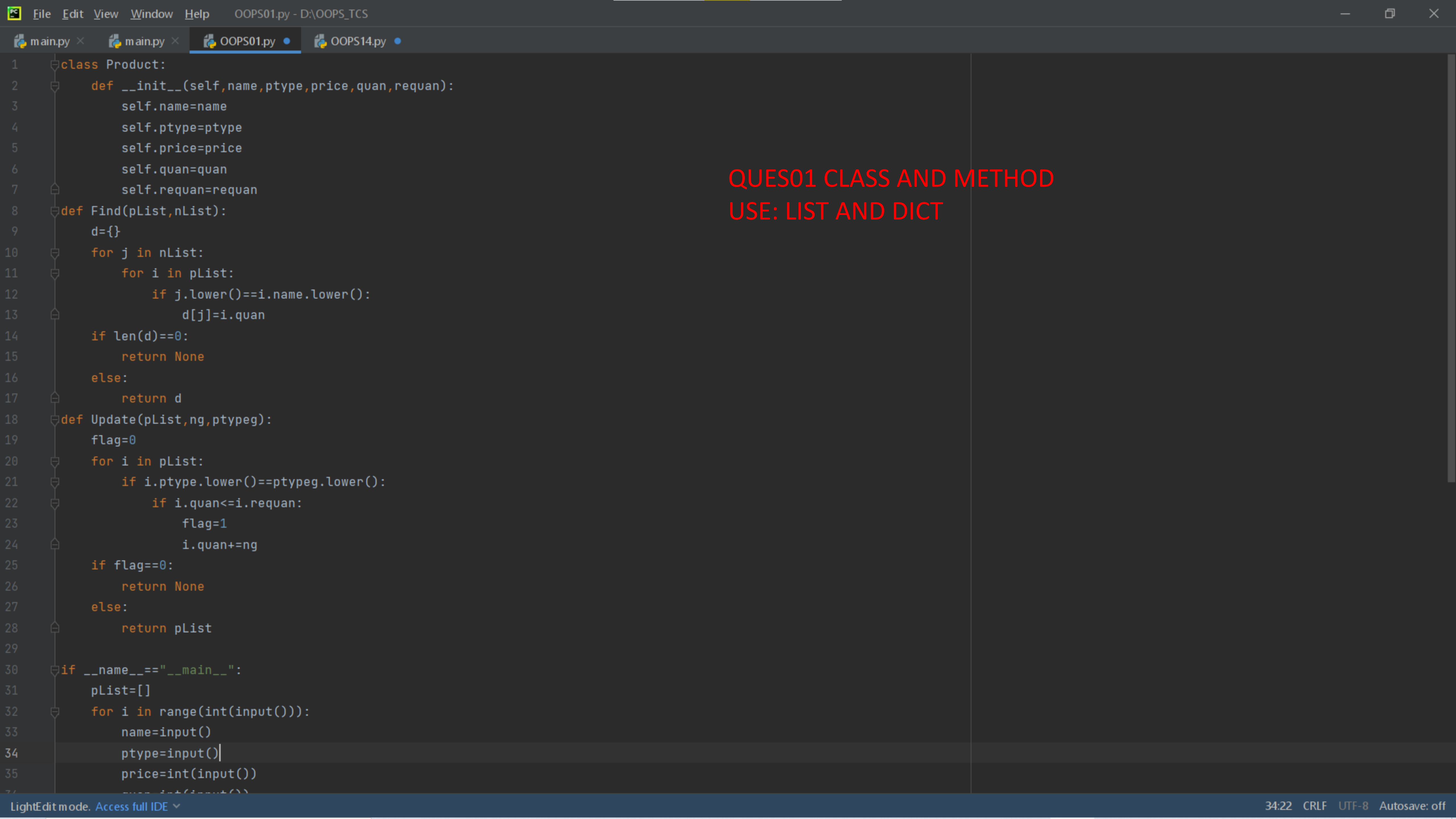
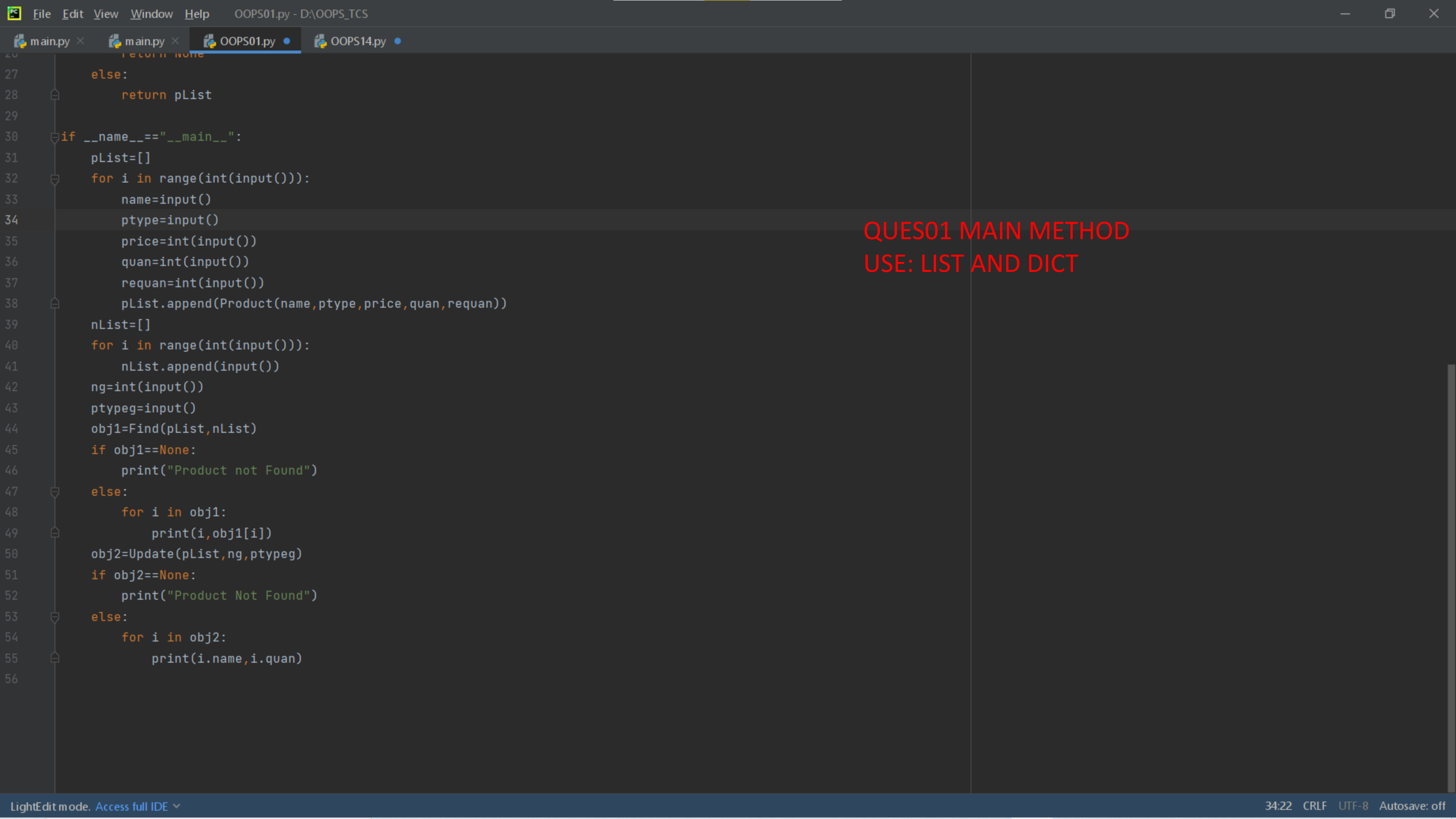
Clear All OOPS Concepts Now


File Edit View Window Help OOPS01.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
LightEdit mode. Access full IDE
34:22 CRLF UTF-8 Autosave: off

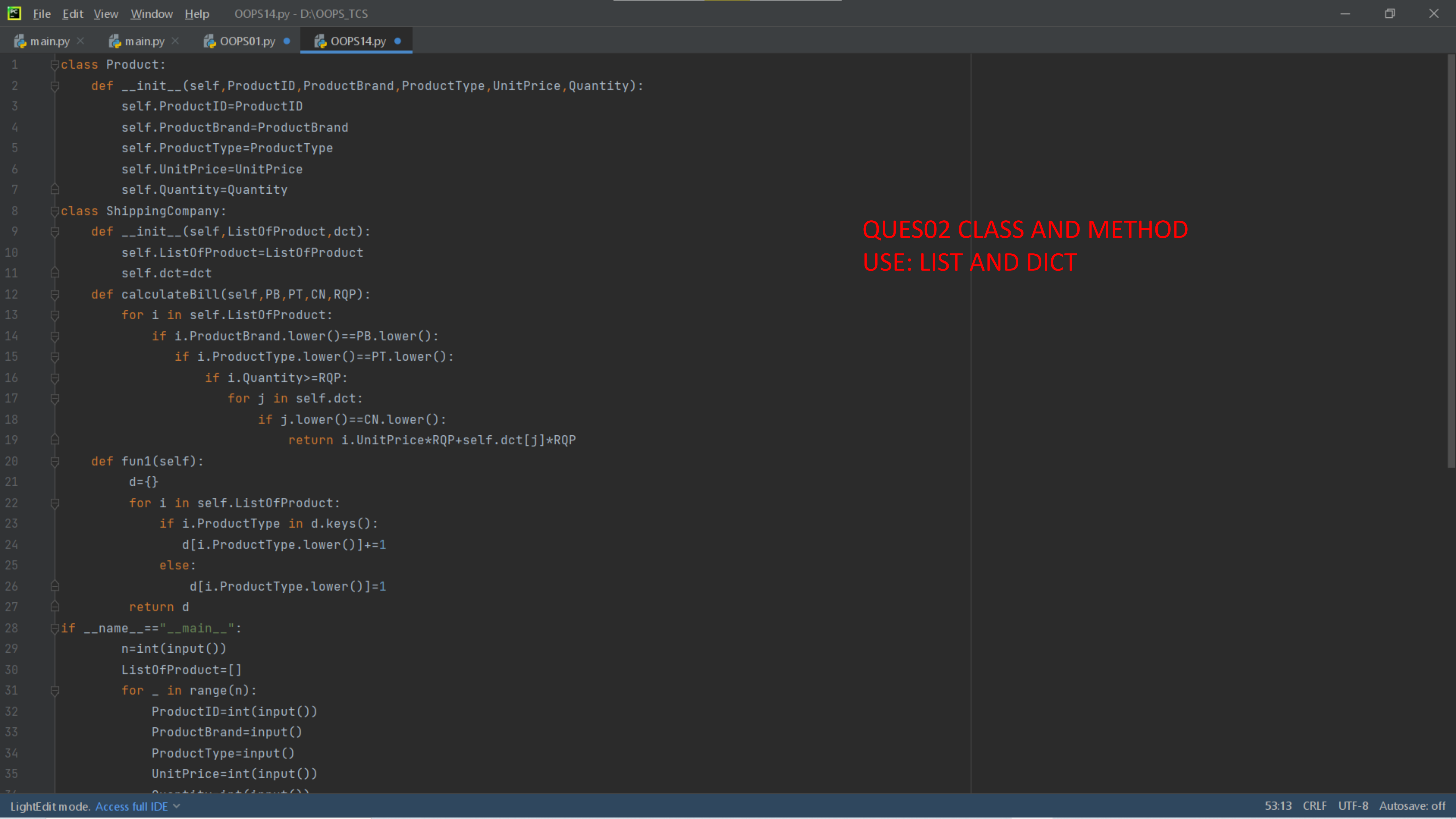
QUES01 CLASS AND METHOD
USE: LIST AND DICT

```python
class Product:
    def __init__(self,name,ptype,price,quan,requan):
        self.name=name
        self.ptype=ptype
        self.price=price
        self.quan=quan
        self.requan=requan
def Find(pList,nList):
    d={}
    for j in nList:
        for i in pList:
            if j.lower()==i.name.lower():
                d[j]=i.quan
    if len(d)==0:
        return None
    else:
        return d
def Update(pList,ng,ptypeg):
    flag=0
    for i in pList:
        if i.ptype.lower()==ptypeg.lower():
            if i.quan<=i.requan:
                flag=1
                i.quan+=ng
    if flag==0:
        return None
    else:
        return pList

if __name__=="__main__":
    pList=[]
    for i in range(int(input())):
        name=input()
        ptype=input()
        price=int(input())
```


File Edit View Window Help OOPS01.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
LightEdit mode. Access full IDE
34:22 CRLF UTF-8 Autosave: off

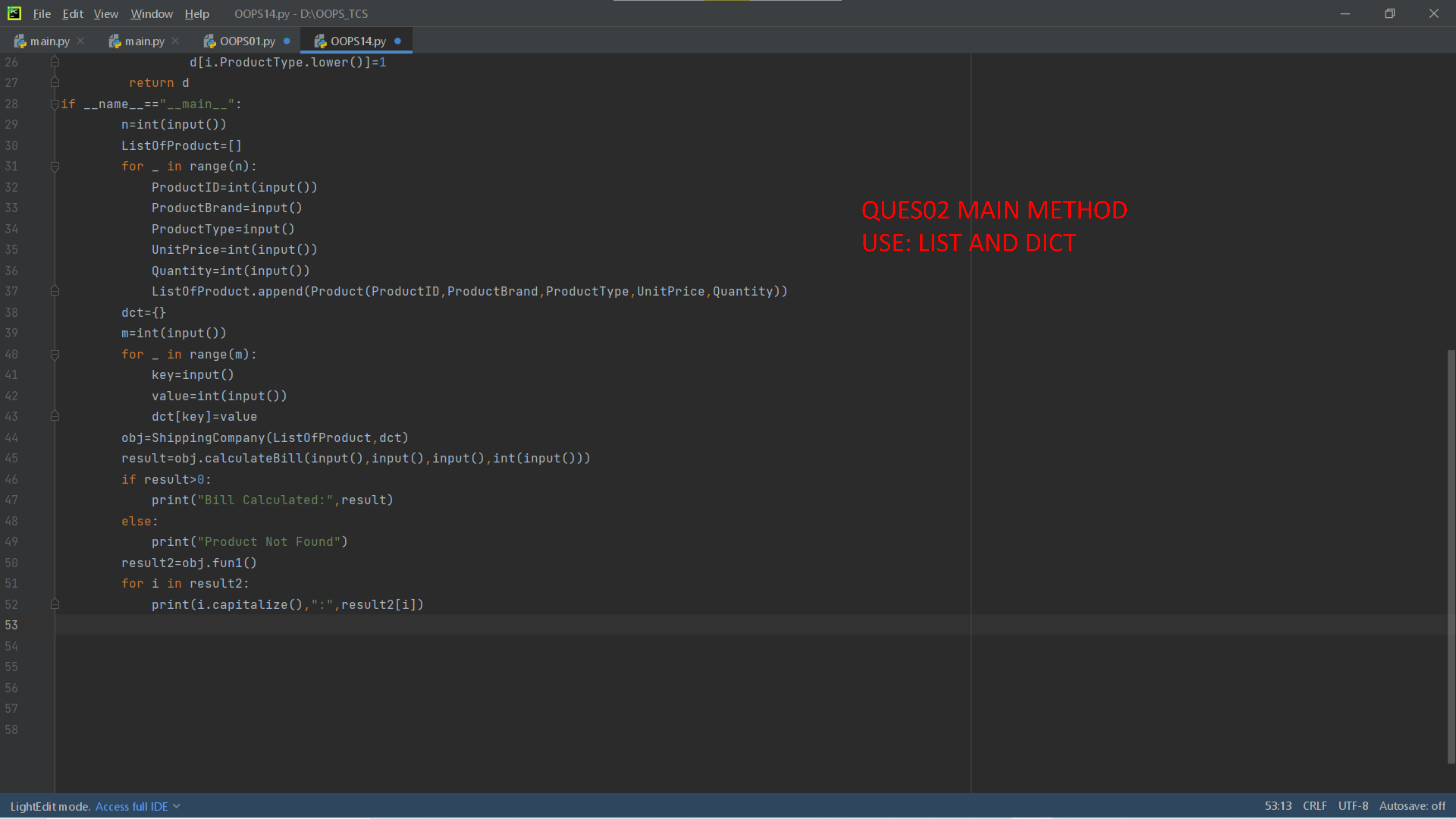
```
    else:
        return pList

if __name__=="__main__":
    pList=[]
    for i in range(int(input())):
        name=input()
        ptype=input()
        price=int(input())
        quan=int(input())
        requan=int(input())
        pList.append(Product(name,ptype,price,quan,requan))
    nList=[]
    for i in range(int(input())):
        nList.append(input())
    ng=int(input())
    ptypeg=input()
    obj1=Find(pList,nList)
    if obj1==None:
        print("Product not Found")
    else:
        for i in obj1:
            print(i,obj1[i])
    obj2=Update(pList,ng,ptypeg)
    if obj2==None:
        print("Product Not Found")
    else:
        for i in obj2:
            print(i.name,i.quan)
```

