**CONTENTS**

PAGE NO

|  | ABSTRACT | 1 |
| --- | --- | --- |
| 1 | INTRODUCTION | 2 |
| 2 | SYSTEM REQUIREMENTS | 9 |
| 3 | OBJECTIVE OF THE PROJECT | 10 |
| 4 | MODULES PURPOSES  AND CODING | 11 |
| 5 | RESULTS AND DISCUSSIONS | 27 |
| 6 | CONCLUSION AND FUTURE  ENHANCEMENT | 36 |
| 7 | BIBLIOGRAPHY | 37 |

**ABSTRACT**

The main objective of this project is to sort out the students in or different subject by giving the dimensions of grid /room in the form of a matrix. For example if there are nine tables in a grid/room the user gives in a form of 3x3 matrix. The second objective of this programme is to make sure that the examinee gets no chance of cheating or do any other malpractices. This programme makes sure the arrangements of the examinees are in such a way that the examiner has clear vision on every single examinee. Programme solves the any matrix dimensions and number of examinees to sort out the apt places for the examinees in the hall/room. This programme is extremely helpful when the number of examinees exceed the digits where a human can’t calculate their seating arrangements.

            This programme follows the principle of **Backtracking** in python. Backtracking is a general algorithm for finding solutions to some computational problems, notably constraint satisfaction problems that incrementally builds candidates to the solutions, and abandons a candidate as soon as it determines that the candidate cannot possibly be completed to a valid solution. Exams are important part of an individual’s life and making sure it’s taken with good spirit and in confidence is more important. This programmes relives the stress of an exam conductor and helps examiner a lot during exams. This programme is also user friendly and easy to use and does not require a system with high specs.

# 1. INTRODUCTION

## 1.1. About python

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant whitespace](https://en.wikipedia.org/wiki/Off-side_rule). Its language constructs and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigms), including [procedural](https://en.wikipedia.org/wiki/Procedural_programming), object-oriented, and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Python was conceived in the late 1980s as a successor to the [ABC language](https://en.wikipedia.org/wiki/ABC_(programming_language)). Python 2.0, released in 2000, introduced features like [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a [garbage collection](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) system capable of collecting [reference cycles](https://en.wikipedia.org/wiki/Reference_cycle). Python 3.0, released in 2008, was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility), and much Python 2 code does not run unmodified on Python 3.

The Python 2 language, i.e. Python 2.7.x, is "sunsetting" on January 1, 2020 (after extension; first planned for 2015), and the Python team of volunteers will not fix security issues, or improve it in other ways after that date. With the [end-of-life](https://en.wikipedia.org/wiki/End-of-life_(product)), only Python 3.5.x and later will be supported.

Python [interpreters](https://en.wikipedia.org/wiki/Interpreter_(computing)) are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). A global community of programmers develops and maintains [CPython](https://en.wikipedia.org/wiki/CPython), an [open source](https://en.wikipedia.org/wiki/Open-source_software) [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation). A [non-profit organization](https://en.wikipedia.org/wiki/Nonprofit_organization), the [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation), manages and directs resources for Python and CPython development.

## 1.2. Overview of Python modules:

### **1.2.1. Overview of Python Modules:**

The Python interpreter has a number of built-in functions. They are loaded automatically as the interpreter starts and are always available. For example, print() and input() for I/O, number conversion functions int(),float(),complex(), data type conversions list(),tuple(), set(), etc.

In addition to built-in functions, a large number of pre-defined functions are also available as a part of libraries bundled with Python distributions. These functions are defined in modules. A module is a file containing definitions of functions, classes, variables, constants or any other Python objects. Contents of this file can be made available to any other program.

Built-in modules are written in C and integrated with the Python interpreter. Each built-in module contains resources for certain system-specific functionalities such as OS management, disk IO, etc. The standard library also contains many Python scripts (with the .py extension) containing useful utilities.

#### **1.2.1.1 Some Predefined Modules of Python:**

|  | **Math** | | These include trigonometric functions, representation functions, logarithmic functions, angle conversion functions, etc. In addition, two mathematical constants are also defined in this module. | |
| --- | --- | --- | --- | --- |
|  | **Numpy** | | Numpy | Is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. |
|  |  |
|  | **Mysql** | | Python needs a MySQL driver to access the MySQL database. | |
|  | **matplotlib** | | Matplotlib is a visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. | |
|  | **tkinter** | | Tkinter is Python's de-facto standard GUI (Graphical User Interface) package. Python with tkinter is the fastest and easiest way to create the GUI applications. | |
| **random** | | Functions in the **random** module depend on a pseudo-random number generator function random(), which generates a random float number between 0.0 and 1.0. | | |

**1.3. Overview of Recursion:**

**1.3.1. ABOUT RECURSION:**

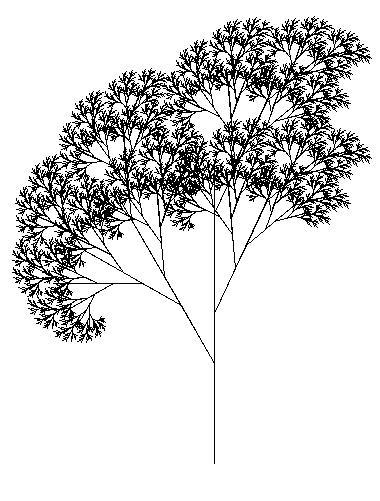
The term Recursion can be defined as the process of defining something in Terms of itself. In simple words, it is a process in which a function calls itself directly or indirectly.

A physical world example would be to place two parallel mirrors facing each Other. Any object in between them would be reflected recursively. If a function definition calls itself, then this function would be known as recursive function. Recursion can be indirect also. When a function calls itself directly from within its body, it is known as indirect recursion.

