Zusätzliche Informationen zur Aufgabe 3:

- Ziel der Aufgabe ist es ein kleines Geoinformationssystem zu entwerfen. Im Test-Programm wird das System zuerst initialisiert, also z.B.:

  ```
  gis = GeoInfoSystem( "welt.jpg", (-180, 180), (-90, 90) )
  gis.readLocations( "Russia.txt" )
  ```

  Anschliessend kann man sich Orte in bestimmten Landstrichen der Welt anzeigen lassen:

  ```
  gis.showLocations( (66.5, 90), (-180, 180) )
  #zeigt alle Orte nördlich des Polarkreises an
  ```

- Die `GeoInfoSystem`-Funktion `readGeoLocations("Dateiname")` soll eine Menge von Geo-Daten einlesen und diese der Liste von `GeoLocation`-Objekten des `GeoInfoSystem`-Objekts hinzufügen.
  So eine Funktion könnte ungefähr folgendermassen aussehen:

  ```
  def readGeoLocations( dateiname ):
       inputFile = open( dateiname )
      # In der ersten Zeile stehen die Spaltennamen
       line = inputFile.readline()
      # Deswegen erst ab der zweiten Zeile Einlesen
      line = inputFile.readline()
      while line:
          tmp = string.split(line, '\t')
         self.geoLocationList.append(GeoLocation( int(tmp[0]),
                                                  float(tmp[3]),
                                                  float(tmp[4]),
                                                  tmp[9],
                                                  tmp[12],
                                                  tmp[23]) )
          line = inputFile.readline()
    inputFile.close()
  ```

- Die `showLocations`-Funktion ruft intern zuerst die Funktion `getLocations` auf. Diese geht einfach die gesamte Liste der `GeoLocation`-Objekte des `GeoInfoSystem`-Objekts durch und vergleicht die Koordinaten mit den Eingabe-Koordinaten des gesuchten Landstrichs. Liegt das `GeoLocation`-Objekt innerhalb, wird es in einer Extra-Liste gespeichert. Diese Liste wird

am Ende, wenn alle `GeoLocation`-Objekte verglichen wurden, als Return-Wert zurückgegeben.
D.h: Nach Aufruf von `getLocations` hat man in der Funktion `showLocations` eine Liste mit allen `GeoLocation`-Objekte die im gesuchten Landstrich liegen.
Nun müssen diese noch auf der Karte eingezeichnet werden:
Dazu geht man die Liste durch und berechnet für jedes der `GeoLocation`-Objekte die entsprechende Pixelkoordinate auf der Karte:
Angenommen die Karte hat die Längen-Koordinaten (-180, 180) und Breiten-Koordinaten (-90, 90) und ist 1000 Pixel lang und 400 Pixel hoch. Dann hat Clausthal (Mit den Koordinaten 51.18 nördliche Breite und 10.34 östliche Länge) auf der Karte die Pixelkoordinate (x,y):
$x = 1000 * \frac{180+10.34}{180-(-180)}$ und $y = 400 * \frac{90+51.18}{90-(-90)}$
Als Markierung wird der entsprechende Pixel mit den wohlbekannten Python-Image-Library-Funktionen einfach schwarz gefärbt.