```
from osgeo import ogr
import matplotlib.pyplot as plt
import matplotlib.patches as rect
import matplotlib.font_manager as font
from descartes import PolygonPatch
from pysal.esda import mapclassify as mc
import numpy as np

# Create a figure plot
fig = plt.figure(1,figsize=[11.5,11.5])
ax = fig.add_subplot(111)

# Open a shapefile
shpFile = ogr.Open("HistoricPop.shp")
layer = shpFile.GetLayer(0)
feature = layer.GetNextFeature()

# First we scan through the attributes and calculate population density
classes =[]
while feature is not None:
   geom = feature.GetGeometryRef()
   pop = feature.GetField('PERSONS200')
   area = float(geom.GetArea())/1000000 #for km2
   popdense = pop/area
   classes.append(popdense)
   feature = layer.GetNextFeature()

# Having appended population density to a list, we give it to pysal
# We're using Jenk's here to find natural breaks in the data
x = np.array(classes)
fj = mc.Fisher_Jenks(x)

# This function allows us the set colours based on the Jenk's breaks
def classify( polygon, value ):
   if value <= fj.bins[0]:
      return PolygonPatch(polygon,fc= '#F1EEF6' ,ec= '#999999')
   elif value <= fj.bins[1]:
      return PolygonPatch(polygon,fc= '#BDC9E1' ,ec= '#999999')
   elif value <= fj.bins[2]:
      return PolygonPatch(polygon,fc= '#74A9CF' ,ec= '#999999')
   elif value <= fj.bins[3]:
      return PolygonPatch(polygon,fc= '#2B8CBE' ,ec= '#999999')
   else:
      return PolygonPatch(polygon,fc= '#045A8D' ,ec= '#999999')

# Reset the shapefile for drawing
layer.ResetReading()
feature = layer.GetNextFeature()
```

```
while feature is not None:
   geom = feature.GetGeometryRef()
   pop = feature.GetField('PERSONS200')
   area = float(geom.GetArea())/1000000 #for km2
   popdense = pop/area
   # This gets the geometry for single features.
   if geom.GetGeometryCount() < 2:
      geomPnt = geom.GetGeometryRef(0)
      points=[]
      for i in range(0,geomPnt.GetPointCount()):
         geomPntX = geomPnt.GetX(i)
         geomPntY = geomPnt.GetY(i)
         points.append([geomPntX,geomPntY])
      # This is a geoJSON-like object
      polygon = {"type": "Polygon","coordinates":[points]}
      patch = classify(polygon, popdense)
      ax.add_patch(patch)

   for geomcnt in range(0,geom.GetGeometryCount()):
      multifeat = geom.GetGeometryRef(geomcnt)
      # This is the loop that deals with multifeature records.
      for i in range(0,multifeat.GetGeometryCount()):
         geomPnt = multifeat.GetGeometryRef(i)
         points = []
         for j in range(0,geomPnt.GetPointCount()):
            geomPntX = geomPnt.GetX(i)
            geomPntY = geomPnt.GetY(i)
            points.append([geomPntX,geomPntY])
      # Unfortunately descartes doesn't support 'MultiPolygon'
      # luckily it's not an issue with this shapefile
      polygon = {"type": "Polygon","coordinates":[points]}
      patch = classify(polygon, popdense)
      ax.add_patch(patch)
   feature = layer.GetNextFeature()

# Set the spatial extent - you could derive this from the shp bounding box
ax.set_xlim([500000,565000])
ax.set_ylim([154000,205000])
# Equal spacing for x and y axes
ax.set_aspect(1)

# Unfortunately we have to manually create the legend and legend labels.
# This is done with rectanges that we draw out of frame.
p = [rect.Rectangle((0,0),1,1,fc='#F1EEF6',ec='#999999'),
    rect.Rectangle((0,0),1,1,fc='#BDC9E1',ec='#999999'),
    rect.Rectangle((0,0),1,1,fc='#74A9CF',ec='#999999'),
    rect.Rectangle((0,0),1,1,fc='#2B8CBE',ec='#999999'),
    rect.Rectangle((0,0),1,1,fc='#045A8D',ec='#999999')
    ]
```

```
labels = ["0 - " + str(round(fj.bins[0],2)),
        str(round(fj.bins[0],2)) + " - " + str(round(fj.bins[1],2)),
        str(round(fj.bins[1],2)) + " - " + str(round(fj.bins[2],2)),
        str(round(fj.bins[2],2)) + " - " + str(round(fj.bins[3],2)),
        str(round(fj.bins[3],2)) + " - " + str(round(fj.bins[4],2)),
        ]
for i in range(0,len(p)):
   ax.add_patch(p[i])

# legend size is based on label font size
legsize = font.FontProperties(size=10)

# Add a legend to the plot
ax.legend(p,labels,loc = 4, title = "People per km$^2$",prop = legsize)

# These 3 lines create a basic scalebar
scalebar = rect.Rectangle((501000,155500),10000,1000,ec= 'black',fc = 'white')
ax.add_patch(scalebar)
plt.text(512000,155400,"10 Kilometres",fontsize = 12)

# Add a title
plt.title("London Population Density, Census 2001")

# Add a north arrow from an image.
imX = plt.imread('NorthArrow.png')
ax.imshow(imX,extent = [501000,505000,158000,165000])

# Either plot the map, or save it to a file.
plt.show()
#plt.savefig('plot1.png')
```