Session 6: Object Oriented Programming.

Q.1[2 mks] Simple Coordinate class.

```
class Coordinate(object):
  """ A coordinate made up of an x and y value """
  def __init__(self, x, y):
    """ Sets the x and y values """
      self.x = x
      self.y = y
  def __str__(self):
    """ Returns a string representation of self """
      return "<" + str(self.x) + "," + str(self.y) + ">"
  def distance(self, other):
    """ Returns the euclidean distance between two points """
      x_diff_sq = (self.x-other.x)**2
      y_diff_sq = (self.y-other.y)**2
      return (x_diff_sq + y_diff_sq)**0.5
```

# Test code

```
c = Coordinate(3,4)
origin = Coordinate(0,0)
print(c.x, origin.x)
print(c.distance(origin))
print(Coordinate.distance(c, origin))
print(origin.distance(c))
print(c)
```

Q.1[4 mks] Write and test a Python class that implement rational number (`Fraction`) that enables us to deal with fractions. Let it support addition, subtraction, and multiplication operations on rational number instances.

Hint:

1- Use (*__add__* , *__sub__* , *__mul__* , …...) to override the required operations.

2- Use *isinstance()* to check if an object is a *`Fraction`* before performing the mention operations.

3- Use Exceptions/Assertions if needed.

```
class Fraction(object):
    """
    Rational Numbers with support for arithmetic operations.

        >>> a = Fraction(1, 2)
        >>> b = Fraction(1, 3)
        >>> print(a + b)
        5/6
        >>> print(a – b)
        1/6
        >>>print( a * b)
        1/6
        >>>print( a/b)
        3/2
    """

def __init__(self, num, denom):

        """ num and denom are integers """

        # write your code here
def __str__(self):
         """ Retunrs a string representation of self """

        # Define your own print method here.
def __add__(self, other):
        """ Returns a new fraction representing the addition """
      top = self.num*other.denom + self.denom*other.num
      bott = self.denom*other.denom
      return Fraction(top, bott)
def __sub__(self, other):
        """ Returns a new fraction representing the subtraction """

        # Define your subtract method here.
def __mul__(self, other):
        # Define your multiplication method here.
def __float__(self):
        """ Returns a float value of the fraction """
      return self.num/self.denom
def inverse(self):
        """ Returns a new fraction representing 1/self """
      return Fraction(self.denom, self.num)
```

# Test code

a = Fraction(1,4)
b = Fraction(3,4)
c = a + b # c is a Fraction object
print(c)
print(float(c))

```
print(Fraction.__float__(c))
print(float(b.inverse()))
##c = Fraction(3.14, 2.7)          # assertion error
##print a*b                        # error, did not define how to multiply two Fraction objects
```

Q.2[3mks] Write and test a Python class named `intSet.`

```
class intSet(object):
    """
        An intSet is a set of integers
        The value is represented by a list of ints, self.vals
        Each int in the set occurs in self.vals exactly once
    """
    def __init__(self):
        """ Create an empty set of integers """
        self.vals = []

    def insert(self, e):
        """ Assumes e is an integer and inserts e into self """
        if not e in self.vals:
            self.vals.append(e)

    def member(self, e):
        """ Assumes e is an integer
            Returns True if e is in self, and False otherwise
        """
        return e in self.vals

    def remove(self, e):
        """
            Assumes e is an integer and removes e from self
            Raises ValueError if e is not in self
        """
        try:
            self.vals.remove(e)
        except:
            raise ValueError(str(e) + ' not found')

    def __str__(self):
        """ Returns a string representation of self """
        self.vals.sort()
        return '{' + ','.join([str(e) for e in self.vals]) + '}'
```

# Test code

```
s = intSet()
print(s)
s.insert(3)
s.insert(4)
s.insert(3)
print(s)
s.member(3)
s.member(5)
s.insert(6)
print(s)
#s.remove(3)  # leads to an error
print(s)
s.remove(3)
```