****

**Course Title: Programming Language II**

**Course Code: CSE111**

**Lab No: 9**

**CSE111 Practice Sheet**

**Task - 1**

**Implement** the design of the **PlayerEarning** class so that the following output is produced:

| **Driver Code** | **Output** |
| --- | --- |
| ***# Write your code here***  **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: 0**  **Player Season Earning After Bonus: 250000**  **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***  **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** |

**Note:** calculateTotal() method takes either 1 or 2 arguments. It takes earning without bonus as the first argument and number of goals as the second argument. Calculate the bonus only if the number of goals is given (see the hint). If the number of goals is not provided, the bonus is 0. Finally calculate the total after bonus.

**Assume** only these 2 ways you can call the calculateTotal() method.

**Hint**:

If Goal > 30, bonus = (5/100) \* earning\_without\_bonus + 10000

else, bonus = (5/100) \* earning\_without\_bonus

**Task - 2**

Design a **myList** class so that the following output is produced upon executing the following code:

| **Driver Code** | **Output** |
| --- | --- |
| 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**

Implement the design of the **Bird** ​class so that the following output is produced:

| **Driver Code** | **Output** |
| --- | --- |
| 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**

Implement the design of the **Account** class so that the following output is produced:

| **Driver Code** | **Output** |
| --- | --- |
| ***# Write your code here***    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**

Write the **Smartphone** class with the required methods to give the following outputs as shown.

| ***# Write your codes here.***    ***# Do not change the following lines of code.***  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("=================================") | ***OUTPUT:***  =================================  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**

Design and implement the **Student** so that the following code gives the expected output

**You are not allowed to change the given code.**

**Hint:**

* **You need to use class/static variables**

| **# Write Your Code Here**  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() | **Output:**  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 Sakura among all students' is: 2  Serial of Sakura in BBA 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 Shikamaru among all students' is: 3  Serial of Shikamaru 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 Deidara among all students' is: 4  Serial of Deidara in BBA department is: 2  #############################  Total Number of Student: 4  Total Number of CSE Student: 2  Total Number of BBA Student: 2 |
| --- | --- |

**Task - 7**

Implement the design of the **fiction** and the **nonfiction** classes that inherit from **book** class so that the following code generates the output below:

| **Driver Code** | **Output** |
| --- | --- |
| 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***  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**

**Implement** the **Intel** and the **AMD** class that inherit from **Processor** class so that the following code generates the output below:

| **Driver Code** | **Output** |
| --- | --- |
| 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***    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: 6  Threads: 12  Price: 17000 taka  ======================  Model: Ryzen 5 3500X  Cores: 6  Threads: 6  Price: 13800 taka  ======================  Model: Ryzen 5 3600  Cores: 6  Threads: 12  Price: 16900 taka |

**Task - 9**

Write the **Mango** and **Jackfruit** class which are derived from the **fruit** class with the required methods to give the following outputs as shown.

**[Hint: total price=weight \* unit price]**

| ***# Do not change the following lines of code.***  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**  class JackFruit(Fruit):  **#write your code here**    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) | ***OUTPUT:***  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 Price of the orders are: 2400  ==================  The total Price of the orders are: 2090 |
| --- | --- |

**Task - 10**

Write the **CSEStudent** class with the required methods to give the following outputs as shown.**Hints:**

1. Each course has 3 credits.
2. GPA = sum( per course grade \* per course credit) / sum(credit attended in that semester)
3. **Grading policy:** mark>=85: 4.0 ; 80<=mark<=84: 3.3;70<=mark<=79:3.0 ;65<=mark<=69: 2.3; 57<=mark<=64:2.0 ; 55<=mark<=56:1.3; 50<=mark<=54:1.0; >50:0.0

| **Driver Code** | **Output** |
| --- | --- |
| 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  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: 3.43  ----------------------------  Carol has taken 4 courses.  CSE470: 2.0  CSE422: 2.3  CSE460: 3.0  CSE461: 4.0  GPA of Carol is: 2.83  ----------------------------  Anny has taken 3 courses.  CSE340: 0.0  CSE321: 4.0  CSE370: 4.0  GPA of Anny is: 2.67 |

**Task - 11**

Design **Bus** class and **Train** class which inherit **Transport** class so that the following code provides the expected output.

