CODING

```
#Task 1
class PlayerEarning():
  def __init__(self,name):
    self.name = name
    self.earning = 0


  def calculateTotal(self,earning,goals = 0):
    self.earning = earning
    if goals > 30:
      self.bonus = (5/100) * earning + 10000
    else:
      self.bonus = (5/100) * earning


  def printDetails(self):
    print(f'''Player Name: {self.name}
Player Season Earning without bonus: {self.earning}
Bonus: {int(self.bonus)}
Player Season Earning After Bonus: {int(self.earning + self.bonus)}''')
```

```
print("**********************")
player1 = PlayerEarning('Buffon')
player1.calculateTotal(250000)
player1.printDetails()

print("\n**********************")
player2 = PlayerEarning('Dybala')
player2.calculateTotal(250000, 31)
player2.printDetails()

print("\n**********************")
player3 = PlayerEarning('Cuadrado')
player3.calculateTotal(250000, 20)
player3.printDetails()
```

```
**********************
Player Name: Buffon
Player Season Earning without bonus: 250000
Bonus: 12500
Player Season Earning After Bonus: 262500

**********************
```

```
Player Name: Dybala
Player Season Earning without bonus: 250000
Bonus: 22500
Player Season Earning After Bonus: 272500

**********
Player Name: Cuadrado
Player Season Earning without bonus: 250000
Bonus: 12500
Player Season Earning After Bonus: 262500
```

```
#Task 2

class myList():
  def __init__(self,*n):
    self.l1 = list(n)
    self.summ = 0
    self.averagee = 0
  def merge(self,*n):
    self.l1 += list(n)
    print(self.l1)
  def sum(self):
    self.summ = 0
    if len(self.l1) == 0:
      self.summ = 0
    else:
      for n in self.l1:
        self.summ += n
    print(f'Sum: {self.summ}')
  def average(self):
    if len(self.l1) == 0:
      self.averagee = 0
    else:
      self.averagee = self.summ/len(self.l1)
    print(f'Average: {self.averagee}')

l1 =  myList(2,3,4,5,6) #you might need a list inside your class to store the values
l1.sum()
l1.merge(4,5,9)
l1.sum()
l1.average()
print('----------------------------')
l2 =  myList()
l2.average()
l2.merge(1,2,4,8)
l2.sum()
```

```
Sum: 20
Sum: 38
Average: 4.75
----------------------------
```

```
Average: 0
Sum: 15
```

```
#Task 3

class Bird():
  def __init__(self,name,can_fly = False):
    self.name = name
    self.can_fly = can_fly
    self.typ = 'Flightless Birds'
  def fly(self):
    if self.can_fly == False:
      print(f'{self.name} can not fly')
    else:
      print(f'{self.name} can fly')
  def setType(self,typ):
    self.typ = typ
  def printDetail(self):
    print(f'''Name: {self.name}
Type: {self.typ}''')


ostrich = Bird('Ostrich')
duck = Bird("Duck", True)
owl = Bird('Owl', True)
print('############################')
ostrich.fly()
duck.fly()
owl.fly()
duck.setType('Water Birds')
owl.setType('Birds of Prey')
print('========================')
ostrich.printDetail()
print('========================')
duck.printDetail()
print('========================')
owl.printDetail()
```

```
############################
Ostrich can not fly
Duck can fly
Owl can fly
========================
Name: Ostrich
Type: Flightless Birds
========================
Name: Duck
Type: Water Birds
========================
Name: Owl
Type: Birds of Prey
```

```
#Task 4
class Account():
  count = 0
  def __init__(self,name,age,occupation,balance):
    self.name = name
    self.age = age
    self.occupation = occupation
    self.balance = balance
    Account.count += 1

  def addMoney(self,money):
    self.balance += money

  def withdrawMoney(self,money):
    if money <= self.balance:
      self.balance -= money

  def printDetails(self):
    print(f'''Name: {self.name}
Age: {self.age}
Occupation:  {self.occupation}
Total Amount:  {self.balance}''')
```

```
print('No of account holders:', Account.count)
print("========================")
p1 = Account("Abdul", 45, "Service Holder", 500000)
p1.addMoney(300000)
p1.printDetails()
print("========================")
p2 = Account("Rahim", 55, "Businessman", 700000)
p2.withdrawMoney(700000)
p2.printDetails()
print("========================")
p3 = Account("Ashraf", 62, "Govt. Officer", 200000)
p3.withdrawMoney(250000)
p3.printDetails()
print("========================")
print('No of account holders:', Account.count)
```

```
No of account holders: 0
========================
Name: Abdul
Age: 45
Occupation:  Service Holder
```

```
Total Amount:  800000
========================
Name: Rahim
Age: 55
Occupation:  Businessman
Total Amount:  0
========================
Name: Ashraf
Age: 62
Occupation:  Govt. Officer
Total Amount:  200000
========================
No of account holders: 3
```

```
#Task 5
class Smartphone():
  def __init__(self,name = None):
    self.name = name
    self.features = {}
  def setName(self,name):
    self.name = name
  def addFeature(self,f1,f2):
    if self.name == None:
      print('Feature can not be added without phone name')
    else:
      if f1 in self.features:
        self.features[f1].append(f2)
      else:
        self.features[f1] = [f2]
  def printDetail(self):
    print(f'Phone Name: {self.name}')
    for k in self.features:
