INFORMATICS INSTITUTE OF TECHNOLOGY DEPARTMENT OF BUSINESS MANAGEMENT

DOC 333 – Introduction to Programming

COURSEWORK – 2023/24

(Foundation September 2023)

Module Leader:	Mr. Nishan Saliya Harankahawa
Submission Date:	25/3/2024

Full Name	Mr. A.A.S. Amer
Student ID	20231461
Course	Foundation Course 2nd Semester
Level	3

Table of Contents

List of Figures

List of Tables

Algorithm for the Python Percolation Process Code

1. Input Grid Dimensions

- If command-line arguments are provided, extract the grid dimensions from `sys.argv[1]`.
- If no arguments are provided, set default grid dimensions to "5x5".
- Ensure that the grid dimensions are within the range of 3x3 and 9x9. If not, print a message and return.

2. Generate Grid:

- Call the `generate_grid(rows, cols)` function from the `process` module.
- Inside `generate_grid()` function:
- Initialize an empty list `grid`.
- Iterate over each row (specified by `rows`):
- For each row, initialize an empty list `row`.
- Iterate over each column (specified by `cols`):
- Generate a random number.
- If the random number falls below 0.3, append `None` (empty cell) to the row.
- Otherwise, append a random integer between 10 and 99 to the row.
- Append the row to the `grid`.
- Call the `display_grid(grid)` function to display the grid.

3. Display Grid:

- Inside `display_grid()` function:
- Initialize a `PrettyTable` instance.
- Configure table properties (style, header, padding).
- Iterate over each row in the grid:
- Initialize an empty list `table_row`.
- Iterate over each cell in the row:
- If the cell is not `None`, append its value to `table_row`. Otherwise, append an empty string.
- Add `table_row` to the table and prints it.
- Display indicators below each column to show if it's filled (`OK`) or not (`NO`).
- Append the indicators to the last row of the grid.
- Call `text.save_txt(grid)` to save the grid to a text file.

4. Save Grid to Text File:

- Inside `text.save_txt(grid)` function:
- Generate a unique filename based on the current date and time.
- Open a file with the generated filename in write mode.
- Iterate over each row in the grid:
- Iterate over each cell in the row:
- Write the cell value (or empty space if `None`) followed by three spaces.
- Write a newline character to move to the next row.
- Close the file.

5. Save Grid to HTML File:

- Inside `text.save_html(table, rows, columns, ok_no)` function:
- Generate a unique filename based on the current date and time.
- Open a file with the generated filename in write mode.
- Write the HTML representation of the table, applying styles for borders, width, and height.
- Write the indicators (OK/NO) below each column.
- Close the file.

This algorithm outlines the main steps performed by the provided code, including generating a grid, displaying it, and saving it to both text and HTML files.

Structure of the Python Code

The Python code consists of three main components:

1. Main Script (`perc.py`):
 - Entry point of the program.
 - Calls functions from the `process` module.

2. Process Module (`process.py`):
 - process.py is a module which was imported to the main python file
 - Generates and displays a grid.
 - Handles grid dimension input and validity checks.
 - Saves the grid to a text file.

3. Text and HTML Handling Functions (`text_html`):
 - text_html is a folder which contains html.py and text.py
 - text_html package is imported to process.py
 - Contains functions to save grids to text and HTML files.