QUES01 MAIN METHOD
USE: LIST AND DICT


File Edit View Window Help
OOPS14.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
LightEdit mode. Access full IDE
53:13 CRLF UTF-8 Autosave: off

QUES02 CLASS AND METHOD
USE: LIST AND DICT

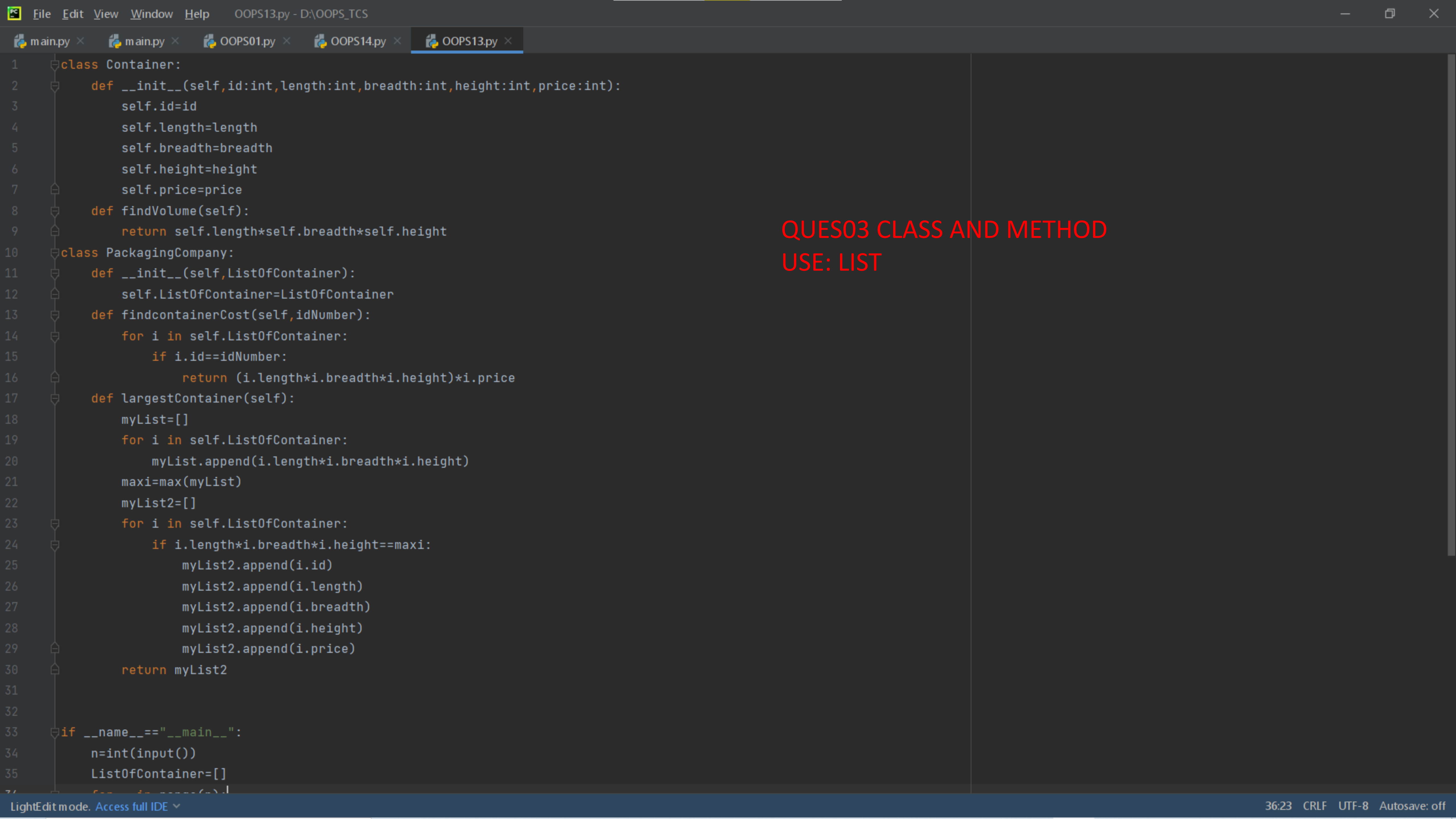
```
class Product:
    def __init__(self,ProductID,ProductBrand,ProductType,UnitPrice,Quantity):
        self.ProductID=ProductID
        self.ProductBrand=ProductBrand
        self.ProductType=ProductType
        self.UnitPrice=UnitPrice
        self.Quantity=Quantity
class ShippingCompany:
    def __init__(self,ListOfProduct,dct):
        self.ListOfProduct=ListOfProduct
        self.dct=dct
    def calculateBill(self,PB,PT,CN,RQP):
        for i in self.ListOfProduct:
            if i.ProductBrand.lower()==PB.lower():
               if i.ProductType.lower()==PT.lower():
                   if i.Quantity>=RQP:
                      for j in self.dct:
                          if j.lower()==CN.lower():
                              return i.UnitPrice*RQP+self.dct[j]*RQP
    def fun1(self):
         d={}
         for i in self.ListOfProduct:
             if i.ProductType in d.keys():
                d[i.ProductType.lower()]+=1
             else:
                 d[i.ProductType.lower()]=1
         return d
if __name__=="__main__":
        n=int(input())
        ListOfProduct=[]
        for _ in range(n):
            ProductID=int(input())
            ProductBrand=input()
            ProductType=input()
            UnitPrice=int(input())
```



```
                d[i.ProductType.lower()]=1
            return d
if __name__=="__main__":
        n=int(input())
        ListOfProduct=[]
        for _ in range(n):
            ProductID=int(input())
            ProductBrand=input()
            ProductType=input()
            UnitPrice=int(input())
            Quantity=int(input())
            ListOfProduct.append(Product(ProductID,ProductBrand,ProductType,UnitPrice,Quantity))
        dct={}
        m=int(input())
        for _ in range(m):
            key=input()
            value=int(input())
            dct[key]=value
        obj=ShippingCompany(ListOfProduct,dct)
        result=obj.calculateBill(input(),input(),input(),int(input()))
        if result>0:
            print("Bill Calculated:",result)
        else:
            print("Product Not Found")
        result2=obj.fun1()
        for i in result2:
            print(i.capitalize(),":",result2[i])
```

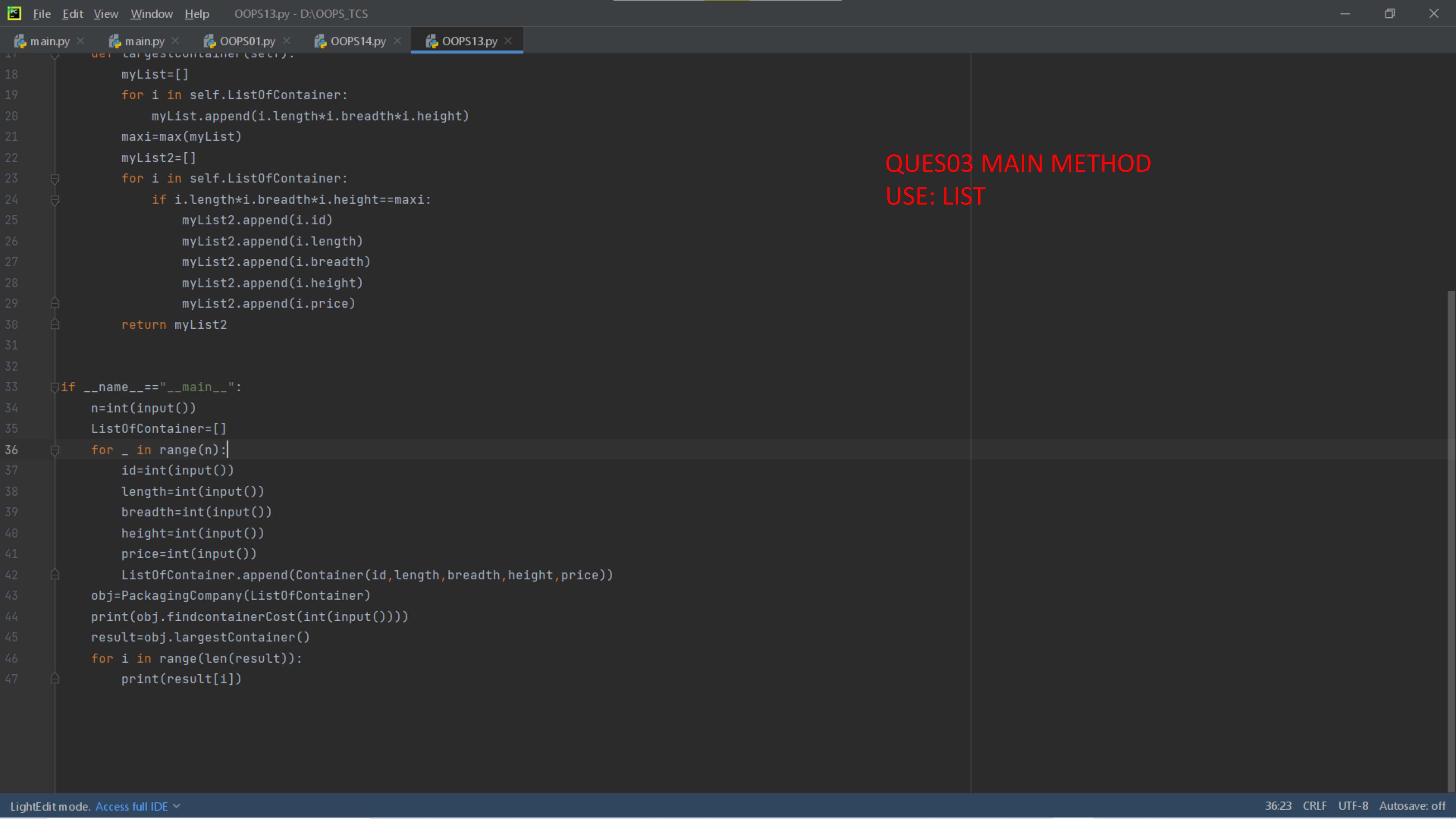
QUES02 MAIN METHOD
USE: LIST AND DICT

LightEdit mode. Access full IDE
53:13 CRLF UTF-8 Autosave: off


File Edit View Window Help
OOPS13.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
OOPS13.py
LightEdit mode. Access full IDE
36:23 CRLF UTF-8 Autosave: off

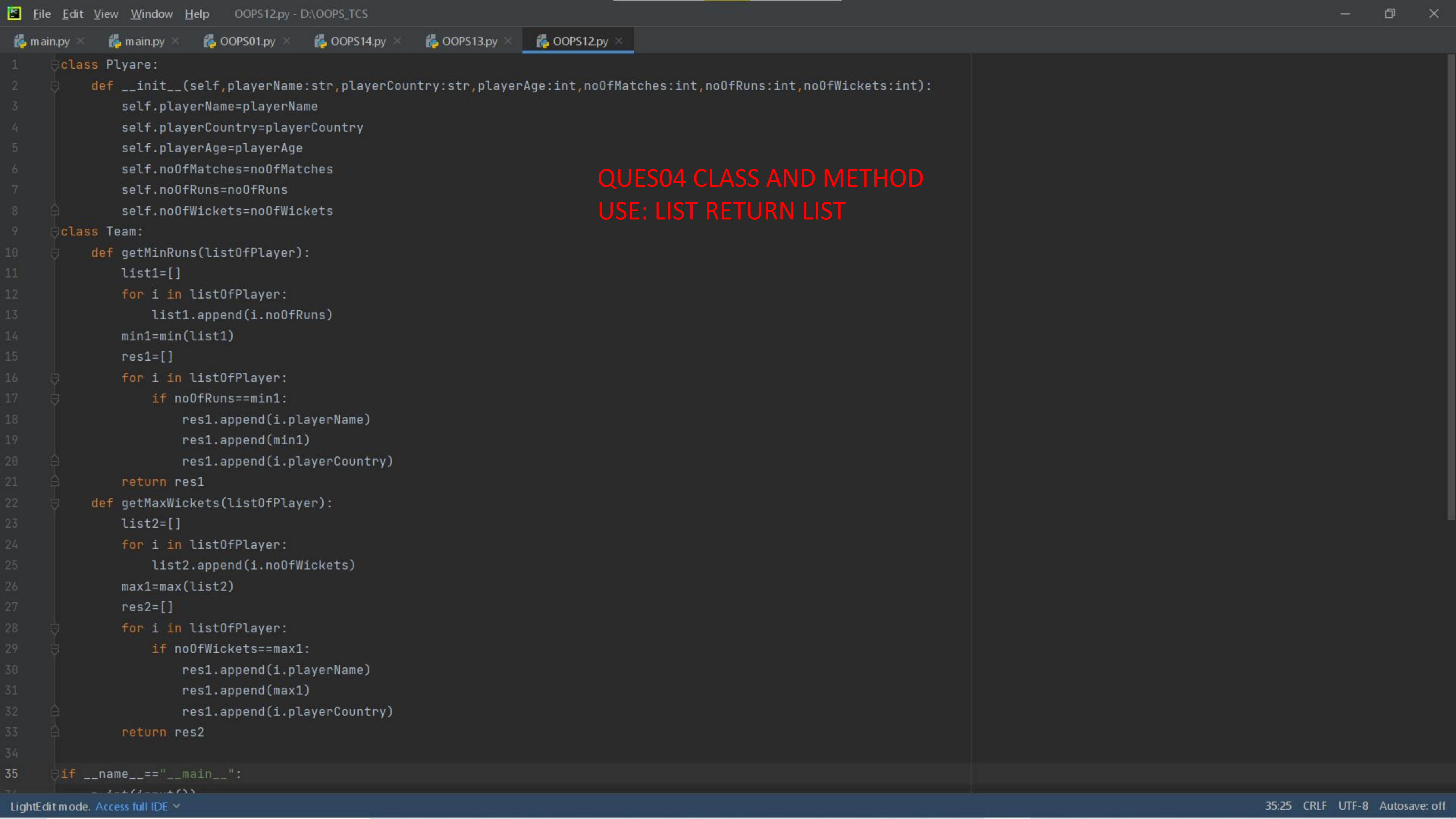
QUES03 CLASS AND METHOD
USE: LIST

```python
class Container:
    def __init__(self,id:int,length:int,breadth:int,height:int,price:int):
        self.id=id
        self.length=length
        self.breadth=breadth
        self.height=height
        self.price=price
    def findVolume(self):
        return self.length*self.breadth*self.height
class PackagingCompany:
    def __init__(self,ListOfContainer):
        self.ListOfContainer=ListOfContainer
    def findcontainerCost(self,idNumber):
        for i in self.ListOfContainer:
            if i.id==idNumber:
                return (i.length*i.breadth*i.height)*i.price
    def largestContainer(self):
        myList=[]
        for i in self.ListOfContainer:
            myList.append(i.length*i.breadth*i.height)
        maxi=max(myList)
        myList2=[]
        for i in self.ListOfContainer:
            if i.length*i.breadth*i.height==maxi:
                myList2.append(i.id)
                myList2.append(i.length)
                myList2.append(i.breadth)
                myList2.append(i.height)
                myList2.append(i.price)
        return myList2


if __name__=="__main__":
    n=int(input())
    ListOfContainer=[]
```


