INTELLIGENT TIME-TABLE PREPARATION

END TERM PROJECT

BY

PRATIK- 11808492

RITIK KUMAR- 11808486

SAMA PRANAY SAI GANESH- 11802220

ANSHUL KUAMR SINGH- 11814863

SECTION: K18KY



DEPARTMENT OF INTELLIGENT SYSTEM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

LOVELY PROFESSIONAL UNIVERSITY, JALANDHAR

APRIL 2019-2020

STUDENT DECLARATION

This is to declare that this report has been written by me/us. No part of the report is copied from other sources. All information included from other sources have been duly acknowledged. I/We aver that if any part of the report is found to be copied, I/we are shall take full responsibility for it.

PRATIK

ROLL:54

RITIK KUMAR

ROLL: 53

SAMA PRANAY GANESH

ROLL: 49

ANSHUL KUMAR SINGH

ROLL:36

PLACE: Lovely Professional University, Jalandhar

DATE: 07-April-2020

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| **TITLE** | **PAGE NO.** |
| 1.STUDENT DECLARATION | 2 |
| 2.BONAFIDE CERTIFICATE | 4 |
| 3. BACKGROUND AND OBJECTIVE OF THE PROJECT | 5-6 |
| 4. DESCRIPTION OF THE PROBLEM | 7-12 |
| 4.1 ABSTRACT | 7 |
| 4.2 PROBLEM | 7 |
| 4.3 CONSTRAINST | 7-8 |
| 4.4 SOLUTIONS | 8 |
| 4.5 ALGORITHM | 8-9 |
| 4.6 OUTPUT | 9-12 |
| 5. WORK DIVISION | 13-14 |
| 6. IMPLEMENTATION OF CODE | 15-17 |
| 7. TECHNOLOGY AND FRAMEWORK | 18 |
| 8. SWOT ANALYSIS | 19 |

BONAFIDE CERTIFICATE

Certified that this project report “INTELLIGENT TIME-TABLE PREPRATION” is the bonafide work of PRATIK, RITIK KUMAR, SAMA PRANAY SAI GANESH AND ANSHUL KUMAR SINGH who carried out the project work under my supervision.

Supervisor: Ms. Shabnam

ID: 25660

Dvision of Admission

**BACKGROUN AND OBJECTIVES OF THE PROJECT**

The class timetabling problem is a scheduling algorithm with great interest and implications in the fields of operational research and artificial intelligence. The problem was first studied by Gotlieb, who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students and one teacher, such that the combination of teacher and students can be chosen freely. Dynamic changes in the context of timetabling problems, had started to be studied. Because of the size of real problem, almost all effective solutions are heuristic in nature, and do not guarantee optimality. Among the well-known results there are several algorithms that deal with various cases of the problem settings. While setting a timetable, importance is given to effective utilization of resources such as the classroom, the teacher, etc. This becomes a very tedious task which needs to be addressed at least once a year by every academic institute. Most institutes deal with this problem manually, i.e. a trial and error method are used to set a timetable. The general timetabling problem comes in many different guises including nurse rostering (e.g. Cheang et al, 2003, Burke et al. 2004), sports timetabling (e.g. Easton, Nemhauser and Trick, 2004), transportation timetabling (e.g. Kwan, 2004) university timetabling (Schaerf, 1999, Burke and Petrovic, 2002, Petrovic and Burke, 2004), etc. A more indirect approach can also be seen in instruction scheduling. Due to the combinatorially explosive nature of the problem, enumeration and other deterministic methods.

**Background:** Timetabling is known to be a non-polynomial complete problem i.e. there is no known efficient way to locate a solution. Also, the most striking characteristic of NP-complete problems is that, no best solution to them is known. Hence, in order to find a solution to a timetabling problem, a heuristic approach is chosen. This heuristic approach, therein, leads to a set of good solutions (but not necessarily the best solution). In a general educational timetabling problem, a set of events (e.g. courses and exams, etc) are assigned into a certain number of timeslots (time periods) subject to a set of constraints, which often makes the problem very difficult to solve in real-world circumstances. In fact, large-scale timetables such as university timetables may need many hours of work spent by qualified people or team in order to produce high quality timetables with optimal constraint satisfaction and optimization of timetable’s objectives at the same time.

**Motivation and Outcomes:** Heuristic optimization methods are explicitly aimed at good feasible solutions that may not be optimal where complexity of problem or limited time available does not allow exact solution. Generally, two questions arise (i) How fast the solution is computed? and (ii) How close the solution is to the optimal one? Trade-off is often required between time and quality which is taken care of by running simpler algorithms more than once, comparing results obtained with more complicated ones and effectiveness in comparing different heuristics. The empirical evaluation of heuristic method is based on analytical difficulty involved in the problem’s worst-case result. In its simplest form the scheduling task consists of mapping class, teacher and room combinations (which have already been pre- allocated) onto time slots. One possible approach is as follows: We define a tuple as a combination of identifiers such as class, teacher and room, which is supplied as an input to the problem. The problem now becomes one of mapping of tuples onto period slots such that tuples which occupy the same period slot are disjoint (have no identifiers in common). If tuples are assigned arbitrarily to periods, then in anything but the most trivial cases, several clashes will exist. We can use the number of clashes in a timetable as an objective measure of the quality of the schedule. Thus, we adopt the number of clashes as the cost of any given schedule. It is simple to measure the cost of a schedule. For each period of the week, we make a count of the number of occurrences of each class, teacher and room identifier. The cost of the entire timetable is the sum of each of the individual costs. This procedure is discussed in more detail in Abramson [21]. The proposed algorithm aids solving the timetabling problem while giving importance to teacher availability. This algorithm uses a heuristic approach to give a general solution to school timetabling problem. It takes the user input of a number of subjects, number of teachers, subjects every teacher takes, number of days in a week for which the timetable needs to be set, number of time slots in a day and the maximum lectures a teacher can conduct in a week. It initially uses randomly generated subject sequence to make a temporary timetable. While generating this sequence, care is taken to avoid repetition of subjects over a day. After this, the teacher availability for each of the subjects allocated for the respective slot is checked. Every time a teacher is available for the subject at the allocated slot, the subject and the teacher are entered the output data structure and marked as final. Before the allocation of this subject to the output data structure, a check is also conducted on the number of maximum lectures a teacher can conduct. If the teacher has been allocated more than the allowed maximum lectures the subject is moved into a Clash data structure.

**Goals and Objectives:** The goal of the algorithm to generate a time-table schedule automatically is satisfied. The algorithm incorporates a number of techniques, aimed to improve the efficiency of the search operation. It should also address the important hard constraint of clashes between the availability of teachers. The non-rigid soft constraints i.e. optimization objectives for the search operation should also be effectively handled. Given the generality of the algorithm operation, it can further be adapted to more specific scenarios, e.g. University, examination scheduling and further be enhanced to create railway timetables. Thus, through the process of automation of the time-table problem, many an-hours of creating an effective timetable have been reduced eventually. The most interesting future direction in the development of the algorithm lies in its extension to constraint propagation. When there is a value assigned to a variable, such assignment can be propagated to unassigned variables to prohibit all values which come into conflict with the current assignments. The information about such prohibited values can be propagated as well.

**DESCRIPTION OF PROJECT**

**Abstract**

This project implements one of possible solutions for generating university schedule using genetic algorithm. The proposed solution is based on methods of evolutionary computing, uses *(1+1) evolutionary strategy* and *simulated hardening*. The success of solution is estimated on fulfilment of given constraints and criteria. Results of testing the algorithm show that all hard constraints are satisfied, while additional criteria are optimized to a certain extent.

**Problem**

The assignment is to find generic solution that will facilitate generating schedule for university/school (this specific problem is adjusted to Faculty of Lovely Professional University.).