**Note:** A passenger can carry upto 2 bags for free. 60 taka will be added if the number of bags is between 3 and 5. 105 taka will be added if the number of bags is greater than 5.

| 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.***  ***# 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 ) | ***OUTPUT:***  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**

Write **MacBookPro2020** class and **iPhone12** class which inherit **AppleProduct** class so that the following code provides the expected output. You need to overwrite necessary methods along with operator overloading.

**Hint:**

* Base price for MacBookPro2020 is 1299
* Base price of iPhone12 is 799
* Total tax = (base price \* rate of tax) / 100
* Total price = base price + total tax

| 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: Tim Cook\nAddress: Apple Inc, 2511 Laguna Blvd, Elk Grove, CA 95758, United States")  return st  def feature(self):  st = (f"Name: {self.name}\nProduct Model: {self.model}\nHardware Quality: Excellent Hardwares\nGuarantee/ Warranty: Apple Care")  return st  def \_\_str\_\_(self):  print('This is apple product.')  def calculatePrice(self):  print('Total Price:', self.base\_price)  ***# Write your codes here.***  ***# 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)) | ***OUTPUT:***  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**

Write the **CSE\_dept and PHR\_dept** class with the required methods to give the following outputs as shown.

| 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.***  ***# 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)) | ***OUTPUT:***    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**

Implement the “Student” class that is derived from the “Library” class.

| 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"  ***#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**

Implement the “FootballPlayer” class that is derived from the “Player” class.

**[Assume that every player's name will consist of 2 words(First name, Last name).]**

| 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**  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) | **Output**  Number of players: 0  Player Database: {}  #################################  ------Details of the player------  Player ID: 1LM10  Name:Lionel Messi  Team:Barcelona  Jersey No:10  Goals Scored:231  Retirement date:Not yet retired  #################################  ------Details of the player------  Player ID: 2CR7  Name:Cristiano Ronaldo  Team:Juventus  Jersey No:7  Goals Scored:215  Retirement date:Not yet retired  #################################  ------Details of the player------  Player ID: 3MK11  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'], '2CR7': ['Cristiano Ronaldo', 'Juventus', 7, 215, 'Not yet retired'], '3MK11': ['Miroslav Klose', 'Lazio', 11, 71, '11 Aug,2014']} |
| --- | --- |

**Task - 16**

**Implement** the “**Vector3D**” class derived from the “**Vector2D**” class so that the following output is generated.

* Length of a 3D vector =
* Unit Vector = /length, where is the vector
* Dot product of v1 and v2 = x1x2 + y1y2 + z1z2
* Two vectors are orthogonal when their dot product is 0