File Edit View Window Help OOPS13.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
OOPS13.py
LightEdit mode. Access full IDE
36:23 CRLF UTF-8 Autosave: off

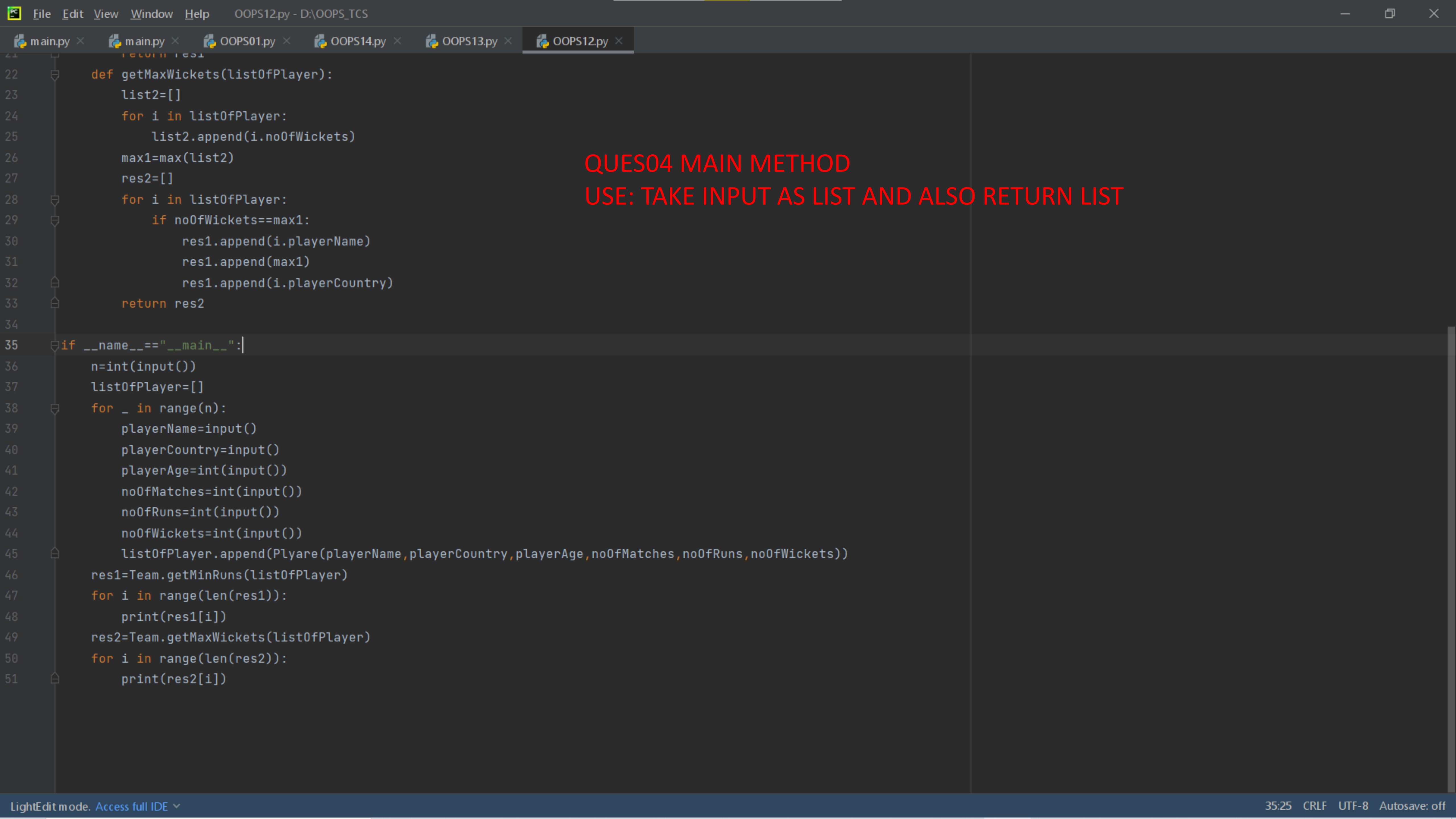
QUES03 MAIN METHOD
USE: LIST

```
        myList=[]
        for i in self.ListOfContainer:
            myList.append(i.length*i.breadth*i.height)
        maxi=max(myList)
        myList2=[]
        for i in self.ListOfContainer:
            if i.length*i.breadth*i.height==maxi:
                myList2.append(i.id)
                myList2.append(i.length)
                myList2.append(i.breadth)
                myList2.append(i.height)
                myList2.append(i.price)
        return myList2


if __name__=="__main__":
    n=int(input())
    ListOfContainer=[]
    for _ in range(n):
        id=int(input())
        length=int(input())
        breadth=int(input())
        height=int(input())
        price=int(input())
        ListOfContainer.append(Container(id,length,breadth,height,price))
    obj=PackagingCompany(ListOfContainer)
    print(obj.findcontainerCost(int(input())))
    result=obj.largestContainer()
    for i in range(len(result)):
        print(result[i])
```


File Edit View Window Help
OOPS12.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
OOPS13.py
OOPS12.py
LightEdit mode. Access full IDE
35:25 CRLF UTF-8 Autosave: off

QUES04 CLASS AND METHOD
USE: LIST RETURN LIST

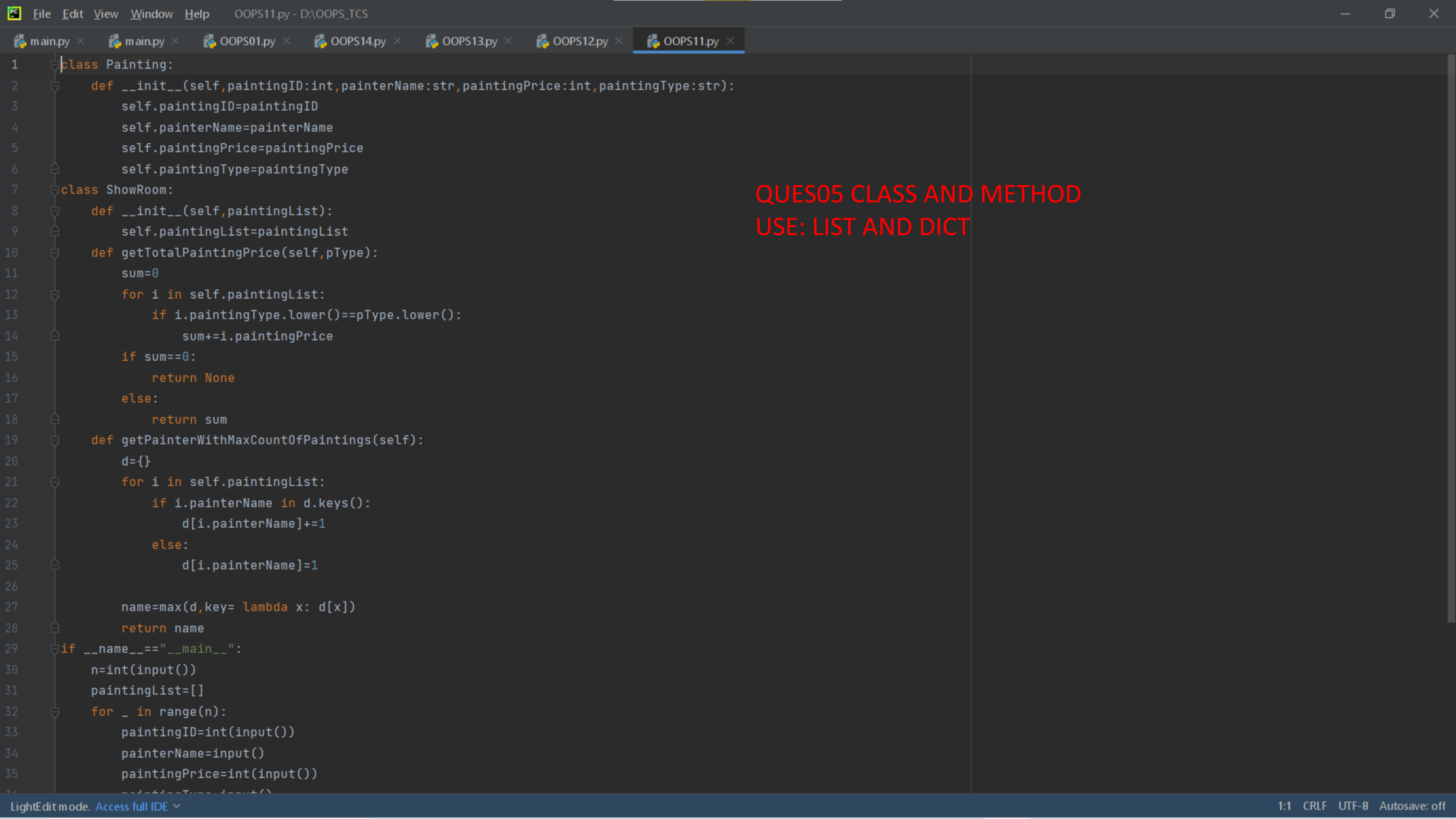
```
class Plyare:
    def __init__(self,playerName:str,playerCountry:str,playerAge:int,noOfMatches:int,noOfRuns:int,noOfWickets:int):
        self.playerName=playerName
        self.playerCountry=playerCountry
        self.playerAge=playerAge
        self.noOfMatches=noOfMatches
        self.noOfRuns=noOfRuns
        self.noOfWickets=noOfWickets
class Team:
    def getMinRuns(listOfPlayer):
        list1=[]
        for i in listOfPlayer:
            list1.append(i.noOfRuns)
        min1=min(list1)
        res1=[]
        for i in listOfPlayer:
            if noOfRuns==min1:
                res1.append(i.playerName)
                res1.append(min1)
                res1.append(i.playerCountry)
        return res1
    def getMaxWickets(listOfPlayer):
        list2=[]
        for i in listOfPlayer:
            list2.append(i.noOfWickets)
        max1=max(list2)
        res2=[]
        for i in listOfPlayer:
            if noOfWickets==max1:
                res1.append(i.playerName)
                res1.append(max1)
                res1.append(i.playerCountry)
        return res2

if __name__=="__main__":
```


File Edit View Window Help OOPS12.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
OOPS13.py
OOPS12.py
LightEdit mode. Access full IDE
35:25 CRLF UTF-8 Autosave: off

```
    def getMaxWickets(listOfPlayer):
        list2=[]
        for i in listOfPlayer:
            list2.append(i.noOfWickets)
        max1=max(list2)
        res2=[]
        for i in listOfPlayer:
            if noOfWickets==max1:
                res1.append(i.playerName)
                res1.append(max1)
                res1.append(i.playerCountry)
        return res2

if __name__=="__main__":
    n=int(input())
    listOfPlayer=[]
    for _ in range(n):
        playerName=input()
        playerCountry=input()
        playerAge=int(input())
        noOfMatches=int(input())
        noOfRuns=int(input())
        noOfWickets=int(input())
        listOfPlayer.append(Plyare(playerName,playerCountry,playerAge,noOfMatches,noOfRuns,noOfWickets))
    res1=Team.getMinRuns(listOfPlayer)
    for i in range(len(res1)):
        print(res1[i])
    res2=Team.getMaxWickets(listOfPlayer)
    for i in range(len(res2)):
        print(res2[i])
```

QUES04 MAIN METHOD

USE: TAKE INPUT AS LIST AND ALSO RETURN LIST



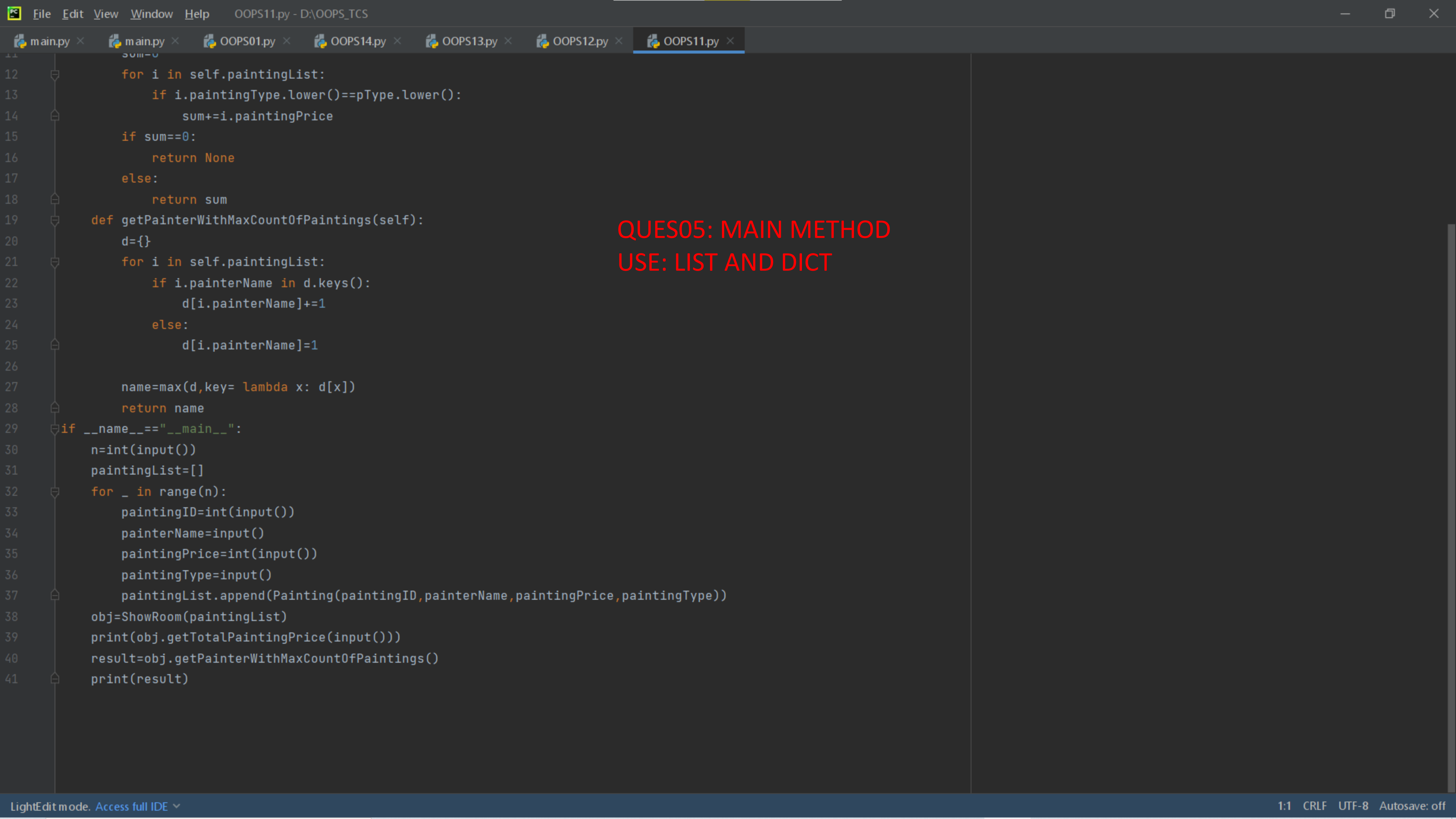
File Edit View Window Help OOPS11.py - D:\OOPS_TCS