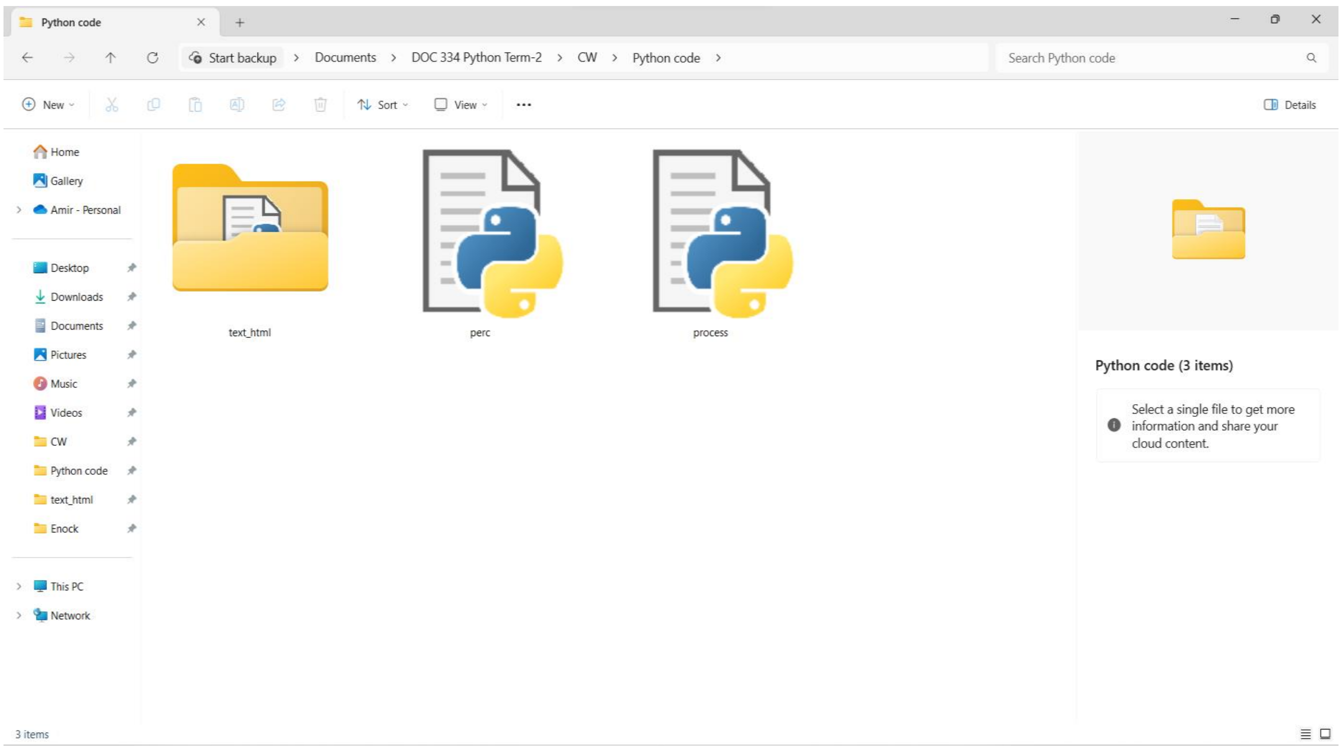


Figure 1: Main python folder.