      x = f'{k}: '
      for i in self.features[k]:
        x += i +', '
      x = x[0:-2:1]
      print(x)


s1 = Smartphone()
print("================================")
s1.addFeature('Display', '6.1 inch')
print("================================")
s1.setName("Samsung Note 20")
s1.addFeature("Display", "6.1 inch")
s1.printDetail()
print("================================")
s2 = Smartphone("Iphone 12 Pro")
s2.addFeature('Display', '6.2 inch')
s2.addFeature("Ram", "6 GB")
print('================================')
s2.printDetail()
```

```
s2.addFeature('Display', 'Amoled panel')
s2.addFeature('Ram', 'DDR5')
print("================================")
s2.printDetail()
print("================================")
```

```
    ================================
    Feature can not be added without phone name
    ================================
    Phone Name: Samsung Note 20
    Display: 6.1 inch
    ================================
    ================================
    Phone Name: Iphone 12 Pro
    Display: 6.2 inch
    Ram: 6 GB
    ================================
    Phone Name: Iphone 12 Pro
    Display: 6.2 inch, Amoled panel
    Ram: 6 GB, DDR5
    ================================
```

```
#Task 6
class Student():
  total_students = 0
  cse_students = 0
  bba_students = 0
  def __init__(self,name,dep):
    self.name = name
    self.dep = dep
    if dep == 'CSE':
      Student.cse_students += 1
    else:
      Student.bba_students += 1
    Student.total_students += 1
    print(f'Creating Student Number: {Student.total_students}')

  def individualInfo(self):
    if self.dep == 'CSE':
      print(f'''{self.name} is from {self.dep} department.
Serial of Naruto among all students' is: {Student.total_students}
Serial of Naruto in CSE department is: {Student.cse_students}''')
    else:
        print(f'''{self.name} is from {self.dep} department.
Serial of Naruto among all students' is: {Student.total_students}
Serial of Naruto in CSE department is: {Student.bba_students}''')

  def totalInfo(self):
    print(f'''Total Number of Student: {self.total_students}
Total Number of CSE Student: {self.cse_students}
Total Number of BBA Student: {self.bba_students}''')
```

```
s1 = Student("Naruto", "CSE")
print('----------------------')
s1.individualInfo()
print('############################')
s1.totalInfo()
print('============================')

s2 = Student("Sakura", "BBA")
print('----------------------')
s2.individualInfo()
print('############################')
s2.totalInfo()
print('============================')

s3 = Student("Shikamaru", "CSE")
print('----------------------')
s3.individualInfo()
print('############################')
s3.totalInfo()
print('============================')

s4 = Student("Deidara", "BBA")
print('----------------------')
s4.individualInfo()
print('############################')
s4.totalInfo()
```

```
Creating Student Number: 1
----------------------
Naruto is from CSE department.
Serial of Naruto among all students' is: 1
Serial of Naruto in CSE department is: 1
############################
Total Number of Student: 1
Total Number of CSE Student: 1
Total Number of BBA Student: 0
============================
Creating Student Number: 2
----------------------
Sakura is from BBA department.
Serial of Naruto among all students' is: 2
Serial of Naruto in CSE department is: 1
############################
Total Number of Student: 2
Total Number of CSE Student: 1
Total Number of BBA Student: 1
============================
Creating Student Number: 3
----------------------
Shikamaru is from CSE department.
```

```
Serial of Naruto among all students' is: 3
Serial of Naruto in CSE department is: 2
##############################
Total Number of Student: 3
Total Number of CSE Student: 2
Total Number of BBA Student: 1
============================
Creating Student Number: 4
----------------------
Deidara is from BBA department.
Serial of Naruto among all students' is: 4
Serial of Naruto in CSE department is: 2
##############################
Total Number of Student: 4
Total Number of CSE Student: 2
Total Number of BBA Student: 2
```

```
#Task 7
class book:
    def __init__(self, name):
        self.name = name
        self.genre='biography'
    def review(self):
         print('This book is just out of the world,mind-blowing!')

# Write your code here

class fiction(book):
  def __init__(self,name,genre = 'biography'):
    super().__init__(name)
    self.genre = genre
  def review(self):
    print(f'{self.name} which is a {self.genre} is just out of the world, mind-blowing!')

class nonfiction(book):
  def __init__(self,name,genre = 'biography'):
    super().__init__(name)
    self.genre = genre
  def review(self):
    print(f'{self.name} which is a {self.genre} is just out of the world, mind-blowing!')
    return


b1 = fiction('The Shining','Psychological horror')
b2 = nonfiction('A Beautiful Mind')
b1.review()
print('========================')
b2.review()
print('========================')
```

```
The Shining which is a Psychological horror is just out of the world, mind-blowing!