main.py × main.py × OOPS01.py × OOPS14.py × OOPS13.py × OOPS12.py × OOPS11.py ×

QUES05 CLASS AND METHOD
USE: LIST AND DICT

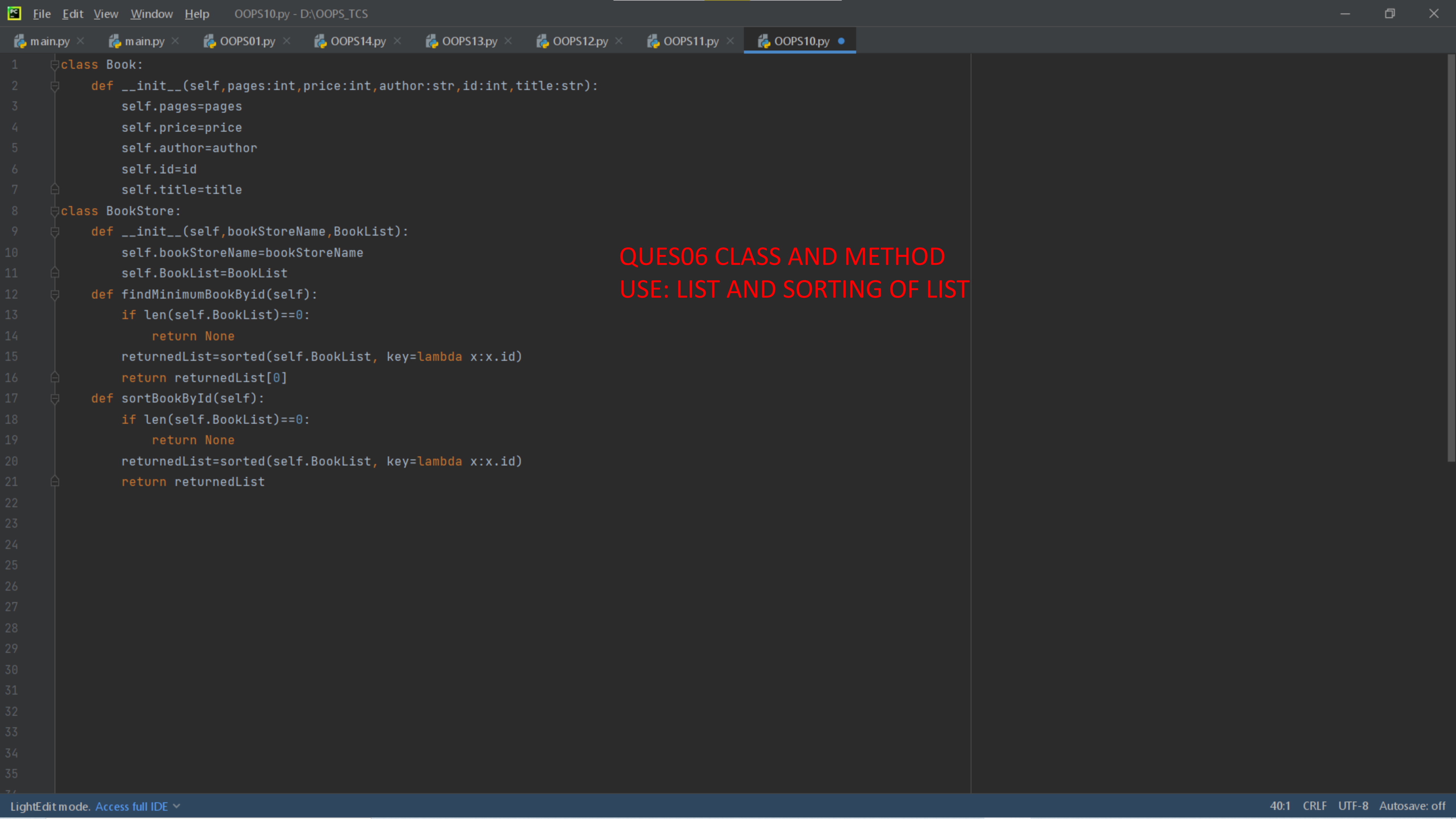
```python
class Painting:
    def __init__(self,paintingID:int,painterName:str,paintingPrice:int,paintingType:str):
        self.paintingID=paintingID
        self.painterName=painterName
        self.paintingPrice=paintingPrice
        self.paintingType=paintingType
class ShowRoom:
    def __init__(self,paintingList):
        self.paintingList=paintingList
    def getTotalPaintingPrice(self,pType):
        sum=0
        for i in self.paintingList:
            if i.paintingType.lower()==pType.lower():
                sum+=i.paintingPrice
        if sum==0:
            return None
        else:
            return sum
    def getPainterWithMaxCountOfPaintings(self):
        d={}
        for i in self.paintingList:
            if i.painterName in d.keys():
                d[i.painterName]+=1
            else:
                d[i.painterName]=1

        name=max(d,key= lambda x: d[x])
        return name
if __name__=="__main__":
    n=int(input())
    paintingList=[]
    for _ in range(n):
        paintingID=int(input())
        painterName=input()
        paintingPrice=int(input())
```

LightEdit mode. Access full IDE 1:1 CRLF UTF-8 Autosave: off


File Edit View Window Help
OOPS11.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
OOPS13.py
OOPS12.py
OOPS11.py
LightEdit mode. Access full IDE
1:1 CRLF UTF-8 Autosave: off

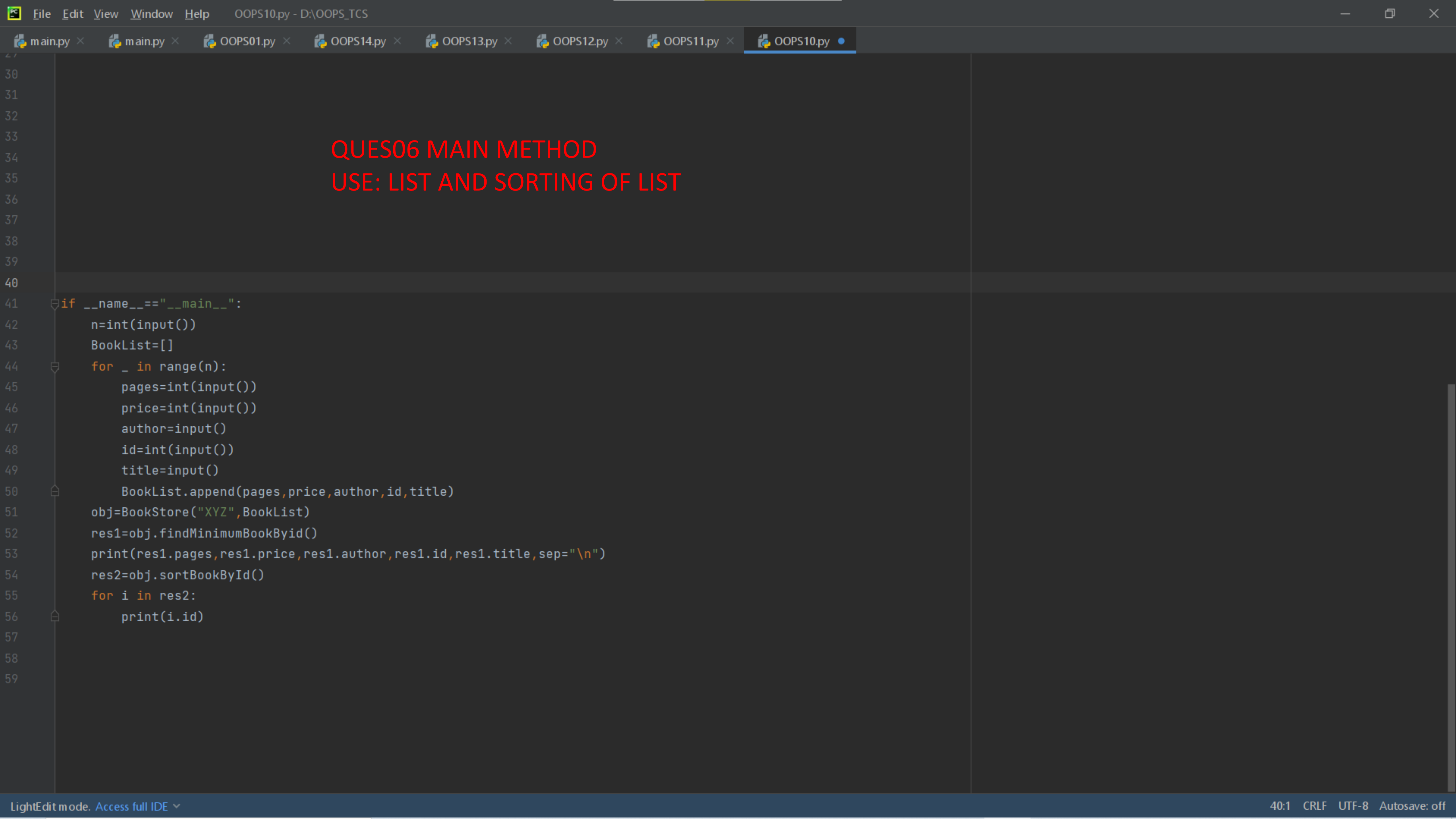
```python
        sum=0
        for i in self.paintingList:
            if i.paintingType.lower()==pType.lower():
                sum+=i.paintingPrice
        if sum==0:
            return None
        else:
            return sum
    def getPainterWithMaxCountOfPaintings(self):
        d={}
        for i in self.paintingList:
            if i.painterName in d.keys():
                d[i.painterName]+=1
            else:
                d[i.painterName]=1

        name=max(d,key= lambda x: d[x])
        return name
if __name__=="__main__":
    n=int(input())
    paintingList=[]
    for _ in range(n):
        paintingID=int(input())
        painterName=input()
        paintingPrice=int(input())
        paintingType=input()
        paintingList.append(Painting(paintingID,painterName,paintingPrice,paintingType))
    obj=ShowRoom(paintingList)
    print(obj.getTotalPaintingPrice(input()))
    result=obj.getPainterWithMaxCountOfPaintings()
    print(result)
```

QUES05: MAIN METHOD
USE: LIST AND DICT


File Edit View Window Help OOPS10.py - D:\OOPS_TCS
main.py
main.py
OOPS01.py
OOPS14.py
OOPS13.py
OOPS12.py
OOPS11.py
OOPS10.py
LightEdit mode. Access full IDE
40:1 CRLF UTF-8 Autosave: off

```python
class Book:
    def __init__(self,pages:int,price:int,author:str,id:int,title:str):
        self.pages=pages
        self.price=price
        self.author=author
        self.id=id
        self.title=title
class BookStore:
    def __init__(self,bookStoreName,BookList):
        self.bookStoreName=bookStoreName
        self.BookList=BookList
    def findMinimumBookByid(self):
        if len(self.BookList)==0:
            return None
        returnedList=sorted(self.BookList, key=lambda x:x.id)
        return returnedList[0]
    def sortBookById(self):
        if len(self.BookList)==0:
            return None
        returnedList=sorted(self.BookList, key=lambda x:x.id)
        return returnedList
```

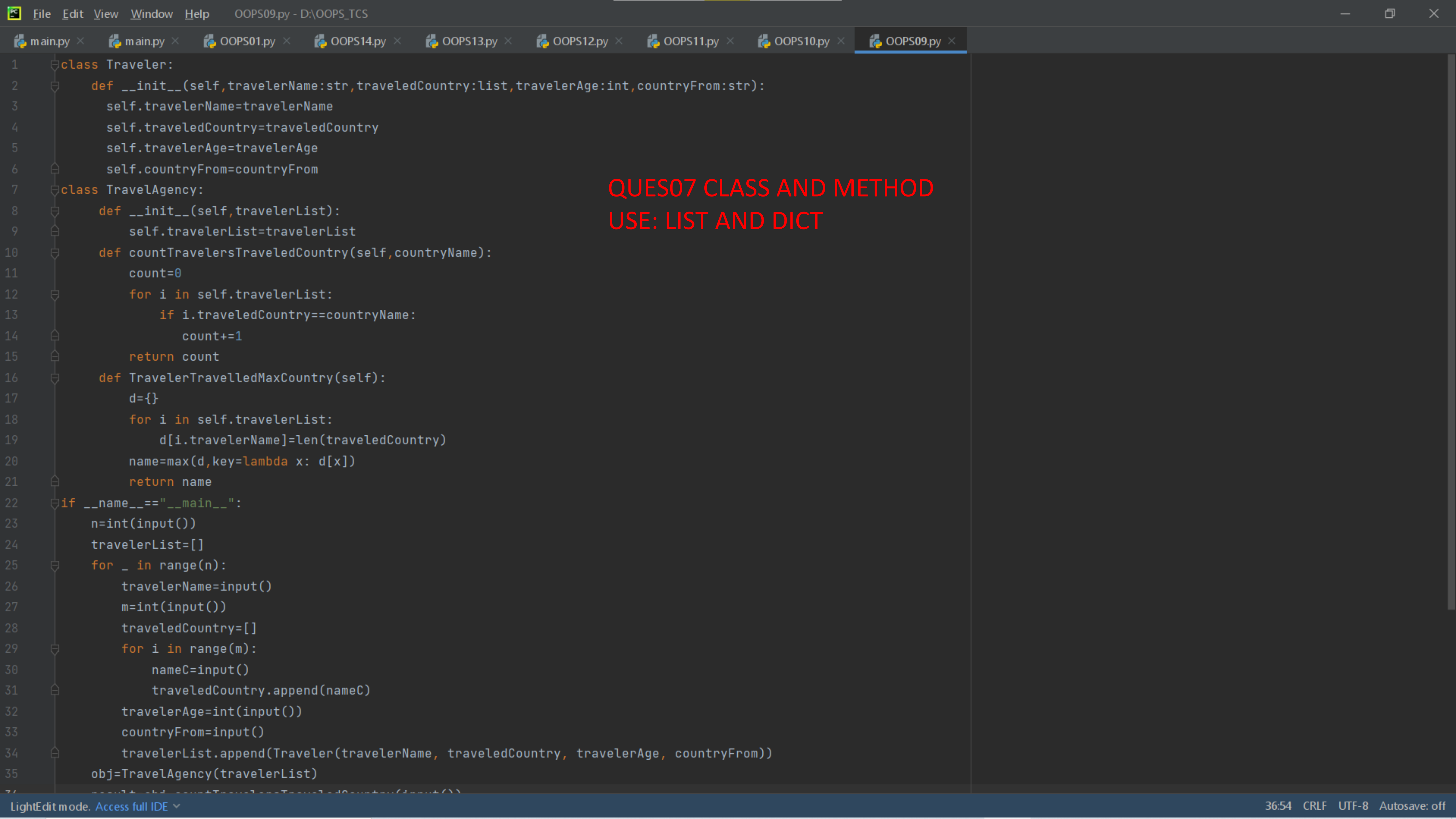
QUES06 CLASS AND METHOD
USE: LIST AND SORTING OF LIST