| **class Vector2D:**  **def \_\_init\_\_(self, Xcomponent, Ycomponent, vec\_type = 'Default'):**  **self.Xcomponent = Xcomponent**  **self.Ycomponent = Ycomponent**  **self.vec\_type = vec\_type**    **def \_\_str\_\_(self):**  **return str(self.Xcomponent)+"i + "+str(self.Ycomponent)+ "j"**  **#Write your code here**  **force1 = Vector3D(1, 3, 5, 'force')**  **print('---------------------------------------')**  **print(force1)**  **print('---------------------------------------')**  **print('length of force1 vector is: ', force1.calculate\_length())**  **force1\_unit\_vector = force1.calculate\_unit\_vector()**  **print('unit vector of force1 vector is: ', force1\_unit\_vector)**  **print('---------------------------------------')**  **displacement1 = Vector3D(5, -5, 2, 'displacement')**  **print(displacement1)**  **print('----------------------------------------')**  **summ = force1 + displacement1**  **print('result of addition:', summ)**  **print('----------------------------------------')**  **force2 = Vector3D(7, 3, -2, 'force')**  **summ = force1 + force2**  **print('result of addition:', summ)**  **print('----------------------------------------')**  **diff = force1 - force2**  **print('result of subtraction:', diff)**  **print('----------------------------------------')**  **work = force1 \* displacement1**  **print('work done by force1 and displacement1 is:', work)**  **print('-----------------------------------------------')**  **print('force1 and displacement1 are orthogonal:', force1.is\_ortho(displacement1))**  **print('force2 and displacement1 are orthogonal:', force2.is\_ortho(displacement1))** | **Output**  **--------------------------------**  **1i + 3j + 5k**  **--------------------------------**  **length of force1 vector is: 5.916079783099616**  **unit vector of force1 vector is: 0.1690308509457033i + 0.50709255283711j + 0.8451542547285166k**  **--------------------------------**  **5i + -5j + 2k**  **--------------------------------**  **result of addition: Different type of vectors cannot be added**  **--------------------------------**  **result of addition: 8i + 6j + 3k**  **--------------------------------**  **result of subtraction: -6i + 0j + 7k**  **--------------------------------**  **work done by force1 and displacement1 is: 0**  **--------------------------------**  **force1 and displacement1 are orthogonal: True**  **force2 and displacement1 are orthogonal: False** |
| --- | --- |

**Task - 17**

**Implement** the “Quidditch\_Player” class that is derived from the “Magical\_SportsPerson” class.

[Assume that every player's name will consist of 2 words (First name, Last name).]

| class Magical\_SportsPerson:  database = {}  playerNo = 0  def \_\_init\_\_(self,name,team,jerseyNo):  self.name = name  self.team = team  self.jerseyNo = jerseyNo    def \_\_str\_\_(self):  return "Name:{}\nTeam:{}\nJerseyNo:{}".format(self.name, self.team, self.jerseyNo)  #Write your code here  print("Number of players:",Magical\_SportsPerson.playerNo)  print("Player Database:",Magical\_SportsPerson.database)  print("#################################")  p1 = Quidditch\_Player("Harry Potter","Gryffindor", 8, 523)  print("------Details of the player------")  print(p1)  print("#################################")  p2 = Quidditch\_Player("Ronald Weasley","Gryffindor", 13, 5)  print("------Details of the player------")  print(p2)  print("#################################")  p3 = Quidditch\_Player.createPlayer("George Weasley","Gryffindor", 12, 11, "11 Magical Year, 1886")  print("------Details of the player------")  print(p3)  print("#################################")  print("Number of players:",Magical\_SportsPerson.playerNo)  print("Player Database:",Magical\_SportsPerson.database)  print("Player Database:",Player.database) | **Output:**  Number of players: 0  Player Database: {}  #################################  ------Details of the player------  Player ID:1HP8,  Name:Harry Potter,  Team:Gryffindor,  Jersey No:8,  Goals Scored:523,  Retirement date:Not yet retired  #################################  ------Details of the player------  Player ID:2RW13,  Name:Ronald Weasley,  Team:Gryffindor,  Jersey No:13,  Goals Scored:5,  Retirement date:Not yet retired  #################################  ------Details of the player------  Player ID:3GW12,  Name:George Weasley,  Team:Gryffindor,  Jersey No:12,  Goals Scored:11,  Retirement date:11 Magical Year, 1886  #################################  Number of players: 3  Player Database: {'1HP8': ['Harry Potter', 'Gryffindor', 8, 523, 'Not yet retired'],  '2RW13': ['Ronald Weasley', 'Gryffindor', 13, 5, 'Not yet retired'],  '3GW12': ['George Weasley', 'Gryffindor', 12, 11, '11 Magical Year, 1886']} |
| --- | --- |