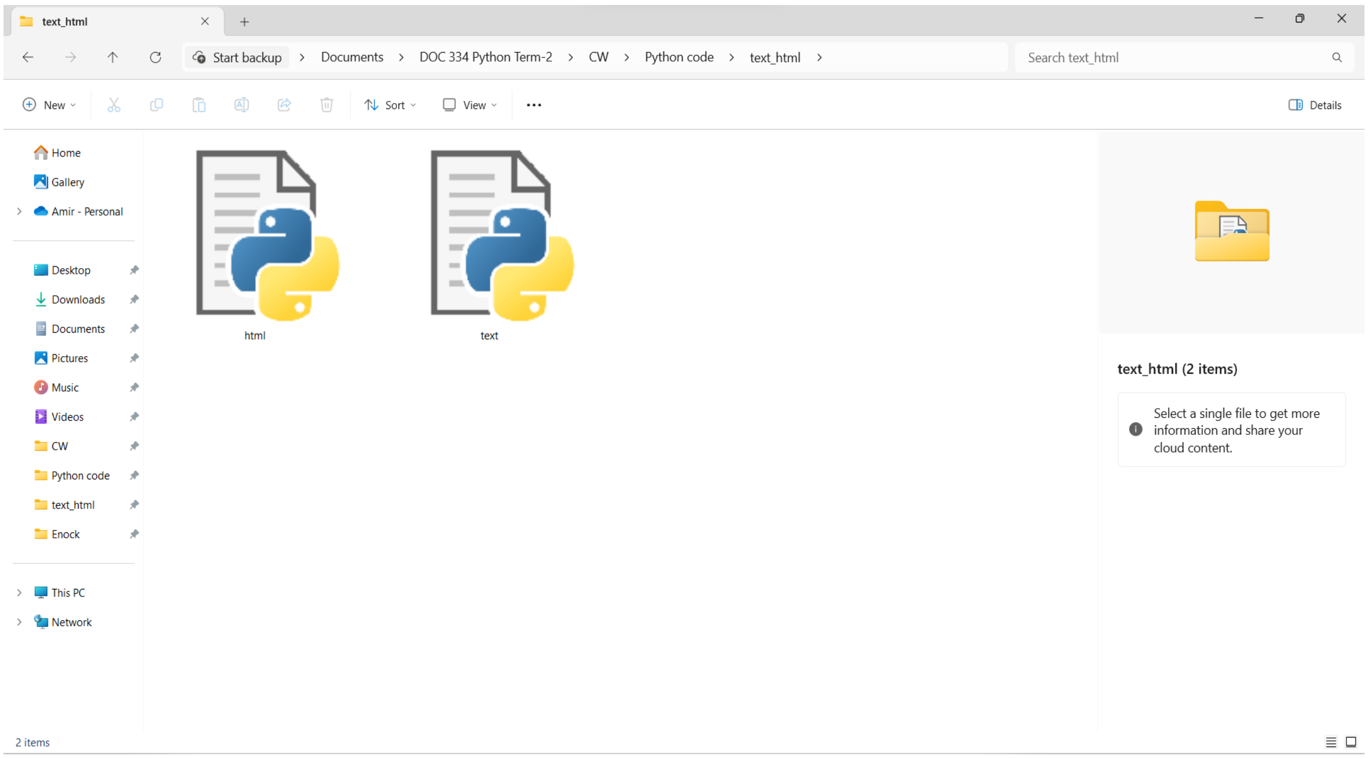


Figure 2: Sub folder called text_html in main python folder

Test Cases

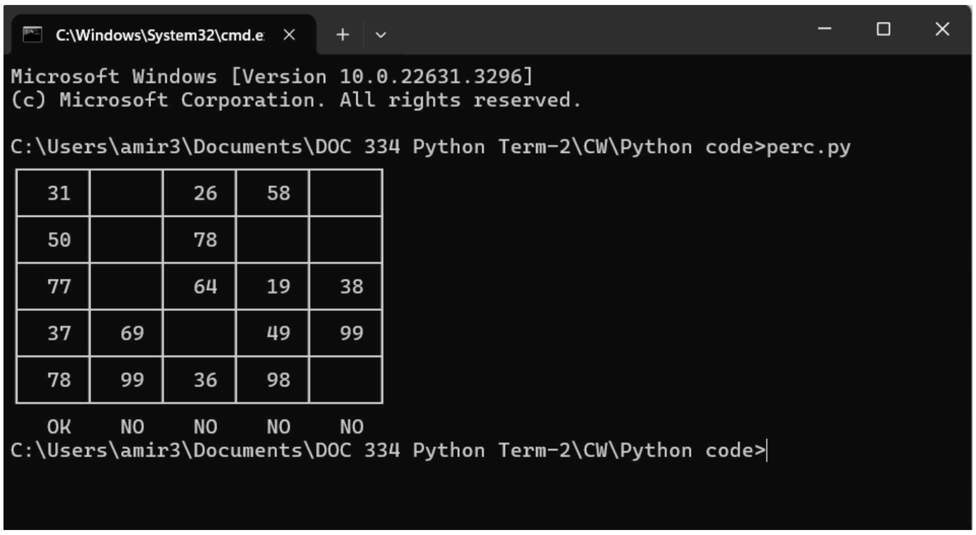
Test Case 1: Command prompt

1.1.Table test case in command prompt.

NO.	Input	Expected Outcomes	Actual Outcomes	Result
1.	perc.py	To show a default 5x5 grid and check whether percolation is possible	Shows a default 5x5 grid and checks whether percolation is possible.	Pass
2.	perc.py 9x9	To show a 9x9 grid and check whether percolation is possible	Shows a 9x9 grid and checks whether percolation is possible	Pass
3.	perc.py 10x13	The above will generate a proper error message to user	The above generates a proper error message to user	Pass

Table 1: Table test case in command prompt.