**1.3.2. APPLICATIONS OF RECURSION**:

Most programming problems are solvable without recursion. So, strictly speaking, recursion usually isn't necessary. However, some situations particularly lend themselves to a self-referential definition. To such a case programmatically, a recursive solution would likely be cleaner and more concise. Traversal of tree-like data structures is another good example. Because these are nested structures, they readily fit a recursive definition. A non-recursive algorithm to walk through a nested structure is likely to be somewhat clunky, while a recursive solution will be relatively elegant. Recursion has many, many applications. We use recursion to compute the factorial function, to determine whether a word is a palindrome, to compute powers of a number, to draw a type of fractal, and to solve the ancient Towers of Hanoi problem, Fibonacci Series, Factorial etc. . . .

**A Basic Example would be as follows:**



Tree created using the Logo programming language and relying heavily on recursion. Each branch can be seen as a smaller version of a tree.

**1.3.3. Recursion vs. Iteration:**

Recursion and loops are actually related concepts. Generally, anything you can do with a loop, you can do with recursion, and vice versa. Sometimes one way is simpler to write, and sometimes the other is, but in principle method calls are treated by a programming language compiler. When a loop repeats, it uses same memory locations for variables and repeats the same unit of code. Whereas in recursion, instead of repeating the same unit of code and using the same memory locations for variables, fresh memory space is allocated for each recursive cell. As it happens any problem that can be solved via iteration can be solved using recursion and any problem that can be solved via recursion can be solved using iteration. Iteration is preferred by programmers for most recurring events, reserving recursion for instances where the programming solution would be greatly simplified. In a programming language, recursion involves an additional cost in terms of the space used in RAM by each recursive call to a function and in time used by the function call. Because of extra memory stack manipulation, recursive versions of functions often run slower and use more memory than their iterative counterparts. But this is not the always the case, and recursion can sometimes make code easier to understand. They are interchangeable. Although loop and recursion are interchangeable, yet there are some examples where recursion is indeed the best way to approach a problem. Even for problems that can be treated equally well through iteration and recursion, there is a subtle difference which is because of the way loops and

**1.4. OVERVIEW OF MySQL:**

MySQL is an open-source relational database management system. Its name is a combination of "My", the name of co-founder Michael Wideness’s daughter, and "SQL", the abbreviation for Structured Query Language.

## 1.4.1. What is MySQL?

MySQL is currently the most popular database management system software used for managing the relational database. It is open-source database software, which is supported by Oracle Company. It is fast, scalable, and easy to use database management system in comparison with Microsoft SQL Server and Oracle Database. It is commonly used in conjunction with [PHP](https://www.javatpoint.com/php-tutorial) scripts for creating powerful and dynamic server-side or web-based enterprise applications.

MySQL is a [Relational Database Management System](https://www.javatpoint.com/what-is-rdbms) (RDBMS) software that provides many things, which are as follows:

* It allows us to implement database operations on tables, rows, columns, and indexes.
* It defines the database relationship in the form of tables (collection of rows and columns), also known as relations.
* It provides the Referential Integrity between rows or columns of various tables.
* It allows us to updates the table indexes automatically.
* It uses many SQL queries and combines useful information from multiple tables for the end-users.

**1.4.2. WORKING with MySQL:**

**1.4.2.1. CREATING A DATABASE AND ACCESSING IT:**

To create a database, we use the following command in MySQL:

mysql> create database grxii;

Query OK, 1 row affected (0.00 sec)

Before you start making queries upon the data in tables of a database, you need to open the database for use. For this, after logging into MySQL, you need to issue a command:

mysql> use grxii;

Database changed

To insert tables into the database grxii we use the following command:

mysql> create table xiib(admno int primary key, sname char(20),gender char(2),address varchar(20),mark int);

Query OK, 0 rows affected (0.00 sec)

Then next to insert values into a table we use the following query:

insert into xiib values(1,'subbu','m','anbu nagar','475');

Query OK, 1 row affected (0.01 sec)

The select command of SQL lets you make queries on the database. This can be used to retrieve a subset of rows or columns from one or more tables. In its simplest form, select statement is used as given below:

select \* from xiib;

+-------+-------+--------+------------+------+

| admno | sname | gender | address | mark |

+-------+-------+--------+------------+------+

| 1 | subbu | m | anbu nagar | 475 |

+-------+-------+--------+------------+------+

1 row in set (0.00 sec)

**1.4.2.2. REMOVING AND DELETING:**

Removing a database

Sometimes, you need to remove a database when you don’t need it anymore. But before making this decision, do make sure that you don’t need data stored in different tables of the database. This is because, when you drop a database, all its tables also get removed along with the database. Following is the command:

mysql> drop database grxii;

Query OK, 1 row affected (0.09 sec)

To delete a table from a database we use the following command:

mysql> drop table xiib;

Query OK, 0 rows affected (0.01 sec)

While working with tables, we may reach at a situation where we no longer need some roes of data. In such a case, we would like to remove such rows. This can be done by using delete command. The delete command removes rows from a table. This removes the entire rows, not individual field values, so no field arguments is needed or accepted. The delete statement takes the following general form:

mysql> Delete from <tablename> [where <predictate>];

To remove all the contents of items table, you use the command:

mysql> DELETE FROM items;

**1.4.2.3. MODIFYING DATA:**

1. Update

Sometimes you need to change some or all of the values in an existing row. This can be done using the update command of MySQL. The update command specifies the rows to be changed using the where clause, and the new data using the SET keyword. The new data can be a specified content, an expression or data from other tables.

For Example:

mysql> update xiib set mark=470 where sname=subbu;

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

2. Alter

The alter table command is used to change definitions of existing tables. Usually, it can add columns to a table. Sometimes it can delete columns (depending on privileges) or change their sizes.

