INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF BUSINESS MANAGEMENT

DOC334 –Computer Programming

COURSEWORK – 2023/24

(Foundation September 2023)

Module Leader:	Mr. Nishan Saliya Harankahawa
Group:	D
Submission Date:	30th March 2024

Module	**DOC334 – Computer Programming**
Assessment Type:	**Individual Coursework**
Project Title:	**Simple Percolation Process.**
Student Name:	**M.A.M Basim**
Student ID:	**20231628**

Abstract

This report examines a python code that mimics the percolation process, percolation is a process of a liquid slowly passing through a filter. Initially this code will ask a grid size from user, if the user inputs in between the range of “3x3” and “9x9”, it will generate a gride according to the user input. If he failed to input or didn’t input anything it will print the default “5x5” grid size. Furthermore, if he went out of the given range it will show a proper error massage; according to the user inputs it will generate 2digit random numbers. If the all the columns are filled with numbers, at the bottom it will print “Ok”, if any empty columns remain it will show as “No”.

some in-built modules are used to enhance this programming. And the result of the program will save as HTML and Text file. Meanwhile, program will ask whether the user wants to Continue or End the program, according to the user preference it will work.

Acknowledgements

I would like to express my heartfelt gratitude to everyone who helped make this report a success.

First and foremost, I would like to sincerely thank Mr. Nishan Saliya Harankahawa, my module leader. His continuous support and priceless assistance in improving my python codes were crucial in helping me stay focused on the task at hand.

I want to thank Mr. Namal Malalasena, the lecturer in charge of my group tutorial, for his invaluable support. His commitment to leading practice sessions on matrices and providing me with a wide variety of questions to answer improved my learning process and made a substantial contribution to the quality of this report.

I also owe a special thank you to my Group D colleagues. Their distinct viewpoints and thoughts really enhanced my learning experience. Working together not only made it possible for me to recognize and correct my shortcomings, but it also promoted a collaborative mindset that improved my learning process.

I would like to express my sincere gratitude to everyone who helped make this endeavor a reality once more. I sincerely appreciate the chance to have collaborated with such committed people, and your help and advice have been invaluable.

Table of Contents

List of figures

List Of Tables

01.INTRODUCTION

This report contains a brief overview of a Python program which was arranged to run a simple percolation process. Percolation is the process of a liquid slowly passing through a filter. This is how coffee is usually made. This report contains a python code that mimics this concept.

Percolation process is Coded according generate a random 2 digits number by asking the user about the grid size, the maximum size of the grid is 9x9 and the minimum size is 3x3, if the user enter a higher or a lower value, it will show a proper error massage. User failed or didn't choose a grid size; it will print the default size 5x5.

The percolation process is possible for a column if the entire column consists of numbers from top to bottom. Likewise the percolation is not possible for a column if the column is consisting of one or more empty spaces from top to bottom. If the percolation is possible at the bottom of the table, it will display "Ok" if not it will display "No".

56	11	23	70	33
76	65	55		16
	43	18		33
32	98		91	29
18	45	10	61	95
No	OK	No	No	Ok

Table 1: Percolation Process

02.Algorithms

1. Start
2. First and foremost, import the in-built module random because it plays a crucial part on this percolation process.

3. For the good and enhanced grid line appearance import pretty table module by the package pip.

4. For saving file to the notepad on the format of year month and day, for that import date time inbuilt module.

5. Create a function " create_grid" that takes two parameter, number of rows and columns for grid; on that function create a empty list called grid to store the rows of the grid. within the function, it iterates through each cell and randomly decides whether to fill with 2 digit number or leave it empty with the probability of 20%.the function returns the generated grids as list of lists.

6. Creating another function called "display_grid" that initializes a pretty table object named table. That add each row the table, and it print the grid.

7. Another one called "display_grid" both are not same; both have different implementation and finctionalities. By this function iterate each row in the input grid. And use a "if" and "else" statement to to check whether the cell is empty or not if its empty it will display "No" else it will display "Ok" this will print bellow the pretty table that got printed.

8. Create a folder inside that folder, create another one and name it as "sub", inside the sub, save the above code. I used to save it as "function".

9. Get a new file from python idle and start to code the main program.

10. As a first thing import the module for this coding also;

    - import random
    - import sub.function
    - import sys
    - from prettytable import PrettyTable
    - import datetime.

11. Open a while true command, inside that asking the user to enter a grid size if he chooses above "9x9" or below "3x3" it will choose a proper error message. If the

user failed or did not type any grid size as a default it will come up with a “5x5” grid size as default.

12. Use error handling for better user experience and debugging.

13. Use "create_grid” function to generate the grid.

14. Pretty tables visually represent the grid, add columns to the table and align them appropriately print the table to display the grid to user.

15. Check the column status by going to each cell whether its empty or not, the status depends on the empty cell according to it print.

16. Write the file content on to a text file with generated file name.

17. For the extra enhancing, saving to HTML.

18. Generate an HTML file with the same current as the table and save it using a similar conversion as file name.

19. Continue to exit prompt; user can choose to continue or end the programming, if the user wish to continue then it will loop into 11th step, else the program will come to an end, without valid input for the given prompt will print an error and loop the prompt.

03. Python codes for the above percolation process:

1. python codes in module “function” from the package “percolation” :

```
import random
from prettytable import PrettyTable
import datetime

#takes two parameters (rows and columns)
def create_grid(rows, columns):
    grid = []
    for i in range(rows):
        row = []
        for j in range(columns):
            if random.random() < 0.20:  # Changed threshold to 20%
                row.append(" ")
            else:
                value = random.randint(10, 99)#integer between 10 and 99 (2digit
integer)
                row.append(value)
        grid.append(row)
    return grid

#assigning the value to pretty table by calling it
def display_grid(grid):
    table = PrettyTable()
    for row in grid:
        table.add_row(row)
    print(table)#assigned into pretty table


def display_grid(grid):
    for row in grid:#display the grid row by row
        for value in row:
            print(value, end=" ")
        print()
    for j in range(len(grid[0])):#Provide column information
        empty = any(row[j] == ' ' for row in grid)
        if empty:
            print("    No  ", end=" ")  # Adjusted spacing for "No"
        else:
```

```
        print("    Ok  ", end=" ")  # Adjusted spacing for "Ok"
    print()
```

