```
#1.

class shape:

  def __init__(self , radius = None , length = None, breadth = None):
    self.radius = radius
    self.length = length
    self.breadth = breadth

class circle(shape):
  def Area(self):
    return self.radius * self.radius * 3.14

class square(shape):
  def Area(self):
    return self.length * self.breadth

class rectangle(shape):
  def Area(self):
    return self.length * self.breadth

#We have three methods with the same name, but they perform different tasks and give differen
#this is method overriding and an example of polymorphism

circle1 = circle(10)
print(circle1.Area())

square1 = square(0 , 10 , 10)
print(square1.Area())

rectangle1 = rectangle(0 , 10 , 20)
print(rectangle1.Area())
```

```
314.0
100
200
```

```
#2.

class Travel:

  def __init__(self , number_of_passengers , distance_traveled , mode_of_transport):
    self._number_of_passengers = number_of_passengers #private
    self.distance_traveled  = distance_traveled
    self.mode_of_transport = mode_of_transport

  @property
  def number_of_passengers(self):
    return self._number_of_passengers

class Train(Travel):

  def cost_of_trip(self):
    return self.number_of_passengers * 60

class Bus(Travel):

  def cost_of_trip(self):
    return self.number_of_passengers * 100


# also this comes under polymorphism


p1 = Train(10 , 25 , 'Train')
print(p1.cost_of_trip())

p2 = Bus(20 , 30 , 'Bus')
print(p2.cost_of_trip())
```

```
600
2000
```

```
#3.
```

```
class Car:

  def __init__(self , model_number):
    self.model_number = model_number

  def swap(self , other):
    temp = other.model_number
    other.model_number = self.model_number
    self.model_number = temp

c1 = Car(100)
c2 = Car(200)

print(c1.model_number)
print(c2.model_number)

print("\nAfter Swapping\n")

c1.swap(c2)
```

```
print(c1.model_number)
print(c2.model_number)
```

```
100
200

After Swapping

200
100
```

✓ 0s completed at 9:27 PM