Each class on faculty is represented as block (lasts arbitrary number of hours, mostly from 1 to 4). For conducting every class required are: *teacher*, *classroom*, *start time*, *duration* and *groups* which attend the class. It is also know in advance which groups attend which class and all classrooms are the same size (each group can fit to a classroom).  
Teaching is done on faculty from 9AM until 9PM on each work day.

**Input data** for each class are *teachers’ name*, *subject*, *type*, *duration* and list of *allowed classrooms*.  
**Output data** are classroom and time for each class. Time is determined by day (Monday to Friday) and start hour of the class.

**Constraints**

1. Resources can not overlap timewise
   * No teacher can hold two classes at the same time
   * No group can listen for two classes at the same time
   * No classroom can receive two classes at the same time

Note: under the term "same time" is not meant only at the beginning of the class, it should be taken into account the duration of the class. If the resource is busy at the moment T1 and the class lasts t1, then the resource can only be re-occupied at the moment T2 = T1 + t1.

1. Class should take place in one of the allowed classrooms
2. If the subject has several forms of teaching, the preferred order for each group is the *lectures*, *exercises*, and *laboratory exercises*.

Constraints 1 and 2 must be met, while the 3rd limit is "soft" and allowed to be violated.  
Additional criteria for estimating solution (used also for cost function):

* Fulfill hard constraints (1 & 2)
* Fulfill soft constraints (3)
* Minimize total "idle" for each group (eliminating pause between classes)
* Minimize total "idle time" for each teacher (elimination of pause between classes)
* Provide one hour on a teaching week where no one has classes

**Solution**

The algorithm for the timetable is represented as table with its columns as class, rooms, departments, courses and instructor and rows allowed times for classes, while the elements of the table are the specific class held in proper classroom and given time.

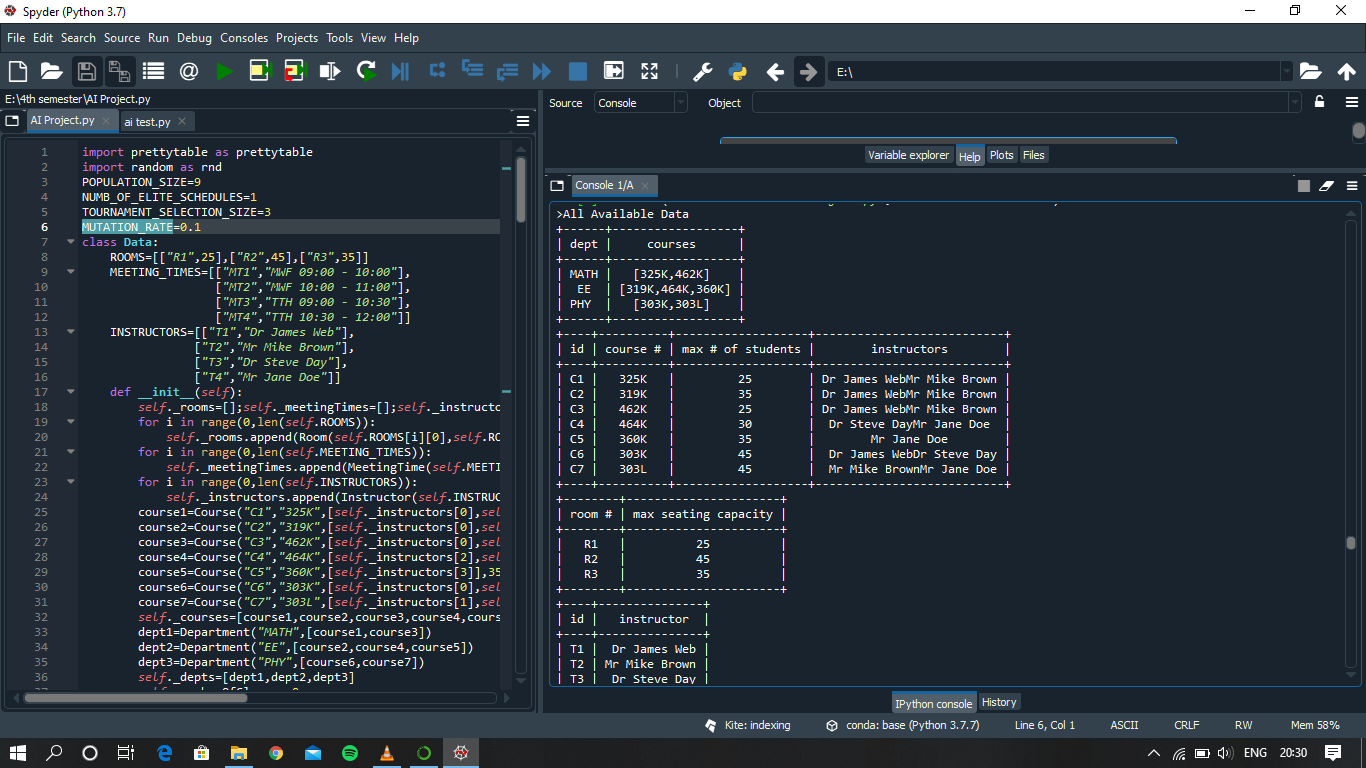
Element of matrix can be empty if there is no teaching held in given time and classroom.   
Index of the class is determined by: *teachers’ name*, *subject*, *type* (lectures, exercises or laboratory exercises), *duration*, *allowed classrooms* and *groups*. Because of different duration of classes, the same indexes appear in the table multiple times and are always in vertical blocks (they take consecutive fields in the same column).