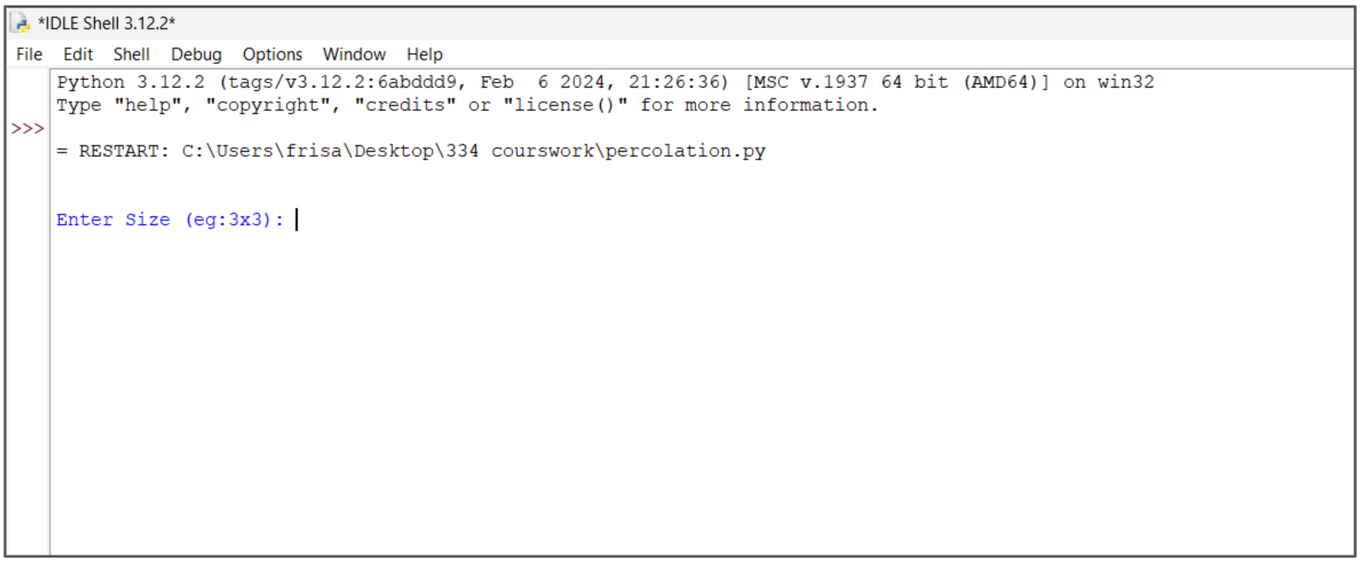
2. Main “percolation” code

```
import random
import sub.function
import sys
from prettytable import PrettyTable
import datetime

while True:
    grid_size = input("\n\nEnter Size (eg:3x3): ")
    print(grid_size)

    if not grid_size.strip():
        rows, columns = 5, 5  # default size
    else:
        rows, columns = map(int, grid_size.split("x"))

    try:
        if not (3 <= rows <= 9 and 3 <= columns <= 9):
            raise ValueError("ERROR")

        grid = sub.function.create_grid(rows, columns)  # Corrected function
import

        # Create pretty table
        table = PrettyTable()

        # Add columns to pretty table with alignment
        for i in range(columns):
            table.add_column(f"Column {i + 1}", [grid[j][i] for j in range(rows)],
align="c")

        # Print the table
        print(table)

        # Create a list to store the status of each column (Ok or No)
        column_status = []
        for i in range(columns):
            column_width = max(len(str(row[i])) for row in grid)
            if any(row[i] == ' ' for row in grid):
                column_status.append("No")
                print(" " * column_width + "    No  ", end=" ")
            else:
                column_status.append("Ok")
                print(" " * column_width + "    Ok  ", end=" ")

        # Add the column status row to the table
        table.add_row(column_status)

        # Generating File Name
        current_date = datetime.datetime.now().strftime("%Y_%m_%d")
        rand_num = random.randint(1000, 9999)
        filename = f"{current_date}_{rand_num}.txt"

        # Save table to file
        with open(filename, "w") as file:
            file.write(str(table))
            print("\n\nTable saved as file")

        # Save table to html
        table.border = True
        htmlfile = f"{current_date}_{rand_num}.html"
        with open(htmlfile, "w") as file:
            file.write(table.get_html_string())
            print("Table saved as html")

        while True:
            Run = str(input("\n\nTo continue press 'C' and to end the program
press 'E': "))
            if Run.lower() == 'c':
                break  # Breaks out of the inner loop and continues to the outer
loop
            elif Run.lower() == 'e':
                sys.exit()  # Exits system
            else:
                print("Invalid Input... Try again !!!")
                continue

    except ValueError:
        print("Error Occurred...")
```

04.Test cases

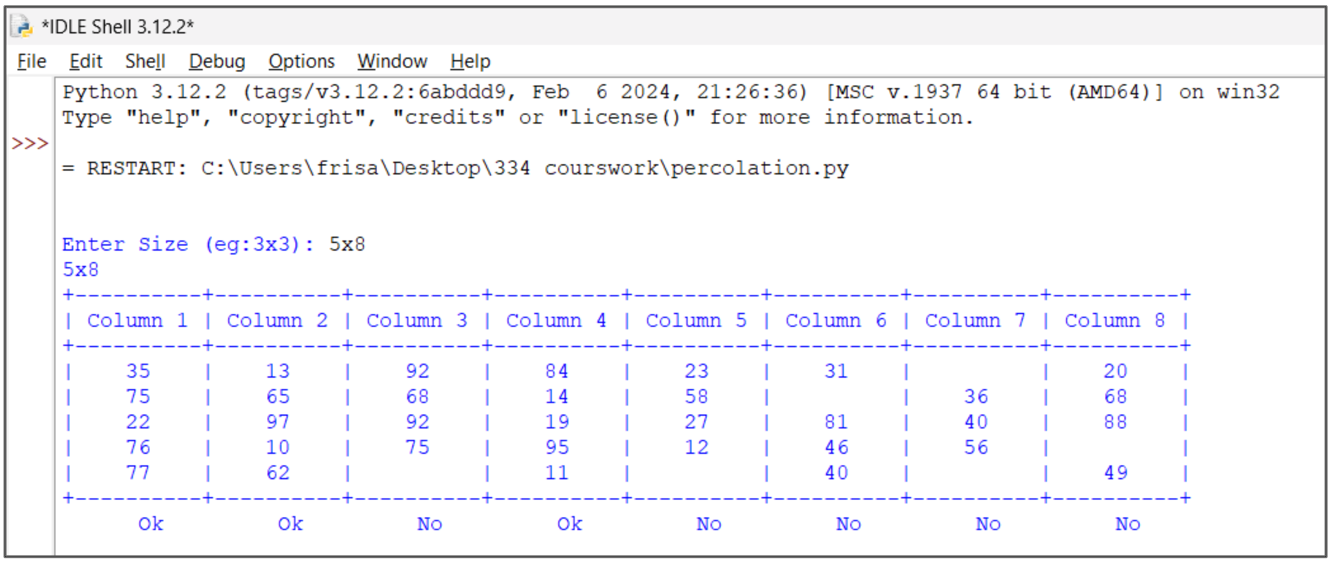
Test case. No.	Input	Expected outcome	Actual outcome	Result
1	When user run the program	The word "Enter Size (eg:3x3:)" should be pop up.	As Expected,	Pass
2	When user Input's grid size	It will print the grid according to the user inputs.	As Expected,	Pass
Test case. No.	**Test**	**Expected outcome**	**Actual outcome**	**Result**
3	If the user failed or didn't input a grid size	It will print "5x5" grid (default)	As Expected,	Pass
4	If the user failed to input in between the range of "3x3" to "9x9"	It will Conway a proper error massage	As Expected,	Pass
5	Review the table status whether its empty or not	It will display the "Ok" or "No" according to the percolation status	As Expected,	Pass
6	When user try it run on command prompt	It will successfully run it	As Expected,	Pass