For example:

mysql> alter table xiib modify sname varchar(40);

Query OK, 10 rows affected (0.03 sec)

Records: 10 Duplicates: 0 Warnings: 0

**1.4.2.4. INTERFACE PYTHON WITH MYSQL:**

In order to connect to a database from within python, you need a library that provides connectivity functionality. There are many different libraries available for python to accomplish this. MySQL connector is an example that we use.

First of all you need to import mysql.connector package in your python scripts. For this, write import command as shown below:

import mysql.connector

Next you need to establish a connection to a MySQL database using connect() function of mysql.connector package. The connect() function of mysql.connector establishes connection to a MySQL database and requires four parameters, for example:

dbms=mc.connect(host='localhost',user='root',passwd='1996',database='grxii')

Normally when you connect to a database from within a script/diagram, then the query gets sent to the server, where it gets executed, and the result set(the set of records that retrieved as per query) is sent over the connection to you, in one burst of activity i.e., in one go. But you may want to access the retrieved data, one row at a time. But query processing cannot happen as one row at a time, so a special type of control structure called database cursor can be created that gets all the access of all the record retrieved as per query(called the result set) and allows you to traverse the result set row by row.

To create this we use the following command syntax:

<cursorobject>=<connectionobject>.cursor()

Once you have created a cursor, you can execute SQL query using execute() function with cursor object as per following syntax:

<cursorobject>.execute(<sql query string>)

Once the result of query is available in the form of a result set stored in cursor object, you can extract data from the result set using any of the following fetch....() functions :

1.fetchall():

The fetchall() method will return all the rows from the result set in the form of a tuple containing the records.

data=curobj.fetchall()

For example, the above command will work to fetch all the records:

(1, 'subbu', 'm', 'anbu nagar', 475)

(2, 'sanjay vikas', 'm', 'erode', 478)

2.fetchone():

The fetchone() method will return only one row from the result set in the form of a tuple containing a record.

data=curobj.fetchone()

This would work as:

(1, 'subbu', 'm', 'anbu nagar', 475)

3.fetchmany():

The fetchmany() method will return only the <n> number of rows from the result set in the form of a tuple containing the records.

data=curobj.fetchmany(2)

This would work as to print two rows as:

(1, 'subbu', 'm', 'anbu nagar', 475)

(2, 'sanjay vikas', 'm', 'erode', 478)

After you are through all the processing, in this final step, you need to close the connection established.

This you can do as follows:

connectionobject.close()

**2. SYSTEM REQUIREMNETS**

* Operating System: Windows 7, 8 and 10.
* Processor: Dual Core 1.4GHz Processor
* RAM: Minimum 1 GB
* Hard Disk Space: Minimum 750MB space required.
* Software: Python 3.7.0, MySQL 5.5

3. **OBJECTIVE OF THE PROJECT**

This project is to sort out the students writing an exam of any type in or different subject so that each student write the exam honestly and simultaneously making it easy for the exam head/invigilator to sort without the need of calculating where to place the student so that he/she doesn't sit next to his/her same subject partner/friend or to cheat. This project also makes it easy to input the dimensions of the room or room size in matrix order. For example if there are 3 rows and 3 columns then the room is basically 3x3 dimension. This project takes multiple subject students and sort them.

It also displays the seating arrangement in a graphical unit input which makes it easier and sleek to see/use by the user (here exam invigilator). The sorted arrangement can be displayed to the students to check their placing arrangement and that makes it easy for the student who are entering the exam hall/place.

This is accessed by the exam head or invigilator.

An invigilator can:

• Enter the dimensions of the room. The dimensions should be in matrix order as explained above.

• Enter different subjects of the students.

• Enter total number of students.

The school, college etc. can store their student’s information in the backend using MySQL. The student is automatically stored in backend. The student data can be even stored first in MySQL and then automatically sorting process can be executed. All the details of the sorted students is shown/displayed properly in the graphic part. The user can easily sort the students with ease.

Therefore, this project helps to save time and importantly honesty is maintained during any examination written.

**4. MODULES PURPOSES AND CODING**

**4.1. Modules Used**

🡪 mysql.connector

🡪 random

**4.2. Modules Purposes**

**• MySQL.Connector**: This module is a standardized database driver provided

By MySQL.

**• random:** The random module is a built-in module to generate the pseudo-random variables. It can be used perform some action randomly such as to get a random number, selecting a random elements from a list, shuffle elements randomly, etc.

**4.3. Source code:**

import random as ra

import mysql.connector as mc

while True:

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print(' Welcome To The Examinee Place Alotter')

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

print('Select "1" to know whats this program about')

print('Select "2" to sort the students')

print('Select "3" to Exit')

ch=int(input('Enter your choice'))

if ch==1:

print('The motto of this project is to sort the students of various categories writing an exam obtained from a given\

data considering the classroom as a rectangular array so that student of similar categories doesnt sit together')

if ch==2:

print('Provide your sql data to fetch the students data from')

print('')

print('Note:The given sql data should be in the format of',sep=':')

print('[roll number,student name,student exam category]')

i\_user = input('enter your username')

i\_password=input('enter your password')

i\_database = input('enter your database')

table\_name = input('enter the table name')

co = mc.connect(host='localhost', user=i\_user,password=i\_password, database=i\_database)

cu = co.cursor()

cu.execute(f"select \* from {table\_name}")

sql\_lst = cu.fetchall()

while True:

r = int(input("enter the number of rows"))

c = int(input('enter the number of coloumns'))

if c > 0 and r > 0:

break

else:

print("Try again the entered dimensions are not valied")

def create\_table(m, n): # m= no of rowss, n=no of coloumns

l = []

for i in range(m): # 3(0,1,2)

z = []

for j in range(n): # 2 (0,1) l=[[0,0],[0,0],[0,0]]

z.append(0)

l.append(z)

return l

def print\_board(board, l=1):

print('[',)

for i in range(len(board)):

for j in range(len(board[0])):