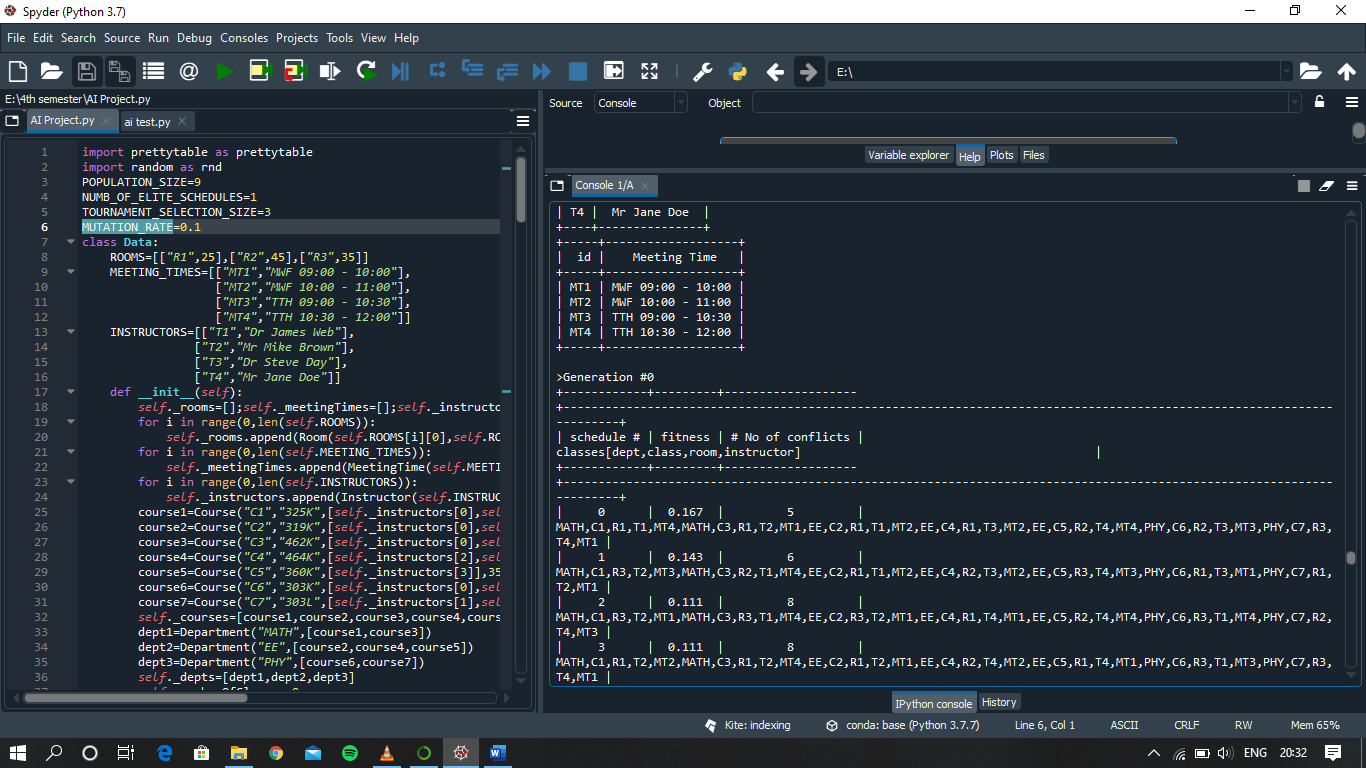
Algorithm

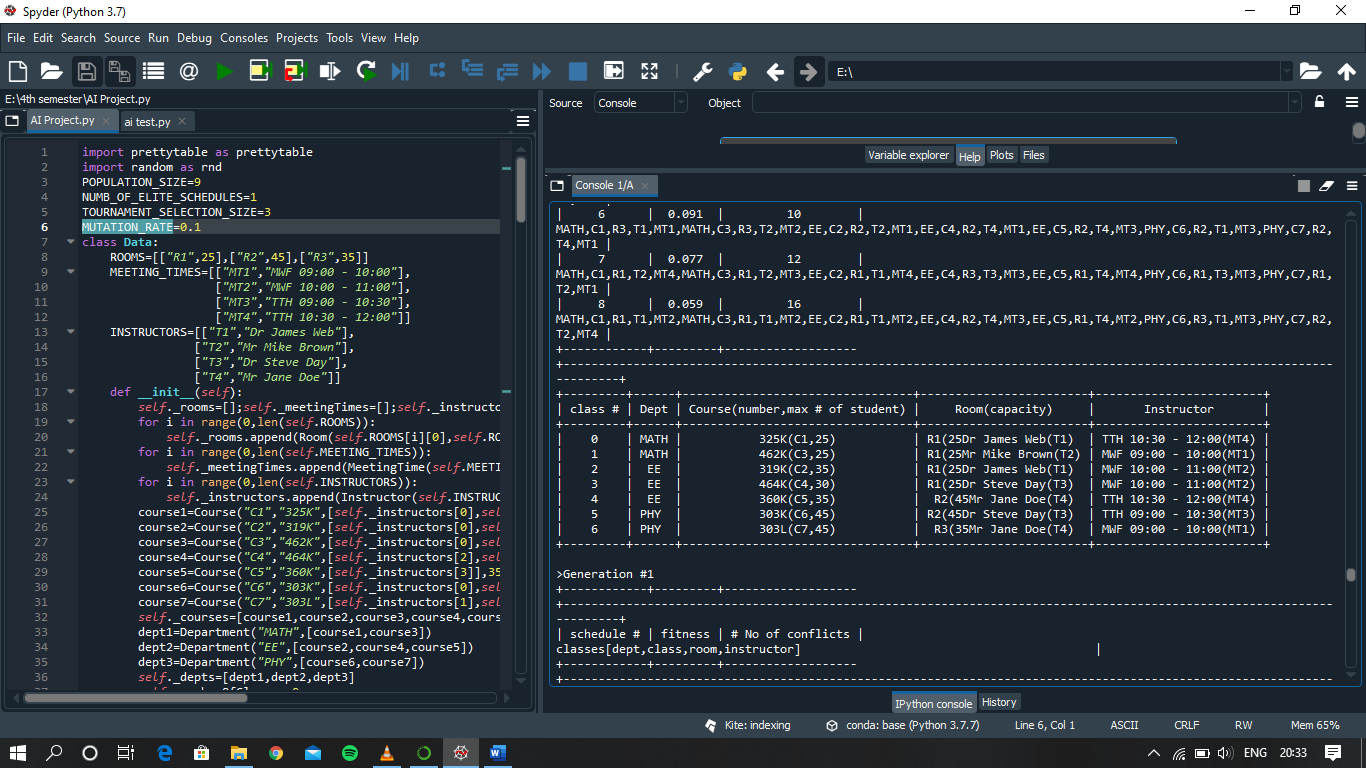
The algorithm is comprised in the following way:

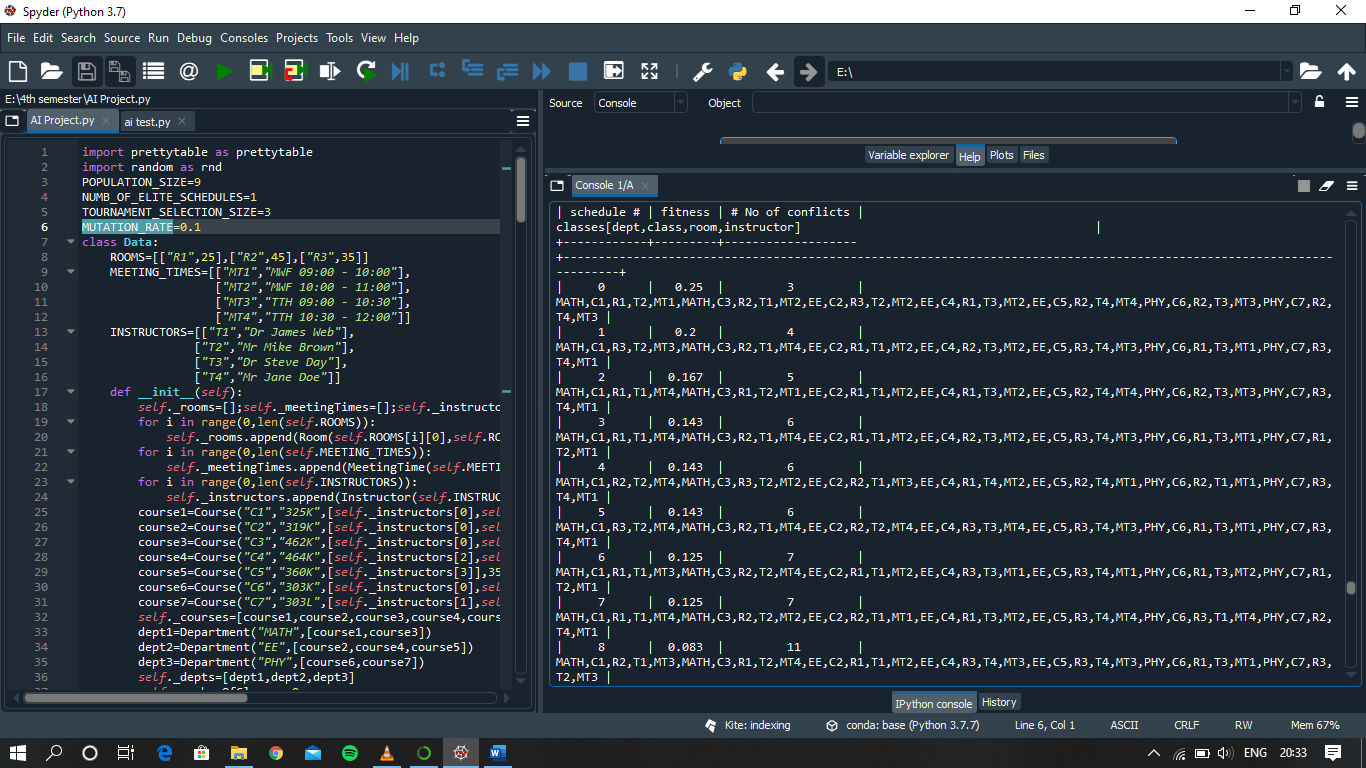
1. **Loading and processing the data**  
   Load all the data from the input file and process it so that each class in the timetable has a list of acceptable classrooms instead of a type of classroom.
2. **1+1 evolutionary strategy (hard constraints)**  
   This is the phase of the algorithm were we generate an arbitrary number of schedules that try to optimize for hard constraints, hence this is shotgun hill-climbing. They way we do this for each of these schedules is: Firstly, we generate a completely random timetable. Then, we use the 1+1 evolutionary strategy to improve out solution. The way in which find a neighbouring solution (one that we compare with the current one) is by using a mutation operator. As part of the mutation, we search for all classes that violate some hard constraint (with any resource) and we randomly choose one of them. Then we transfer that class in an unoccupied time frame, in one of the allowed classrooms for that class. If there exists no such combination of time frame and classroom, we transfer the class into a random time frame in one of the allowed classrooms. Also, we are careful of not accidentally placing a class to overlap days (start Monday evening, finish Tuesday morning) which is possible since we are representing the whole week as an array. If there are no more classes that violate hard constraints, we choose a random one and transfer it to an unoccupied time frame (in this phase we also optimize for the soft constraint of preferred order). Of all of the timetables we get by shotgun hill-climbing (recommended number is 5) we choose the best one (the one with the lowest cost function) to propagate to the next phase. Also, the recommended number of iterations for the 1+1 evolutionary strategy is 5000.
3. **1+1 evolutionary strategy (soft constraints)**  
   In this phase we optimize for soft constraints only, but we are wary of not violating any hard constraints in the process. The way we do this is similar as in the previous step. We run 15000 iterations of 1+1 evolutionary strategy on the previously obtained timetable. The difference compared to the previous step is that we use a different cost function (which takes into account all of the soft constraints, as well as the hard ones). Furthermore, we mutate the chromosome in a different way: We pick two classes at random and swap their places and assigned times. Besides this, we check if the two classes are compatible for swapping (if they use the same type of classrooms). After 15000 iterations, we assert that the algorithm has converged, and we have our solution.
4. **Saving the solution and displaying statistics**  
   In the final step, we save the obtained schedule in a output data and we display all the relevant metrics regarding our solution. We also show metrics individually for all subjects, groups and professors.