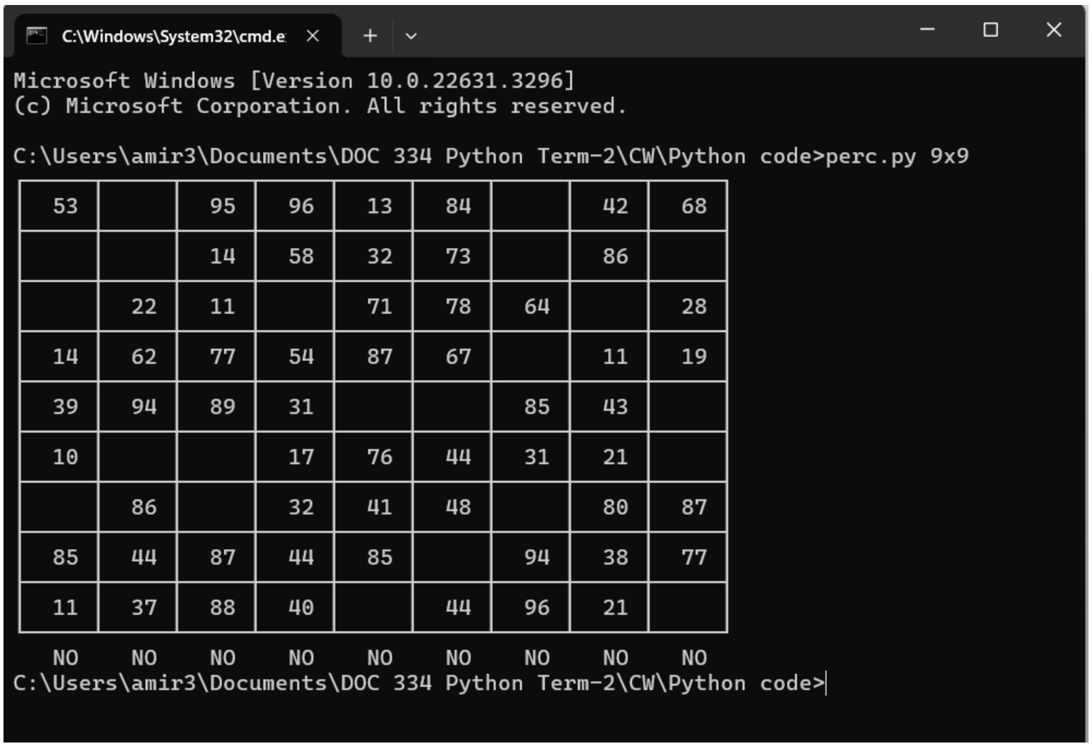
1.2.Test Screenshot



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

C:\Users\amir3\Documents\DOC 334 Python Term-2\CW\Python code>perc.py
| 31 |    | 26 | 58 |    |
| 50 |    | 78 |    |    |
| 77 |    | 64 | 19 | 38 |
| 37 | 69 |    | 49 | 99 |
| 78 | 99 | 36 | 98 |    |
  OK   NO   NO   NO   NO
C:\Users\amir3\Documents\DOC 334 Python Term-2\CW\Python code>
```

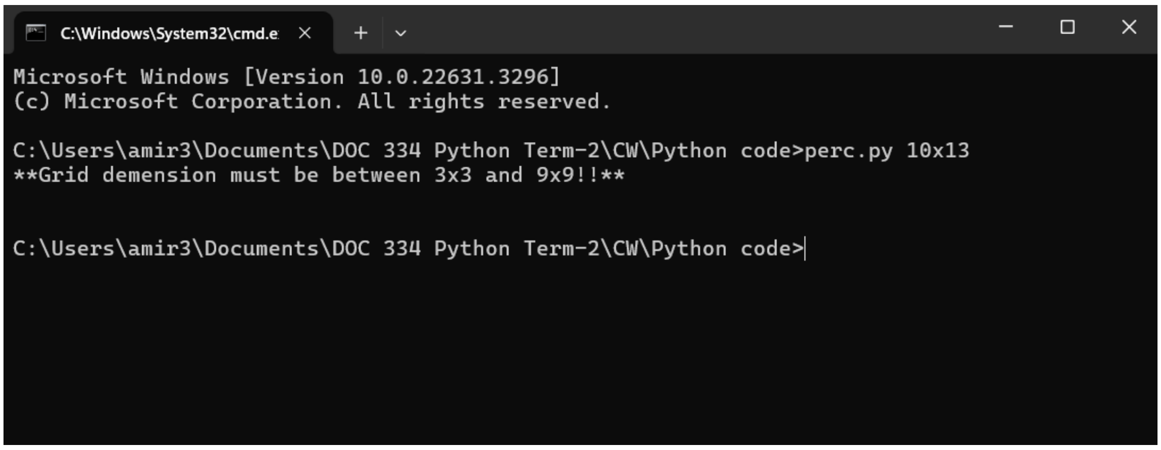
Figure 3: When perc.py is typed in command prompt.



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

C:\Users\amir3\Documents\DOC 334 Python Term-2\CW\Python code>perc.py 9x9
| 53 |    | 95 | 96 | 13 | 84 |    | 42 | 68 |
|    |    | 14 | 58 | 32 | 73 |    | 86 |    |
|    | 22 | 11 |    | 71 | 78 | 64 |    | 28 |
| 14 | 62 | 77 | 54 | 87 | 67 |    | 11 | 19 |
| 39 | 94 | 89 | 31 |    |    | 85 | 43 |    |
| 10 |    |    | 17 | 76 | 44 | 31 | 21 |    |
|    | 86 |    | 32 | 41 | 48 |    | 80 | 87 |
| 85 | 44 | 87 | 44 | 85 |    | 94 | 38 | 77 |
| 11 | 37 | 88 | 40 |    | 44 | 96 | 21 |    |
  NO   NO   NO   NO   NO   NO   NO   NO   NO
C:\Users\amir3\Documents\DOC 334 Python Term-2\CW\Python code>
```

Figure 4: When perc.py 9x9 is typed in command prompt.



