4/15/2023

CS-351-Artificial Intelligence

2020244 - Mohsin Zia
2020474 - Syed Irtaza Haider

INSTRUCTOR: PROFESSOR ZAHID HALIM

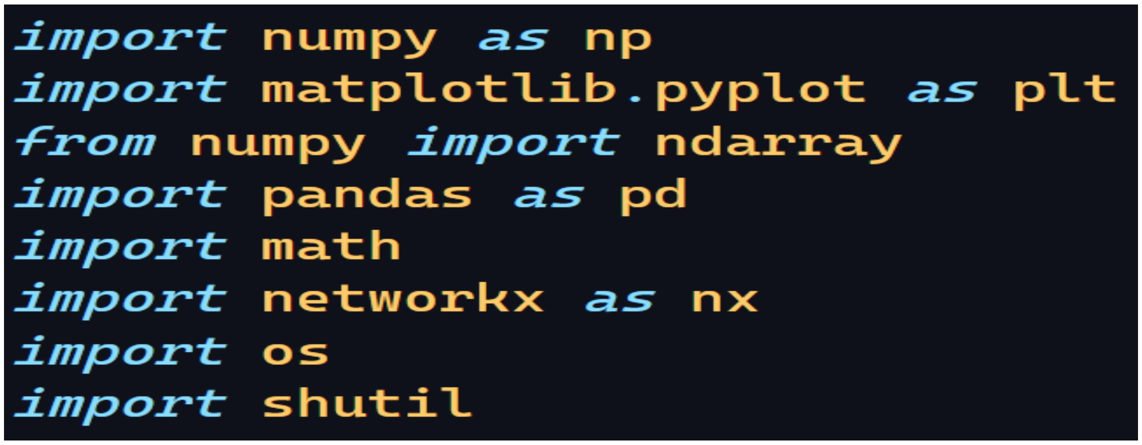
Clustering Techniques!

HOW TO RUN?

1. Open the folder in vsCode.
2. Go to main.py.
3. Click on run python file.

EXPLAINATION:

➢ **UTILITY LIBRARIES USED:**

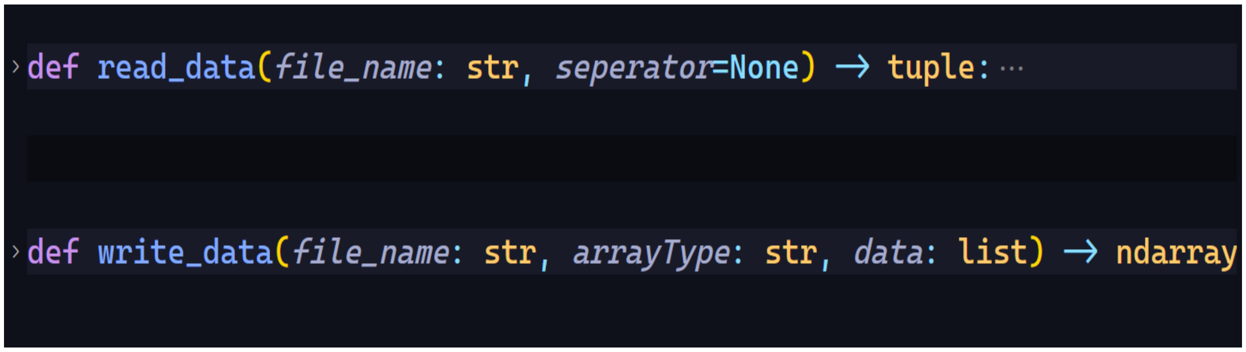


```python
import numpy as np
import matplotlib.pyplot as plt
from numpy import ndarray
import pandas as pd
import math
import networkx as nx
import os
import shutil
```

