Problem-1: Class inheritence

Create a **Bus** child class that inherits from the Vehicle class. The default fare charge of any vehicle is seating capacity * 100. If Vehicle is Bus instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the final amount = total fare + 10% of the total fare.

Note: The bus seating capacity is 50. so the final fare amount should be 5500. You need to override the fare() method of a Vehicle class in Bus class.

```
# Write code here

class Vehicle:

  def __init__(self,type,capacity):
    self.type = type
    self.capacity = capacity

  def fare(self):
    return 100*self.capacity

class Bus(Vehicle):

  def fare(self):
    base_fare = super().fare()
    bus_fare = base_fare + 0.1*base_fare
    return bus_fare

bus = Bus('school bus',50)
print(bus.fare())
```

```
5500.0
```

Problem-2: Class Inheritence

Create a Bus class that inherits from the Vehicle class. Give the capacity argument of *Bus.seating_capacity()* a default value of 50.

Use the following code for your parent Vehicle class.

```
# Write code here
```

Problem-3: Write a program that has a class Point. Define another class Location which has two objects (Location & Destination) of class Point. Also define a function in Location that prints the reflection of Destination on the x axis.

```
# Write code here
class Point:

  def __init__(self,x,y):
    self.x = x
    self.y = y

  def show_point(self):
    return '{},{}'.format(self.x,self.y)

class Location:

  def __init__(self,x1,y1,x2,y2):
    self.source = Point(x1,y1)
    self.destination = Point(x2,y2)

  def show(self):
    print('source is',self.source.show_point())
    print('destination is',self.destination.show_point())

  def reflection(self):
    self.destination.y = -self.destination.y
    print('Relection is',self.destination.show_point())

L = Location(0,0,1,1)
L.show()
L.reflection()
```

```
 source is 0,0
 destination is 1,1
 Relection is 1,-1
```

`Problem-4:` Write a program that has an abstract class Polygon. Derive two classes Rectangle and Triamgle from Polygon and write methods to get the details of their dimensions and hence calculate the area.

```
# Write code here
from abc import ABC,abstractmethod

class Polygon(ABC):

  @abstractmethod
  def get_data():
    pass

  @abstractmethod
  def area():
    pass

class Rectangle(Polygon):

  def get_data(self,l,b):
    self.l = l
    self.b = b

  def area(self):
    return self.l*self.b

class Triangle(Polygon):

  def get_data(self,b,h):
    self.b = b
    self.h = h

  def area(self):
    return 0.5 * self.b*self.h

rect = Rectangle()
rect.get_data(4,5)
print(rect.area())

tri = Triangle()
tri.get_data(4,5)
tri.area()
```

```
20
10.0
```

Problem-5: Write a program with class Bill. The users have the option to pay the bill either by cheque or by cash. Use the inheritance to model this situation.

```
# Write code here
class Bill:

  def __init__(self,items,price):
    self.total = 0
    self.items = items
    self.price = price

    for i in self.price:
      self.total = self.total + i

  def display(self):
    print('Item \t\t\t Price')
    for i in range(len(self.items)):
      print(self.items[i],'\t',self.price[i])
    print("*"*10)

    print("Total",self.total)

class CashPayment(Bill):

items = ["External Hard Disk", "RAM", "Printer", "Pen Drive"]
price = [5000, 2000, 6000, 800]

deno = [10, 20, 50, 100, 500, 2000]
value = [1, 1, 1, 20, 4, 5]
cash = CashPayment(items, price, deno, value)
cash.show_cash_payment()
```

```
Item                    Price
External Hard Disk      5000
RAM     2000
Printer         6000
Pen Drive       800
**********
Total 13800
10 * 1 = 10
20 * 1 = 20
50 * 1 = 50
100 * 20 = 2000
500 * 4 = 2000
2000 * 5 = 10000
```

```
    super().display()

items = ["External Hard Disk", "RAM", "Printer", "Pen Drive"]
price = [5000, 2000, 6000, 800]
option = int(input("Would you like to pay by cheque or cash (1/2): "))
```