Table 2 Test cases



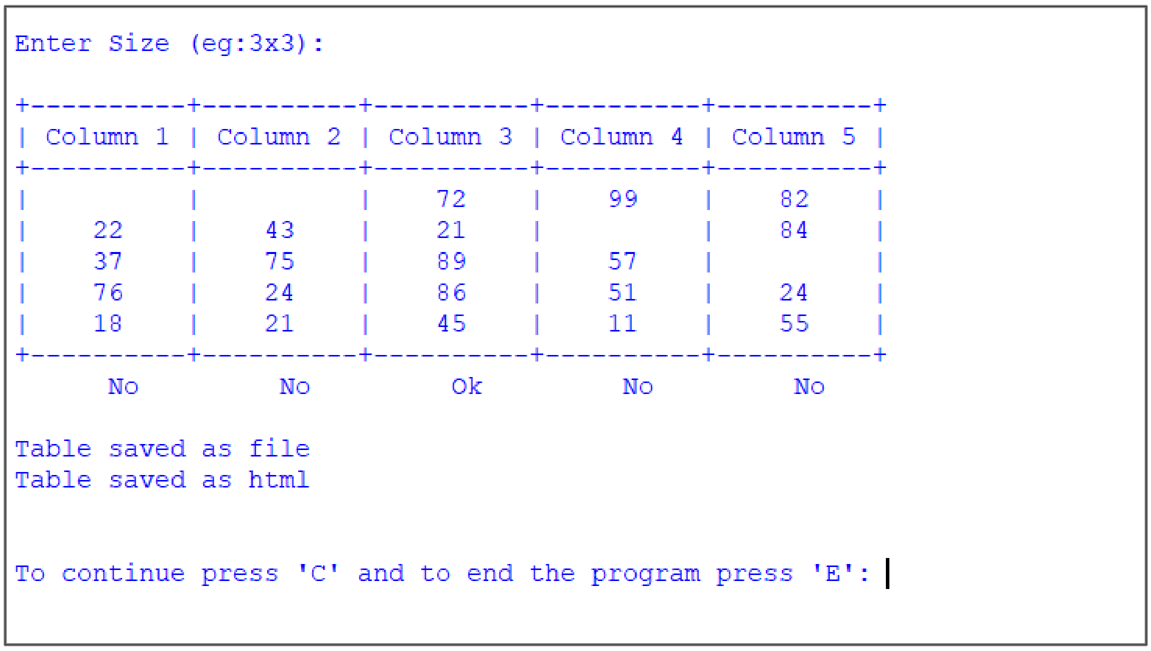
```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\frisa\Desktop\334 courswork\percolation.py


Enter Size (eg:3x3):
```

Figure 1: Test Case No. 1



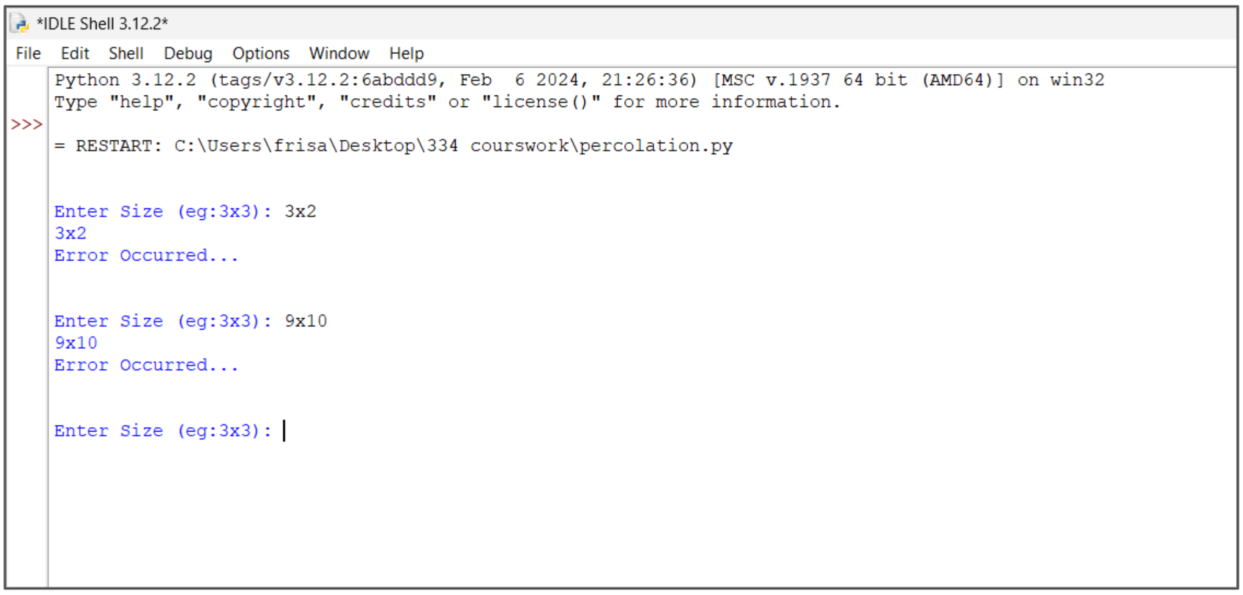
```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\frisa\Desktop\334 coursework\percolation.py


Enter Size (eg:3x3): 5x8
5x8
+----------+----------+----------+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 | Column 8 |
+----------+----------+----------+----------+----------+----------+----------+----------+
|    35    |    13    |    92    |    84    |    23    |    31    |          |    20    |
|    75    |    65    |    68    |    14    |    58    |          |    36    |    68    |
|    22    |    97    |    92    |    19    |    27    |    81    |    40    |    88    |
|    76    |    10    |    75    |    95    |    12    |    46    |    56    |          |
|    77    |    62    |          |    11    |          |    40    |          |    49    |
+----------+----------+----------+----------+----------+----------+----------+----------+
      Ok         Ok         No         Ok         No         No         No         No
```