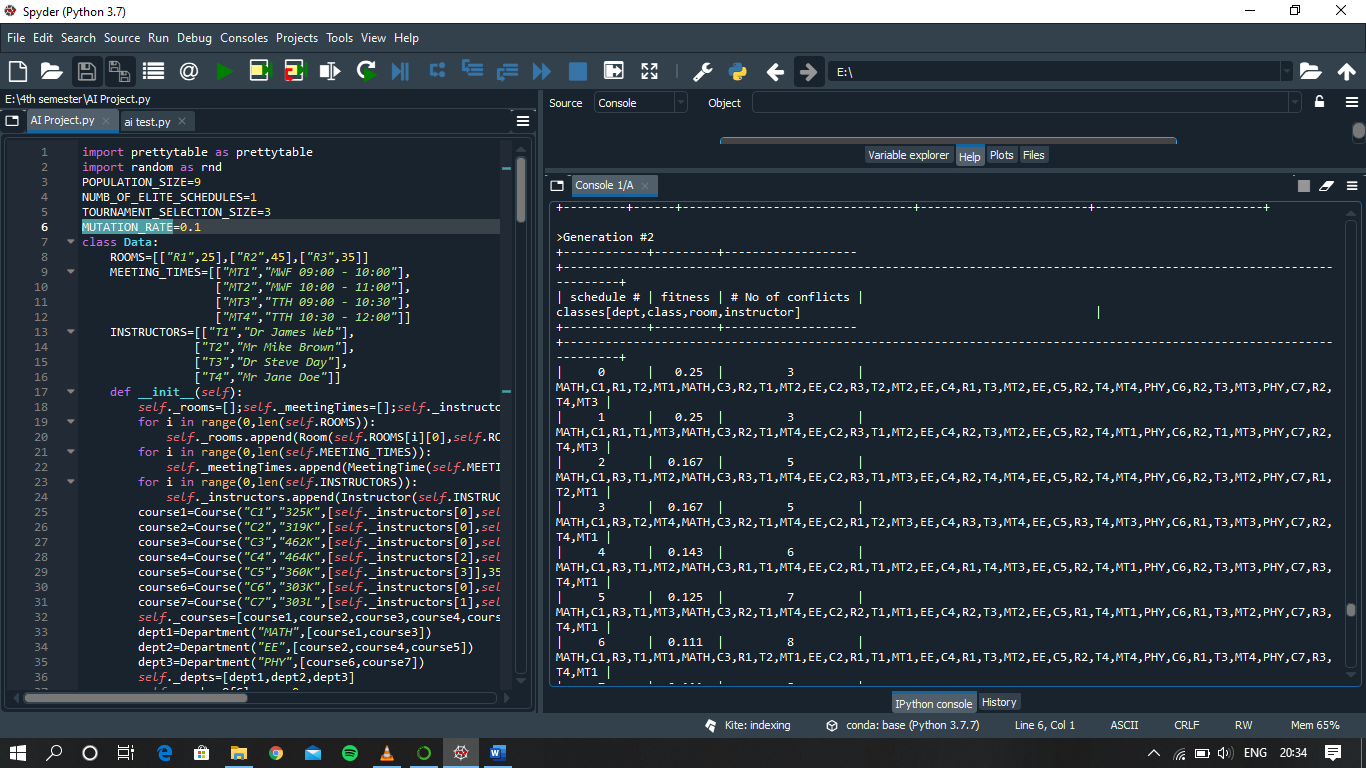
**OUTPUTS:**

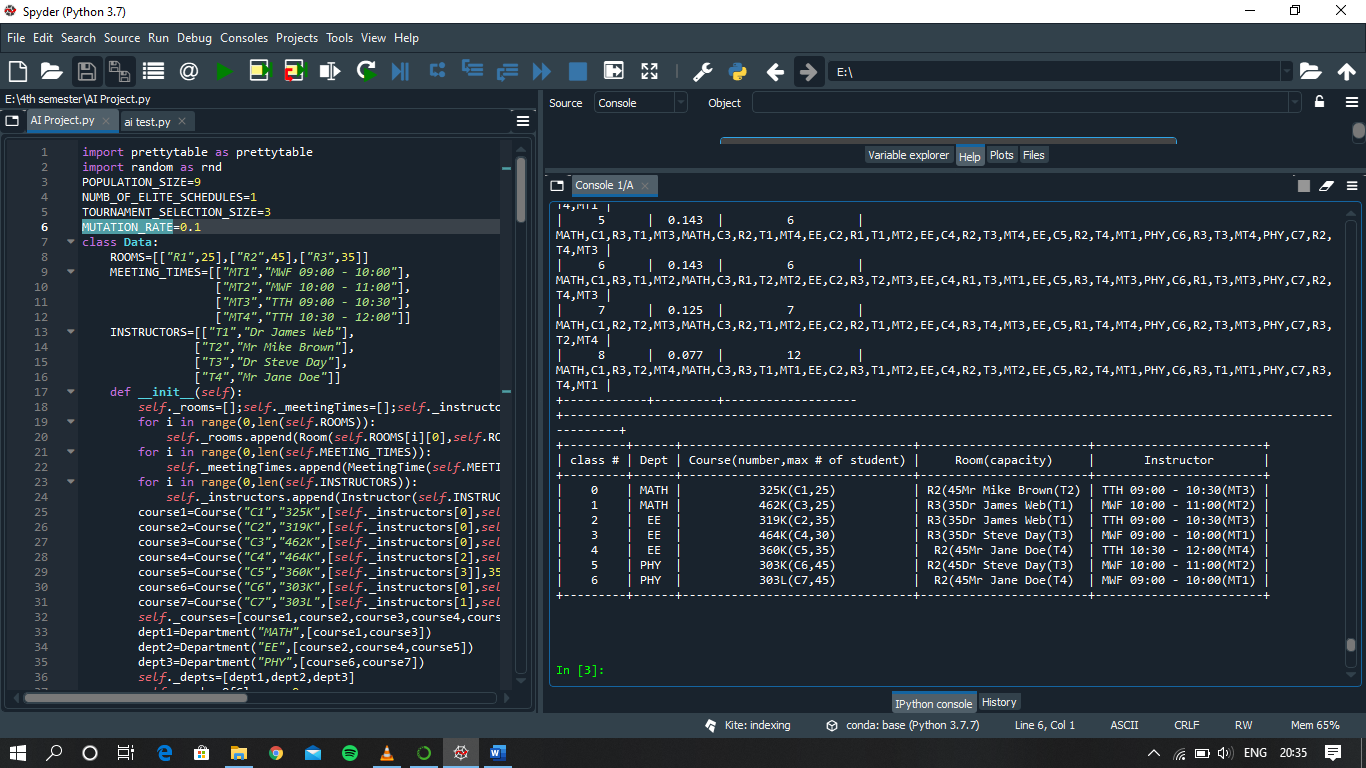












**WORK DIVISION**

**CONTRIBUTION IN PROJECT:**

1. **Group Member: Pratik**

We have followed a total object-oriented programming by making a respective classes for each of the factors that were essential in making our time table generator project.

The three main modules/classes were handled by Pratik named “Data”, “Genetic Algorithm”, and “DisplayMgr” classes.

The **Data Class** consist all the data which are required in generating the timetable. The data are as follows: Instructors, Rooms, Meeting time, Courses and departments. All the data are first stored in list and then I have generated getter function for all these so that it can return the the given specific data.

The **Genetic Algorithm** class handles the main crux of the program also that main part of genetic algorithm that is mutation and crossover. The mutation has the job to create solutions till a solution of required fitness is obtained.

The **DisplayMgr class** dealt with all the output generated. Since we had several state space till we reach our final result it was mandatory to huge amount of data we had should be dealt properly and have proper tabular form through which we could easily see the conflict showing us the reason why the processor moved to next generation in order to find the solution.

1. **Group Member : Ritik Kumar**

Two modules/classes were handled by Ritik named Schedule class and “Class” class.

The **Class** class dealt with returning the assigned values of Room, Instructor, Courses, Department and the Meeting time every time the mutation process generated a new solution.