#print([board[i][j], len([board[i][j]]), l])

if len(board[i][j]) < l:

a = board[i][j]+' '\*(l-len(board[i][j]))

elif len(board[i][j]) == l:

a = board[i][j]

print(a, end=' | ')

print('')

def print\_coordinateboard(board):

for i in range(len(board)):

for j in range(len(board[0])):

print((i, j), end='|')

print('')

def find\_empty(board, l=None):

for i in range(len(board)):

for j in range(len(board[0])):

if board[i][j] == 0 and l == None:

return i, j

def segregation(a):

category = []

number\_category = []

for j in a:

if j[2] not in category:

category.append(j[2])

number\_category.append(category.index(j[2])+1)

namelist = []

for j in category:

l1 = []

for i in a:

if i[2] == j:

l1.append(i[1])

namelist.append(l1)

return namelist, number\_category, category

def dist\_list(a):

l=[]

for i in a:

if i not in l:

l.append(i)

return l

def check(num, row, col, b):

if True:

if col+1 != len(b[0]) and col-1 != -1:

if b[row][col+1] == num or b[row][col-1] == num:

return False

elif col+1 == len(b[0]):

if b[row][col-1] == num:

return False

elif col-1 == -1:

if b[row][col+1] == num:

return False

if True:

if row+1 != len(b) and row-1 != -1:

if b[row-1][col] == num or b[row+1][col] == num:

return False

elif row+1 == len(b):

if b[row-1][col] == num:

return False

elif row-1 == -1:

if b[row+1][col] == num:

return False

return True

def limit(m,n):

if m%2!=0:

a=(m/2+0.5)

if m%2==0:

a=(m/2)

if n%2!=0:

b=(n/2+0.5)

if n%2==0:

b=(n/2)

#finding max value

if r%2==0 and c%2==0:

maximum=a\*b+a\*b

if r%2==0 and c%2!=0:

maximum=a\*b+a\*(b-1)

if r%2!=0 and c%2==0:

maximum=a\*b+(a-1)\*b

if r%2!=0 and c%2!=0:

maximum=a\*b+(a-1)\*(b-1)

return int(maximum)

lim=limit(r,c)

def elemental\_list(): # [3,2,1] ---->[1,1,1,2,2,3]

el = []

dict = {}

for i in range(len(fetch\_lst)):

dict[b[i]] = fetch\_lst[i]

a = sorted(dict.items(), key=lambda x: x[1], reverse=True)

for i in a:

el = el+[i[0]]\*i[1]

#print(el)

#print(dict)

return el

tlist = []

initial\_cu = [0, 0]

BOARD = create\_table(r, c)

a, b, ca = segregation(sql\_lst)

lim = limit(r, c)

def fetchlist():

fetch\_lst=[]

while True:

print('Hey there enter the number of records from each category needed to be fetched,\

the maximum number of students that can be fetched for a particular category in'\

,len(b),'categories for the given table ', r,'\*',c,'is', lim)

print('')

for i in ca:

while True:

print('enter the number of students needed to be fetched for ',i.upper(),'category ')

print('Note:The remaining number of entries are',r\*c-sum(fetch\_lst),'Max limit=',lim)

print('')

inp=int(input('enter'))

cnt=0

while True:

if inp>lim:

print('enter again as number student exceeded the max limit ')

break

if sum(fetch\_lst)+inp>r\*c:

print('enter last value again as enterd value is greater than seat limit')

break

if inp>len(a[ca.index(i)]):

print('Enter again as number of student in this category exceeded the record limit' )

else:

fetch\_lst.append(inp)

cnt=1

break

if cnt==1:

break

break

return fetch\_lst

def solve(board): # (BOARD)

#find=find\_empty(board)

#print\_board(board)

#print('Elelist',ele\_list)

#print('TLSIT',tlist)

if not ele\_list:

#print('first ret')

return True # To validate the if condition

else:

if len(ele\_list)+len(tlist) == r\*c:

R, C = find\_empty(board)

for i in dist\_list(ele\_list): # [1,2,3,4,5]

if check(i, R, C, board):

board[R][C] = i

a = ele\_list.pop(ele\_list.index(i))

tlist.append(a)

if solve(board):

#print('2nd ret')

return board

else:

board[R][C] = 0

ele\_list.append(tlist.pop())

#print('ELELIST',ele\_list)

else:

if find\_empty(board) == None and len(ele\_list) >= 1:

return False

else:

R, C = find\_empty(board)

#print(R,C)

for i in dist\_list(ele\_list): # [1,2,3,4,5]

# and ((R+1==r and C+1==c and len(ele\_list)==1) or (R+1!=r or C+1!=c)):

if check(i, R, C, board):

#print('h1')

board[R][C] = i

a = ele\_list.pop(ele\_list.index(i))

tlist.append(a)

if solve(board):

#print('2nd ret')

return board

else: # backtracking

board[R][C] = 0

ele\_list.append(tlist.pop())

#print('ELELIST',ele\_list)

for i in dist\_list(ele\_list):

if C+1 == c:

R1 = R+1

C1 = 0

if C+1 != c:

R1 = R

C1 = C+1

if (R+1 != r or C+1 != c) and check(i, R1, C1, board) and len(dist\_list(ele\_list)) == 1:

#print('hi2',i)

board[R1][C1] = i

a = ele\_list.pop(ele\_list.index(i))

tlist.append(a)

board[R][C] = 'EMPTY'

if solve(board):

#print('2nd ret')

return board

else: # backtracking

board[R][C] = 0

board[R1][C1] = 0

ele\_list.append(tlist.pop())

#print('ELELIST',ele\_list)

return False

fetch\_lst =fetchlist()

ele\_list = elemental\_list()

len\_ca = len(ca)