```

```
========================
A Beautiful Mind which is a biography is just out of the world, mind-blowing!
========================
```

```
#Task 8
class Processor:
    def __init__(self, model, thread, core):
        self.model = model
        self.core = core
        self.thread = thread
    def getInfo(self):
        return 'Model:'+self.model+ '\nCores:'+str(self.core)+ '\nThreads:'+ str(self.thread)
# Write your code here

class Intel(Processor):
  def __init__(self,model,thread,core,price):
    super().__init__(model,thread,core)
    self.price = price
  def getInfo(self):
    print(super().getInfo())
    print(f'Price: {self.price}')

class AMD(Processor):
  def __init__(self,model,thread,core,price):
    super().__init__(model,thread,core)
    self.price = price
  def getInfo(self):
    print(super().getInfo())
    print(f'Price: {self.price}')

p1 = Intel("Intel i5 10th Gen",6,12,17000)
p2 = AMD("Ryzen 5 3500X",6,6,13800)
p3 = AMD("Ryzen 5 3600",6,12,16900)
print('======================')
p1.getInfo()
print('======================')
p2.getInfo()
print('======================')
p3.getInfo()
```

```
======================
Model:Intel i5 10th Gen
Cores:12
Threads:6
Price: 17000
======================
Model:Ryzen 5 3500X
Cores:6
Threads:6
Price: 13800
======================
```

```
        Model:Ryzen 5 3600
        Cores:12
        Threads:6
        Price: 16900


    #Task 9
    class Fruit:
      Total_order=0

      def __init__(self, Order_ID, weight):
        self.Order_ID=Order_ID
        self.weight=weight
        Fruit.Total_order=Fruit.Total_order+1

      def __str__(self):
        return self.Order_ID+", Weight: "+str(self.weight)

    class Mango(Fruit):
        #write your code here

      def __init__(self,Order_ID, weight, variety, price):
        super().__init__(Order_ID, weight)
        self.variety = variety
        self.price = weight * price

      def __add__(self,other):
        return f'The total of the orders are {self.price + other.price}'

      def __str__(self):
        return super().__str__() + ', Variety: ' + self.variety + ', Total Price: ' + str(self.pr

    class JackFruit(Fruit):
      #write your code here

      def __init__(self,Order_ID, weight, price):
        super().__init__(Order_ID, weight)
        self.price = weight * price

      def __add__(self,other):
        return f'The total of the orders are {self.price + other.price}'

      def __str__(self):
        return super().__str__() +', Total Price: ' + str(self.price)




    m1=Mango("Order Id 1", 5,"GopalVog",250)
    print(m1)
    m2=Mango("Order Id 2", 5,"HariVanga", 230)
    print(m2)
```

```
j1=JackFruit("Order Id 3", 5,250)
print(j1)
j2=JackFruit("Order Id 4", 4,210)
print(j2)
print("Total number of Orders: "+str(Fruit.Total_order))
print("==================")
print(m1+m2)
print("==================")
print(j1+j2)
```

```
 Order Id 1, Weight: 5, Variety: GopalVog, Total Price: 1250
 Order Id 2, Weight: 5, Variety: HariVanga, Total Price: 1150
 Order Id 3, Weight: 5, Total Price: 1250
 Order Id 4, Weight: 4, Total Price: 840
 Total number of Orders: 4
 ==================
 The total of the orders are 2400
 ==================
 The total of the orders are 2090
```

```
#Task 10
class Student:
    def __init__(self,name,ID):
        self.name = name
        self.ID = ID
    def Details(self):
        return "Name: "+self.name+"\n"+"ID: "+self.ID+"\n"
#Write your code here

class CSEStudent(Student):
  def __init__(self,name,ID,sem):
    super().__init__(name,ID)
    self.sem = sem
    self.courses = {}
    self.no_of_courses = 0
    self.total_credits = 0
    self.sum_gpa = 0
  def Details(self):
    return super().Details() + 'Current semester: ' + self.sem
  def addCourseWithMarks(self,*n):
    for i in range(0,len(n),2):
      self.courses[n[i]] = n[i+1]
      self.no_of_courses += 1
      self.total_credits += 3
  def showGPA(self):
    print(f'{self.name} has taken {self.no_of_courses} courses')
    for c in self.courses:
      if self.courses[c] >= 85:
        self.sum_gpa += 4.0
        x = 4.0
```

```
        elif self.courses[c] >= 80:
          self.sum_gpa += 3.3
          x = 3.3
        elif self.courses[c] >= 70:
          self.sum_gpa += 3.0
          x = 3.0
        elif self.courses[c] >= 65:
          self.sum_gpa += 2.3
          x = 2.3
        elif self.courses[c] >= 57:
          self.sum_gpa += 2.0
          x = 2.0
        elif self.courses[c] >= 55:
          self.sum_gpa += 1.3
          x = 1.3
        elif self.courses[c] >= 50:
          self.sum_gpa += 1.0
          x = 1.0
        else:
          self.sum_gpa += 0.0
          x = 0.0
        print(f'{c}: {x}')
      print(f'GPA of {self.name} is {(self.sum_gpa * 3)/self.total_credits}')


Bob = CSEStudent("Bob","20301018",'Fall 2020')