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

C:\Users\amir3\Documents\DOC 334 Python Term-2\CW\Python code>perc.py 10x13
**Grid demension must be between 3x3 and 9x9!!**


C:\Users\amir3\Documents\DOC 334 Python Term-2\CW\Python code>
```

Figure 5: When perc.py 10x13 is typed in command prompt.

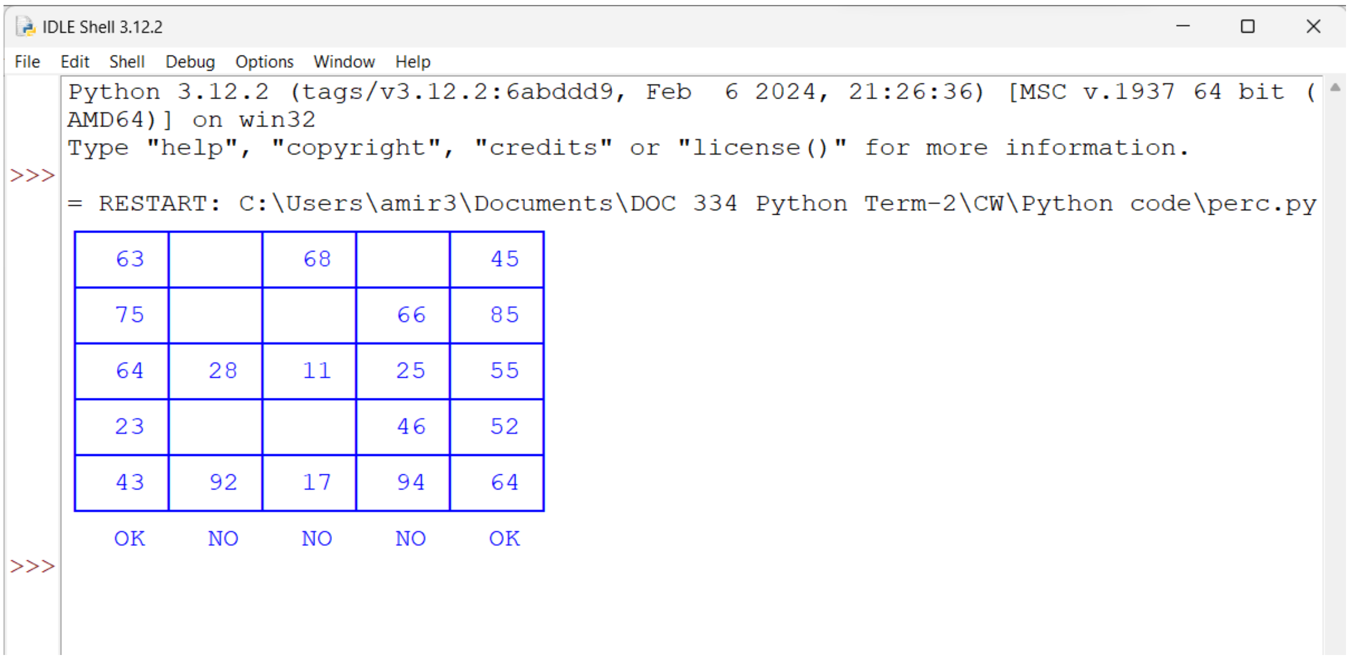
Test Case 2: Python Code

2.1.Table test case in python IDLE.

NO.	Input	Expected Outcomes	Actual Outcomes	Result
1.	When python code is runed in IDLE	To show a default 5x5 grid and check whether percolation is possible	Shows a default 5x5 grid and checks whether percolation is possible.	Pass

Table 2: Table test case in python IDLE.

2.2.Test Screenshot