Figure 2:Test Case No.2



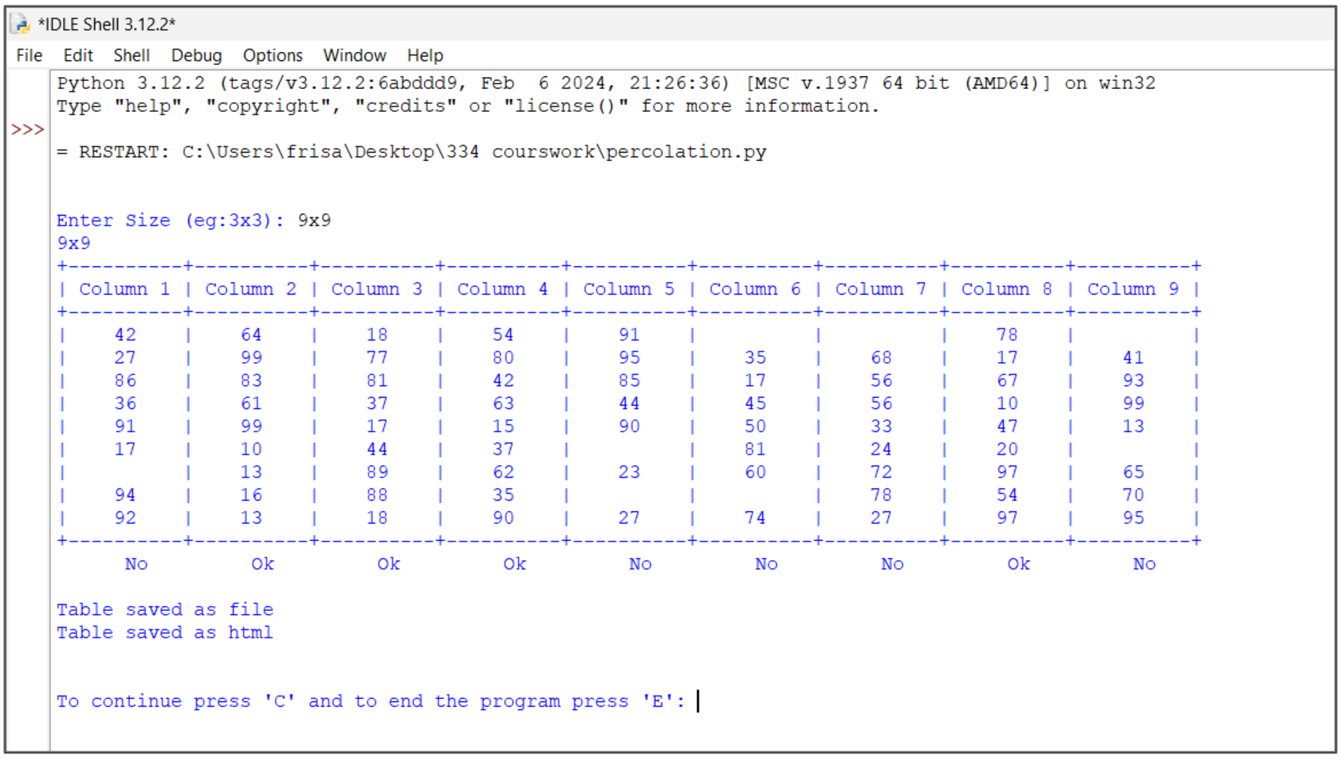
```
Enter Size (eg:3x3):

+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 |
+----------+----------+----------+----------+----------+
|          |          |    72    |    99    |    82    |
|    22    |    43    |    21    |          |    84    |
|    37    |    75    |    89    |    57    |          |
|    76    |    24    |    86    |    51    |    24    |
|    18    |    21    |    45    |    11    |    55    |
+----------+----------+----------+----------+----------+
      No         No         Ok         No         No

Table saved as file
Table saved as html


To continue press 'C' and to end the program press 'E':
```