File Edit View Window Help OOPS10.py - D:\OOPS_TCS
main.py main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py
LightEdit mode. Access full IDE
40:1 CRLF UTF-8 Autosave: off

QUES06 MAIN METHOD

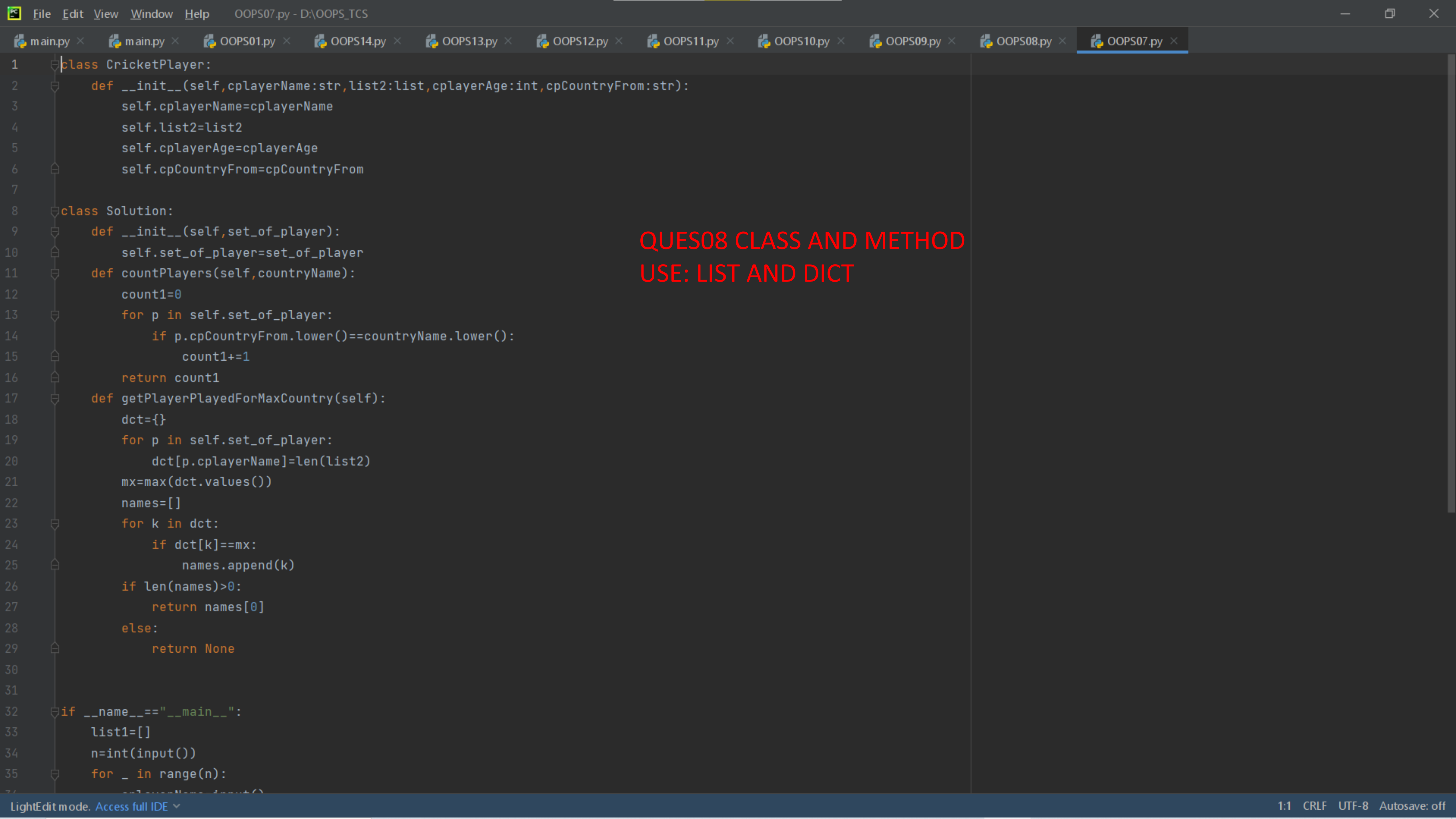
USE: LIST AND SORTING OF LIST

```python
if __name__=="__main__":
    n=int(input())
    BookList=[]
    for _ in range(n):
        pages=int(input())
        price=int(input())
        author=input()
        id=int(input())
        title=input()
        BookList.append(pages,price,author,id,title)
    obj=BookStore("XYZ",BookList)
    res1=obj.findMinimumBookByid()
    print(res1.pages,res1.price,res1.author,res1.id,res1.title,sep="\n")
    res2=obj.sortBookById()
    for i in res2:
        print(i.id)
```



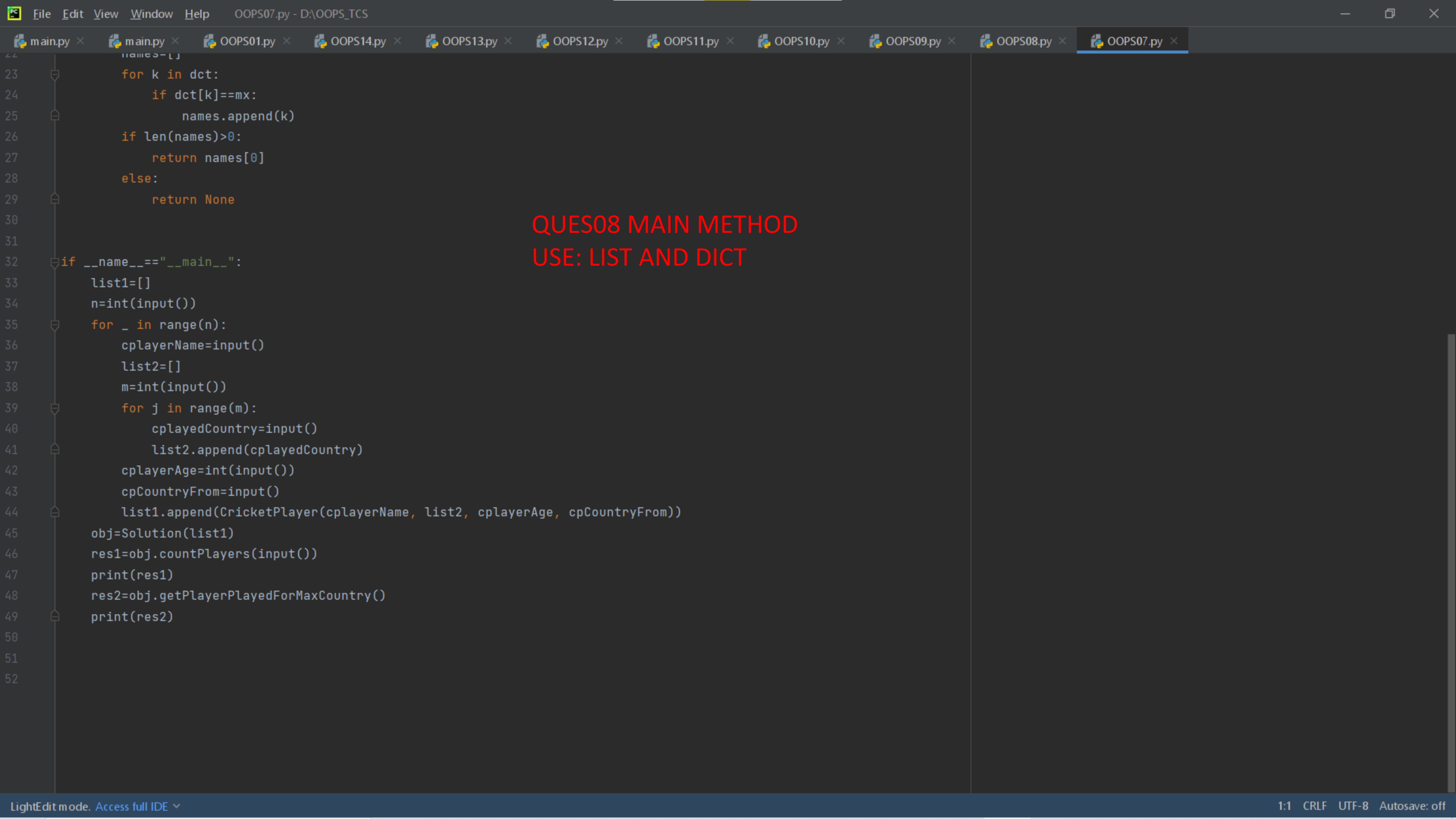
QUES07 CLASS AND METHOD
USE: LIST AND DICT

```
class Traveler:
    def __init__(self,travelerName:str,traveledCountry:list,travelerAge:int,countryFrom:str):
      self.travelerName=travelerName
      self.traveledCountry=traveledCountry
      self.travelerAge=travelerAge
      self.countryFrom=countryFrom
class TravelAgency:
     def __init__(self,travelerList):
         self.travelerList=travelerList
     def countTravelersTraveledCountry(self,countryName):
         count=0
         for i in self.travelerList:
             if i.traveledCountry==countryName:
                count+=1
         return count
     def TravelerTravelledMaxCountry(self):
         d={}
         for i in self.travelerList:
             d[i.travelerName]=len(traveledCountry)
         name=max(d,key=lambda x: d[x])
         return name
if __name__=="__main__":
    n=int(input())
    travelerList=[]
    for _ in range(n):
        travelerName=input()
        m=int(input())
        traveledCountry=[]
        for i in range(m):
            nameC=input()
            traveledCountry.append(nameC)
        travelerAge=int(input())
        countryFrom=input()
        travelerList.append(Traveler(travelerName, traveledCountry, travelerAge, countryFrom))
    obj=TravelAgency(travelerList)
```


File Edit View Window Help OOPS07.py - D:\OOPS_TCS
main.py main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py
LightEdit mode. Access full IDE
1:1 CRLF UTF-8 Autosave: off

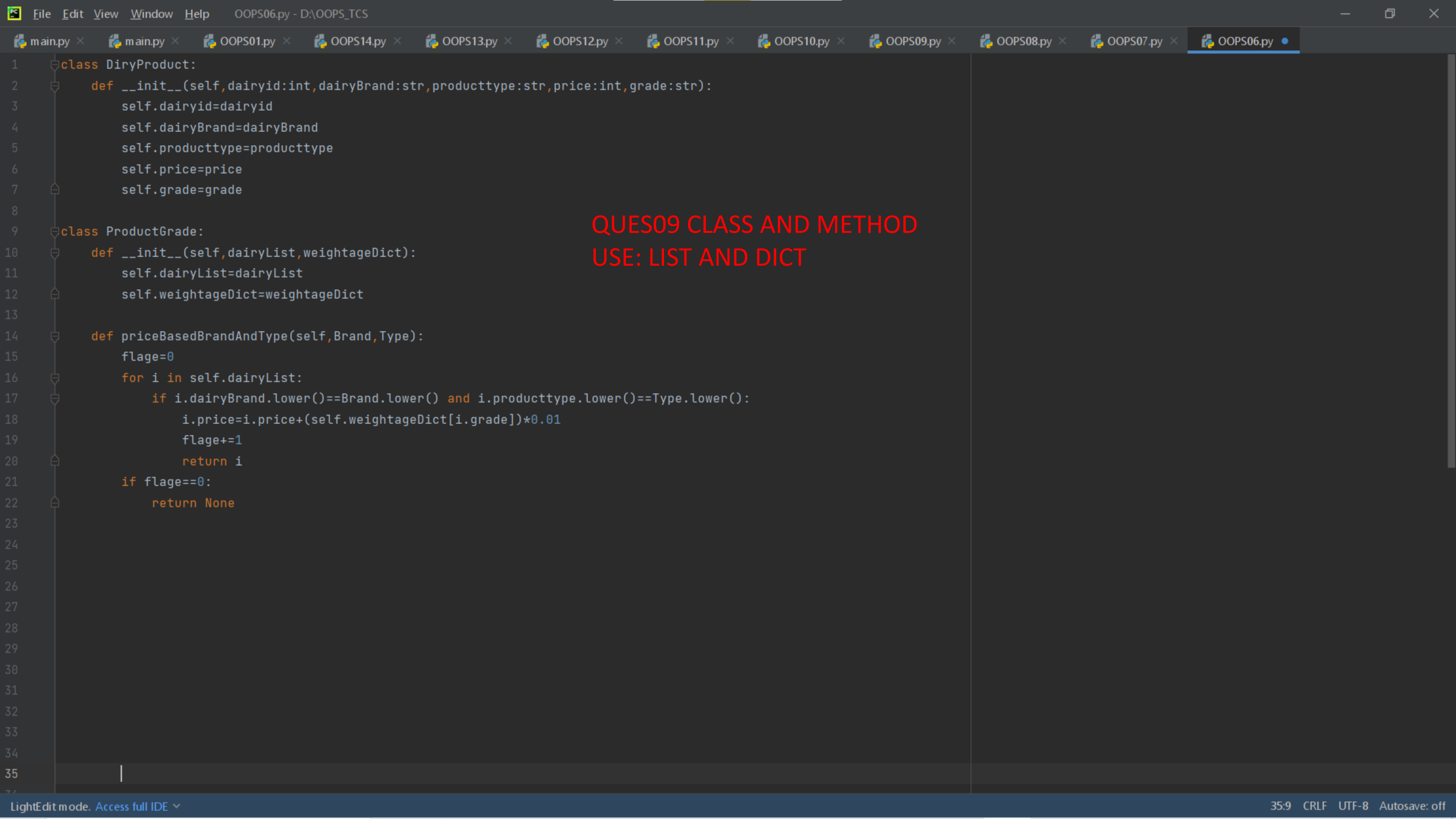
```python
class CricketPlayer:
    def __init__(self,cplayerName:str,list2:list,cplayerAge:int,cpCountryFrom:str):
        self.cplayerName=cplayerName
        self.list2=list2
        self.cplayerAge=cplayerAge
        self.cpCountryFrom=cpCountryFrom

class Solution:
    def __init__(self,set_of_player):
        self.set_of_player=set_of_player
    def countPlayers(self,countryName):
        count1=0
        for p in self.set_of_player:
            if p.cpCountryFrom.lower()==countryName.lower():
                count1+=1
        return count1
    def getPlayerPlayedForMaxCountry(self):
        dct={}
        for p in self.set_of_player:
            dct[p.cplayerName]=len(list2)
        mx=max(dct.values())
        names=[]
        for k in dct:
            if dct[k]==mx:
                names.append(k)
        if len(names)>0:
            return names[0]
        else:
            return None


if __name__=="__main__":
    list1=[]
    n=int(input())
    for _ in range(n):
```