The **Schedule class** also had main role in our time table preparing algorithm which had the job of calculating the fitness of each new solution generated. It assigned random values of Meeting time, instruction and rooms and calculated the fitness of the solution which tells about how much conflict that we can have in our solution.

The formula for calculating the fitness of the solution was also proposed by Ritik which is

1/ (1.0\*no. of conflicts +1).

1. **Group Member : Sama Pranay Sai Ganesh**

The class Population, Courses and Meeting time was handles by Ganesh.

The **Population class** dealt with what information needs to be provided during the process of mutation in order till a required state is not reached.

The **Meeting Time class** defines the meeting time available in our time table algorithm and code had getter function for returning the time table if any instances of the class required or demanded the information

The **Courses class** also maintains the courses that are available in our code and has the getter function which returns the courses oi any instance demanded it.

1. **Group Member : Anshul Kumar Singh**

The classes Instructor, Departments and the main function was handled by Anshul.

The **Instructor class** defines the instructors available in our time table algorithm and code had getter function for returning the time table if any instances of the class required or demanded the information

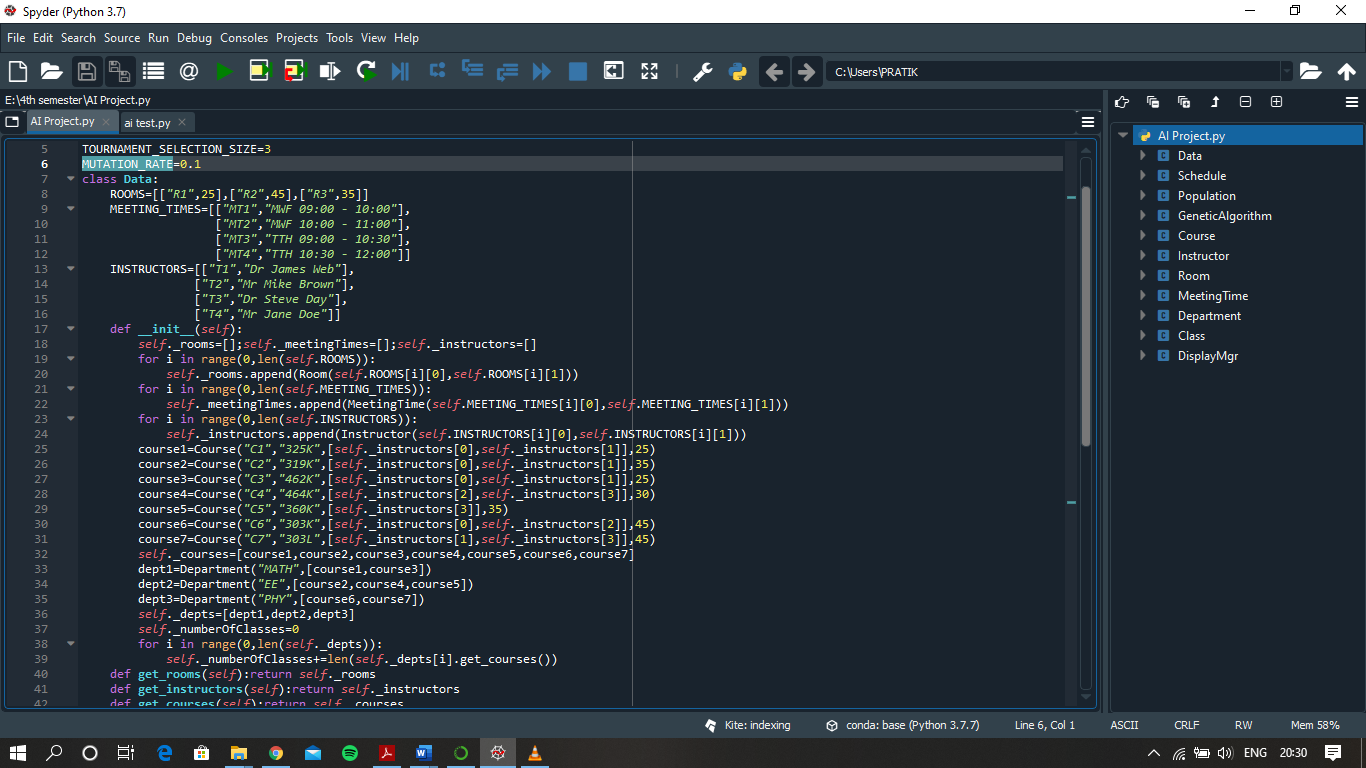
The **Department class** also maintains the courses that are available in our code and has the getter function which returns the departments if any instance demanded it.

