Week3 Set5

Q1 Employee payroll system

```
Employee name
Employee ID
Basic pay
HRA (House Rent Allowance)
TA (Travelling Allowance)
DA (Dearness Allowance)
Grade pay: Deduction 5% from Basic pay
```

Find the Gross pay, Net pay (Grade pay + deductions) and print all the details given above, using class and object of get () method and cal( )method.

In [3]:
```
import random

class Payroll:
    def __init__(self, empName, basicPay, hra, ta, da):
        self.empName = empName
        self.empId = random.randint(100, 999)
        self.basicPay = basicPay
        self.hra = hra
        self.ta = ta
        self.da = da
    def cal(self):
        self.grossPay = self.basicPay + self.hra + self.ta + self.da
        self.gradePay = 0.95 * self.basicPay
        self.netPay = self.grossPay - self.gradePay
    def get(self):
        print('Employee Name: ' + self.empName)
        print('Employee ID: ' + str(self.empId))
        print('Basic Pay: ' + str(self.basicPay))
        print('HRA (House Rent Allowance): ' + str(self.hra))
        print('TA (Travelling Allowance): ' + str(self.ta))
        print('DA (Dearness Allowance): ' + str(self.da))
        print('Grade Pay: ' + str(self.gradePay))
        print('Gross Pay: ' + str(self.grossPay))
        print('Net Pay: ' + str(self.netPay))

emp1 = Payroll('Aman', 7500000, 150000, 100000, 98000)
emp1.cal()
emp1.get()
```

```
Employee Name: Aman
Employee ID: 961
Basic Pay: 7500000
HRA (House Rent Allowance): 150000
TA (Travelling Allowance): 100000
DA (Dearness Allowance): 98000
Grade Pay: 7125000.0
Gross Pay: 7848000
Net Pay: 723000.0
```

Q2 Student management system

Student name, Student id as inputs with marks for 5 Subjects, calculate and round of the subject marks to 10 and then find CGPA, Grade using object and class concept of oops. Solve the above by creating a base class and derived class use inheritance to solve it

In [6]:
```
class Person:
    def __init__(self, name):
        self.name = name

class Student(Person):
    def __init__(self, name, stId, mrk1, mrk2, mrk3, mrk4, mrk5):
        super().__init__(name)
        self.stId = stId
        self.mrk1 = mrk1
        self.mrk2 = mrk2
        self.mrk3 = mrk3
        self.mrk4 = mrk4
        self.mrk5 = mrk5
    def cal(self):
        total = self.mrk1 + self.mrk2 + self.mrk3 + self.mrk4 + self.mrk5
        self.cgpa = total/50
    def display(self):
        print('The CGPA of ' + self.name + ' is: ' + str(self.cgpa))

std1 = Student('Arjun', 'RA891', 98, 78, 89, 68, 92)
std1.cal()
std1.display()
```

```
The CGPA of Arjun is: 8.5
```

Q3

Instantiate the X class using self and then using self in the get () method and to achieve composition you can instantiate other objects in the class and then use those instances that Generate output of the total code which has both inheritance and composition

CLASSES: When there is an ***IS A*** relationship between parent and child
COMPOSITION: Deligating responsibility from one class to another ... Content-Container relationship

BELOW IS AN EXAMPLE OF COMPOSITION

In [22]:
```
class Salary:
    def __init__(self, pay, bonus):
        self.pay = pay
        self.bonus = bonus
    def annualSalary(self):
        return (self.pay*12) + self.bonus


class Employee:
    def __init__(self, name, age, pay, bonus):
        self.name = name
        self.age= age
        self.objSalary = Salary(pay, bonus)
    def totalSalary(self):
        return self.objSalary.annualSalary()


emp = Employee('Max', 20, 15000, 2000)
print('Salary of the employee is: ' + str(emp.totalSalary()))
```

```
Salary of the employee is: 182000
```

Q4

An Internet service provider has three different subscription packages for its customers:

Package A: for 9.95/month, 10 hours of access are provided. Additional hours are 2.00/hour.
Package B: for 14.95/month, 20 hours of access are provided. Additional hours are 1.00/hour.
Package C: for 19.95/month, unlimited access is provided.

Write a program using objects and classes that calculates a customer's monthly bill. It should ask which package the customer has purchased and how many hours were used. It should then display the total amount due. Input validation: be sure the user only selects package A, B, or C. Also, the number of hours used in a month cannot exceed 744.

In [19]:
```
class Bill:
    def __init__(self, package, hours):
        if package in ['A', 'B', 'C']:
            self.package = package
        else:
            print('The package chosen does not exist')
        if hours <= 744:
            self.hours = hours
        else:
            print('The number of hours used in a month cannot exceed more than 744')
    def cal(self):
        try:
            if self.package is 'A':
                if self.hours <= 10:
                    self.bill = 9.95
                else:
                    self.bill = 9.95 + (self.hours-10)*2.00
            elif self.package is 'B':
                if self.hours <= 20:
                    self.bill = 14.95
                else:
                    self.bill = 14.95 + (self.hours-20)*1.00
            elif self.package is 'C':
                self.bill = 19.95
        except:
            print('Cannot calculate the value of Bill')
    def display(self):
        try:
            print('The package chosen is: ' + self.package)
            print('The total monthly bill is: ' + str(self.bill))
        except:
            print('Cant display the bill')

cust1 = Bill('A', 12)
cust1.cal()
cust1.display()

print()
cust2 = Bill('D', 987)
cust2.cal()
cust2.display()
```

```
The package chosen is: A
The total monthly bill is: 13.95

The package chosen does not exist
The number of hours used in a month cannot exceed more than 744
Cannot calculate the value of Bill
Cant display the bill
```

Q5

Write a program using object-oriented concepts that calculates and displays a person's body mass index (BMI). The BMI is often used to determine whether a person with sedentary lifestyle is overweight or underweight for his or her height. A person's BMI is calculated with the following formula:

BMI = weight * 703/height^2

where weight is measured in pounds and height is measured in inches. The program should display a message indicating whether the person has optimal weight, is underweight, or is overweight. A sedentary person's weight is considered to be optimal if his or her BMI is between 18.5 and 25. If the BMI is less than 18.5, the person is considered to be underweight. If the BMI value is greater than 25, the person is considered to be overweight.

In [20]:
```
class BMI:
    def __init__(self, height, weight):
        self.height = height
        self.weight = weight
    def compare(self):
        self.bmi = (self.weight * 703)/self.height**2
        if self.bmi<18.5:
            print('The person is underweight')
        elif self.bmi>=18.5 and self.bmi<=25:
            print('This person has an optimal BMI')
        elif self.bmi>25:
            print('This person is overweight')

person1 = BMI(4.6, 83)
person1.compare()
```

```
This person is overweight
```