```
'''
 ### Information for use of AddTextToDXF ###

@author Hans-Christian Ringstad

This script was made to be used in the Bachelor Thesis Manulab spring 2020, it will put a logo image and a text on a
dxf-file template. Files used in this program needs to be put in the same folder as the program. For the source code
Python 3.8.1 was used.

 ### Files needed to run the program ###
info.csv: Contains information for the program to add text and logo image. This csv-file uses ";" as separator and the
          program will only read the info in the last row.
      This file will need to contain these fields;
    # Name of field # Datatype # Comment
    finalFileName   : String   : Name of the dxf file that will be created
    templateName    : String   : Name of the template to be used as background
    logoFileName    : String   : Name of the logo PNG-file
    xInsertPointLogo: float    : x-coordinate of the insert point of logo, the insertion point will be at the bottom right
    yInsertPointLogo: float    : y-coordinate of the insert point of logo, the insertion point will be at the bottom right
    xPixelSize      : int      : x length in pixels
    yPixelSize      : int      : y length in pixels
    xSizeInmm       : float    : x length in millimeter
    ySizeInmm       : float    : y length in millimeter
    rotationLogo    : float    : Rotation of logo
    textToInsert    : String   : Text to insert on the dxf file
    textStyle       : String   : Text style
    xInsertPointText: float    : x-coordinate of the insert point of text, the insertion point will be at the top center
    yInsertPointText: float    : y-coordinate of the insert point of text, the insertion point will be at the top center
    rotationText    : float    : Rotation of text
    alignmentText   : int      : Alignement for the text*
    textWidth       : float    : Width of text fields, does
```

```
31  not cut ind ividual words
32     textHeight      : float    : Height of text
33     #               #          #
34 *See doc for more info: https://ezdxf.mozman.at/docs/
   dxfentities/mtext.html#ezdxf.entities.MText.dxf.
   attachment_point
35
36 templateName.dxf: The file containing the dxf template to
   be used by the program, the name of the file is to be
   decided
37                   in info.csv
38
39 logoFileName.png: The file containing the the png of the
   logo in use to be used by the program, the name of the file
    is
40                   to be decided in info.csv
41
42  ### Files created by the program ###
43 status.csv: Contains information on status of the program.
   This csv-file uses ";" as separator and the program will
44             write the status at the bottom row read the
   info in the last row.
45         This file will contain these fields;
46     # Name of field # Datatype # Comment
47     working         : boolean  : True when active, false if
    inactive
48     done            : boolean  : True when the dxf file was
    created succesfully
49     error           : boolean  : True when an error has
   occured
50     #               #          #
51
52 finalFileName.dxf: The final dxf file created by the
   specifications in info.csv,the name of the file is to be
   decided in
53                    info.csv
54
55 errLog.txt: All expected errors will be printed to this
   file with an error message of what has caused it.
56 '''
57 import datetime
58 import sys
59 import ezdxf
60 import os
61 import csv
62 import time
```

```
 63
 64 '''
 65 Error log
 66 '''
 67
 68
 69 def errLog(errType, errMsg, stopProg, updateStatus):
 70     with open("errLog.txt", "a") as text_file:
 71         __ = text_file.write(str(datetime.datetime.now
    ()) + "  |  " + str(errType) + ": " + errMsg + "\n")
 72     if updateStatus:
 73         __ = updateStatus(_statusFileName, False, False,
    True)
 74     if stopProg:
 75         sys.exit()
 76     return True
 77
 78
 79 '''
 80  Write to status.csv
 81 '''
 82
 83
 84 def updateStatus(statusFileName, working, done, error):
 85     returnVal = False
 86     attempts = 0
 87     maxAttempts = 50
 88     while (not returnVal) & (attempts < maxAttempts):
 89         try:
 90             with open(statusFileName, 'w', newline='') as
    csvfile:
 91                 sep = ';'
 92                 statusWriter = csv.writer(csvfile,
    delimiter=sep, quotechar=' ', quoting=csv.QUOTE_MINIMAL)
 93                 statusWriter.writerow(['sep=' + sep])
 94                 statusWriter.writerow(['working', 'done',
    'error'])
 95                 statusWriter.writerow([working, done,
    error])
 96                 returnVal = True