#print\_board(BOARD)

solved\_board = solve(BOARD)

print('This is just the coordinates of each table for referential purpose')

print\_coordinateboard(solved\_board)

while True:

ans=int(input('1)do you want to autofill students or 2)customfill,choice=1or2:'))

if ans==1 or ans==2:

break

else:

print('enterd choice doesnt excist try again')

t\_table = []

sql\_op\_lst=[]

added\_names=[]

max = 0

for i in range(len(solved\_board)):

l = []

for j in range(len(solved\_board[0])):

if solved\_board[i][j] != 'EMPTY' and solved\_board[i][j] != 0:

if ans==1:

element = ra.choice(a[solved\_board[i][j]-1])

added\_names.append(a[solved\_board[i][j]-1].pop(a[solved\_board[i][j]-1].index(element)))

if ans==2:

while True:

print('enter the student to be seated in coordinate',i+1,',',j+1)

print('')

print('available students are in the given coordinates are' )

print('')

print(a[solved\_board[i][j]-1])

print('NOTE:please be careful to not enter duplicate student entry')

print('')

element=input('enter')

if element not in a[solved\_board[i][j]-1] :

print('enter again as enterd student name does not excist or cant be seated in the given place')

else:

added\_names.append(a[solved\_board[i][j]-1].pop(a[solved\_board[i][j]-1].index(element)))

break

if len(element) > max:

max = len(element)

for k in sql\_lst:

if k[1]==element:

sno=k[0]

sql\_op\_lst.append([sno,element, ca[solved\_board[i][j]-1], i+1, j+1])

else:

element = 'EMPTY'

sql\_op\_lst.append(['EMPTY',element, 'EMPTY', i+1, j+1])

l.append(element)

t\_table.append(l)

print('Here comes the graphical representation of sorted classroom')

print\_board(t\_table, max)

'''

cu.execute("create table finaltable (admissionno varchar(20),studentname varchar(20),category varchar(20),rownumber int,coloumnnumber int,primary key(rownumber,coloumnnumber));")

'''

for i in sql\_op\_lst:

cu.execute(f"insert into finaltable values('{i[0]}','{i[1]}','{i[2]}',{i[3]},{i[4]})")

co.commit()

print('Successfully updated the final record in your sql table')

if ch==3:

break

#print('')

#swapping

'''

nl1=[]

nl2=[]

while True:

for i in solved\_board:

for j in i:

nl1.append(j)

a=nl1.sort()

for k in b:

nl2.append(nl1.count(k))

if nl2==fetch\_lst:

print('test passed')

break

else:

print('test failed',nl2)

break

print(fetch\_lst)

print(ele\_list)

'''

'''

if len(fetch\_lst)==len(ca)-1:

print(' remaining item',r\*c-sum(fetch\_lst),'is considerd for category',i)

fetch\_lst.append(r\*c-sum(fetch\_lst))

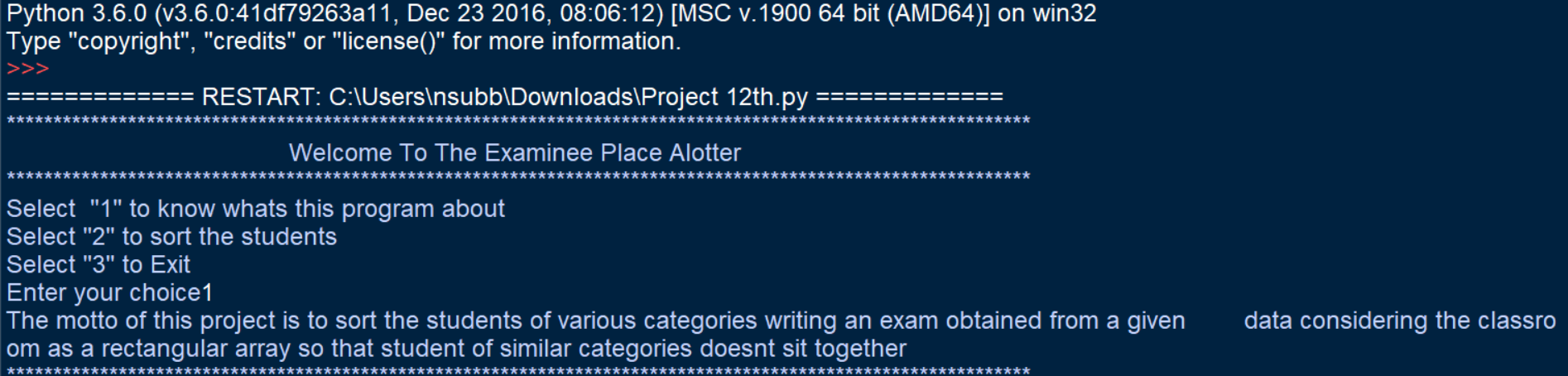
break

'''