The main function involves the overall integration of all classes/modules which include proper object creation of classes and proper function call. It includes lambda expression and while statements for proper function of mutation and crossover process.

**IMPLEMENTATION OF PROJECT**

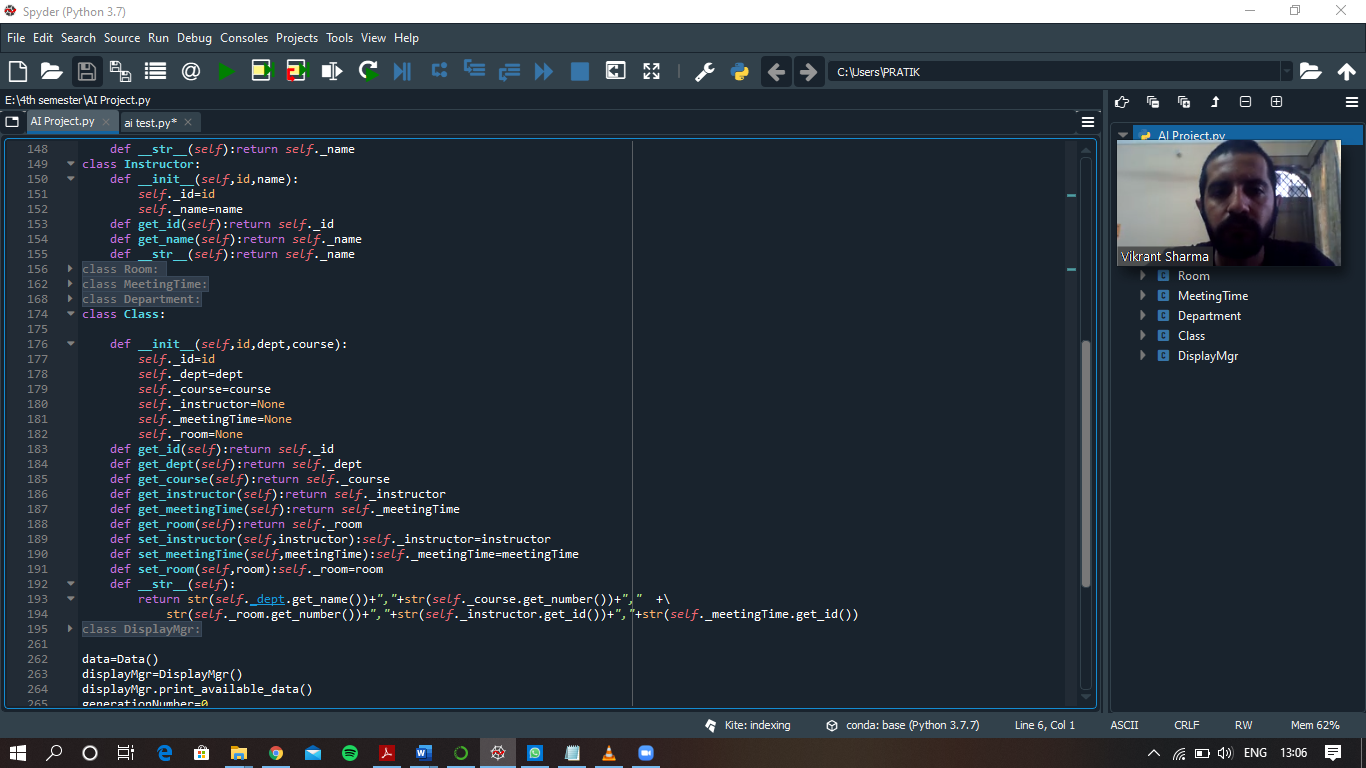
**class: Data**

It is the class with data of Instructors,Room, Meeting Time, Courses and Departments.



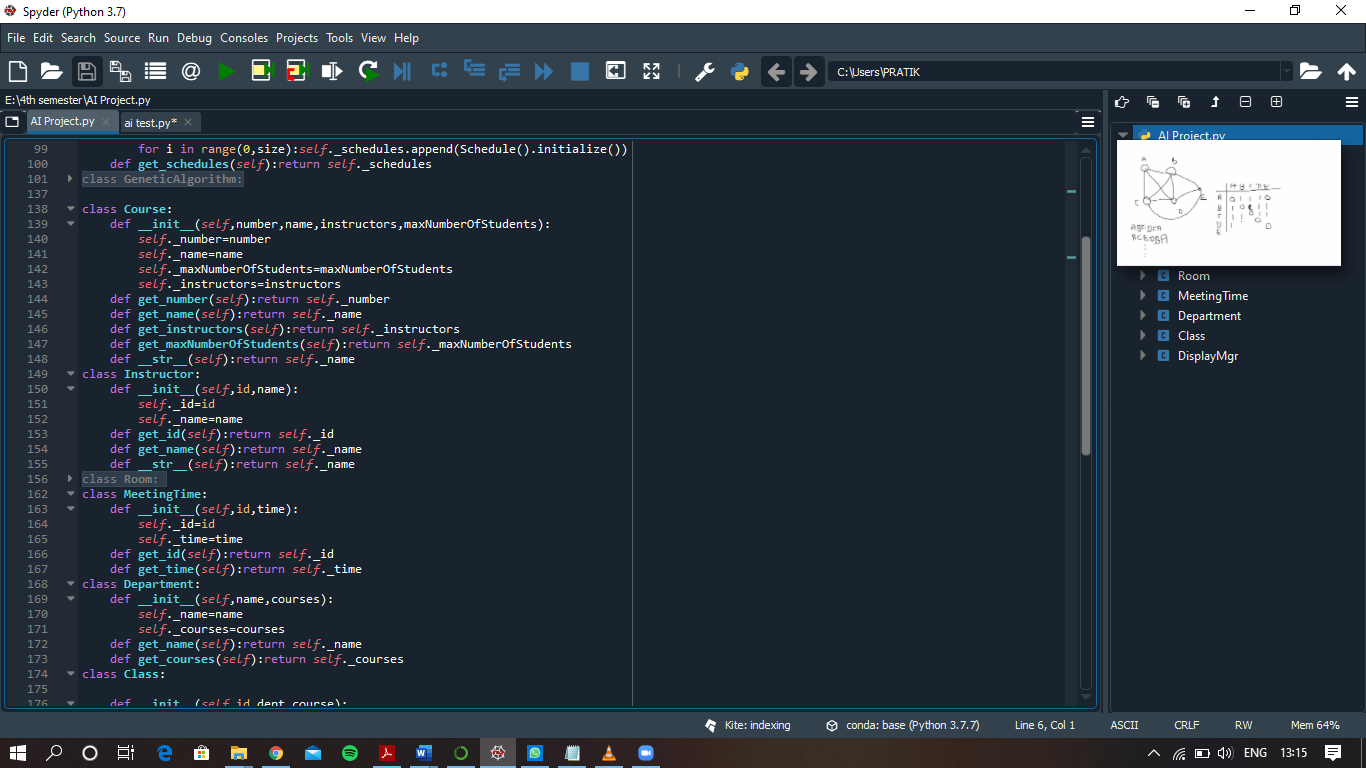
**class : Class**

It returns the assigned values of Room, Instructor, Courses, Department and the Meeting time every time the mutation process generated a new solution.