Figure 3: Test Case No.3



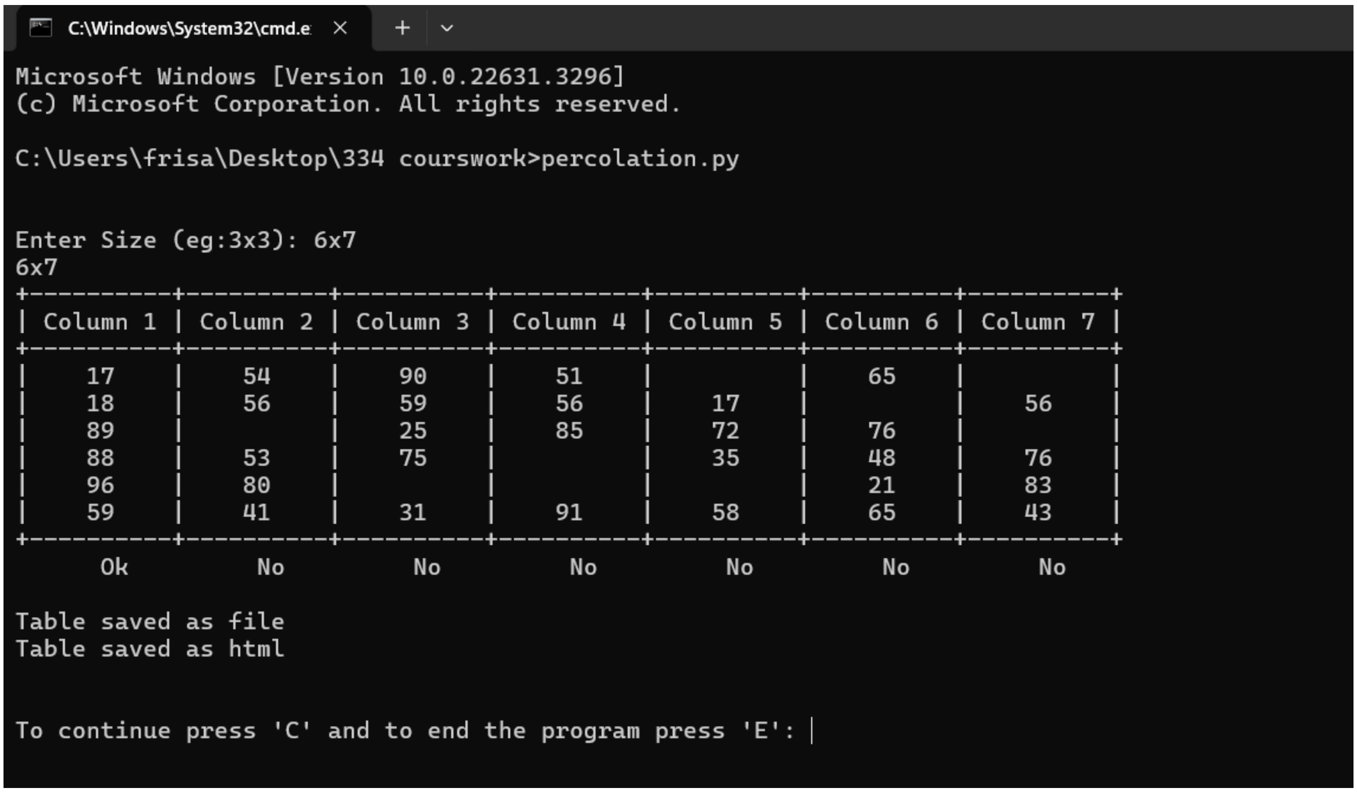
```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\frisa\Desktop\334 courswork\percolation.py


Enter Size (eg:3x3): 3x2
3x2
Error Occurred...


Enter Size (eg:3x3): 9x10
9x10
Error Occurred...


Enter Size (eg:3x3): 
```

Figure 4: Test Case No.4



```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\frisa\Desktop\334 courswork\percolation.py


Enter Size (eg:3x3): 9x9
9x9
+----------+----------+----------+----------+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 | Column 8 | Column 9 |
+----------+----------+----------+----------+----------+----------+----------+----------+----------+
|    42    |    64    |    18    |    54    |    91    |          |          |    78    |          |
|    27    |    99    |    77    |    80    |    95    |    35    |    68    |    17    |    41    |
|    86    |    83    |    81    |    42    |    85    |    17    |    56    |    67    |    93    |
|    36    |    61    |    37    |    63    |    44    |    45    |    56    |    10    |    99    |
|    91    |    99    |    17    |    15    |    90    |    50    |    33    |    47    |    13    |
|    17    |    10    |    44    |    37    |          |    81    |    24    |    20    |          |
|          |    13    |    89    |    62    |    23    |    60    |    72    |    97    |    65    |
|    94    |    16    |    88    |    35    |          |          |    78    |    54    |    70    |
|    92    |    13    |    18    |    90    |    27    |    74    |    27    |    97    |    95    |
+----------+----------+----------+----------+----------+----------+----------+----------+----------+
      No         Ok         Ok         Ok         No         No         No         Ok         No

Table saved as file
Table saved as html


To continue press 'C' and to end the program press 'E': 
```

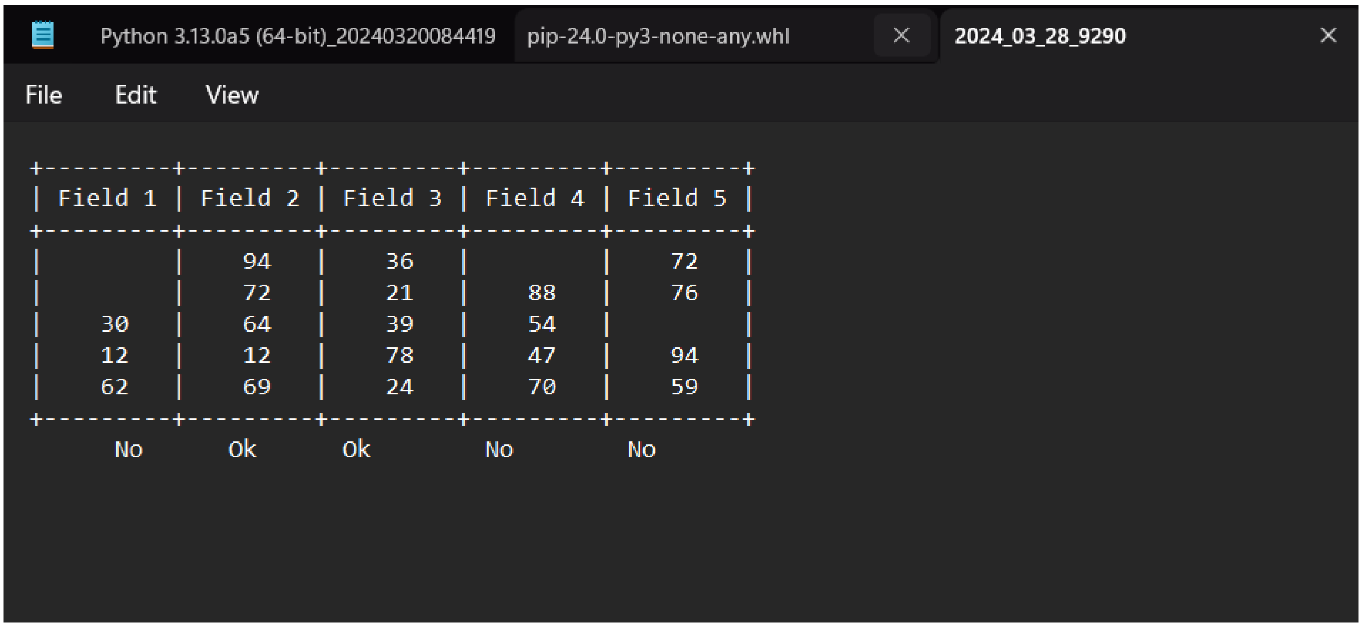
Figure 5: Test Case No.5



```
Microsoft Windows [Version 10.0.22631.3296]
(c) Microsoft Corporation. All rights reserved.

C:\Users\frisa\Desktop\334 courswork>percolation.py


Enter Size (eg:3x3): 6x7
6x7
+----------+----------+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
+----------+----------+----------+----------+----------+----------+----------+
|    17    |    54    |    90    |    51    |          |    65    |          |
|    18    |    56    |    59    |    56    |    17    |          |    56    |
|    89    |          |    25    |    85    |    72    |    76    |          |
|    88    |    53    |    75    |          |    35    |    48    |    76    |
|    96    |    80    |          |          |          |    21    |    83    |
|    59    |    41    |    31    |    91    |    58    |    65    |    43    |
+----------+----------+----------+----------+----------+----------+----------+
      Ok         No         No         No         No         No         No

Table saved as file
Table saved as html


To continue press 'C' and to end the program press 'E':
```