➢ **TASK 01:**

✓ **FUNCTIONS FOR READ/WRITE FROM/TO A TEXT FILE:**

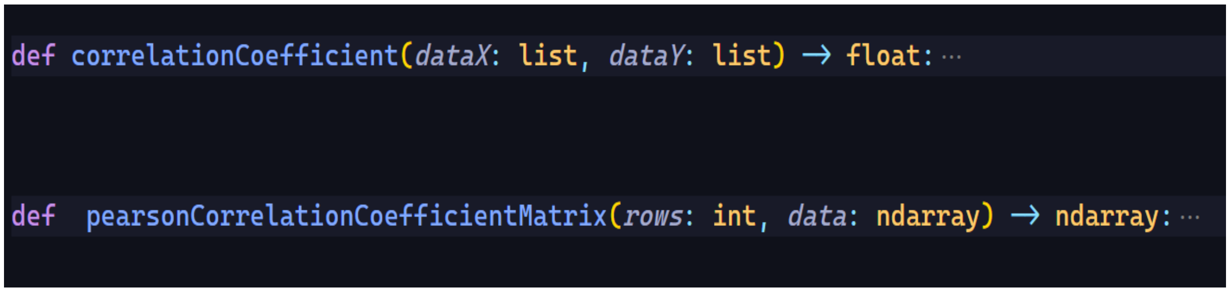
- Function read_data reads data from a text file and returns a tuple containing (numberOfRows, numberOfColumns, Data).
- Function write_data writes data from matrix to a text file including (numberOfRows, numberOfColumns, Data).



```python
def read_data(file_name: str, seperator=None) -> tuple: ...


def write_data(file_name: str, arrayType: str, data: list) -> ndarray
```

✓ **FUNCTIONS FOR COMPUTING PEARSON'S CORRELATION COEFFICIENT:**

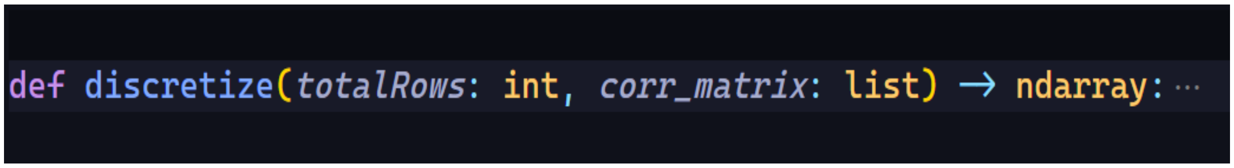
- Function correlationCoefficient computes and returns the correlation value between two rows using Pearson's correlation coefficient formula.
- Function pearsonCorrelationCoefficientMatrix computes and returns the correlation matrix of NxN dimension.



```
def correlationCoefficient(dataX: list, dataY: list) → float: ···


def  pearsonCorrelationCoefficientMatrix(rows: int, data: ndarray) → ndarray: ···
```

✓ **FUNCTION FOR COMPUTING DISCRETIZED MATRIX:**

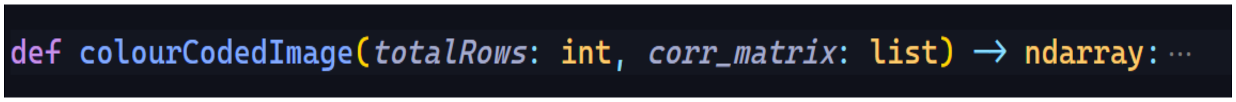
- Function discretize computes and returns the discretize matrix from the correlation matrix data. If the value in the correlation matrix is ≥ mean of it's respective column, then it's replaced with 0 else it is kept 1. Here doing vice versa of what we should've made us closer to sample output.



```
def discretize(totalRows: int, corr_matrix: list) → ndarray: ···
```

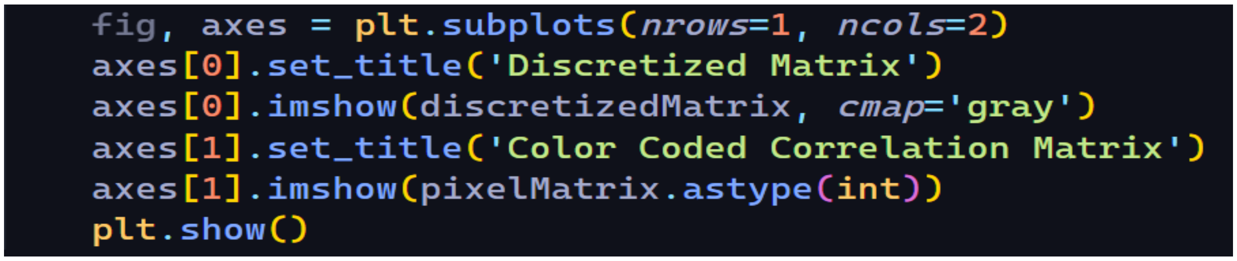
✓ **FUNCTION FOR COMPUTING PIXEL MATRIX FOR COLOR CODED IMAGE:**

- Function colourCodedImage computes and returns the Pixel Matrix of the correlation matrix data. It is calculated by dividing each element by maximum value of their respective column and then multiplying each value with 255. Then it's placed in 3-D Matrix [R, G, B]. We compute the value of G here as we are required to print sample image in green shade and then place it like [0 G 0].



```
def colourCodedImage(totalRows: int, corr_matrix: list) → ndarray: ···
```