**Task - 18**

| **Driver Code** | **Output** |
| --- | --- |
| user1 = User("Brooks", "Banani", "Shared")  user2 = User("Jocelyn", "Uttara")  user3 = User("Robert", "Gulshan", "Shared")  user4 = User("Langdon", "Mohakhali", "Shared")  user1.status()  user2.status()  user3.status()  user4.status()  print("----------------------------------")  car1 = Uber("0K32BH", "Shared", "Mohakhali", "Banani", "Nikunja", "Uttara")  car1.details()  print("----------------------------------")  car1.pick(user1,user2,user3,user4)  print("----------------------------------")  user1.status()  user2.status()  user3.status()  user4.status()  print("----------------------------------")  car2 = Uber("5GD2BD", "Single", "Uttara")  car3 = Uber("4T12FR", "Shared", "Gulshan", "Bashundhara")  car2.details()  car3.details()  print("----------------------------------")  car2.pick(user2, user3)  print("----------------------------------")  car3.pick(user3)  print("----------------------------------")  user2.status()  user3.status() | Status: Brooks is looking for a shared ride!  Status: Jocelyn is looking for a single ride!  Status: Robert is looking for a shared ride!  Status: Langdon is looking for a shared ride!  ----------------------------------  Car number: 0K32BH  Type: Shared  Routes: Mohakhali --> Banani --> Nikunja --> Uttara  ----------------------------------  Brooks has been picked up.  Jocelyn is looking for a different ride.  Robert's destination is different from this car's route.  Langdon has been picked up.  ----------------------------------  Status: Brooks boarded in car 0K32BH!  Status: Jocelyn is looking for a single ride!  Status: Robert is looking for a shared ride!  Status: Langdon boarded in car 0K32BH!  ----------------------------------  Car number: 5GD2BD  Type: Single  Routes: Uttara  Car number: 4T12FR  Type: Shared  Routes: Gulshan --> Bashundhara  ----------------------------------  Jocelyn has been picked up.  Robert is looking for a different ride.  ----------------------------------  Robert has been picked up.  ----------------------------------  Status: Jocelyn boarded in car 5GD2BD!  Status: Robert boarded in car 4T12FR! |

**Task - 19**

**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**

**Consider the following code:**

**q1 = Quiz1()**

**q1.methodA()**

**q1.methodA()**

**Quiz1.temp += 2**

**q2 = Quiz1(2)**

**q2.methodA()**

**q2.methodA()**

**Task - 20**

**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**

**What is the output of the following code sequence?**

**q2 = Scope()**

**q2.met1()**

**q2.met2()**

**q2.met1()**

**q2.met2()**

**Task - 21**

**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**

**Write the output of the following code:**

**[Answer on the question paper]**

| **q = Q5()**  **q.methodA()** | **x** | **y** | **sum** |
| --- | --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# 

**Task - 22**

**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}")**

**Consider the following code:**

**a1 = A()**

**b1 = B()**

**b2 = B(b1)**

**b1.methodA(3,2)**

**b2.methodB(1,2)**

**Task - 23**

**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**

**Consider the following code:**

| **q = Q5()**  **q.methodA()** | **x** | **y** | **sum** |
| --- | --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Task - 24**

**class A:**

**temp = 8**

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

**self.y = A.temp - 5**

**self.sum = self.temp + 3**

**self.temp += 2**

**def methodA(self, m, n):**

**x = 4**

**self.y = self.y + m + (A.temp)**

**x = x - 2 + n**

**print(x, self.y, self.sum)**

**x = self.y + self.methodB(4, -3)**

**self.sum = self.sum + x + A.temp**

**self.methodB(-4, self.sum, 3)**

**def methodB(self, m, n):**

**y = 5**

**y = y + self.y**

**self.sum = B.x + y + n**

**print(B.x, y, self.sum)**

**class B(A):**

**x = -3**

**def \_\_init\_\_(self, obj=None):**

**super().\_\_init\_\_()**

**if obj != None:**

**obj.sum = self.temp + 13**

**self.y = A.temp + 3**

**self.sum = 6 + A.temp + B.x**

**def methodB(self, m, n, y=0):**

**y = y + self.y + n**

**B.x = m + self.y + n**

**A.temp+=2**

**self.sum = B.x + y + A.temp**

**print(B.x, y, self.sum)**

**return y**

**b1 = B()**

**b2 = B(b1)**

**b1.methodA(-4, 5)**