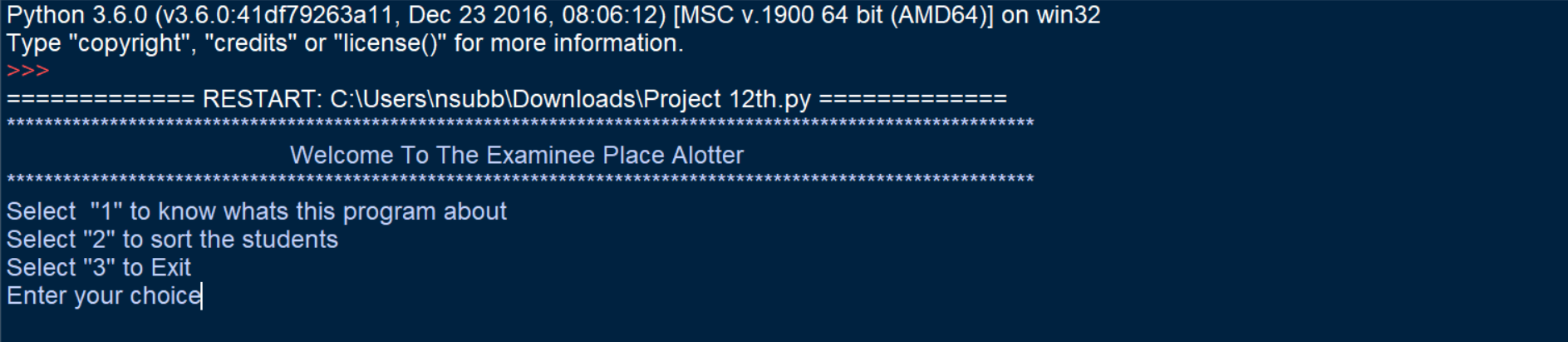
**5. RESULTS AND DISCUSSION**

**5.1. Results**

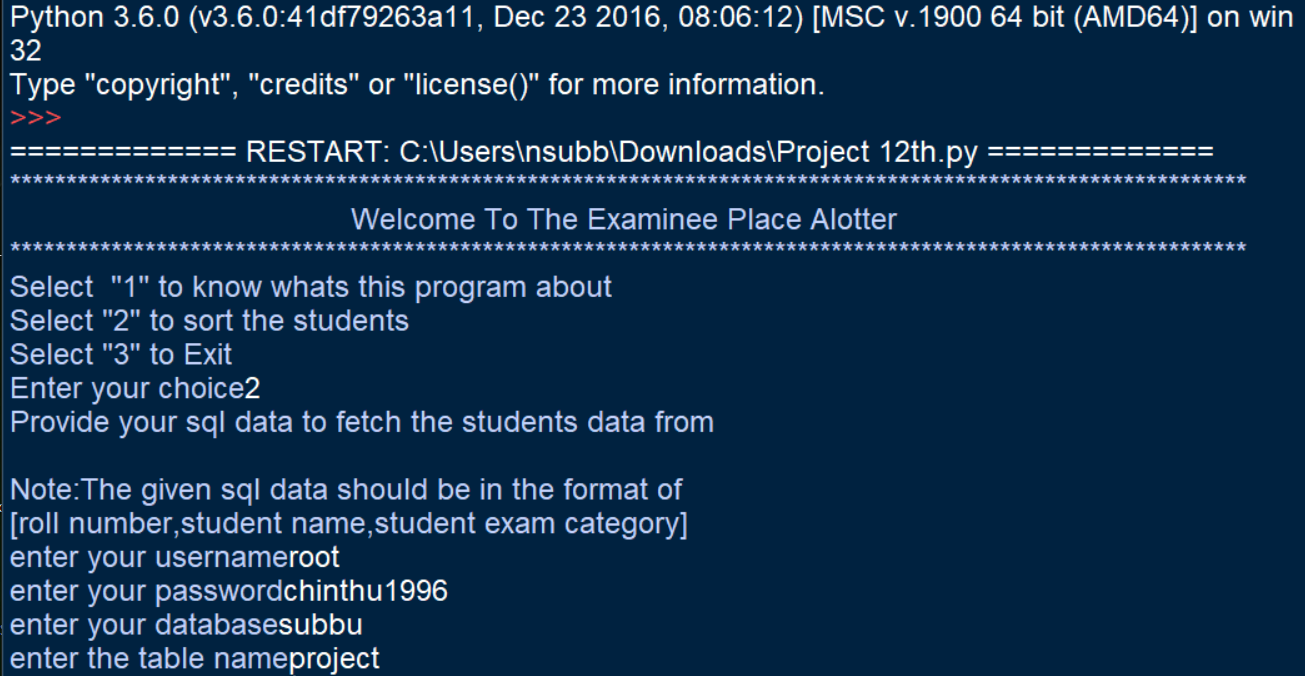
Entering menu to know what’s this program is about



