Session 5

Question 1

Write Point class, each point consists of two coordinates x, y, include the class the following:

- **init** : initialize x, y.
- getLength(point): return the length between two points.
- **eq** (point): check if the two points are same.
- **str** (): return "(x, y)"

In [7]:
```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def getLength(self, p):
        return ((self.x - p.x)**2 + (self.y - p.y)**2)**0.5
    def __eq__(self, p):
        return self.x == p.x and self.y == p.y
    def __str__(self):
        return "({}, {})".format(self.x, self.y)
```

In [8]:
```
p1 = Point(3, 4)
p2 = Point(5, 9)
print(p1.getLength(p2))
if p1 == p2:
    print(p1)
else:
    print(p2)
```

```
5.385164807134504
(5, 9)
```

Question 2

Write Polygon class, each polygon consists of list of points, include the class the following:

- **init** : initialize an empty points list
- addPoint(point): add point to the list.
- isClosed(): return True if the polygon is closed (first point equals the last one)
- **len** : return the number of points.

In [13]:
```
class Polygon:
    def __init__(self):
        self.points = []

    def addPoint(self, p):
        self.points.append(p)
    def isClosed(self):
        if len(self.points) < 2:
            return False
        else:
            if self.points[0] == self.points[-1]:
                return True
            else:
                return False

    def __len__(self):
        return len(self.points)
```

In [15]:
```
poly = Polygon()
poly.addPoint(Point(1, 3))
poly.addPoint(Point(2, 4))
poly.addPoint(Point(1, 6))
poly.addPoint(Point(1, 3))
print(poly.isClosed())
print(len(poly))
```

```
True
4
```

Question 3

UPDATE the Polygon class to be an iterator over its points.

In [18]:
```
class Polygon:
    def __init__(self):
        self.points = []

    def addPoint(self, p):
        self.points.append(p)

    def isClosed(self):
        if len(self.points) < 2:
            return False
        else:
            if self.points[0] == self.points[-1]:
                return True
            else:
                return False

    def __len__(self):
        return len(self.points)

# Make the Polygon be an Iterator over its points
    def __iter__(self):
        self.current = 0
        return self

    def __next__(self):
        if self.current >= len(self):
            raise StopIteration
        else:
            p = self.points[self.current]
            self.current += 1
            return p
```

In [20]:
```python
poly = Polygon()
poly.addPoint(Point(1, 3))
poly.addPoint(Point(2, 4))
poly.addPoint(Point(1, 6))
poly.addPoint(Point(1, 3))

for p in poly:
    print(p)
```

```
(1, 3)
(2, 4)
(1, 6)
(1, 3)
```

Homework

UPDATE the iterator to iqnore the last point if the polygon is closed.

In [ ]: