Project Structure – Single objective optimization

1. Build – contains compiled cython (C + python) files

2. Data – Contains data files of benchmark problems and inspired from real-life production problems
    a. Given_data : Production problems
        i. Job_info.csv - information of job, operations, demand (no of pieces to produce), operation sequence, required machines
        ii. Machine_info.csv - information of machine id, machine processing speed, setup time. (Note: setup time is not considered in experiments)
        iii. Sequence_dependecy_matrix – operation sequence dependency matrix (not considered)
    b. Benchmark – directory for benchmark problem files

3. Example_output – Used to store schedule output from algorithm in excel format and HTML files that contains plots of algorithms performance.

4. Optimizer – Repository of implemented hybrid algorithm
    a. Genetic_alg – Contains implemented genetic algorithm files
        i. _ga_helpers.pyx – helper file for the Genetic algorithm (used cython for faster computation -> heap structure in C)
        ii. Genetic_alg.py - Contains developed Genetic algorithm code
    b. Simulated_annealing - Contains implemented simulated annealing algorithm (not used in hybrid approach)
        i. _generate_neighbor.pyx - cython file contains function to generate neighbourhood of seed solution
        ii. Simulated_annealing.py - Contains developed Simulated Annealing algorithm code
    c. Solution – Contains files to generate solutions for the algorithms, and to create schedule in time-framed format.
        i. _makespan.pyx - contains code to evaluate makespan of the solution or schedule (not used)
        ii. _schedule_creator.py - contains code to generate time-framed schedule in excel format
        iii. factory.py - contains code to generate initial population or set of solutions based on randomess and dispatching rules. (Chromosome generation)
        iv. Solution.py - contains code to decode the chromosome representation and to evaluate objective function values
    d. Tabu_search – contains implemented tabu search files

i. _generate_neighbor.pyx - cython file contains function to generate neighbourhood of seed solution
ii. Tabu_search.py - Contains developer tabu search code

e. Template – contains template for benchmark result plots in HTML format
f. Benchmark_plotter.py - Python code to plot algorithm performance results
g. Coordinator.py - Important file in the developed algorithm code. Gets hyperparameter and sequential ensemble information from main executioner and executes Genetic algorithm and parallel tabu search in the mentioned order.
h. Data_fjs.py - Python code to extract the inspired real-life problem's data
i. Data_normal_job_shop.py – Python code to extract data from given benchmark problem's file
j. Data.py - Acts as super class for data extraction files
k. Exception.py - Custom exception code
l. Job.py - Python code to initialize jobs based on extracted data
m. Operation.py - Python code initialize operations based on extracted data
n. Utility.py - Helper functions

5. Main.py - Starting point of algorithm execution

6. Setup.py - Code and information about software installation (required python packages)