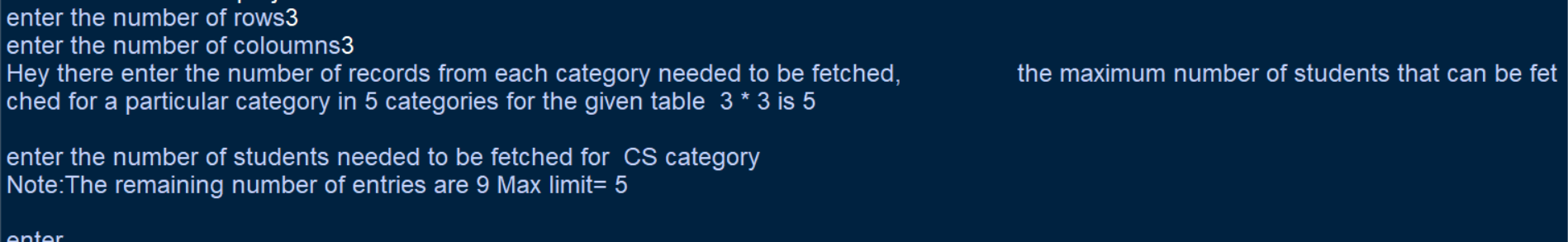
Main Window —



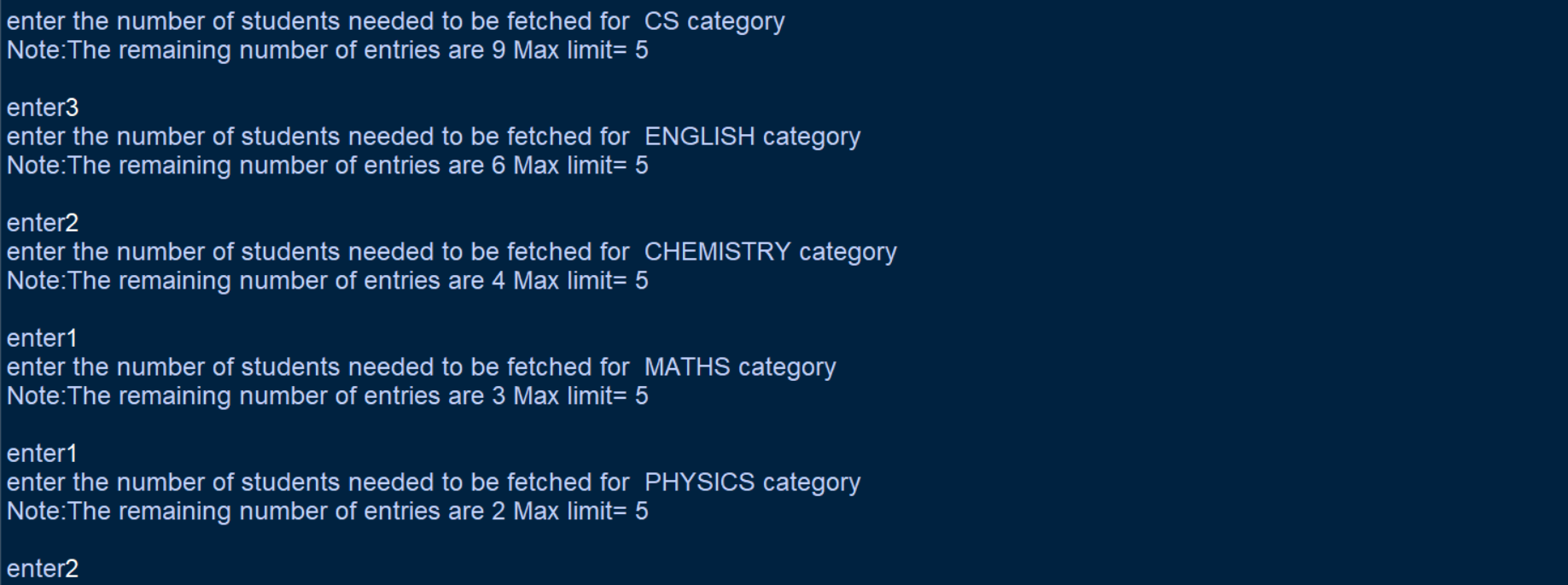
Selecting Sort the student menu



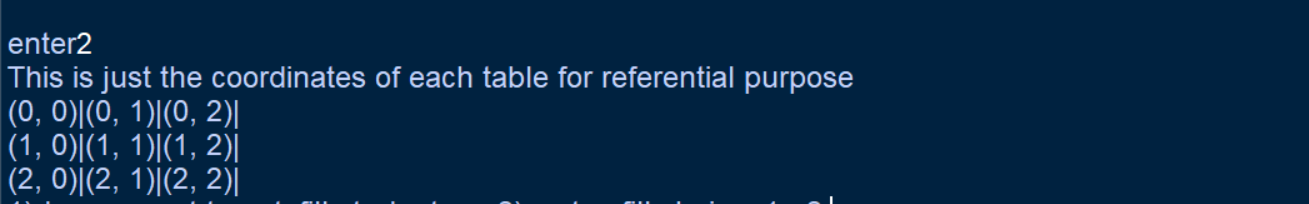
rows and columns



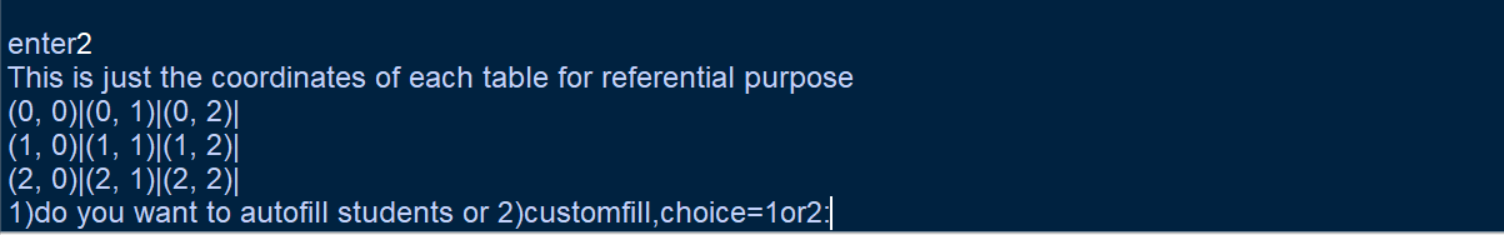
entering students needed to be fetched for the particular category (subject)



