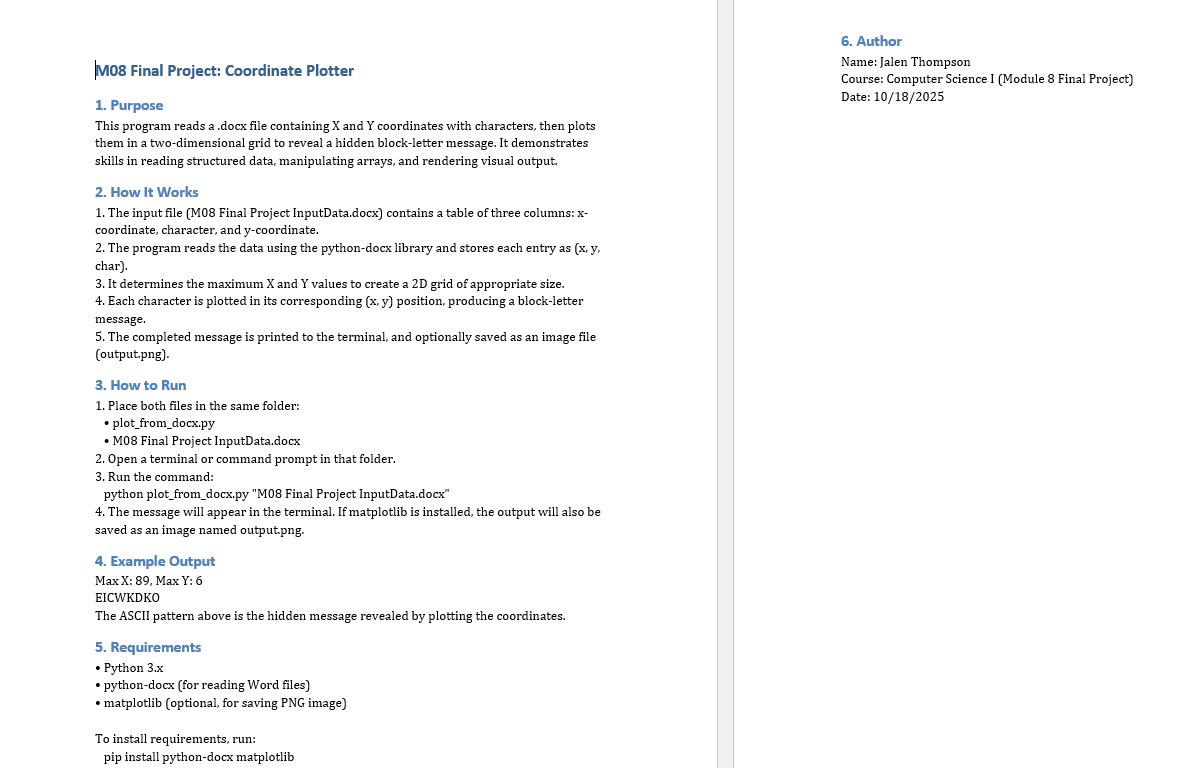
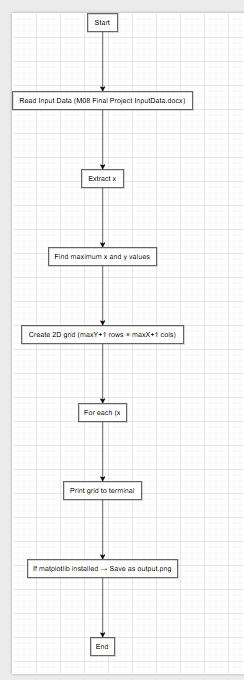
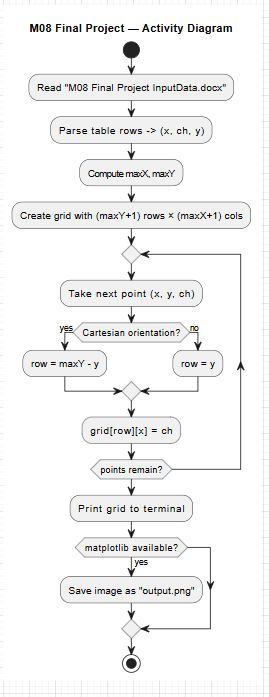
Jalen Thompson

1. ReadMe Documentation
2. Flowchart
3. UML Diagram 

4+5. Source Code (with documentation)

import sys

from collections import namedtuple

def read\_points\_from\_docx(path):

try:

from docx import Document

except ImportError:

print("Error: python-docx is not installed. Run: pip install python-docx")

sys.exit(1)

doc = Document(path)

Point = namedtuple("Point", "x ch y")

points = []

for tbl in doc.tables:

# Expect 3 columns: x-coordinate | Character | y-coordinate

# Skip header row if present

for i, row in enumerate(tbl.rows):

cells = [c.text.strip() for c in row.cells]

if len(cells) < 3:

continue

# Try to parse header detection

if i == 0 and ("x" in cells[0].lower() and "y" in cells[-1].lower()):

continue

# Cells may contain stray text like "x: 89" — pull first int out of each needed cell

def first\_int(s):

num = ''

sign = 1

found = False

for j, ch in enumerate(s):

if ch == '-' and (j + 1 < len(s)) and s[j+1].isdigit() and not found:

sign = -1

found = True

elif ch.isdigit():

num += ch

found = True

elif found and not ch.isdigit():

break

return sign \* int(num) if num else None

x = first\_int(cells[0])

y = first\_int(cells[-1])

# Character column might have a solid block, shaded block, or '#'

ch\_raw = cells[1]

ch = ch\_raw.strip()[:1] if ch\_raw.strip() else '#'

# Normalize common block symbols to printable

# (your console may not show ▀/█ consistently—'#' is safest)

if ch in ('', '\u00A0'):

ch = '#'

if x is None or y is None:

continue

points.append(Point(x, ch, y))

if not points:

print("No points parsed. Check the column order or export to CSV.")

sys.exit(1)

return points

def build\_canvas(points, cartesian=True):

maxX = max(p.x for p in points)

maxY = max(p.y for p in points)

# Make grid of spaces

grid = [[' ']\*(maxX+1) for \_ in range(maxY+1)]

for p in points:

r = (maxY - p.y) if cartesian else p.y

c = p.x

if 0 <= r <= maxY and 0 <= c <= maxX:

grid[r][c] = p.ch if p.ch.strip() else '#'

return grid, maxX, maxY

def print\_canvas(grid):

for row in grid:

print(''.join(row))

def save\_png(grid, out\_path="output.png", dpi=200):

# Render as an image using matplotlib (no special styles/colors)

import matplotlib.pyplot as plt

import numpy as np

# Convert to 0/1 mask for visibility

h, w = len(grid), len(grid[0])

mask = np.zeros((h, w))

for i in range(h):

for j in range(w):

if grid[i][j] != ' ':

mask[i, j] = 1

plt.figure(figsize=(w/10, h/10))

plt.imshow(mask, cmap="gray\_r", interpolation="nearest")

plt.axis('off')

plt.tight\_layout(pad=0)

plt.savefig(out\_path, dpi=dpi, bbox\_inches='tight', pad\_inches=0)

plt.close()

def main():

if len(sys.argv) < 2:

print("Usage: python plot\_from\_docx.py input.docx")

sys.exit(1)

path = sys.argv[1]

points = read\_points\_from\_docx(path)

grid, maxX, maxY = build\_canvas(points, cartesian=True)

print(f"Max X: {maxX}, Max Y: {maxY}")

print\_canvas(grid)

try:

save\_png(grid, "output.png")

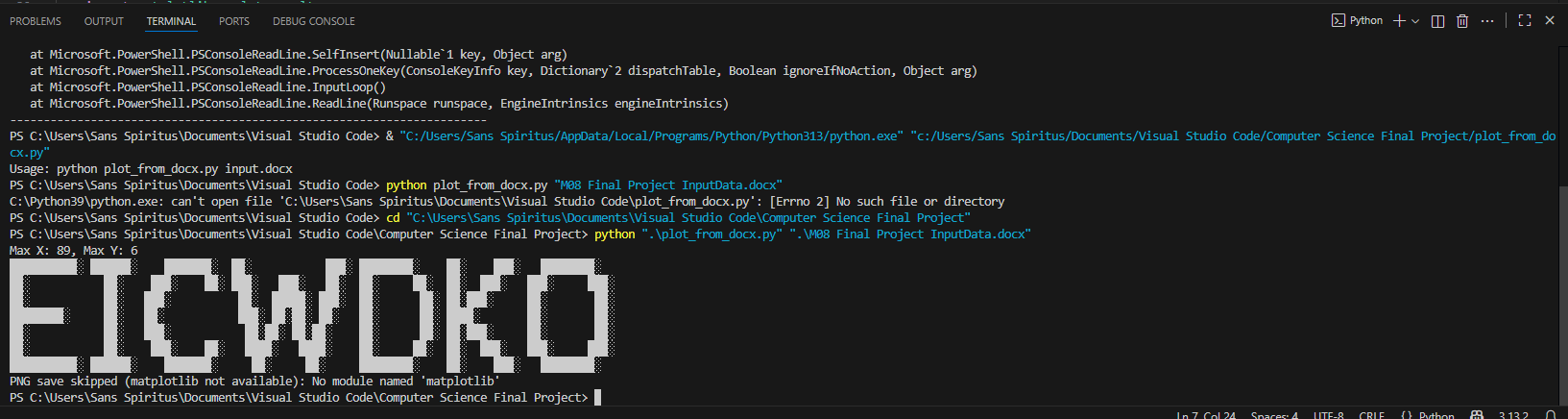
print("Saved image: output.png")

except Exception as e:

print("PNG save skipped (matplotlib not available):", e)

if \_\_name\_\_ == "\_\_main\_\_":

main()

6. 

7. https://github.com/Sans-Spiritus/Computer-Science-I