Carol = CSEStudent("Carol","16301814",'Fall 2020')
Anny = CSEStudent("Anny","18201234",'Fall 2020')
print("##########################")
print(Bob.Details())
print("##########################")
print(Carol.Details())
print("##########################")
print(Anny.Details())
print("##########################")
Bob.addCourseWithMarks("CSE111",83.5,"CSE230",73.0,"CSE260",92.5)
Carol.addCourseWithMarks("CSE470",62.5,"CSE422",69.0,"CSE460",76.5,"CSE461",87.0)
Anny.addCourseWithMarks("CSE340",45.5,"CSE321",95.0,"CSE370",91.0)
print("---------------------------")
Bob.showGPA()
print("---------------------------")
Carol.showGPA()
print("---------------------------")
Anny.showGPA()


      ##########################
      Name: Bob
      ID: 20301018
      Current semester: Fall 2020
      ##########################
      Name: Carol
```

```
ID: 16301814
Current semester: Fall 2020
#########################
Name: Anny
ID: 18201234
Current semester: Fall 2020
#########################
---------------------------
Bob has taken 3 courses
CSE111: 3.3
CSE230: 3.0
CSE260: 4.0
GPA of Bob is 3.4333333333333336
---------------------------
Carol has taken 4 courses
CSE470: 2.0
CSE422: 2.3
CSE460: 3.0
CSE461: 4.0
GPA of Carol is 2.8250000000000006
---------------------------
Anny has taken 3 courses
CSE340: 0.0
CSE321: 4.0
CSE370: 4.0
GPA of Anny is 2.6666666666666665
```

```
#Task 11
class Transport:
    total_traveller = 0

    def __init__(self, name, fare):
        self.name = name
        self.baseFare = fare

    def __str__(self):
        s =  'Name: '+self.name+", Base fare: "+str(self.baseFare)
        return s

# Write your codes here.

class Bus(Transport):
  def __init__(self,name,fare):
    super().__init__(name,fare)
    self.passengers = {}
    print(f'Base-fare of {self.name} is {self.baseFare} taka')
  def addPassengerWithBags(self,*n):
    for i in range(0,len(n),2):
      if n[i+1] > 5:
        x = self.baseFare + 105
      elif n[i+1] >= 3:
        x = self.baseFare + 60
```

```
        else:
          x = self.baseFare
        self.passengers[n[i]] = x
        Transport.total_traveller += 1
    def __str__(self):
      s = super().__str__()
      s += f'''\nTotal Passenger(s): {len(self.passengers)}
Passenger details:'''
      for p,f in self.passengers.items():
        s+= f'\nName: {p}, Fare: {f}'
      return s
class Train(Transport):
    def __init__(self,name,fare):
      super().__init__(name,fare)
      self.passengers = {}
      print(f'Base-fare of {self.name} is {self.baseFare} taka')
    def addPassengerWithBags(self,*n):
      for i in range(0,len(n),2):
        if n[i+1] > 5:
          x = self.baseFare + 105
        elif n[i+1] >= 3:
          x = self.baseFare + 60
        else:
          x = self.baseFare
        self.passengers[n[i]] = x
        Transport.total_traveller += 1
    def __str__(self):
      s = super().__str__()
      s += f'''\nTotal Passenger(s): {len(self.passengers)}
Passenger details:'''
      for p,f in self.passengers.items():
        s+= f'\nName: {p}, Fare: {f}'
      return s

# Do not change the following lines of code.
t1 = Bus('Volvo', 950)
print("================================")
t1.addPassengerWithBags('David', 6,  'Mike', 1, 'Carol', 3)
print("================================")
print(t1)
print("================================")
t2 = Train('Silk City', 850)
print("================================")
t2.addPassengerWithBags('Bob', 2, 'Simon', 4)
print("================================")
print(t2)
print("================================")
print('Total Passengers in Transport: ', Transport.total_traveller )
```

```
Base-fare of Volvo is 950 taka
```

```
      ================================
      ================================
      Name: Volvo, Base fare: 950
      Total Passenger(s): 3
      Passenger details:
      Name: David, Fare: 1055
      Name: Mike, Fare: 950
      Name: Carol, Fare: 1010
      ================================
      Base-fare of Silk City is 850 taka
      ================================
      ================================
      Name: Silk City, Base fare: 850
      Total Passenger(s): 2
      Passenger details:
      Name: Bob, Fare: 850
      Name: Simon, Fare: 910
      ================================
      Total Passengers in Transport:  5
```

```
#Task 12
class AppleProduct:
  def __init__(self, name, model, base_price):
    self.name = name
    self.model = model
    self.base_price = base_price
  def companyInfo(self):
    st = ("Company Name: Apple\nFouder: Steve Jobs, Steve Wozniak, Ronald Wayne\nCurrent CEO:
    return st
  def feature(self):
    st = (f"Name: {self.name}\nProduct Model: {self.model}\nHardware Quality: Excellent Hardw
    return st
  def __str__(self):
    print('This is apple product.')