```
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\amir3\Documents\DOC 334 Python Term-2\CW\Python code\perc.py
 63        68        45
 75             66   85
 64   28   11   25   55
 23             46   52
 43   92   17   94   64
 OK   NO   NO   NO   OK
>>>
```

Figure 6: When python code is run in IDLE.

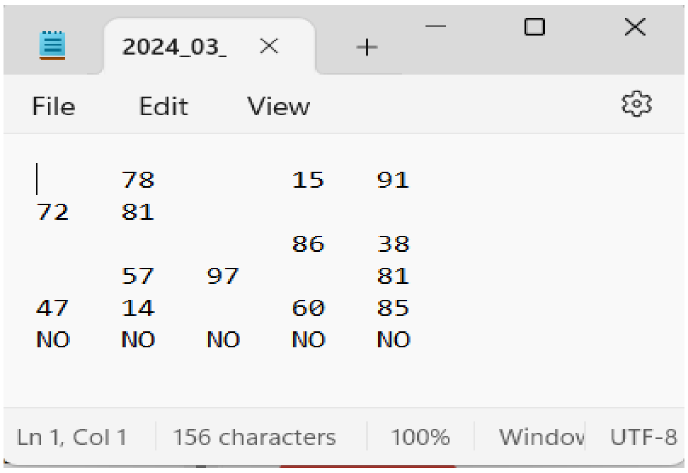
Test Case 3: Text and html file

3.1.Table test case for Text and html file.

NO.	Input	Expected Outcomes	Actual Outcomes	Result
1.	perc.py	To save a default 5x5 grid and check whether percolation is possible and save it in a text and html file	Shows a default 5x5 grid and checks whether percolation is possible and saves it in a text and html file	Pass
2.	perc.py 9x9	To show a 9x9 grid and check whether percolation is possible and save it in a text and html file	Shows a 9x9 grid and checks whether percolation is possible and saves it in a text and html file	Pass

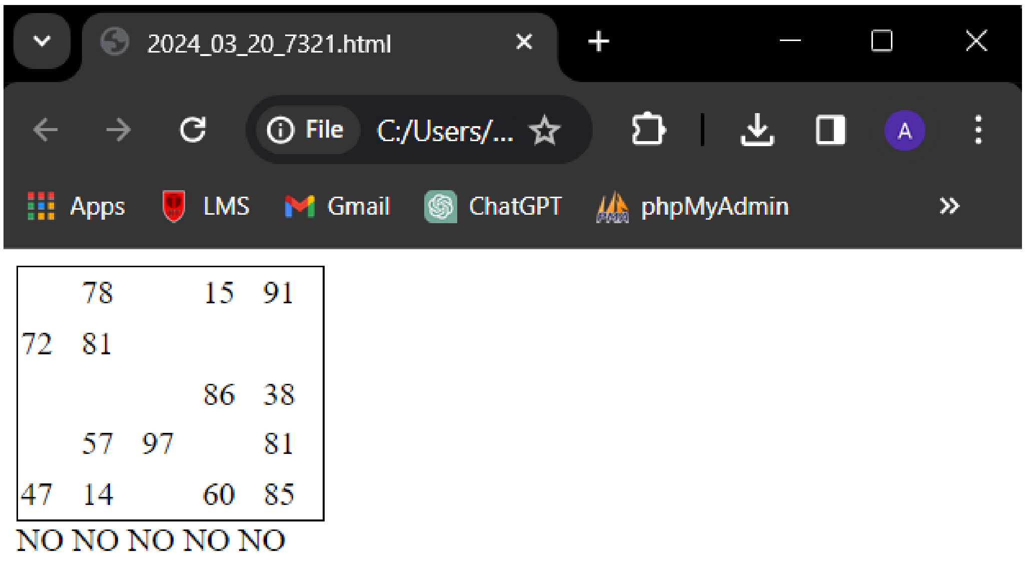
Table 3: Table test case for Text and html file.

3.2.Test Screenshot



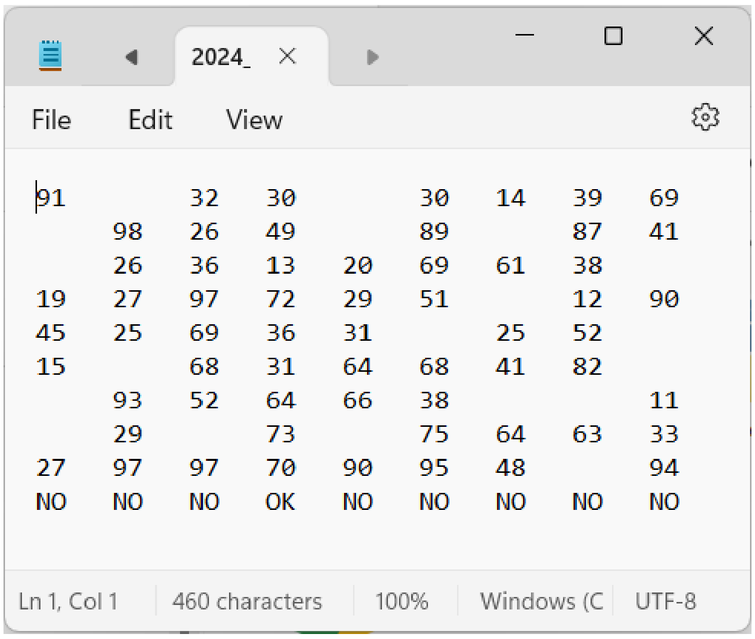
```
    78      15  91
72  81
            86  38
    57  97      81
47  14      60  85
NO  NO  NO  NO  NO
```

Figure 7: Saves the default 5x5 grid in text file.



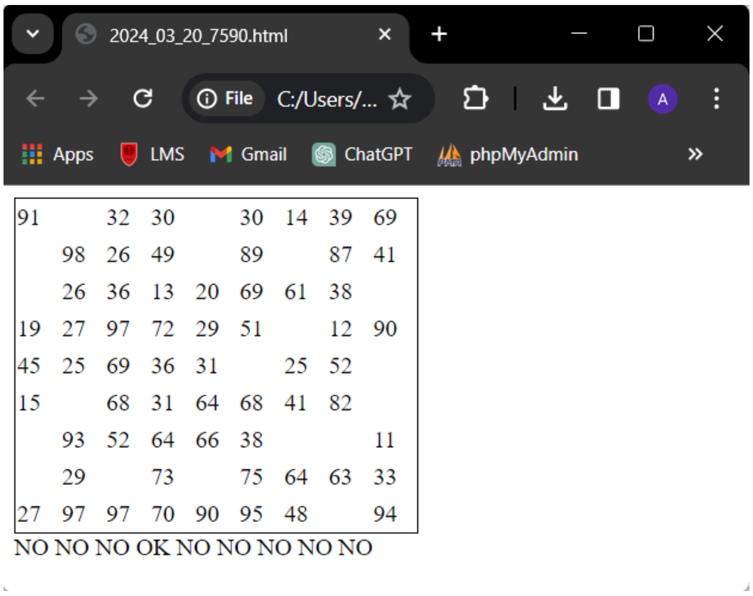
```
    78      15  91
72  81
            86  38
    57  97      81
47  14      60  85
NO NO NO NO NO
```

Figure 8: : Saves the default 5x5 grid in html file.