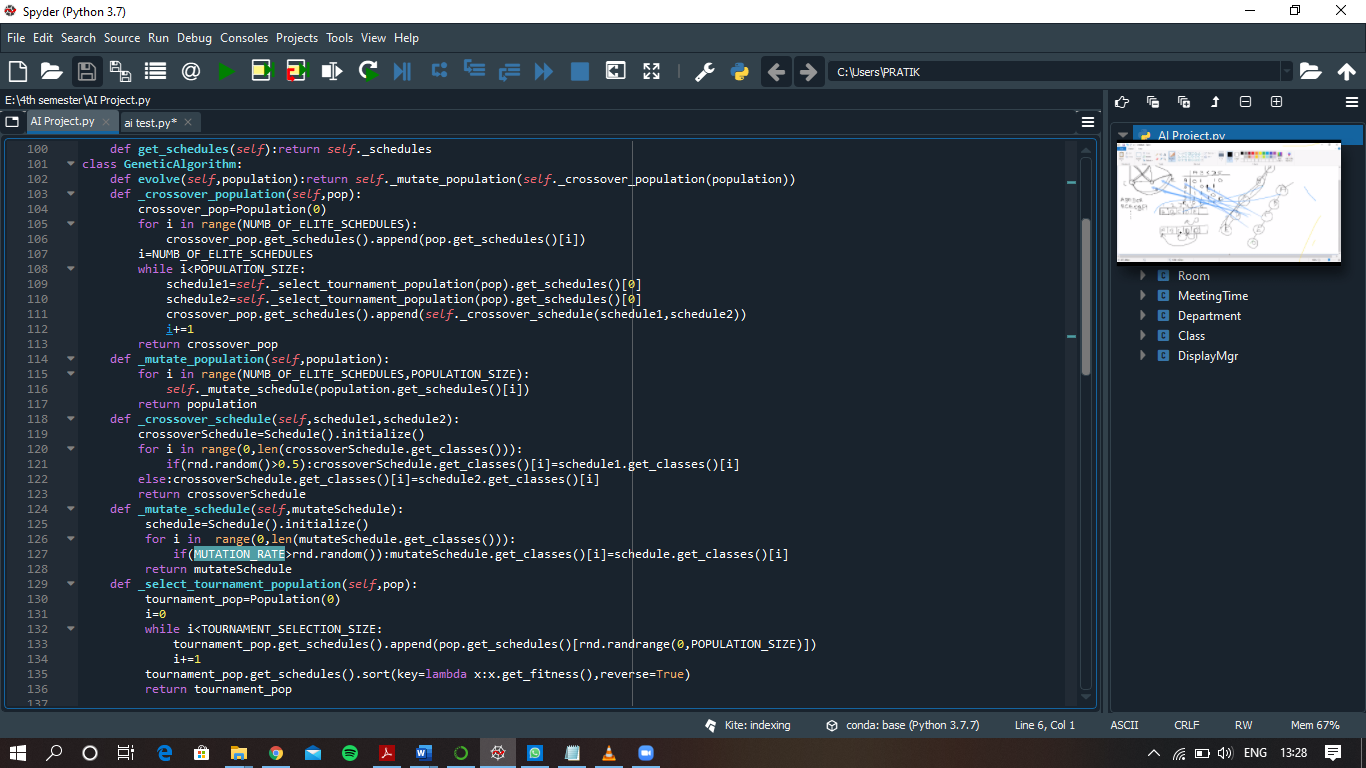
**Class : Courses, Instructor, Room and Department**

These all classes have Courses, instructros , rooms and department information respectively and getter function for returning the values.