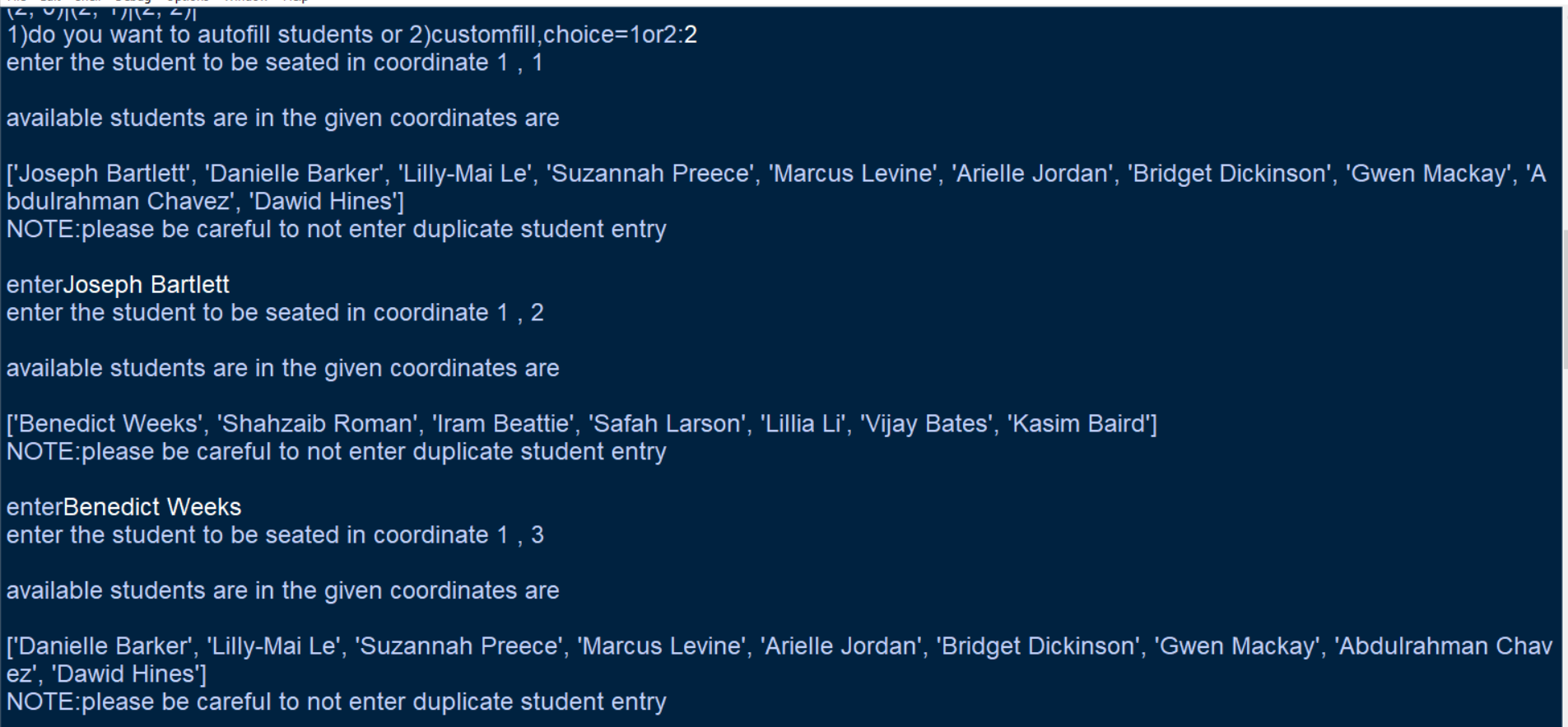
Displaying the output solved seating arrangement

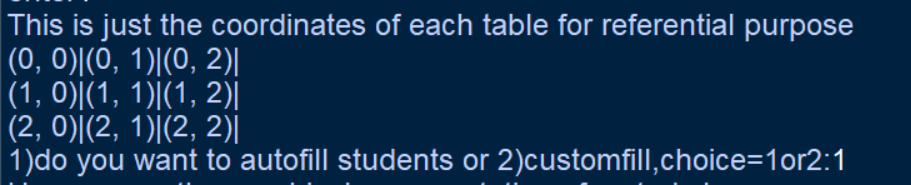


Choice to autofill or customfill students

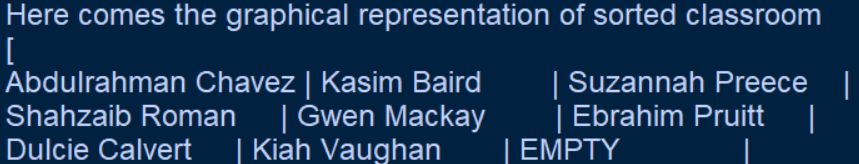


When choice is customfill and autofill

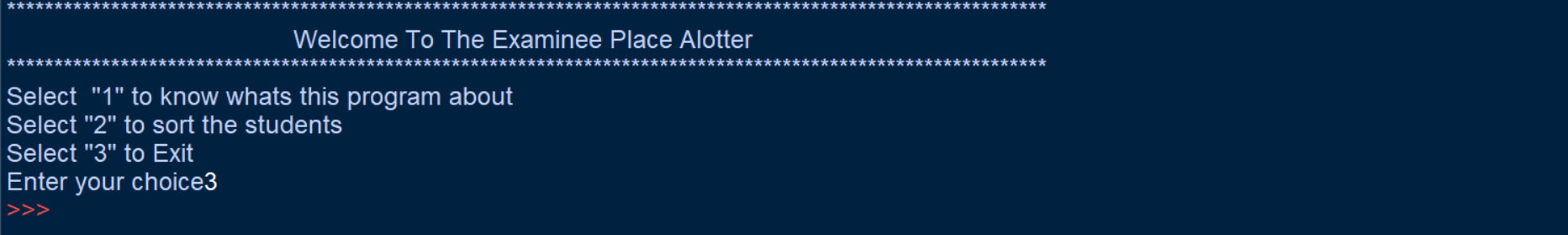




Resultant solved seating arrangement with names for both customfill and autofill (Here customfill means that user entered the names respectively and autofill means that it is autogenerated)



Exiting the program



**5.2. Discussion**

This project has many advantages:

**•** Security: Invigilator can easily login and use the service

**•** Efficiency: Admin can easily enter the dimensions of the room.

**•** Modifiable: User can make specific changes in the arrangement

if not satisfied.

**•** Simplicity: the working of the code is pretty simple and not

complex

**•** Understandable: The output is understandable and easy

A few disadvantages:

**•** There is no alternative method for logging in.

**•** Does not support cross platform

**6. CONCLUSION AND FUTURE ENHANCEMENT**

**6.1. Conclusion**

This software has its own advantages and disadvantages. But it is

quite successful in resolving the scandal of arranging a large number

of candidates at a single go. Future enhancements will be made to

make this project much better. All the information are secure and

modifiable.

**6.2. Future Enhancement**

The project is left for future enhancement.

**•** A new refined GUI.

**•** A alternative way for logging in.

**7. BIBLIOGRAPHY**

1. Computer Science with Python,Sumita Arora

2. www.stackoverflow.com

3. www.geeksforgeeks.org