# SVG_converter_basic1

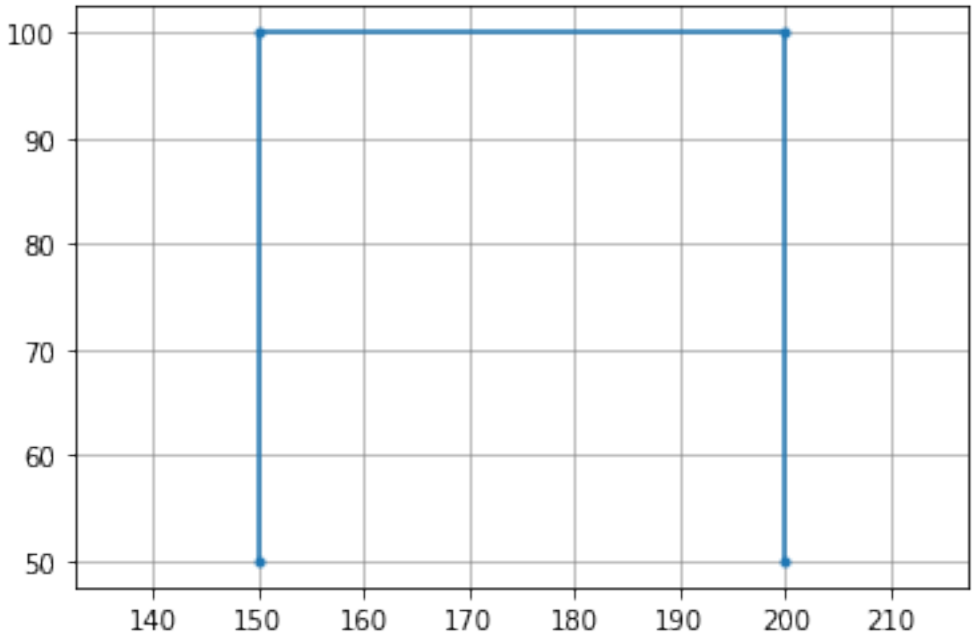
December 24, 2022

[1]:
```
## Python basics for novice data scientists, supported by Wagatsuma Lab@Kyutech
#
# The MIT License (MIT): Copyright (c) 2022 Hiroaki Wagatsuma and Wagatsuma␣
 ↪Lab@Kyutech
#
# Permission is hereby granted, free of charge, to any person obtaining a copy␣
 ↪of this software and associated documentation files (the "Software"), to␣
 ↪deal in the Software without restriction, including without limitation the␣
 ↪rights to use, copy, modify, merge, publish, distribute, sublicense, and/or␣
 ↪sell copies of the Software, and to permit persons to whom the Software is␣
 ↪furnished to do so, subject to the following conditions:
# The above copyright notice and this permission notice shall be included in␣
 ↪all copies or substantial portions of the Software.
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR␣
 ↪IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,␣
 ↪FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE␣
 ↪AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER␣
 ↪LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING␣
 ↪FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS␣
 ↪IN THE SOFTWARE. */
#
# # @Time    : 2022-12-24
# # @Author  : Hiroaki Wagatsuma
# # @Site    : https://github.com/hirowgit/2BO_python_optmization_course
# # @IDE     : Python 3.9.14 (main, Sep  6 2022, 23:29:09) [Clang 13.1.6␣
 ↪(clang-1316.0.21.2.5)] on darwin
# # @File    : SVG_converter_basic1.py


import numpy as np
import matplotlib.pyplot as plt
import os

from svg.path import parse_path
from svg.path.path import Line
from xml.dom import minidom
```

```
from time import time
import pandas as pd

datafol_in='data'
datafol_out='output'

inputF='draw1.svg'

doc = minidom.parse(os.path.join(datafol_in,inputF))
# path_strings = [path.getAttribute('d') for path
#                 in doc.getElementsByTagName('path')]
# path_strings = [path.getAttribute('d') for path
#                 in doc.getElementsByTagName('polyline')]

path_strings = [path.getAttribute('points') for path
                in doc.getElementsByTagName('polygon')]

doc.unlink()

points_np_all2=[]
points_np_all=np.empty((len(path_strings)),dtype=object)

for k in range(len(path_strings)):
# for k in range(KL):
#for path_string in path_strings:
#     path = parse_path(path_strings[k])
    points_np_merge=np.empty((0,2), float)

#     path = path_strings[k].split(' ')
    kd=path_strings[k]
    pointDstr=[d.split(',') for d in kd.split(' ') if len(d)>0]
    pointDtmp=np.array([(d[0]) for d in pointDstr])
    pointD=pointDtmp.astype(np.float64)
    vNum=int(np.shape(pointD)[0]/2)
    pointD2=np.reshape(pointD,[vNum,2])
#     print(pointD2)
    points_np_all[k]=pointD2
#     k2=1

# points_np_all
print(len(points_np_all))

for k in range(len(points_np_all)):
    print(' %d : %d dots' % (k,len(points_np_all[k])))

fig1, ax = plt.subplots()
# plt.grid(color='k', linestyle='-', linewidth=0.5)
```

```
plt.grid(color=[0.5,0.5,0.5], linestyle='-', linewidth=0.5)

for k in range(len(points_np_all)):
    points_np=points_np_all[k]
    plt.plot(points_np[:, 0], points_np[:, 1], '.-')
ax.axis('equal')
plt.show()

maxL=max(len(points_np_all[k]) for k in range(len(points_np_all)))

label=np.empty([],dtype='unicode')
print("label size = %d" % (label.size))
label=[]
for k in range(len(points_np_all)):
    label=np.append(label,["x%d"%(k+1),"y%d"%(k+1)])
dat_df = pd.DataFrame([],columns=label)
for k in range(len(points_np_all)):
    points_np=points_np_all[k]
    tmp0=np.zeros([maxL,2])
    tmp0[0:points_np.shape[0],:]=points_np
    dat_df["x%d"%(k+1)] = tmp0[:,0]
    dat_df["y%d"%(k+1)] = tmp0[:,1]

print(dat_df.shape)
dat_df

outF=inputF.split('.')[0]+'_full.csv'
dat_df.to_csv(os.path.join(datafol_out,outF))
```

```
1
 0 : 4 dots
```




```
label size = 1
(4, 2)
```