QUES08 CLASS AND METHOD
USE: LIST AND DICT


File Edit View Window Help OOPS07.py - D:\OOPS_TCS
main.py main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py
LightEdit mode. Access full IDE 1:1 CRLF UTF-8 Autosave: off

```
        for k in dct:
            if dct[k]==mx:
                names.append(k)
        if len(names)>0:
            return names[0]
        else:
            return None


if __name__=="__main__":
    list1=[]
    n=int(input())
    for _ in range(n):
        cplayerName=input()
        list2=[]
        m=int(input())
        for j in range(m):
            cplayedCountry=input()
            list2.append(cplayedCountry)
        cplayerAge=int(input())
        cpCountryFrom=input()
        list1.append(CricketPlayer(cplayerName, list2, cplayerAge, cpCountryFrom))
    obj=Solution(list1)
    res1=obj.countPlayers(input())
    print(res1)
    res2=obj.getPlayerPlayedForMaxCountry()
    print(res2)
```

QUES08 MAIN METHOD
USE: LIST AND DICT


File Edit View Window Help OOPS06.py - D:\OOPS_TCS

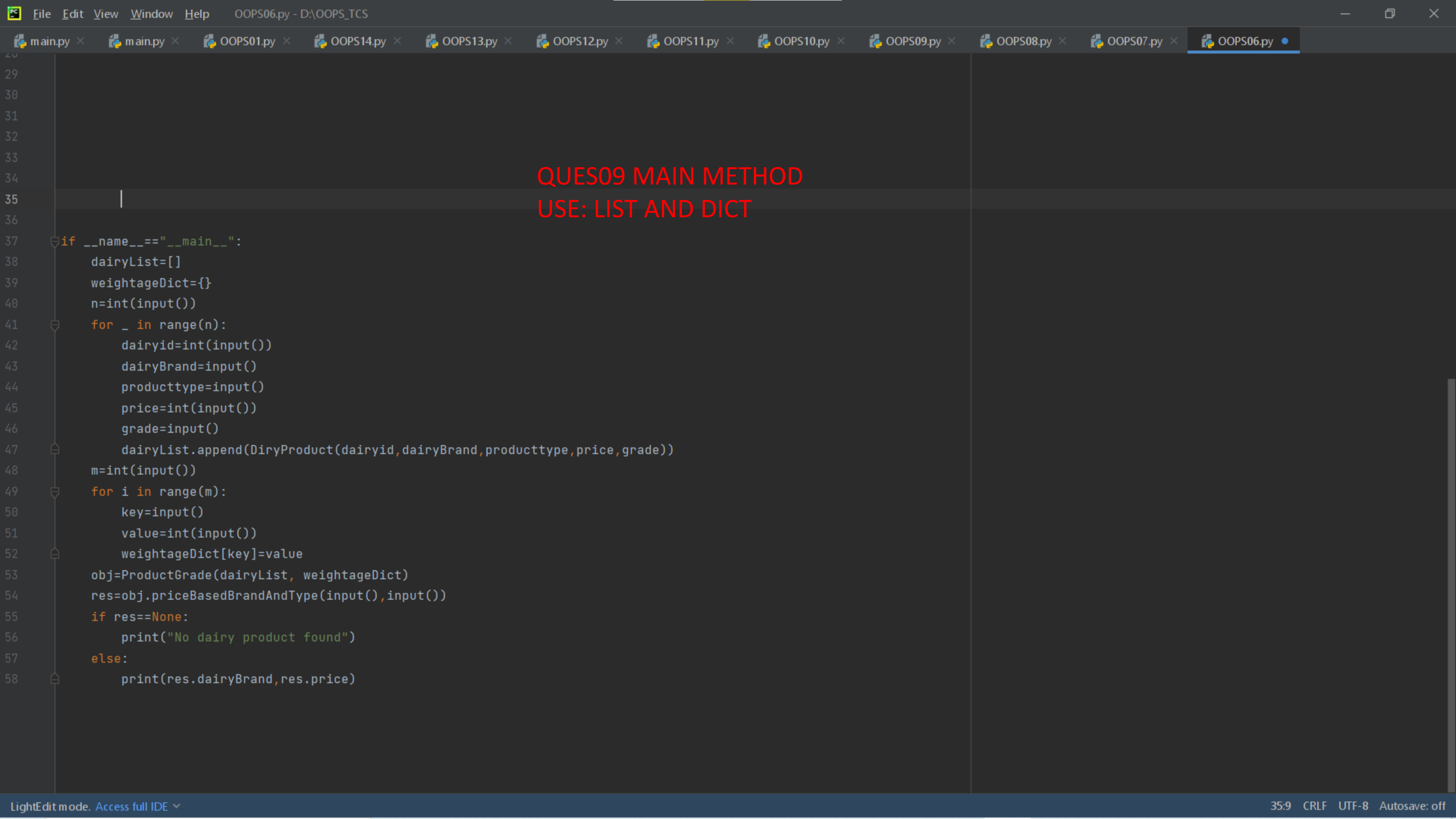
main.py main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py OOPS06.py

```python
class DiryProduct:
    def __init__(self,dairyid:int,dairyBrand:str,producttype:str,price:int,grade:str):
        self.dairyid=dairyid
        self.dairyBrand=dairyBrand
        self.producttype=producttype
        self.price=price
        self.grade=grade

class ProductGrade:
    def __init__(self,dairyList,weightageDict):
        self.dairyList=dairyList
        self.weightageDict=weightageDict

    def priceBasedBrandAndType(self,Brand,Type):
        flage=0
        for i in self.dairyList:
            if i.dairyBrand.lower()==Brand.lower() and i.producttype.lower()==Type.lower():
                i.price=i.price+(self.weightageDict[i.grade])*0.01
                flage+=1
                return i
        if flage==0:
            return None
```

QUES09 CLASS AND METHOD
USE: LIST AND DICT

LightEdit mode. Access full IDE

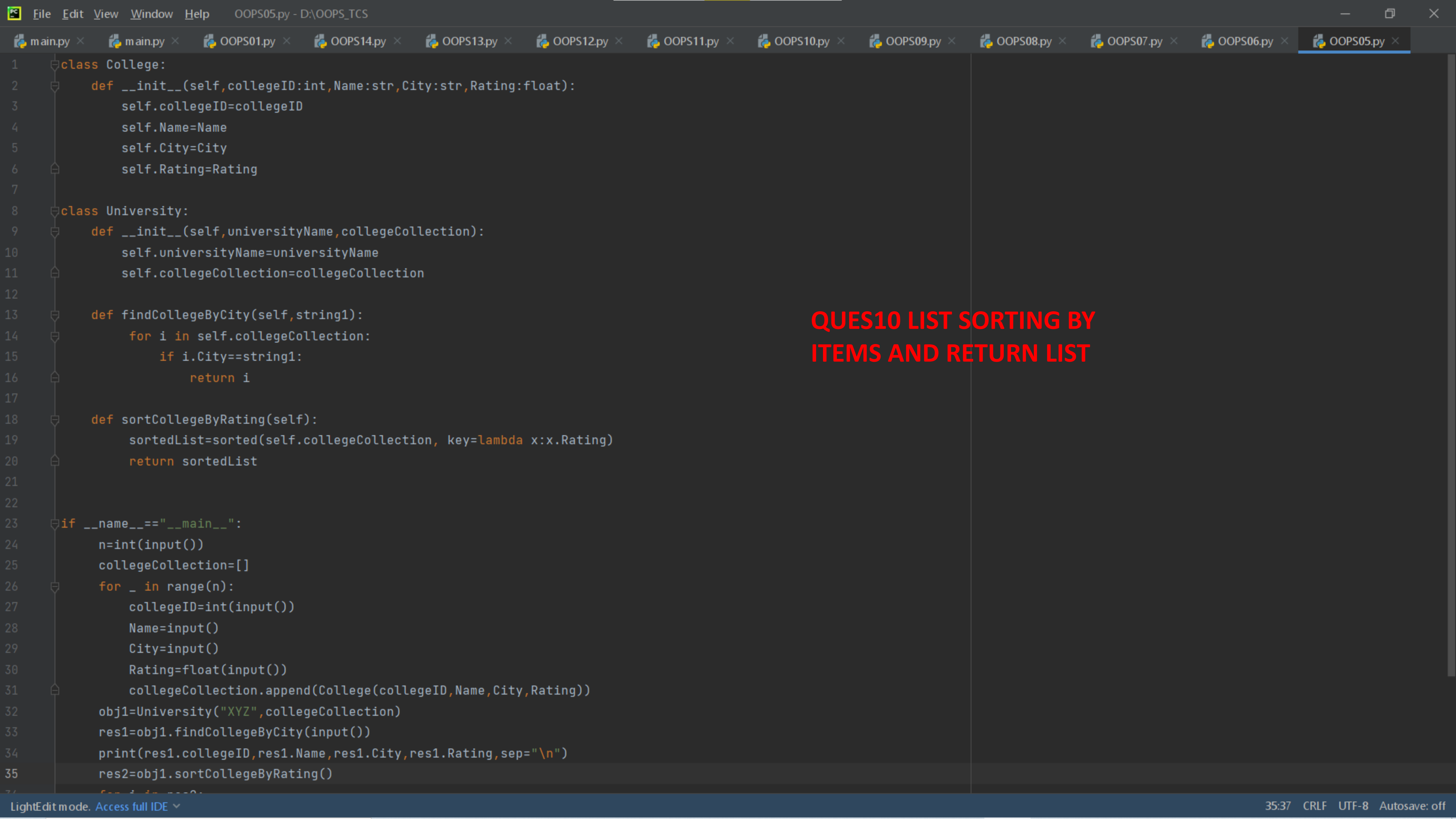
35:9 CRLF UTF-8 Autosave: off


File Edit View Window Help OOPS06.py - D:\OOPS_TCS
main.py main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py OOPS06.py

QUES09 MAIN METHOD
USE: LIST AND DICT

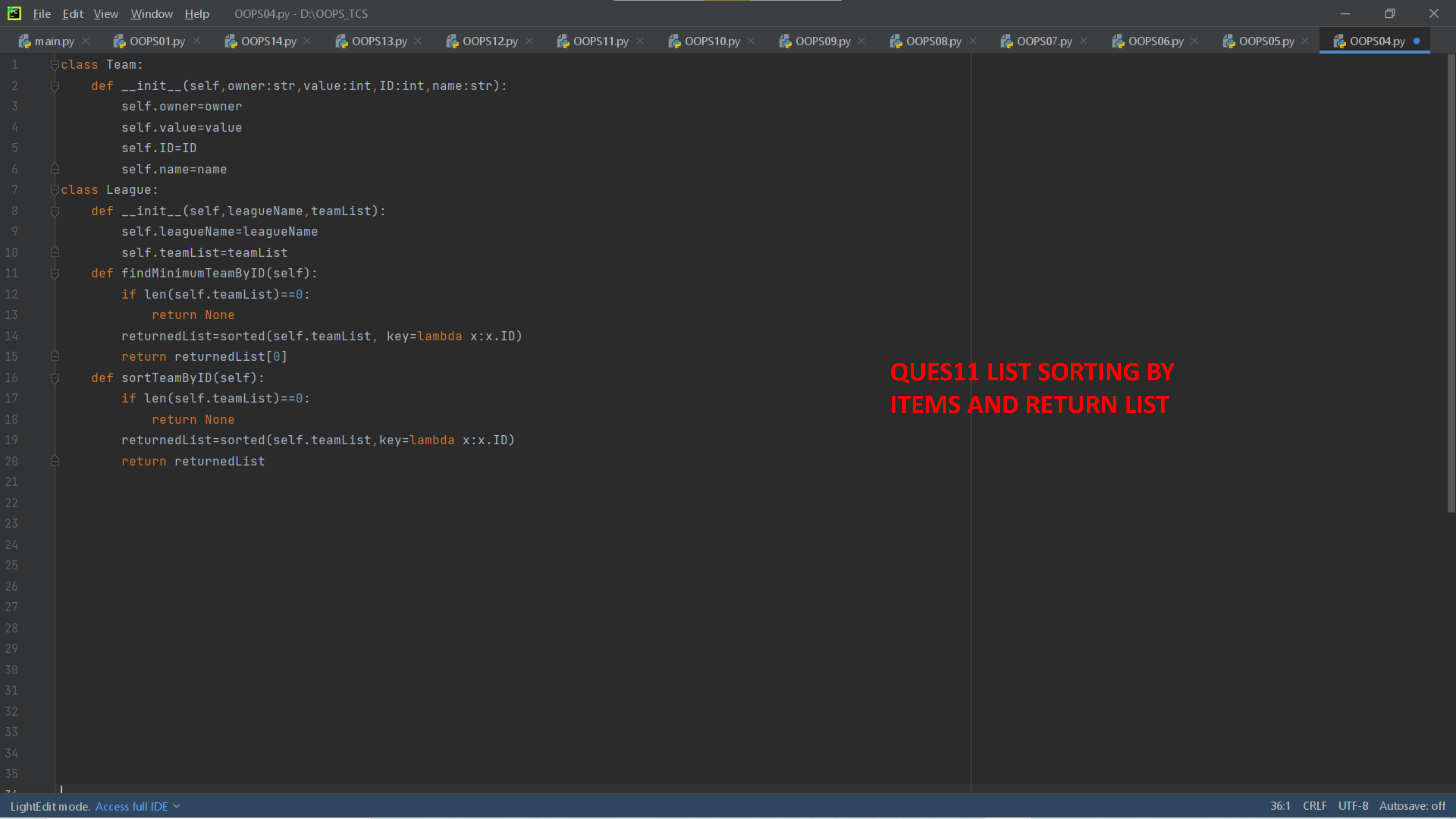
```
if __name__=="__main__":
    dairyList=[]
    weightageDict={}
    n=int(input())
    for _ in range(n):
        dairyid=int(input())
        dairyBrand=input()
        producttype=input()
        price=int(input())
        grade=input()
        dairyList.append(DiryProduct(dairyid,dairyBrand,producttype,price,grade))
    m=int(input())
    for i in range(m):
        key=input()
        value=int(input())
        weightageDict[key]=value
    obj=ProductGrade(dairyList, weightageDict)
    res=obj.priceBasedBrandAndType(input(),input())
    if res==None:
        print("No dairy product found")
    else:
        print(res.dairyBrand,res.price)
```