  def calculatePrice(self):
    print('Total Price:', self.base_price)
# Write your codes here.

class MacBookPro2020(AppleProduct):
  def __init__(self,name,model,ram,chip,tax,base_price = 1299):
    super().__init__(name,model,base_price)
    self.ram = ram
    self.chip = chip
    self.tax = tax
    self.total_price = self.base_price + (self.base_price * self.tax/100)
  def __str__(self):
    return f'''Product Details:
{super().feature()}
Ram: {self.ram}GB
Chip: {self.chip}
Company Details:
{super().companyInfo()}'''
```

```
  def calculatePrice(self):
    print(f'''Calculating Total Price:
Base Price: {self.base_price}
Tax: {self.tax}%
Total Price: {self.total_price}''')
  def __add__(self,other):
    return self.total_price + other.total_price


class iPhone12(AppleProduct):
  def __init__(self,name,model,ram,chip,tax,base_price = 799):
    super().__init__(name,model,base_price)
    self.ram = ram
    self.chip = chip
    self.tax = tax
    self.total_price = self.base_price + (self.base_price * self.tax/100)
  def __str__(self):
    return f'''Product Details:
{super().feature()}
Ram: {self.ram}GB
Chip: {self.chip}
Company Details:
{super().companyInfo()}'''
  def calculatePrice(self):
    print(f'''Calculating Total Price:
Base Price: {self.base_price}
Tax: {self.tax}%
Total Price: {self.total_price}''')


# Do not change the following lines of code.

m1 = MacBookPro2020('MacBook', 'MacBookPro2020', 8, 'M1', 10)
print(m1)
print('====================================')
m1.calculatePrice()
print('####################################')
iphone = iPhone12('iPhone', 'iPhone 12', 8, 'A14', 5)
print(iphone)
print('====================================')
iphone.calculatePrice()
print('####################################')
print('Total Price of these two products: ',end='')
print('%.2f Dollars'%(m1 + iphone))
```

```
Product Details:
Name: MacBook
Product Model: MacBookPro2020
```

```
Hardware Quality: Excellent Hardwares
Guarantee/ Warranty: Apple Care
Ram: 8GB
Chip: M1
Company Details:
Company Name: Apple
Fouder: Steve Jobs, Steve Wozniak, Ronald Wayne
Current CEO: Tim Cook
Address: Apple Inc, 2511 Laguna Blvd, Elk Grove, CA 95758, United States
==================================
Calculating Total Price:
Base Price: 1299
Tax: 10%
Total Price: 1428.9
##################################
Product Details:
Name: iPhone
Product Model: iPhone 12
Hardware Quality: Excellent Hardwares
Guarantee/ Warranty: Apple Care
Ram: 8GB
Chip: A14
Company Details:
Company Name: Apple
Fouder: Steve Jobs, Steve Wozniak, Ronald Wayne
Current CEO: Tim Cook
Address: Apple Inc, 2511 Laguna Blvd, Elk Grove, CA 95758, United States
==================================
Calculating Total Price:
Base Price: 799
Tax: 5%
Total Price: 838.95
##################################
Total Price of these two products: 2267.85 Dollars
```

```
#Task 13
class University:
  name = 'ABC University'
  numberOfStudents = 0
  admissionFee = 28000
  Library = 2000
  def __init__(self,n,i):
    self.stName = n
    self.stId = i
  def payment(self):
    return self.admissionFee + self.Library
  def __str__(self):
    return "Student Name: {}, ID: {}\nFee: {}".format(self.stName, self.stId, self.payment())


# Write your codes here.
class CSE_dept(University):
  SemesterFee = 7700
```

```
    LabFee = 2750
    PerCreditFee = 6600
    def __init__(self,name,id,credits = 6):
      super().__init__(name,id)
      self.credits = credits
      University.numberOfStudents += 1
    def payment(self):
      return super().payment() + self.SemesterFee + self.LabFee + self.PerCreditFee * self.cred
    def payment_details(self):
      print(f'''DETAILS:
Admission Fee: {self.admissionFee}
Library Fee: {self.Library}
Semester Fee: {self.SemesterFee}
Per Credit Fee: {self.PerCreditFee}
Number of credits: {self.credits}
Lab Fee: {self.LabFee}''')
    def __add__(self,other):
      return self.payment() + other.payment()

class PHR_dept(University):
    SemesterFee = 11000
    PerCreditFee = 6600
    def __init__(self,name,id,credits = 9):
      super().__init__(name,id)
      self.credits = credits
      University.numberOfStudents += 1
    def payment(self):
      return super().payment() + self.SemesterFee + self.PerCreditFee * self.credits
    def payment_details(self):
      print(f'''DETAILS:
Admission Fee: {self.admissionFee}
Library Fee: {self.Library}
Semester Fee: {self.SemesterFee}
Per Credit Fee: {self.PerCreditFee}
Number of credits: {self.credits}''')
    def __add__(self,other):
      return self.payment() + other.payment()