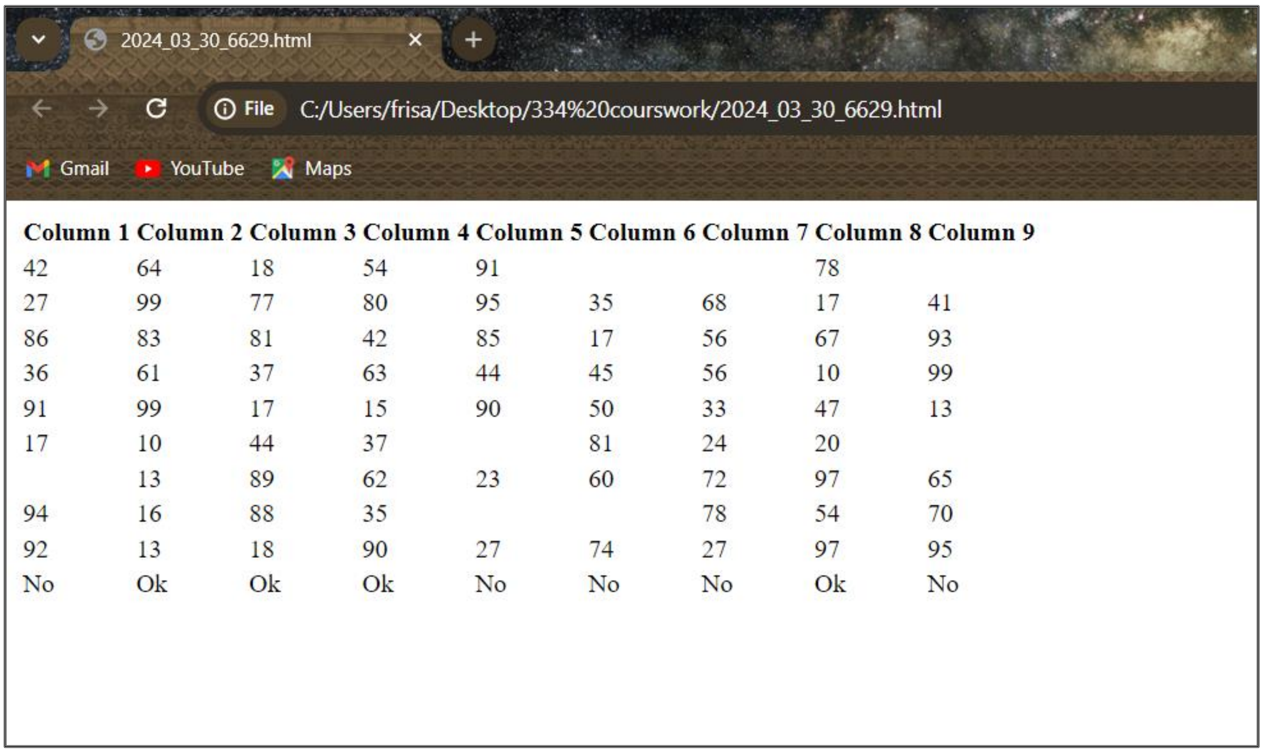
Figure 6: Test Case No.6

Test case. No.	Test	Expected outcome	Actual outcome	Result
7	According to the grid size, it saves as a text file	Automatically saves pretty table and the percolation status and it will pop up " Table saved as file"	As Expected,	Pass
8	According to the grid size it saves as a HTML file	Automatically saves the percolation status and it will pop up "Table saved as html"	As Expected,	Pass
9	Asking user to loop the program by "C" and end the program by "E"	If user select "C" the program will continue and if the user inputs "E" it will end	As Expected,	Pass
10	If the user inputs other than "C" and "E"	It will show an error massage and repeat pop up the user choice options	As Expected,	Pass


Python 3.13.0a5 (64-bit)_20240320084419
pip-24.0-py3-none-any.whl
2024_03_28_9290
File
Edit
View