 97         except PermissionError:
 98             errType = str(PermissionError)
 99             returnVal = False
100             attempts = attempts + 1
101             __ = errLog(errType, " has occured at " +
    statusFileName + '. Attempt ' + str(attempts), False,
```

```
101 False)
102             time.sleep(_permWaitTime)
103     if not returnVal:
104         __ = errLog(errType, " has occured at " +
    statusFileName + ". Force-stops program after " + str(
    attempts)
105                         + " attempts.", True, False)
106     return returnVal
107 
108 
109 '''
110  Removes file from system
111 '''
112 
113 
114 def removeFile(fileName):
115     returnVal = False
116     attempts = 0
117     maxAttempts = 50
118     while (not returnVal) & (attempts < maxAttempts):
119         try:
120             returnVal = True
121             os.remove(fileName)
122         except FileNotFoundError:
123             returnVal = True
124             errType = str(FileNotFoundError)
125             __ = errLog(errType, " File " + fileName + "
    was not found when trying to delete it.", False, False)
126             pass
127         except PermissionError:
128             returnVal = False
129             errType = str(PermissionError)
130             __ = errLog(errType,
131                         " File " + fileName + " is in use
    when trying to delete it. Close all programs using it",
    True,
132                         True)
133     if not returnVal:
134         __ = errLog(errType, " has occured at " +
    _statusFileName + ". Force-stops program after " + str(
    attempts)
135                     + " attempts.", True, False)
136     return returnVal
137 
138 
139 '''
```

```
Get a dxf file
'''


def getDXF(dxfFileName):
    try:
        temp = ezdxf.readfile(dxfFileName)
    except FileNotFoundError:
        __ = errLog(str(FileNotFoundError), " has occured. " + dxfFileName + " were not found", True, True)
    except PermissionError:
        __ = errLog(str(PermissionError), " has occured. Please close all program using " + dxfFileName
                    + " before continuing ", True, True)
    return temp


'''
 Setup
'''
_permWaitTime = 0.1
_errType = "No Error"
_statusFileName = 'status.csv'
_infoFileName = "info.csv"
_strNameLen = 40
'''
 Update status
'''
status = updateStatus(_statusFileName, True, False, False)
'''
 Read info.csv
'''
with open('info.csv') as csvfile:
    _infoReader = csv.reader(csvfile, delimiter=' ', quotechar='|')
    for row in _infoReader:
        _info = row
_info = ', '.join(_info)
_info = _info.replace(',', '')
_info = _info.split(";")

'''
 Deletes the dxf file before saving the new file, also ignores exception FileNotFoundError if file does not exist
'''
_dxfFileName = str(_info[0])  # finalFileName
```

```
182 removeFile(_dxfFileName)
183 '''
184  Get template from file and create the modelspace to add
    design
185 '''
186 _dxfTemplateFileName = str(_info[1])  # templateName
187 _doc = getDXF(_dxfTemplateFileName)
188 _msp = _doc.modelspace()
189 '''
190  Add NTNU Manulab logo
191 '''
192 _logoFileName = str(_info[2])  # logoFileName
193 _xInsert = float(_info[3])  # xInsertPointLogo
194 _yInsert = float(_info[4])  # yInsertPointLogo
195 _xPixel = int(float(_info[5]))  # xPixelSize
196 _yPixel = int(float(_info[6]))  # yPixelSize
197 _xSize = float(_info[7])  # xSizeInmm
198 _ySize = float(_info[8])  # ySizeInmm
199 _rotLogo = float(_info[9])  # rotationLogo
200
201 _my_image_def = _doc.add_image_def(filename=_logoFileName
    , size_in_pixel=(_xPixel, _yPixel))
202 _image = _msp.add_image(image_def=_my_image_def, insert=(
    _xInsert, _yInsert), size_in_units=(_xSize, _ySize),
    rotation=_rotLogo)
203
204 '''
205  Add name text from input
206 '''
207 _textFileName = str(_info[10])  # textToInsert
208 _textStyle = str(_info[11])  # textStyle
209
210 _strName = _textFileName.replace('\n', ' ')
211 _strName = _strName[:_strNameLen]
212
213 _xLoc = float(_info[12])  # xInsertPointText
214 _yLoc = float(_info[13])  # yInsertPointText
215 _rotation = float(_info[14])  # rotationText
216 _alignment = int(float(_info[15]))  # alignmentText
217 _textWidth = float(_info[16])  # textWidth
218 _textHeight = float(_info[17])  # textHeight
219
220 _mtext = _msp.add_mtext(_strName, dxfattribs={'style':
    _textStyle}).set_location((_xLoc, _yLoc), _rotation,
    _alignment)
221 _mtext.dxf.width = _textWidth
```

```
222 _mtext.dxf.char_height = _textHeight
223
224 '''
225  Save the new file
226 '''
227 _doc.saveas(_dxfFileName)
228 status = updateStatus(_statusFileName, False, True, False)
229
```