# Do not change the following lines of code.

c1 = CSE_dept("Mary","5678")
print(c1)
c1.payment_details()
print("============================")
p1 = PHR_dept("Simon","91011")
print(p1)
p1.payment_details()
```

```
print("============================")
c2 = CSE_dept("Adam","1234", 12)
print(c2)
c2.payment_details()
print("============================")
p2 = PHR_dept("David","121314", 15)
print(p2)
p2.payment_details()
print("============================")
print("Total Number of Students:", University.numberOfStudents)
print("Total University Revenue:", (c1 + c2) + (p1 + p2))
print("============================")
print("Due to the pandemic, admission and library fees have been reduced for all departments.
University.admissionFee -= 1000
University.Library -= 100
print("The credit, semester and lab fees have been reduced for the CSE department. ")
CSE_dept.PerCreditFee -= 100
CSE_dept.SemesterFee -= 100
CSE_dept.LabFee -=100
print("The credit and semester fees have been reduced for the PHR department.\n ")
PHR_dept.PerCreditFee -= 100
PHR_dept.SemesterFee -= 1000
print(c1)
print(p1)
print(c2)
print(p2)
print("============================")
print("Total Number of Students:", University.numberOfStudents)
print("Total University Revenue:", (c1 + c2) + (p1 + p2))
```

```
Student Name: Mary, ID: 5678
Fee: 80050
DETAILS:
Admission Fee: 28000
Library Fee: 2000
Semester Fee: 7700
Per Credit Fee: 6600
Number of credits: 6
Lab Fee: 2750
============================
Student Name: Simon, ID: 91011
Fee: 100400
DETAILS:
Admission Fee: 28000
Library Fee: 2000
Semester Fee: 11000
Per Credit Fee: 6600
Number of credits: 9
============================
Student Name: Adam, ID: 1234
Fee: 119650
DETAILS:
```

```
Admission Fee: 28000
Library Fee: 2000
Semester Fee: 7700
Per Credit Fee: 6600
Number of credits: 12
Lab Fee: 2750
==========================
Student Name: David, ID: 121314
Fee: 140000
DETAILS:
Admission Fee: 28000
Library Fee: 2000
Semester Fee: 11000
Per Credit Fee: 6600
Number of credits: 15
==========================
Total Number of Students: 4
Total University Revenue: 440100
==========================
Due to the pandemic, admission and library fees have been reduced for all departments.
The credit, semester and lab fees have been reduced for the CSE department.
The credit and semester fees have been reduced for the PHR department.

Student Name: Mary, ID: 5678
Fee: 78150
Student Name: Simon, ID: 91011
Fee: 97400
Student Name: Adam, ID: 1234
Fee: 117150
Student Name: David, ID: 121314
Fee: 136400
==========================
Total Number of Students: 4
Total University Revenue: 429100
```

```
#Task 14
class Library:
    Total_book = 1000
    borrow_data = {}

    def __init__(self,n,id):
        self.student_name = n
        self.student_id = id

    def borrowbook(self):
        print("A book is borrowed!")

    def __str__(self):
        return "Library: XYZ"


class Student(Library):
  def __init__(self,name,id):
    super().__init__(name,id)
```

```
        self.books = []
      def borrowbook(self,name,id = None):
        if name in super().borrow_data:
          print(f'Sorry {self.student_name} ! {name} book is borrowed by {super().borrow_data[nam
        else:
          super().borrowbook()
          Library.Total_book -= 1
          if id == None:
            print(f''''{name}' book is borrowed by {self.student_name}({self.student_id})
    Number of books available for borrowing = {Library.Total_book}''')
          else:
            print(f'''{name}' book with the unique id {id} is borrowed by {self.student_name}({se
    Number of books available for borrowing = {Library.Total_book}''')
          Library.borrow_data[name] = [self.student_name]
          self.books.append(name)
      def returnAllBooks(self):
        for k in Library.borrow_data.copy():
          if k in self.books:
            del Library.borrow_data[k]
            Library.Total_book += 1
        print(f'All books are returned by {self.student_name}')

      def __str__(self):
        print(super().__str__())
        print(f'Student Name: {self.student_name} ID: {self.student_id}')
        x = 'Books Borrowed: '
        for b in self.books:
          x += b + ', '
        x = x[0:-2:1]
        return x



    #Write your code here
    s1 = Student("Alice",18101259)
    s1.borrowbook("The Alchemist", "Hdw652")
    print("===============")
    print(s1)
    print("===============")
    print(Library.borrow_data)
    print("===============")
    s1.borrowbook("Wuthering Heights")
    print("===============")
    print(s1)
    print("===============")
    s2= Student("David",18141777)
    s2.borrowbook("The Alchemist", "Hdw652")
    print("===============")
    s2.borrowbook("The Vampyre")
    print("===============")
    print(Library.borrow_data)
```

```
print("===============")
s1.returnAllBooks()
print("===============")
print(Library.borrow_data)
```

```
A book is borrowed!
