```
## This is the main.py file.

import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection
import pickle
from pathlib import Path
import time

from intersections import all_pairs_intersections, efficient_intersections


def read_pickle(filename):
    # Can sometimes read pickle3 from python2 by calling twice
    with Path.open(Path(filename), 'rb') as f:
        try:
            return pickle.load(f)
        except UnicodeDecodeError:
            return pickle.load(f, encoding='latin1')

def write_pickle(data, filename):  # NOTE - cannot pickle lambda or nested functions
    with Path.open(Path(filename), 'wb') as f:
        pickle.dump(data, f)

def read_segments(type, index):
    if type =="random":
        segments = read_pickle(f"./data/Random/random_{index}.pkl")
    elif type =="convex":
        segments = read_pickle(f"./data/Convex/convex{index}.pkl")
    elif type =="nonconvex":
        segments = read_pickle(f"./data/NonConvex/non_convex{index}.pkl")
    else:
        ValueError("No such type exist")

    return segments

def visualize_lines_and_intersections(L1,L2, points):
    fig, ax = plt.subplots()
    ax.add_collection(LineCollection(L1,  linestyle='solid', zorder=0))
    ax.add_collection(LineCollection(L2,  linestyle='solid', color="purple",
zorder=1))
    ax.scatter(points[0], points[1], color="red", marker="x", zorder=2)
    ax.set_title('Line collection and their intersections')
    plt.show()

#Create random line segments
def main():
    L1 = read_segments("random", 50)
    L2 = read_segments("convex", 1)
    points =  all_pairs_intersections(L1, L2)
    visualize_lines_and_intersections( L1, L2, points)
    L1 = read_segments("nonconvex", 11)
    L2 = read_segments("convex", 3)
    points =  all_pairs_intersections(L1, L2)
    visualize_lines_and_intersections( L1, L2, points)

    L1 = read_segments("nonconvex", 2)
    L2 = read_segments("convex", 10)
    points =  all_pairs_intersections(L1, L2)
    visualize_lines_and_intersections( L1, L2, points)

    times = {}
    times_efficient = {}
    for N in range(0, 500, 10):
        L1= read_segments("random", N)
        start = time.time()
        points1 = all_pairs_intersections(L1, L1)
        end = time.time()
        times[N]  = end-start

        start = time.time()
        points2 =  efficient_intersections(L1, L1)
        end = time.time()
        times_efficient[N]  = end-start
        if len(points1[0])!=len(points2[0]):
            print(f"Efficient intersections for {N} is not implemented correctly!")

    plt.plot(times.keys(), times.values())
    plt.plot(times_efficient.keys(), times_efficient.values())
    plt.legend(labels = ["All pairs Algorithm", "Efficient Algorithm"])
    plt.show()

if __name__== "__main__":
    main()
```