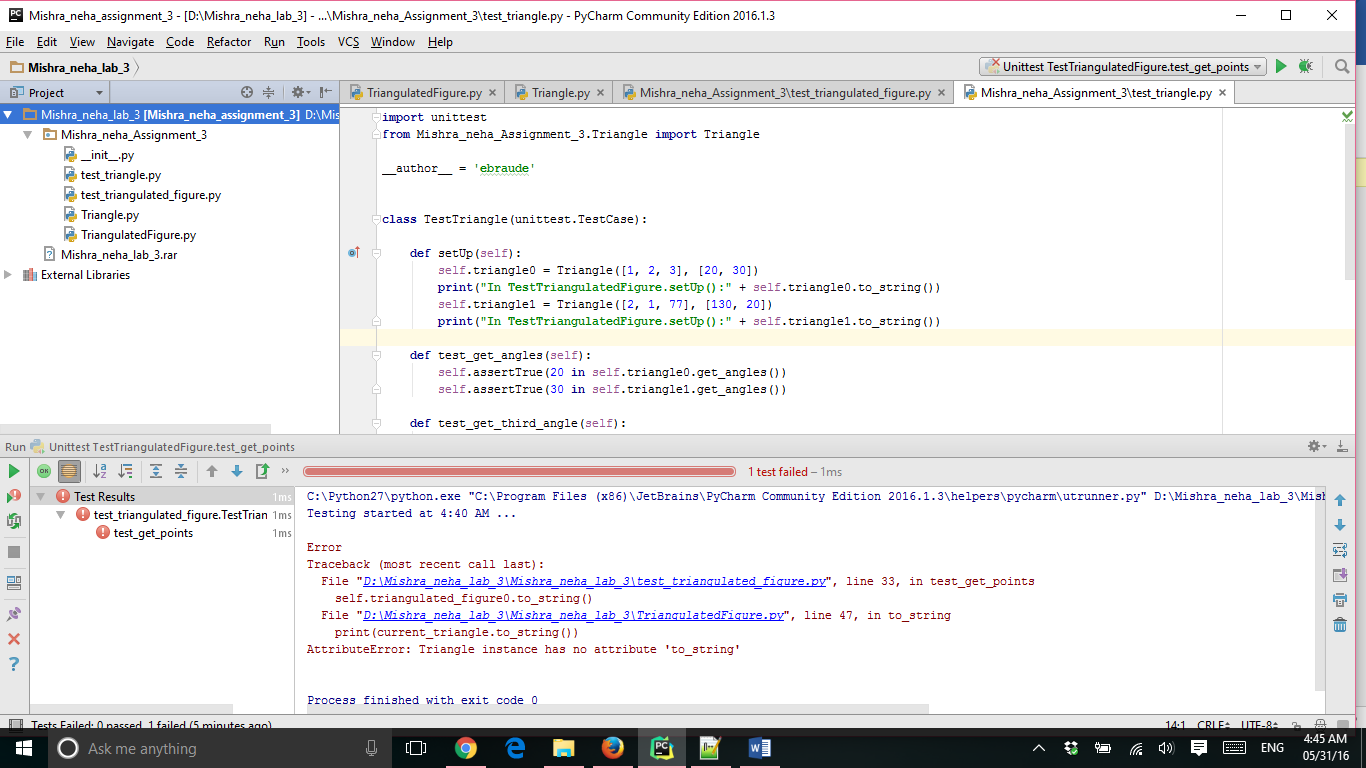
**Assignment 3**

**Information structures with python**

**Deliverables:**



The code is attached in the .zip file

**from** triangle **import** Triangle  
  
  
**class** TriangulatedFigure:  
 *# Class Invariant 1: Every triangle in self.triangles has  
 # a unique set of vertices [Note 1 (at end)]  
  
 # Class Invariant 2: len(self.triangles) < 2 --XOR--  
 # Every Triangle in self.triangles shares two get\_points with another* **def** \_\_init\_\_(self):  
  
  
 self.triangles = [] *# the Triangle objects that make up self* **def** add(self, a\_triangle):  
 *# Precondition 1: a\_triangle is a Triangle instance  
 # Precondition 2: len(self.triangles) < 2  
 # --XOR--  
 # a\_triangle ... is not in self.triangles AND  
 # ... shares two vertices with a Triangle in old(self.triangles)  
 # Postcondition: a\_triangle is in self.triangles* **if** (len(self.triangles) < 2 **or** (a\_triangle **not in** self.triangles) **and not** (  
 len(self.triangles) < 2 **and** (a\_triangle **not in** self.triangles))):  
 *## satisfying the condition first* self.triangles.append(a\_triangle)*## adding the value to the list* **def** get\_points(self):  
 triangles\_points = []  
 **for** triangle **in** self.triangles:  
 triangles\_points.extend(triangle.get\_points())*## attaching by extending the value to traingles.points* **return** set(triangles\_points)  
  
  
  
 **def** to\_string(self):  
 **for** current\_triangle **in** self.triangles:  
 **print**(current\_triangle.to\_string())  
  
 **def** triangles\_with\_vertex(self, a\_point):  
 *# Precondition: At least one triangle in self.triangles contains a\_point  
 # Returns the (contiguous) list of self.triangles containing a\_point  
 # in clockwise order  
 # Example: URL1 (see at end)  
  
 # [Collected]: triangles\_with\_a\_point =  
 # the triangles in self.triangles containing a\_point [Note 3]* triangles\_with\_a\_point = []  
  
 **for** triangles\_ **in** self.triangles:  
 **if** a\_point **in** triangles\_.get\_points():  
 triangles\_with\_a\_point.append(triangles\_)  
 triangles\_in\_order=[triangles\_with\_a\_point[0]]  
 *#triangles\_remaining=triangles\_with\_a\_point[1:].copy()* **return** triangles\_with\_a\_point

output :

only one test case runs and the output is :