The Alchemist' book with the unique id Hdw652 is borrowed by Alice(18101259)
Number of books available for borrowing = 999
===============
Library: XYZ
Student Name: Alice ID: 18101259
Books Borrowed: The Alchemist
===============
{'The Alchemist': ['Alice']}
===============
A book is borrowed!
'Wuthering Heights' book is borrowed by Alice(18101259)
Number of books available for borrowing = 998
===============
Library: XYZ
Student Name: Alice ID: 18101259
Books Borrowed: The Alchemist, Wuthering Heights
===============
Sorry David ! The Alchemist book is borrowed by Alice
===============
A book is borrowed!
'The Vampyre' book is borrowed by David(18141777)
Number of books available for borrowing = 997
===============
{'The Alchemist': ['Alice'], 'Wuthering Heights': ['Alice'], 'The Vampyre': ['David']}
===============
All books are returned by Alice
===============
{'The Vampyre': ['David']}
```

```
#Task 15
class Player:
  database={}
  playerNo = 0
  def __init__(self,name,team,jerseyNo):
    self.name = name
    self.team = team
    self.jerseyNo = jerseyNo
  def __str__(self):
    return "Name:{}\nTeam:{}\nJersey No:{}".format(self.name,self.team,self.jerseyNo)

#Write your code here

class FootballPlayer(Player):
  def __init__(self,name,club,jn,goal,ret = 'Not Yet Retired'):
    super().__init__(name,club,jn)
    self.goals = goal
    self.ret = ret
```

```
        Player.playerNo += 1
        s1 = str(super().playerNo)
        l1 = name.split(' ')
        for i in l1:
          s1 += i[0]
        s1 += str(jn)
        self.id = s1
        super().database[self.id] = [name,club,jn,goal,ret]

    def __str__(self):
      return f'''{super().__str__()}
Goals Scored: {self.goals}
Retirement Date: {self.ret}'''

    @classmethod
    def createPlayer(cls,name,club,jn,goals,ret):
      return FootballPlayer(name,club,jn,goals,ret)




print("Number of players:",Player.playerNo)
print("Player Database:",Player.database)
print("##################################")
p1 = FootballPlayer("Lionel Messi","Barcelona",10,231)
print("------Details of the player------")
print(p1)
print("##################################")
p2 = FootballPlayer("Cristiano Ronaldo","Juventus",7,215)
print("------Details of the player------")
print(p2)
print("##################################")
p3 = FootballPlayer.createPlayer("Miroslav Klose","Lazio",11, 71,"11 Aug,2014")
print("------Details of the player------")
print(p3)
print("##################################")
print("Number of players:",Player.playerNo)
print("Player Database:",Player.database)
```

```
Number of players: 0
Player Database: {}
##################################
------Details of the player------
Name:Lionel Messi
Team:Barcelona
Jersey No:10
Goals Scored: 231
Retirement Date: Not Yet Retired
##################################
------Details of the player------
Name:Cristiano Ronaldo
```

```
Team:Juventus
Jersey No:7
Goals Scored: 215
Retirement Date: Not Yet Retired
##################################
------Details of the player------
Name:Miroslav Klose
Team:Lazio
Jersey No:11
Goals Scored: 71
Retirement Date: 11 Aug,2014
##################################
Number of players: 3
Player Database: {'1LM10': ['Lionel Messi', 'Barcelona', 10, 231, 'Not Yet Retired'], '2
```

TRACING

```
#Task 16
class Quiz1:
    temp = 4

    def __init__(self, p = None):
        if p is None:
            self.y = self.temp - 1
            self.sum = self.temp + 1
            Quiz1.temp += 2
        else:
            self.y = self.temp + p
            self.sum = p + self.temp + 1
            Quiz1.temp -= 1

    def methodA(self):
        x, y = 0, 0
        y = y + self.y
        x = self.y + 2 + self.temp
        self.sum = x + y + self.methodB(x, y)
        print(x, y, self.sum)

    def methodB(self, m, n):
        x = 0
        Quiz1.temp += 1
        self.y = self.y + m + (self.temp)
        x = x + 2 + n
        self.sum = self.sum + x + self.y
        print(x, self.y, self.sum)
        return self.sum

q1 = Quiz1()
q1.methodA()
```

```
q1.methodA()
Quiz1.temp += 2
q2 = Quiz1(2)
q2.methodA()
q2.methodA()
```

```
5 21 31
11 3 45
23 59 127
30 21 178
14 45 72
23 12 107
47 113 267