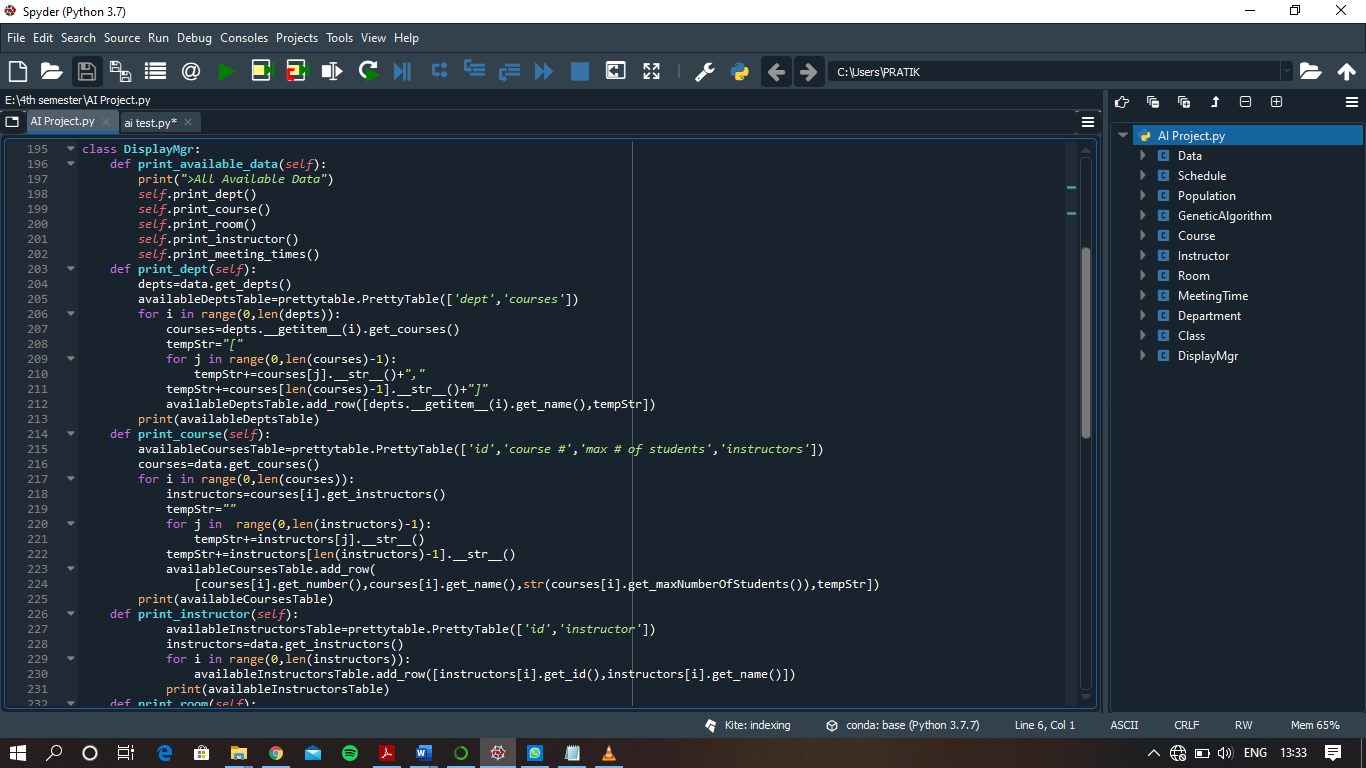
**Class :Genetic Algorithm**

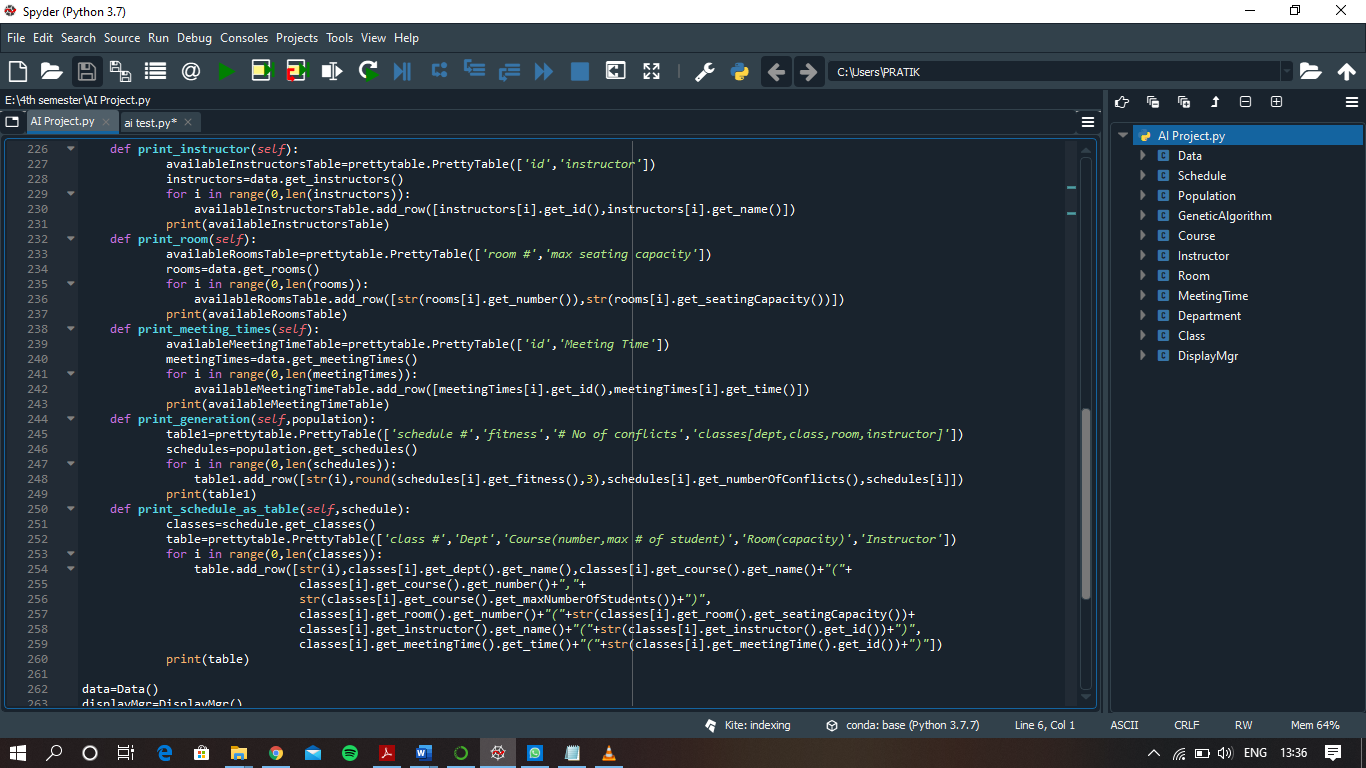
Perform Mutaion and Crossover.



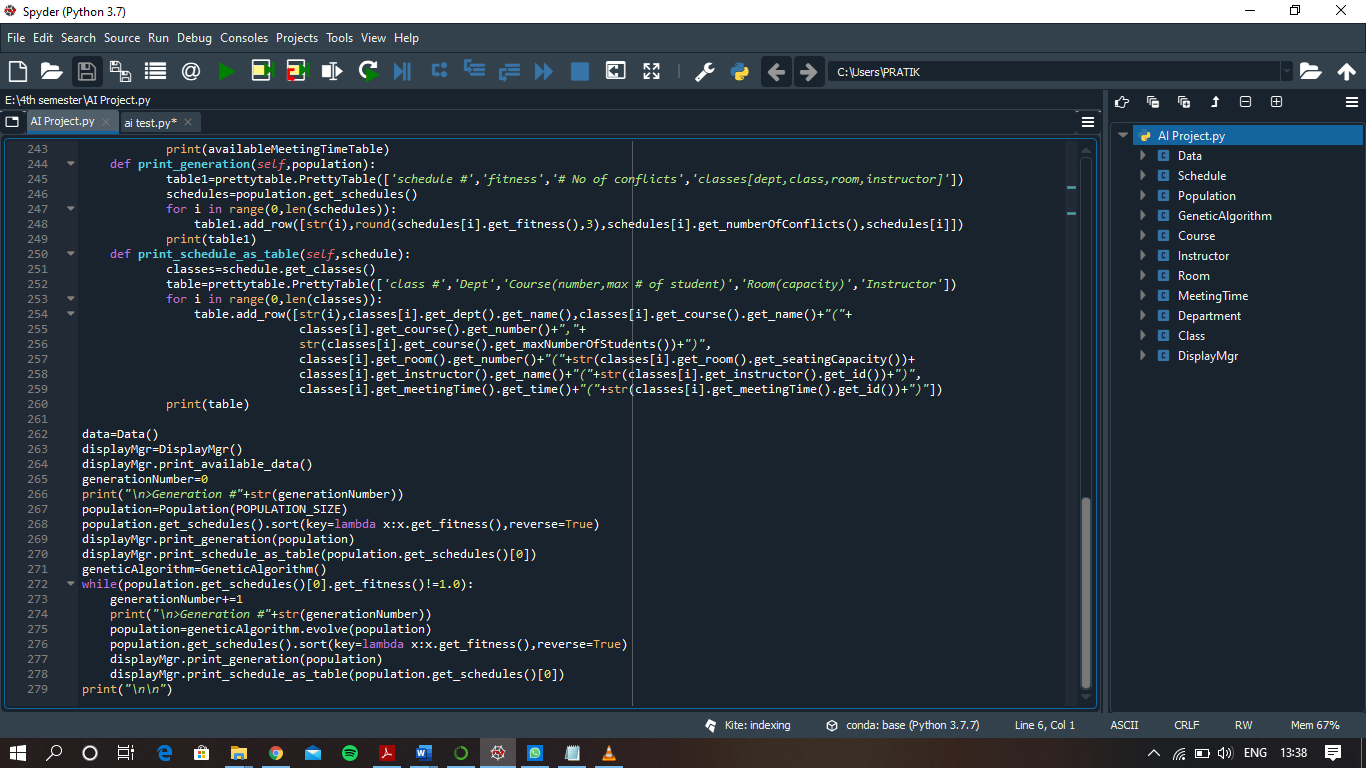
**Class : DisplayMgr**

The tabular form of time table generated and the state space generated from the algorithm with generation ,is handled in this class.

****



**Main function**



**TECHNOLOGY AND FRAMEWORK**

1. The most important point the code uses genetic approach.
2. The code is implemented on anaconda platform on spyder application.
3. The code includes a header called “pretty table” which helps us in a proper display of timetable in tabular form and all the other regarded data.
4. The pretty table header need to be installed explicitly by entering the command “easy\_install pretty table” on the anaconda promt shell.
5. The code made by our group follows the completely object oriented programming using the help of classes. By doing so it made us easy for our work distribution and also the integration of all the classes into one code.
6. The architecture of the system is divided into 4 levels. The first level is use

for acquiring following input from the user: Student data, Teacher data, Subject data, Room data and Timeslot data. The second level is used for linking student data with that of the teacher data and teacher data with that of the subject data.

Once data is linked, memes are generated.From the generated list of memes, genetic operations are performed in the third level.

**SWOT ANALYSIS**

As the previous methods of Timetable generation included a tedious process of assigning each subject to staff manually and scheduling the Timetable as in a way so that no clashes occur. But this process also took great use of time and also us of paperwork which is cost-ineffective. For this approach we decide a solution of using our computing skills and technology to generate the Timetable. It is done using the Automated Timetable generator which contains the involvement of Evolutionary Algorithms (EAs) called Genetic Algorithm. The Genetic Algorithm involves the process of Chromosome Representation to generate the Timetable. The above Solution gives an block model of following processes: The user will enter each of the data as counts of subjects,class-rooms, labs, lectures, students. The admin will assign each subject to their respective staff and assign them classrooms and the students whom they will teach. The Admin will use constraints as given in the algorithm so that no constraints occur. After assigning the Admin will do a verification check so that no anomalies are missed out. If the Admin encounters any mistake or clash that had been gone unnoticed earlier, he/she has the option to edit and then regenerate. After successful reviews the Timetable is uploaded on the college website for the staffs and students to view.