Dataset

The dataset used in this assignment are the **Numpy arrays** of the followings:

- Teacher
- Courses
- Sections
- Duration
- Days
- Rooms

```
8   # Global Arrays Dataset
9   Teachers = np.array(
10      ["Sir Ahmed Nawaz", "Sir Umair Arshad", "Mam Labiba", "Mam Humera", "Mam Amna Irum", "Mam Irum Inayat",
11       "Sir Kifayatullah", "Sir Ahsanullah", "Mam Noorul ain", "Mam Amna Basharat", "Sir Ejaz", "Sir Hassan", "Mam Rabia",
12       "Mam Sarah", "Sir Ahmed Ali", "Mam Shehr Bano", "Sir Usama Rasheed", "Dr Aqeel Ejaz", "Sir Anas Zafar"])
13  Courses = np.array(["ITC", "AI", "HCI", "SMD", "TM", "COAL", "CP", "DB", "AP", "CA", "SE", "FYP-I", "FYP-II"])
14  Sections = np.array(["A", "B", "C", "D", "E", "F", "G"])
15  Duration = np.array(["0830", "0930", "1030", "1130", "1230", "0130", "0230", "0330", "0430"])
16  Day = np.array(["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"])
17  Rooms = np.array([401, 402, 403, 101, 102, 103, 201, 202, 203, 301, 302, 303, 501, 502, 503])
```

Time slot

Using the concepts of the **Genetic Algorithm**, the **timeslot** is actually a **Gene** which includes teacher's name, course's name, section, duration, day and room.

So the **three** main concepts uses in this assignment are:

- **Gene** which is actually a **timeslot** generated randomly
- **Chromosome** which is actually a **timetable** containing multiple genes
- **Population** which comprises of **multiple chromosomes**

Hard & Soft Constraints

The **timetable** (chromosome) generated fulfill the following hard and soft constraints:

- No teacher can hold two classes at the same time
- No section can listen for two classes at the same time
- No classroom can receive two classes at the same time
- No teacher can hold three consecutive classes
- No section can hold three consecutive classes.
- There will be no class before 8:30 am and after 5:00 pm.
- University will remain close as there will be no class on weekends (Sat, Sun)
- There will be no class from 1-2 on Friday.

Working of Genetic Algorithm

1. First of all, **population** is generated of the length of No of chromosomes
2. After that, fitness of each chromosome is calculated which counts number of all conflicts in accordance with the constraints
3. The **fitness** is calculated by **incrementing 1 to overall score**
4. A while loop runs until the best chromosome (timetable) is not found
5. An inner loop runs till the required number of children for that specific iteration are generated
6. Then two parents are selected using the **roulette wheel selection** in order to run the **crossover** to generate two children
7. Then those children are mutated using the **mutation** and are added to the new population
8. This goes on until the inner loop ends

9. After that the fitness of the new population is calculated and the old population is replaced with the new population if fitness of new population is greater than minimum of old population

Methods

fill_populations_array () - Fill the population array

return_random_slot () - Generate random slot (gene)

return_choromosomes_array () - Generate chromosome array

printing_population_array () – Print the population array

calculate_fitness (chromosome) - Calculate fitness of a single chromosome

roulette_wheel_selection (population_array) - Select two best Chromosomes

crossover (cromosome_1, cromosome_2) - Generate two children from the given parent chromosomes

exchange_lowest_fitness (chromosome_1, chromosome_2) - replace old chromosomes with the new chromosomes if fitness of new chromosomes is greater than minimum of old chromosomes

mutation (chromosome) - perform mutation on a given chromosome with a selected mutation rate (in this case 3 %)

return_best_chromosome () - returns chromosome with best fitness from entire population

main () – runs the entire above operations