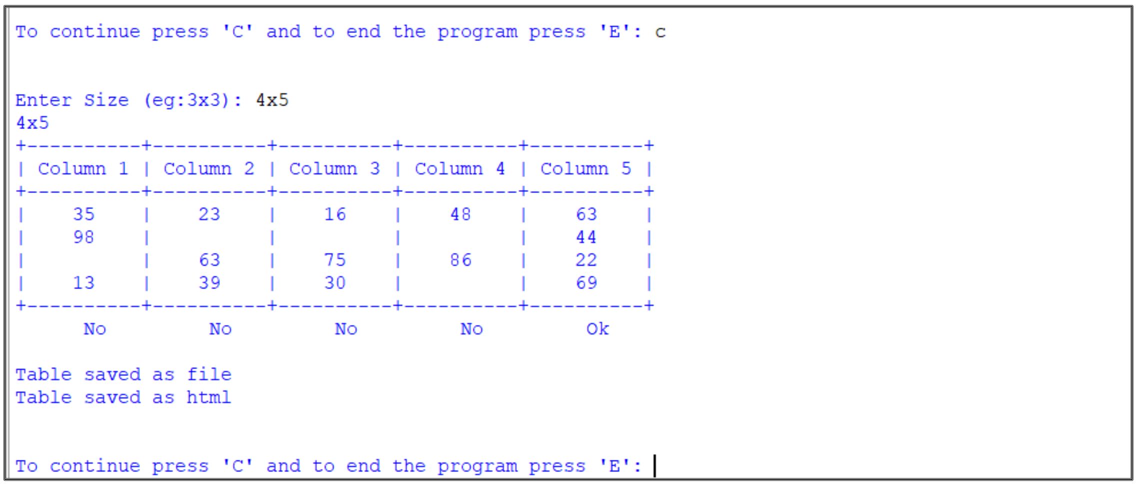
```
+---------+---------+---------+---------+---------+
| Field 1 | Field 2 | Field 3 | Field 4 | Field 5 |
+---------+---------+---------+---------+---------+
|         |    94   |    36   |         |    72   |
|         |    72   |    21   |    88   |    76   |
|    30   |    64   |    39   |    54   |         |
|    12   |    12   |    78   |    47   |    94   |
|    62   |    69   |    24   |    70   |    59   |
+---------+---------+---------+---------+---------+
      No       Ok       Ok         No         No
```

Figure 7: Test Case No.7



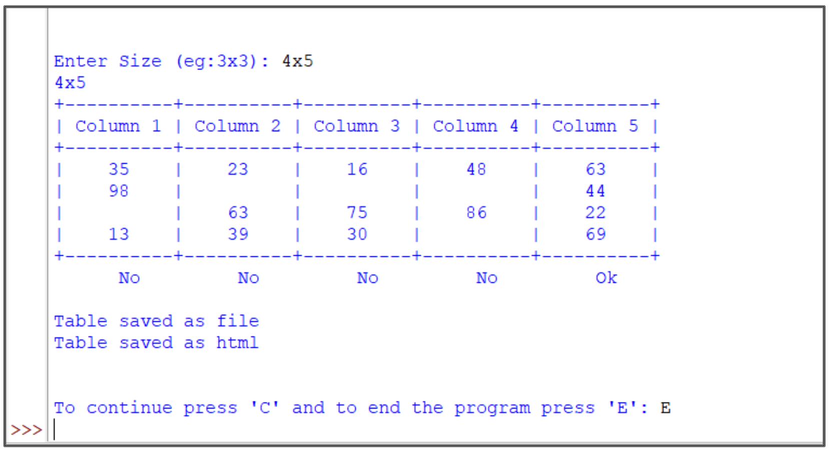
Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8	Column 9
42	64	18	54	91			78	
27	99	77	80	95	35	68	17	41
86	83	81	42	85	17	56	67	93
36	61	37	63	44	45	56	10	99
91	99	17	15	90	50	33	47	13
17	10	44	37		81	24	20	
	13	89	62	23	60	72	97	65
94	16	88	35			78	54	70
92	13	18	90	27	74	27	97	95
No	Ok	Ok	Ok	No	No	No	Ok	No

Figure 9: Test Case No.8



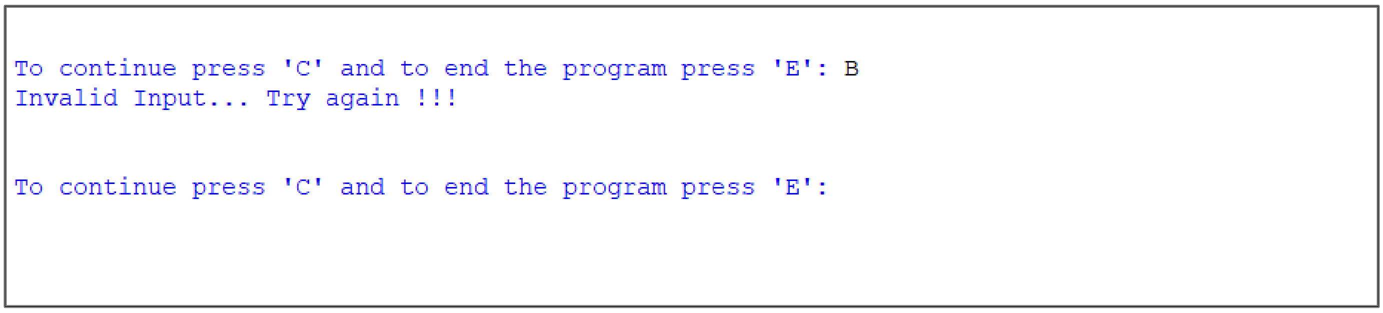
```
To continue press 'C' and to end the program press 'E': c


Enter Size (eg:3x3): 4x5
4x5
+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 |
+----------+----------+----------+----------+----------+
|    35    |    23    |    16    |    48    |    63    |
|    98    |          |          |          |    44    |
|          |    63    |    75    |    86    |    22    |
|    13    |    39    |    30    |          |    69    |
+----------+----------+----------+----------+----------+
      No         No         No         No         Ok

Table saved as file
Table saved as html


To continue press 'C' and to end the program press 'E': 
```

Figure 8: Test Case No.09



```
Enter Size (eg:3x3): 4x5
4x5
+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 |
+----------+----------+----------+----------+----------+
|    35    |    23    |    16    |    48    |    63    |
|    98    |          |          |          |    44    |
|          |    63    |    75    |    86    |    22    |
|    13    |    39    |    30    |          |    69    |
+----------+----------+----------+----------+----------+
      No         No         No         No         Ok

Table saved as file
Table saved as html


To continue press 'C' and to end the program press 'E': E
>>> 
```

