When you find an interesting code, I was you to copy and paste it into a word document and then answer the following questions in a paragraph:

|  |
| --- |
| #!/usr/bin/env pythonw |
|  | # encoding: utf-8 |
|  | # requires BeautifulSoup |
|  | # based on Flowing Data blog post: |
|  | # http://flowingdata.com/2009/11/12/how-to-make-a-us-county-thematic-map-using-free-tools/ |
|  | # |
|  | # Usage: |
|  | # $ ./colorize\_svg.py -f county\_counts.txt > twitter\_users.svg |
|  | # |
|  | # Expects a file of county\_counts containing two columns: |
|  | # fipscode, count (integers) |
|  | # |
|  | # 51770 1 |
|  | # 13089 1 |
|  | # 54011 1 |
|  | # 54039 3 |
|  | # 12117 2 |
|  | # |
|  | # Also require a baseline svg file in the same directory called counties.svg |
|  | # |
|  |  |
|  | import csv |
|  | from BeautifulSoup import BeautifulSoup, Tag |
|  | from math import log |
|  | import time |
|  | import getopt |
|  | import os, sys |
|  |  |
|  | class Usage(Exception): |
|  | def \_\_init\_\_(self, msg): |
|  | self.msg = msg |
|  |  |
|  | def load\_intensities(filename): |
|  | intensities = {} |
|  | reader = csv.reader(open(filename), delimiter="\t") |
|  | for row in reader: |
|  | try: |
|  | fips = row[0] |
|  | intensities[fips] = int(row[1]) |
|  | except: |
|  | pass |
|  | return intensities |
|  |  |
|  | def generate\_heatmap(intensities): |
|  | # Load the SVG map |
|  | svg = open('counties.svg', 'r').read() |
|  | # Load into Beautiful Soup |
|  | soup = BeautifulSoup(svg, selfClosingTags=['defs','sodipodi:namedview']) |
|  | # Find counties |
|  | paths = soup.findAll('path') |
|  | colors = ["#DEEBF7", "#C6DBEF", "#9ECAE1", "#6BAED6", "#4292C6", "#2171B5", "#08519C", "#08306B"] |
|  | min\_value = min(intensities.values()) |
|  | max\_value = max(intensities.values()) |
|  | scalefactor = (len(colors)-1)/(log(max\_value +1)-log(min\_value +1)) |
|  | # County style |
|  | path\_style = 'font-size:12px;fill-rule:nonzero;stroke:#FFFFFF;stroke-opacity:1;stroke-width:0.1;stroke-miterlimit:4;stroke-dasharray:none;stroke-linecap:butt;marker-start:none;stroke-linejoin:bevel;fill:' |
|  | # we will append this hover tooltip after each county path |
|  | hover\_text = '''<text id="popup-%s" x="%s" y="%s" font-size="10" fill="black" visibility="hidden">%s (%s)<set attributeName="visibility" from="hidden" to="visible" begin="%s.mouseover" end="%s.mouseout"/></text>''' |
|  | for p in paths: |
|  | if p['id'] not in ["State\_Lines", "separator"]: |
|  | try: |
|  | count = intensities[p['id']] |
|  | except: |
|  | count = 0 |
|  | x, y = (p['d'].split()[1]).split(',') |
|  | # insert a new text tag for the county hover tooltip... |
|  | p.parent.insert(0, Tag(soup, 'text', [("id", 'popup-'+p['id'])])) |
|  | hover = soup.find("text", { "id" : 'popup-'+p['id'] }) |
|  | hover.insert(1, "%s (%s)" % (p['inkscape:label'], str(count))) |
|  | # add attributes to that text tag... |
|  | hover['x'] = 250 |
|  | hover['y'] = 20 |
|  | hover['font-size'] = "20" |
|  | hover['fill'] = "black" |
|  | hover['visibility'] = "hidden" |
|  | hover.insert(0, Tag(soup, 'set', [("begin", p['id']+'.mouseover')])) |
|  | set\_tag = soup.find("set", { "begin" : p['id']+'.mouseover' }) |
|  | set\_tag['attributeName'] = "visibility" |
|  | set\_tag['from'] = "hidden" |
|  | set\_tag['to'] = "visible" |
|  | set\_tag['end'] = p['id']+'.mouseout' |
|  | color\_class = min(int(scalefactor\*log(count +1)), len(colors)-1) |
|  | # color\_class = int((float(len(colors)-1) \* float(count - min\_value)) / float(max\_value - min\_value)) |
|  | # if count > 0: |
|  | # print color\_class |
|  | color = colors[color\_class] |
|  | p['style'] = path\_style + color |
|  | print soup.prettify() |
|  |  |
|  | def main(argv=None): |
|  | if argv is None: |
|  | argv = sys.argv |
|  | try: |
|  | try: |
|  | opts, args = getopt.getopt(argv[1:], "hf:v", ["help", "filename="]) |
|  | except getopt.error, msg: |
|  | raise Usage(msg) |
|  |  |
|  | # option processing |
|  | for option, value in opts: |
|  | if option == "-v": |
|  | verbose = True |
|  | if option in ("-h", "--help"): |
|  | raise Usage(help\_message) |
|  | if option in ("-f", "--filename"): |
|  | filename = value |
|  |  |
|  | # main processing |
|  | intensities = load\_intensities(filename) |
|  | generate\_heatmap(intensities) |
|  |  |
|  | except Usage, err: |
|  | print >> sys.stderr, sys.argv[0].split("/")[-1] + ": " + str(err.msg) |
|  | print >> sys.stderr, "\t for help use --help" |
|  | return 2 |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | sys.exit(main()) |

Link for the code <https://github.com/datawrangling/spatialanalytics/blob/master/visualization/colorize_svg.py>

1. What does this code do?

This colorizes the thematic map. The color intensifies based on the file “BeautifulSoup” and compares the intensities by counties.

1. Do the author of the code have any interesting comments about how to use the code?

The author does not make any notes throughout the code, but they do have a small description in the beginning.

1. Is the code seem easy to modify for you to use?

At this time, it is not that easy to modify, but I plan to search for what the arguments mean and how to modify this code.

1. Is there anything interesting? #general comments

Even though this code is really old, 10 years old, it looks like it will still run perfectly fine if you have the correct files. The online tutorial is also really informative too. <https://flowingdata.com/2009/11/12/how-to-make-a-us-county-thematic-map-using-free-tools/>