LightEdit mode. Access full IDE
35:9 CRLF UTF-8 Autosave: off



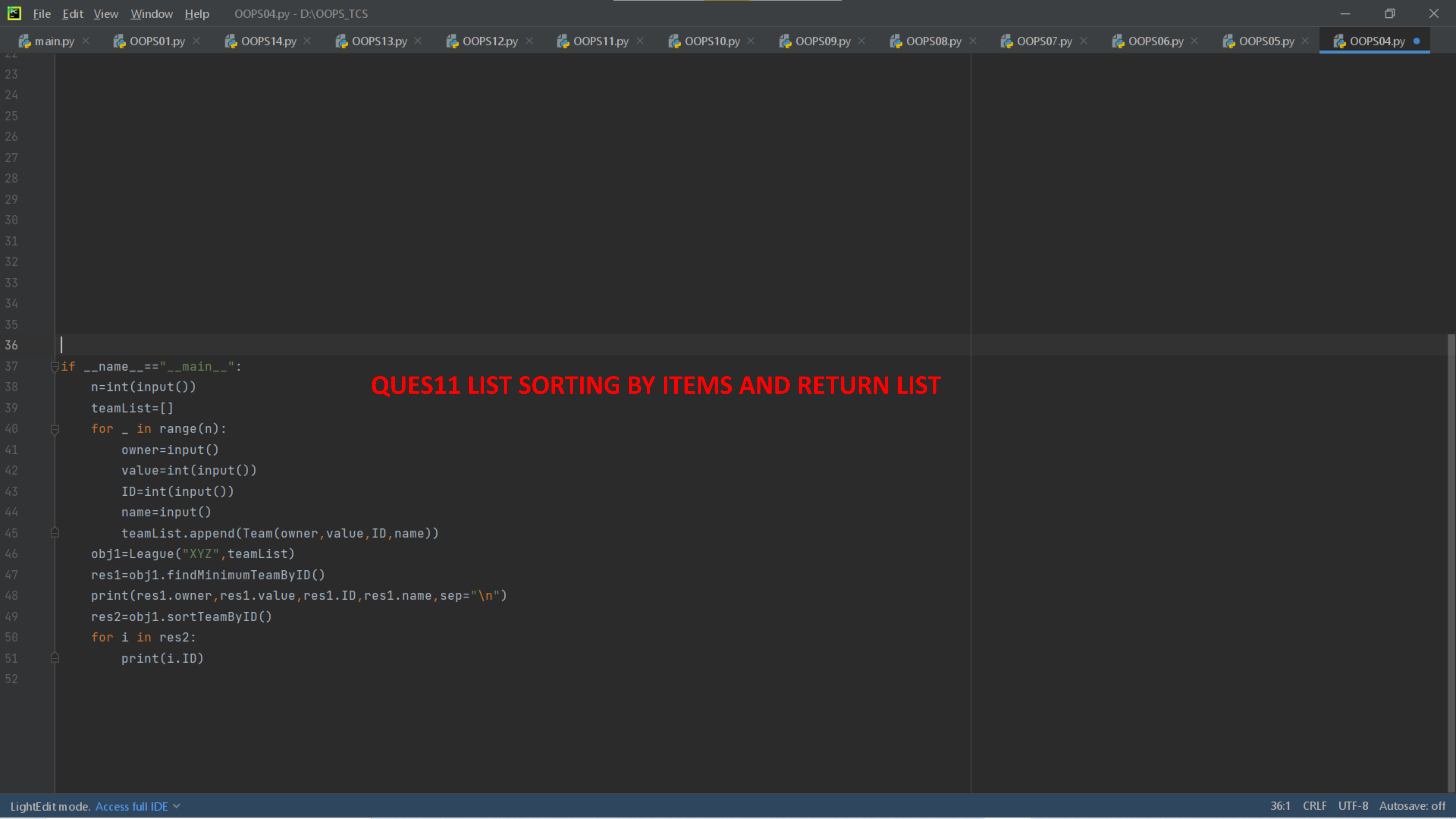
```python
class College:
    def __init__(self,collegeID:int,Name:str,City:str,Rating:float):
        self.collegeID=collegeID
        self.Name=Name
        self.City=City
        self.Rating=Rating

class University:
    def __init__(self,universityName,collegeCollection):
        self.universityName=universityName
        self.collegeCollection=collegeCollection

    def findCollegeByCity(self,string1):
         for i in self.collegeCollection:
             if i.City==string1:
                 return i

    def sortCollegeByRating(self):
         sortedList=sorted(self.collegeCollection, key=lambda x:x.Rating)
         return sortedList


if __name__=="__main__":
     n=int(input())
     collegeCollection=[]
     for _ in range(n):
         collegeID=int(input())
         Name=input()
         City=input()
         Rating=float(input())
         collegeCollection.append(College(collegeID,Name,City,Rating))
     obj1=University("XYZ",collegeCollection)
     res1=obj1.findCollegeByCity(input())
     print(res1.collegeID,res1.Name,res1.City,res1.Rating,sep="\n")
     res2=obj1.sortCollegeByRating()
```

QUES10 LIST SORTING BY ITEMS AND RETURN LIST


File Edit View Window Help OOPS04.py - D:\OOPS_TCS
main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py OOPS06.py OOPS05.py OOPS04.py
LightEdit mode. Access full IDE
36:1 CRLF UTF-8 Autosave: off

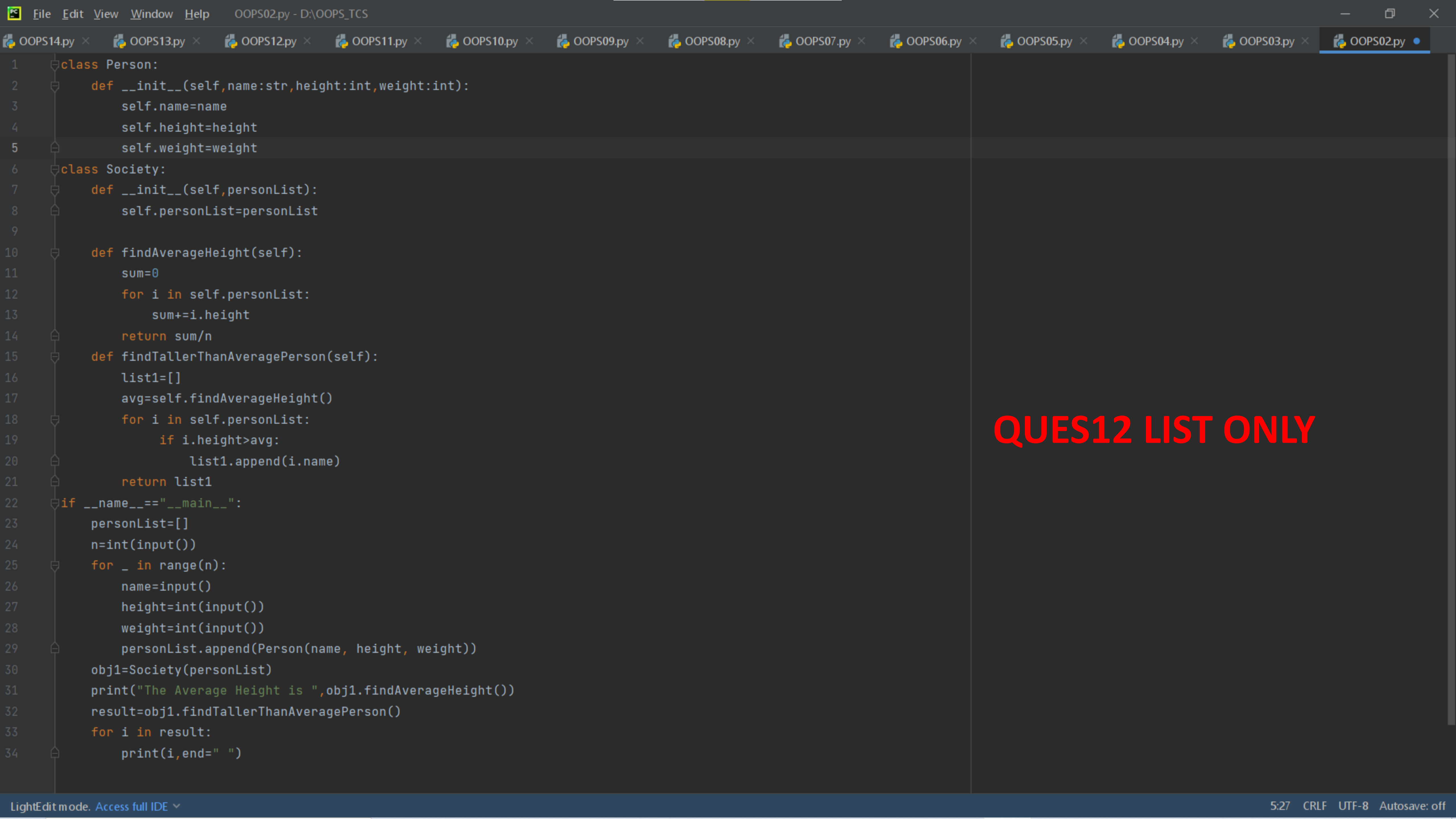
```python
class Team:
    def __init__(self,owner:str,value:int,ID:int,name:str):
        self.owner=owner
        self.value=value
        self.ID=ID
        self.name=name
class League:
    def __init__(self,leagueName,teamList):
        self.leagueName=leagueName
        self.teamList=teamList
    def findMinimumTeamByID(self):
        if len(self.teamList)==0:
            return None
        returnedList=sorted(self.teamList, key=lambda x:x.ID)
        return returnedList[0]
    def sortTeamByID(self):
        if len(self.teamList)==0:
            return None
        returnedList=sorted(self.teamList,key=lambda x:x.ID)
        return returnedList
```

QUES11 LIST SORTING BY ITEMS AND RETURN LIST


File Edit View Window Help OOPS04.py - D:\OOPS_TCS
main.py OOPS01.py OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py OOPS06.py OOPS05.py OOPS04.py
LightEdit mode. Access full IDE
36:1 CRLF UTF-8 Autosave: off

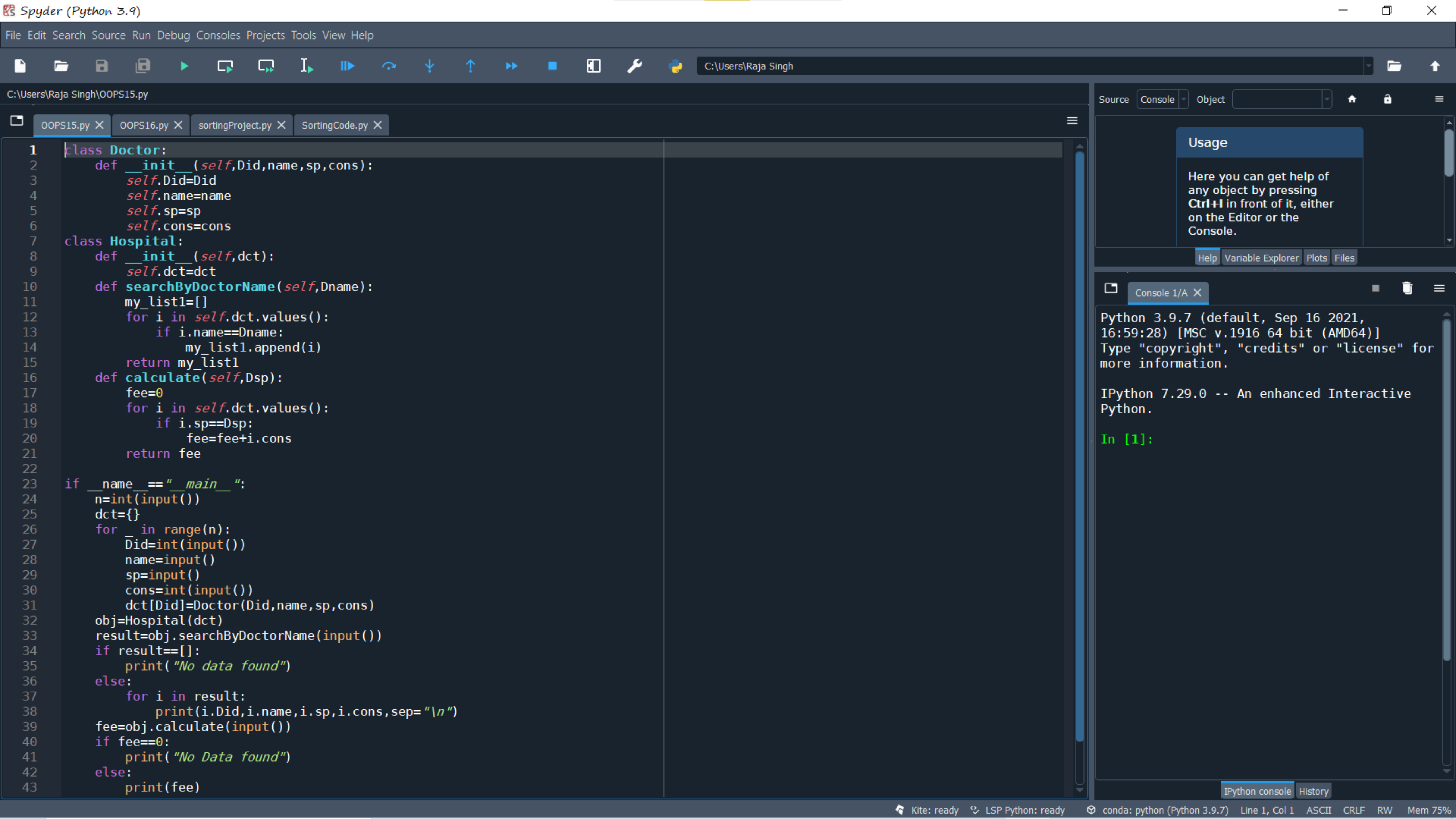
QUES11 LIST SORTING BY ITEMS AND RETURN LIST

```python

if __name__=="__main__":
    n=int(input())
    teamList=[]
    for _ in range(n):
        owner=input()
        value=int(input())
        ID=int(input())
        name=input()
        teamList.append(Team(owner,value,ID,name))
    obj1=League("XYZ",teamList)
    res1=obj1.findMinimumTeamByID()
    print(res1.owner,res1.value,res1.ID,res1.name,sep="\n")
    res2=obj1.sortTeamByID()
    for i in res2:
        print(i.ID)
```


File Edit View Window Help OOPS02.py - D:\OOPS_TCS
OOPS14.py OOPS13.py OOPS12.py OOPS11.py OOPS10.py OOPS09.py OOPS08.py OOPS07.py OOPS06.py OOPS05.py OOPS04.py OOPS03.py OOPS02.py
LightEdit mode. Access full IDE
5:27 CRLF UTF-8 Autosave: off

```python
class Person:
    def __init__(self,name:str,height:int,weight:int):
        self.name=name
        self.height=height
        self.weight=weight
class Society:
    def __init__(self,personList):
        self.personList=personList

    def findAverageHeight(self):
        sum=0
        for i in self.personList:
            sum+=i.height
        return sum/n
    def findTallerThanAveragePerson(self):
        list1=[]
        avg=self.findAverageHeight()
        for i in self.personList:
             if i.height>avg:
                 list1.append(i.name)
        return list1
if __name__=="__main__":
    personList=[]
    n=int(input())
    for _ in range(n):
        name=input()
        height=int(input())
        weight=int(input())
        personList.append(Person(name, height, weight))
    obj1=Society(personList)
    print("The Average Height is ",obj1.findAverageHeight())
    result=obj1.findTallerThanAveragePerson()
    for i in result:
        print(i,end=" ")
```