```
91      32  30      30  14  39  69
    98  26  49      89      87  41
    26  36  13  20  69  61  38
19  27  97  72  29  51      12  90
45  25  69  36  31      25  52
15      68  31  64  68  41  82
    93  52  64  66  38          11
    29      73      75  64  63  33
27  97  97  70  90  95  48      94
NO  NO  NO  OK  NO  NO  NO  NO  NO
```

Figure 9: : Saves the 9x9 grid in text file.



91		32	30		30	14	39	69
	98	26	49		89		87	41
	26	36	13	20	69	61	38	
19	27	97	72	29	51		12	90
45	25	69	36	31		25	52	
15		68	31	64	68	41	82	
	93	52	64	66	38			11
	29		73		75	64	63	33
27	97	97	70	90	95	48		94

NO NO NO OK NO NO NO NO NO

Figure 10: Saves the 9x9 grid in html file.

Test Case 4: To save Text and html file

4.1.Table test case for saving Text and html file.

NO.	Expected Outcomes	Actual Outcomes	Result
1.	To save the text file named as the current date and random numbers	Saves the text file named as the current date and random numbers	Pass
2.	To save the html file named as the current date and random numbers	Saves the text file named as the current date and random numbers	Pass

Table 4: Table test case for saving Text and html file.

4.2.Test Screenshot

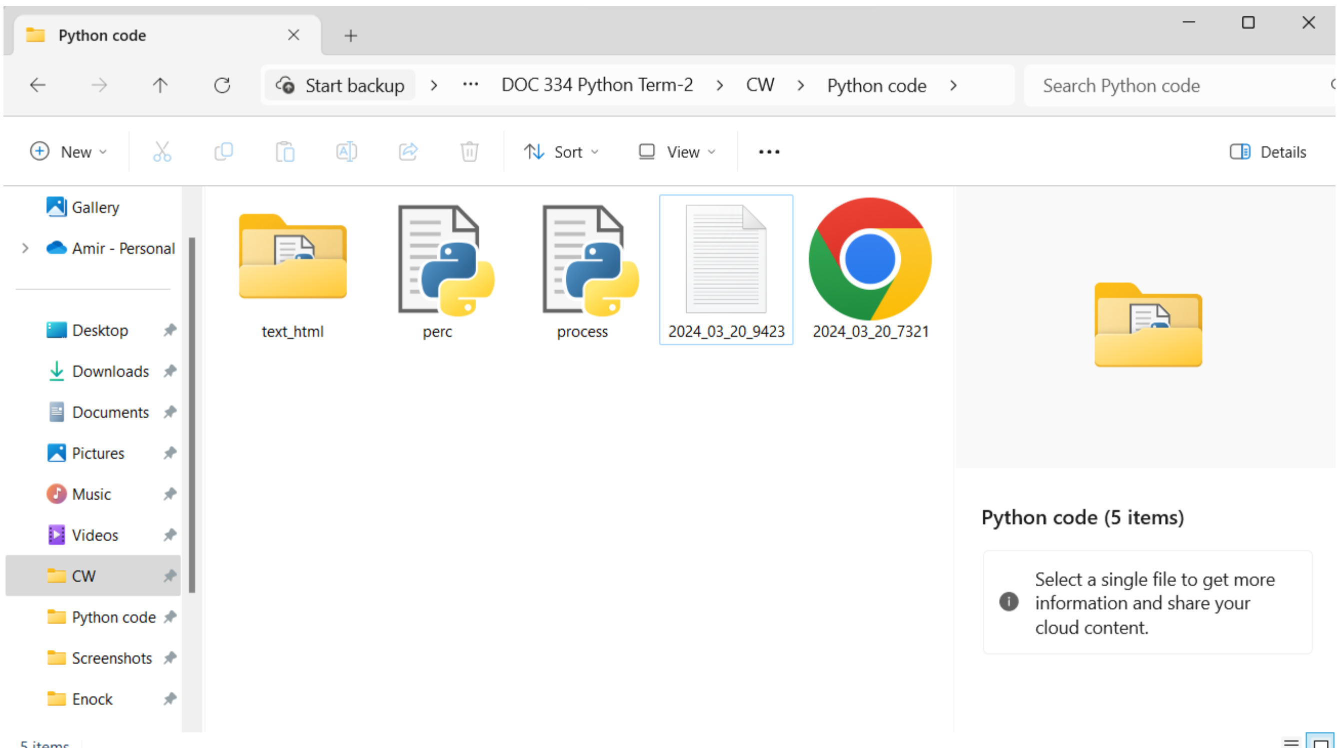


Figure 11: Nameing for text and html file.

Python Codes

perc.py (main)