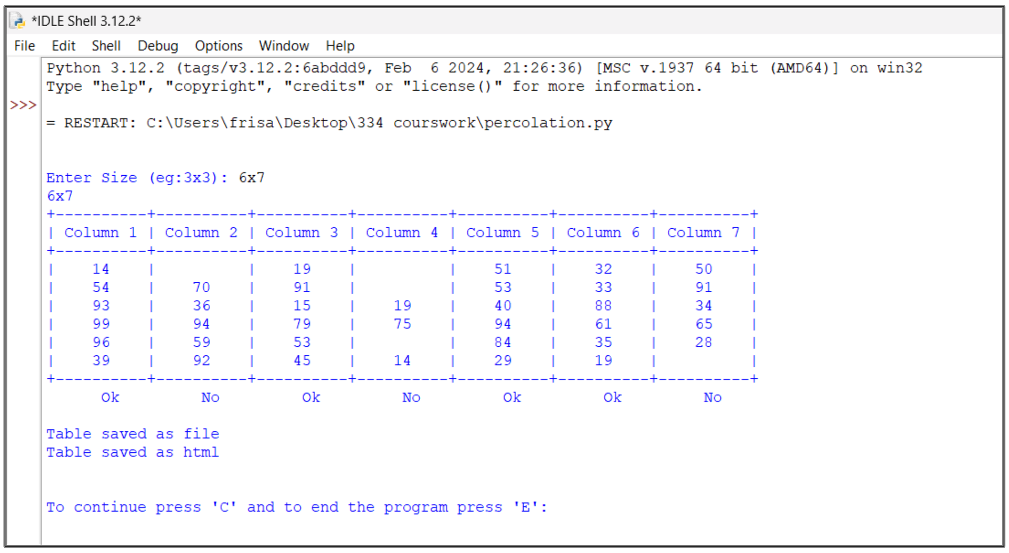
Figure 10: Test Case No.9



```
To continue press 'C' and to end the program press 'E': B
Invalid Input... Try again !!!


To continue press 'C' and to end the program press 'E':
```

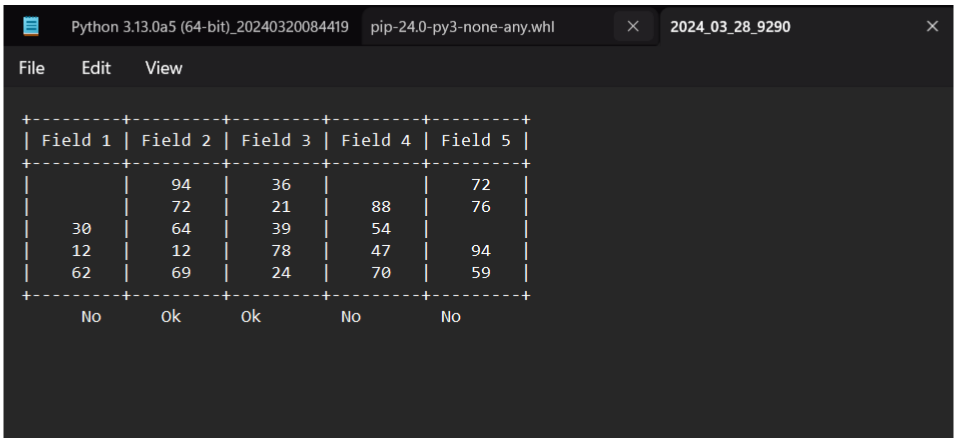
Figure 11: Test Case No.10

05.Program Output



```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\frisa\Desktop\334 courswork\percolation.py


Enter Size (eg:3x3): 6x7
6x7
+----------+----------+----------+----------+----------+----------+----------+
| Column 1 | Column 2 | Column 3 | Column 4 | Column 5 | Column 6 | Column 7 |
+----------+----------+----------+----------+----------+----------+----------+
|    14    |          |    19    |          |    51    |    32    |    50    |
|    54    |    70    |    91    |          |    53    |    33    |    91    |
|    93    |    36    |    15    |    19    |    40    |    88    |    34    |
|    99    |    94    |    79    |    75    |    94    |    61    |    65    |
|    96    |    59    |    53    |          |    84    |    35    |    28    |
|    39    |    92    |    45    |    14    |    29    |    19    |          |
+----------+----------+----------+----------+----------+----------+----------+
      Ok         No         Ok         No         Ok         Ok         No

Table saved as file
Table saved as html


To continue press 'C' and to end the program press 'E':
```

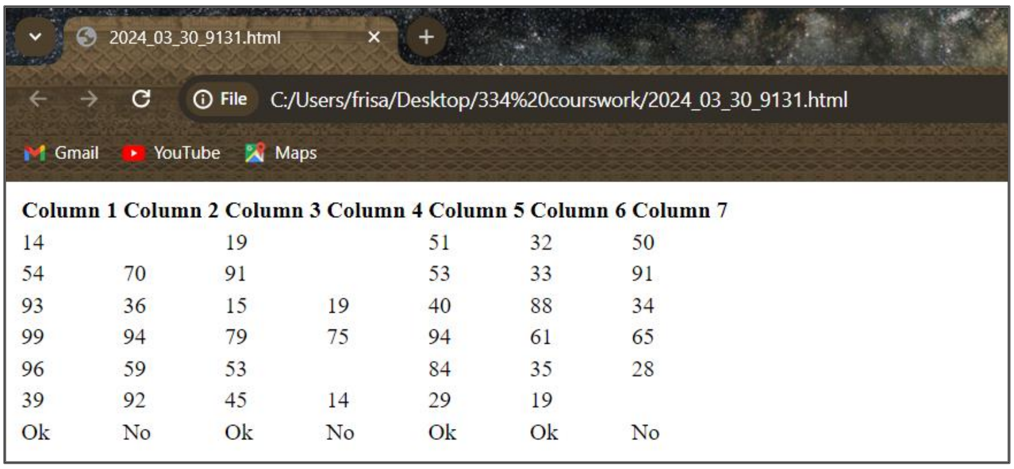
Figure 12: Outcome.



Python 3.13.0a5 (64-bit)_20240320084419
pip-24.0-py3-none-any.whl
2024_03_28_9290
File Edit View

```
+---------+---------+---------+---------+---------+
| Field 1 | Field 2 | Field 3 | Field 4 | Field 5 |
+---------+---------+---------+---------+---------+
|         |   94    |   36    |         |   72    |
|         |   72    |   21    |   88    |   76    |
|   30    |   64    |   39    |   54    |         |
|   12    |   12    |   78    |   47    |   94    |
|   62    |   69    |   24    |   70    |   59    |
+---------+---------+---------+---------+---------+
      No       Ok      Ok        No        No
```

Figure 13: Outcome.



C:/Users/frisa/Desktop/334%20courswork/2024_03_30_9131.html

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7
14		19		51	32	50
54	70	91		53	33	91
93	36	15	19	40	88	34
99	94	79	75	94	61	65
96	59	53		84	35	28
39	92	45	14	29	19	
Ok	No	Ok	No	Ok	Ok	No

Figure 14: Outcome.