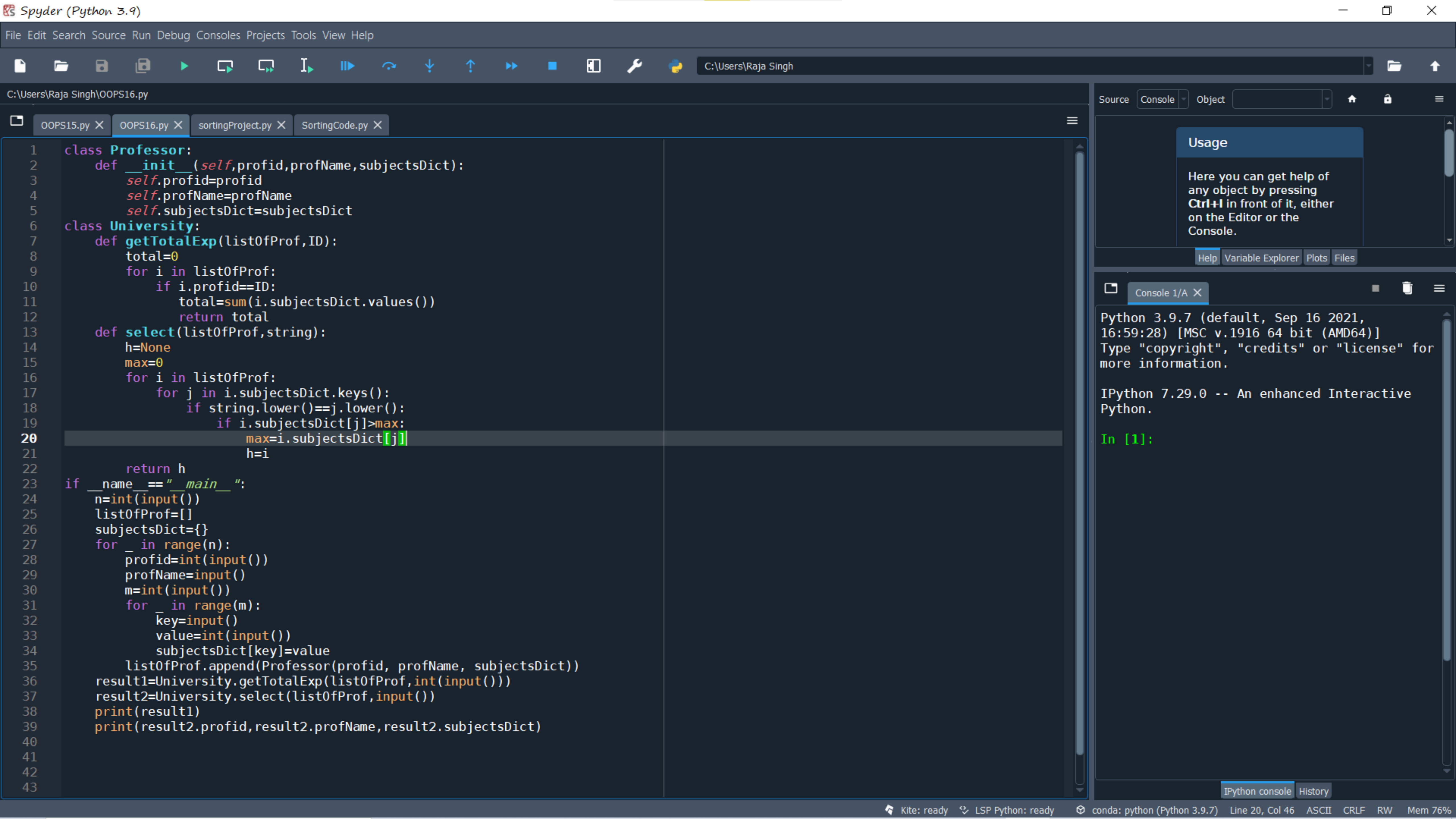
QUES12 LIST ONLY


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Raja Singh
C:\Users\Raja Singh\OOPS15.py
OOPS15.py
OOPS16.py
sortingProject.py
SortingCode.py
Source Console Object
Usage
Help Variable Explorer Plots Files
Console 1/A
IPython console History
Kite: ready LSP Python: ready conda: python (Python 3.9.7) Line 1, Col 1 ASCII CRLF RW Mem 75%

```python
class Doctor:
    def __init__(self,Did,name,sp,cons):
        self.Did=Did
        self.name=name
        self.sp=sp
        self.cons=cons
class Hospital:
    def __init__(self,dct):
        self.dct=dct
    def searchByDoctorName(self,Dname):
        my_list1=[]
        for i in self.dct.values():
            if i.name==Dname:
                my_list1.append(i)
        return my_list1
    def calculate(self,Dsp):
        fee=0
        for i in self.dct.values():
            if i.sp==Dsp:
                fee=fee+i.cons
        return fee

if __name__=="__main__":
    n=int(input())
    dct={}
    for _ in range(n):
        Did=int(input())
        name=input()
        sp=input()
        cons=int(input())
        dct[Did]=Doctor(Did,name,sp,cons)
    obj=Hospital(dct)
    result=obj.searchByDoctorName(input())
    if result==[]:
        print("No data found")
    else:
        for i in result:
            print(i.Did,i.name,i.sp,i.cons,sep="\n")
    fee=obj.calculate(input())
    if fee==0:
        print("No Data found")
    else:
        print(fee)
```

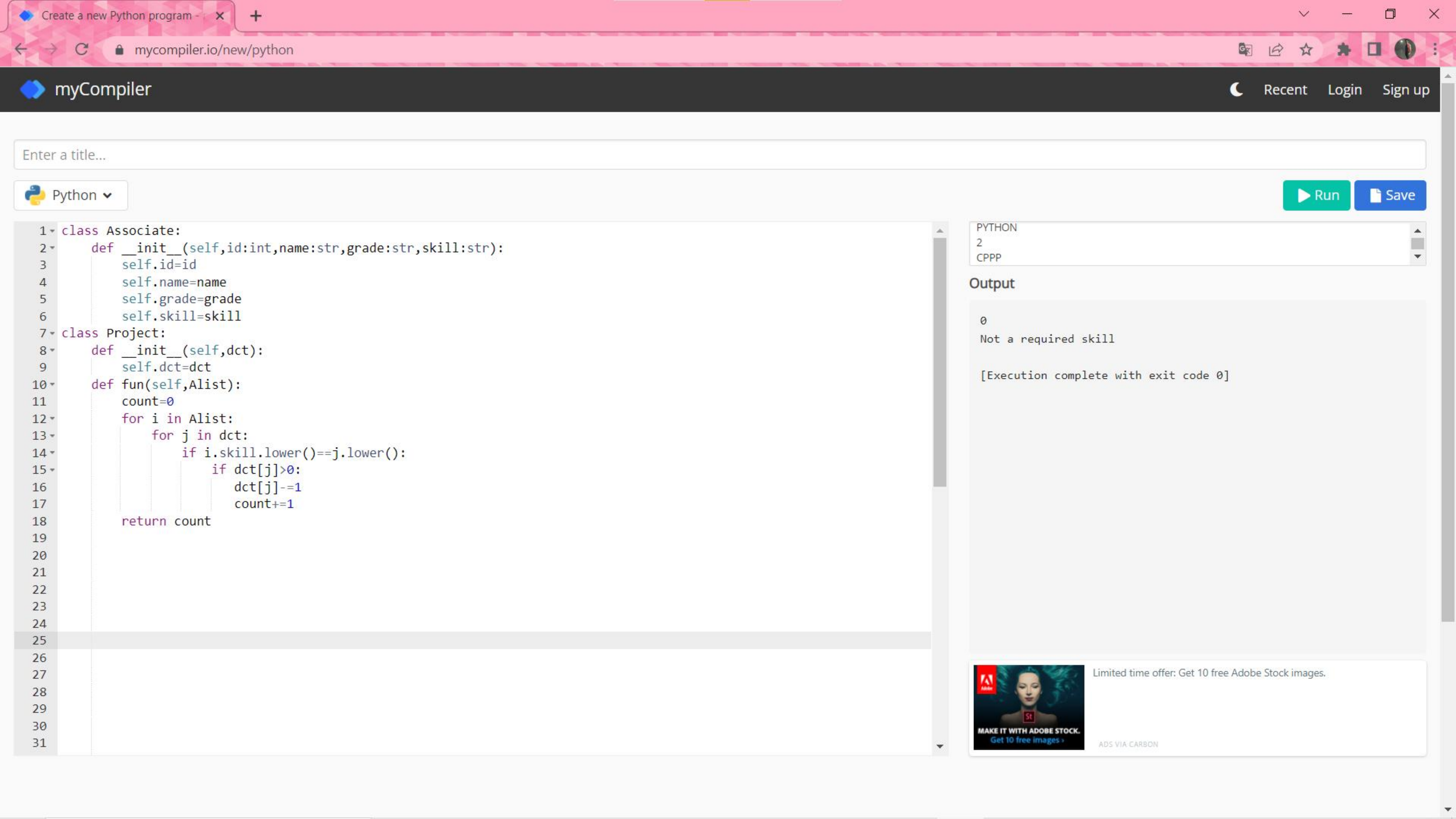
Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

```
Python 3.9.7 (default, Sep 16 2021,
16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.29.0 -- An enhanced Interactive
Python.

In [1]:
```


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Raja Singh
C:\Users\Raja Singh\OOPS16.py
OOPS15.py OOPS16.py sortingProject.py SortingCode.py
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help Variable Explorer Plots Files
Console 1/A
IPython console History
Kite: ready LSP Python: ready conda: python (Python 3.9.7) Line 20, Col 46 ASCII CRLF RW Mem 76%

```python
class Professor:
    def __init__(self,profid,profName,subjectsDict):
        self.profid=profid
        self.profName=profName
        self.subjectsDict=subjectsDict
class University:
    def getTotalExp(listOfProf,ID):
        total=0
        for i in listOfProf:
            if i.profid==ID:
               total=sum(i.subjectsDict.values())
               return total
    def select(listOfProf,string):
        h=None
        max=0
        for i in listOfProf:
            for j in i.subjectsDict.keys():
                if string.lower()==j.lower():
                    if i.subjectsDict[j]>max:
                       max=i.subjectsDict[j]
                       h=i
        return h
if __name__=="__main__":
    n=int(input())
    listOfProf=[]
    subjectsDict={}
    for _ in range(n):
        profid=int(input())
        profName=input()
        m=int(input())
        for _ in range(m):
            key=input()
            value=int(input())
            subjectsDict[key]=value
        listOfProf.append(Professor(profid, profName, subjectsDict))
    result1=University.getTotalExp(listOfProf,int(input()))
    result2=University.select(listOfProf,input())
    print(result1)
    print(result2.profid,result2.profName,result2.subjectsDict)
```

```
Python 3.9.7 (default, Sep 16 2021,
16:59:28) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.29.0 -- An enhanced Interactive
Python.

In [1]:
```

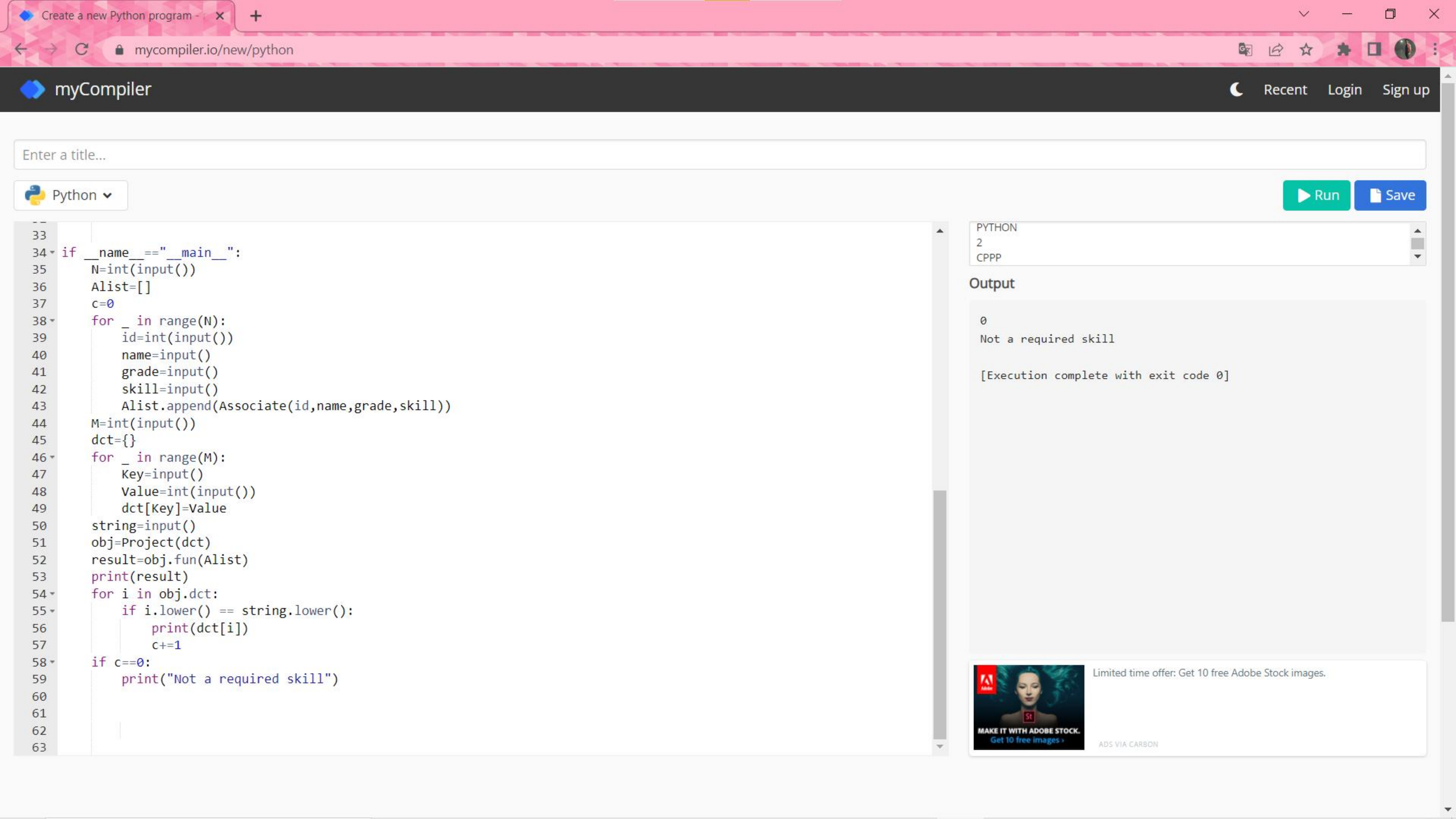


```python
class Associate:
    def __init__(self,id:int,name:str,grade:str,skill:str):
        self.id=id
        self.name=name
        self.grade=grade
        self.skill=skill
class Project:
    def __init__(self,dct):
        self.dct=dct
    def fun(self,Alist):
        count=0
        for i in Alist:
            for j in dct:
                if i.skill.lower()==j.lower():
                    if dct[j]>0:
                       dct[j]-=1
                       count+=1
        return count
```

```
PYTHON
2
CPPP
```

Output

```
0
Not a required skill

[Execution complete with exit code 0]
```



```python

if __name__=="__main__":
    N=int(input())
    Alist=[]
    c=0
    for _ in range(N):
        id=int(input())
        name=input()
        grade=input()
        skill=input()
        Alist.append(Associate(id,name,grade,skill))
    M=int(input())
    dct={}
    for _ in range(M):
        Key=input()
        Value=int(input())
        dct[Key]=Value
    string=input()
    obj=Project(dct)
    result=obj.fun(Alist)
    print(result)
    for i in obj.dct:
        if i.lower() == string.lower():
            print(dct[i])
            c+=1
    if c==0:
        print("Not a required skill")
```

```
PYTHON
2
CPPP
```

```
0
Not a required skill

[Execution complete with exit code 0]
```