```
import process
import sys
def grid_demension():
    if len(sys.argv) > 1:
        grid = sys.argv[1]
    else :
        grid="5x5"
    split = grid.split('x')
    cols = int(split[0])
    rows = int(split[1])
    if cols < 2 or cols > 9 or rows < 2
or rows > 9:
        print("**Grid demension must
be between 3x3 and 9x9!!**")
        print()
        return
    process.generate_grid(rows,cols)




grid_demension()
```

process.py

```
import random
import sys
import os
from datetime import datetime
import prettytable
from text_html import text
from text_html import html

def generate_grid(rows, cols):
    row_num=rows
    col_num=cols
    grid = []
    for _ in range(rows):
        row = []
        for _ in range(cols):
            if random.random() < 0.3:  # 30% chance of an empty cell
                row.append(None)
            else:
                row.append(random.randint(10, 99))
        grid.append(row)
    display_grid(grid, row_num,col_num)

def display_grid(grid,row_num,col_num):
    rows=row_num
    cols=col_num
    table = prettytable.PrettyTable()
    table.set_style(prettytable.SINGLE_BORDER)
    table.hrules = prettytable.ALL
    table.left_padding_width=2
    table.header = False
    num_columns = len(grid[0])
    filled_columns = []
    for i in range(num_columns):
        column_filled = all(row[i] is not None for row in grid);
        filled_columns.append(column_filled);

    for row in grid:
        table_row = []
        for cell in row:
            if cell != None:
                table_row.append(cell)
            else:
                table_row.append('')
        table.add_row(table_row)
    # Display the table
    print(table)

    # Display the indicators below each column
    ok_no=[]#Creating a list to save ok or no for text file
    for i in range(num_columns):
        if filled_columns[i]:
            indicator = '   OK '
            ok_no.append("OK")
        else:
            indicator = '   NO '
            ok_no.append("NO")
        print(indicator, end="")
    grid.append(ok_no)
    text.save_txt(grid)
    html.save_html(table,rows,cols,ok_no)
```

text_html (folder)

text.py

```
import random
from datetime import datetime



def save_txt (grid):
    now = datetime.now()
    filename = now.strftime("%Y_%m_%d_") + str(random.randint(1000, 9999)) + ".txt"
    file=open(filename,'w')
    for i in grid :
        for x in i:
            num=str(x)
            if num == "None":
                file.write("     ")
            else:
                file.write(num)
                file.write('   ')
        file.write('\n')
    file.close()
```

html.py

```
import random
from datetime import datetime



def save_html(table, rows, columns,ok_no):
    now = datetime.now()
    filename = now.strftime("%Y_%m_%d_") + str(random.randint(1000, 9999)) + ".html"
    with open(filename, 'w') as file:
        file.write(table.get_html_string(attributes={"style": "border: 1px solid black; border-collapse: collapse;
width:{}px; height:{}px;".format(30 * columns, 25 * rows)}))  # Applying style to the table
        for x in ok_no:
            file.write(x)
            file.write(' ')
        file.close()
```

---XX---