57 45 369
```

```
#Task 17
class Scope:
   def __init__(self):
      self.x=1
      self.y=100
   def met1(self):
       x = 3
       x = self.x + 1
       self.y = self.y + self.x + 1
       x = self.y + self.met2(x+self.y) + self.y
       print(x)
       print(self.y)
   def met2(self,y=0):
       print(self.x)
       print(y)
       self.x = self.x + y
       self.y = self.y + 200
       return self.x + y


q2 = Scope()
q2.met1()
q2.met2()
q2.met1()
q2.met2()
```

```
1
104
613
302
105
0
105
714
2949
808
```

```
819
0
819
```

```
#Task 18
class msgClass:
    def __init__(self):
        self.content = 0
class Q5:
  def __init__(self):
    self.sum = 1
    self.x= 2
    self.y = 3
  def methodA(self):
    x, y = 1, 1
    msg = []
    myMsg = msgClass()
    myMsg.content = self.x
    msg.append(myMsg)
    msg[0].content = self.y + myMsg.content
    self.y = self.y + self.methodB(msg[0])
    y = self.methodB(msg[0]) + self.y
    x = y + self.methodB(msg[0], msg)
    self.sum = x + y + msg[0].content
    print(x," ", y," ", self.sum)
  def methodB(self, mg1, mg2 = None):
    if mg2 == None:
      x, y = 5, 6
      y = self.sum + mg1.content
      self.y = y + mg1.content
      x = self.x + 7 +mg1.content
      self.sum = self.sum + x + y
      self.x = mg1.content + x +8
      print(x, " ", y," ", self.sum)
      return y
    else:
      x = 1
      self.y += mg2[0].content
      mg2[0].content = self.y + mg1.content
      x += 4 + mg1.content
      self.sum += x + self.y
      mg1.content = self.sum - mg2[0].content
      print(self.x, " ",self.y," ", self.sum)
      return self.sum


q = Q5()
q.methodA()
```

```
14   6   21
```

```
39   26   86
52   36   168
225   57   409
```

```
#Task 19
class A:
  temp = -5
  def __init__(self):
    self.sum = 0
    self.y = 0
    self.y = self.temp - 3
    self.sum = A.temp + 2
    A.temp -= 2
  def methodA(self, m ,n):
    x = 1
    A.temp += 1
    self.y = self.y + m + self.temp
    x = x + 1 + n
    self.sum = self.sum + x + self.y
    print(f"{x} {self.y} {self.sum}")

class B(A):
  x = -10
  def __init__(self, b = None):
    super().__init__()
    self.y = 4
    self.temp = -5
    self.sum = 2
    if b == None:
      self.y = self.temp + 3
      self.sum = 3 + self.temp + 3
      self.temp -= 2
    else:
      self.sum = b.sum
      B.x = b.x
      b.methodB(1,3)
  def methodA(self, m, n):
    x = 1
    self.temp += 1
    self.y = self.y + m + self.temp
    x = x + 7 + n
    super().methodA(x, m)
    self.sum = self.sum + x + self.y
    print(f"{x} {self.y} {self.sum}")
  def methodB(self, m, n):
    y = 3
    y = y + self.y
    B.x = self.y + 3 + self.temp
    self.methodA(B.x, y)
    self.sum = self.x + y + self.sum
    print(f"{B.x} {y} {self.sum}")
```

```
a1 = A()
b1 = B()
b2 = B(b1)
b1.methodA(3,2)
b2.methodB(1,2)
```

```
     -4 -11 -14
     9 -11 -16
     -6 1 -21
     5 -8 -24
     10 -8 -22
     4 13 18
     15 13 46
     2 7 55
```

```
#Task 20
class msgClass:
  def __init__(self):
    self.content = 0
class Q5:
  def __init__(self):
    self.sum = 3
    self.y = 6
    self.x = 1
  def methodA(self):
    x = 1
    y = 1
    msg = [msgClass()]
    myMsg = msgClass()
    myMsg.content = self.x
    msg[0] = myMsg
    msg[0].content = self.y + myMsg.content
    self.y = self.y + self.methodB(msg[0])
    y = self.methodB(msg[0]) + self.y
    x = y + self.methodB(msg, msg[0])
    self.sum = x + y + msg[0].content
    print(f"{x} {y} {self.sum}")

  def methodB(self, *args):
    if len(args) == 1:
      x = 1
      y = 1
      y = self.sum + args[0].content
      self.y = y + args[0].content
      x = self.x + 3 + args[0].content
      self.sum = self.sum + x + y
      Q5.x = args[0].content + x + 2
      print(f"{x} {y} {self.sum}")
```

```
        return y
      else:
        x = 1
        self.y = self.y + args[0][0].content
        args[0][0].content = self.y + args[1].content
        x = x + 3 + args[1].content
        self.sum = self.sum + x + self.y
        args[1].content = self.sum - args[0][0].content
        print(f"{Q5.x} {self.y} {self.sum}")
        return self.sum

q =  Q5()
q.methodA()
```

```
11 10 24
11 31 66
20 45 167
236 69 420
```