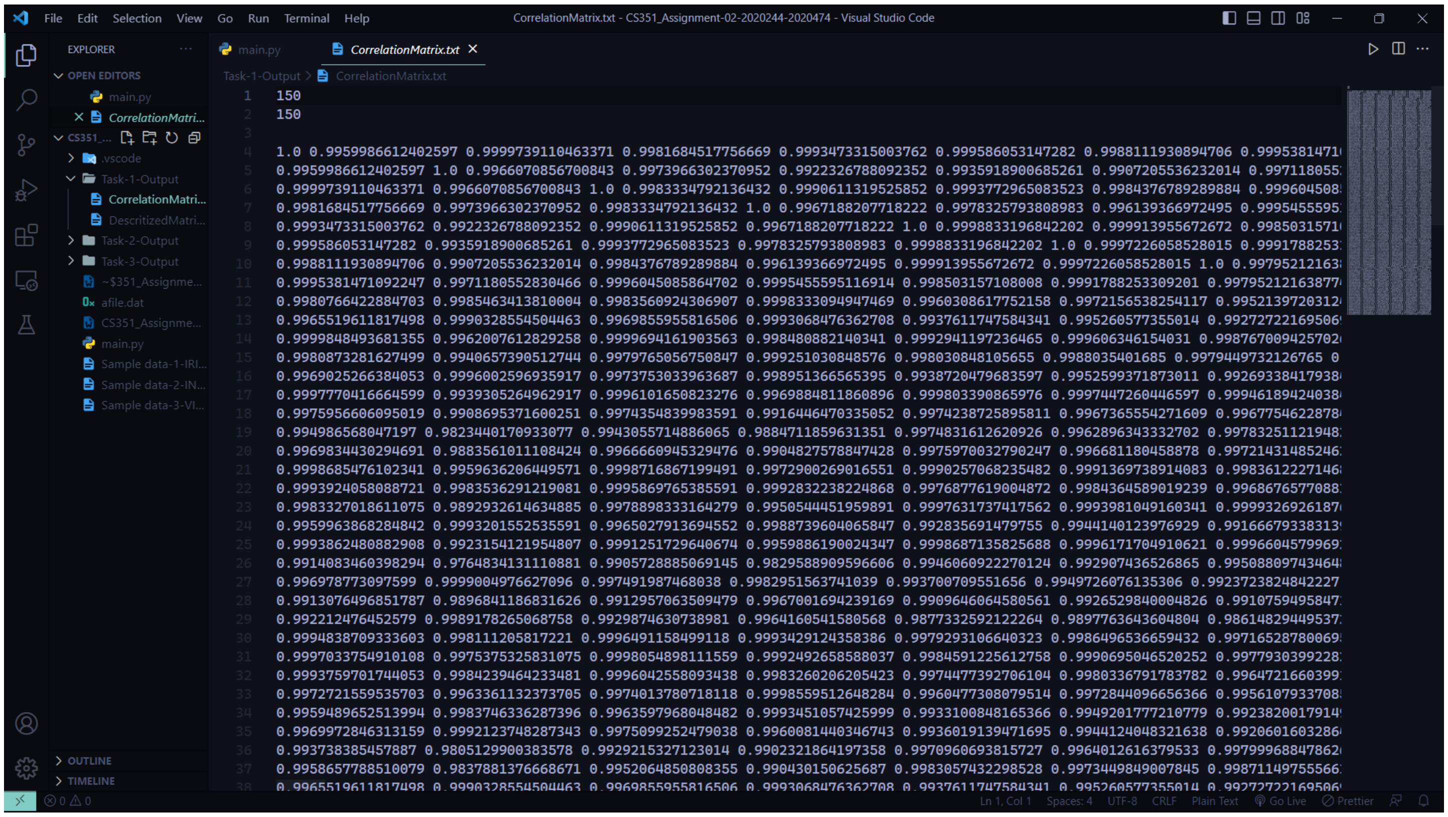
✓ **USING MATPLOTLIB FOR VISUALIZATION:**



```
    fig, axes = plt.subplots(nrows=1, ncols=2)
    axes[0].set_title('Discretized Matrix')
    axes[0].imshow(discretizedMatrix, cmap='gray')
    axes[1].set_title('Color Coded Correlation Matrix')
    axes[1].imshow(pixelMatrix.astype(int))
    plt.show()
```

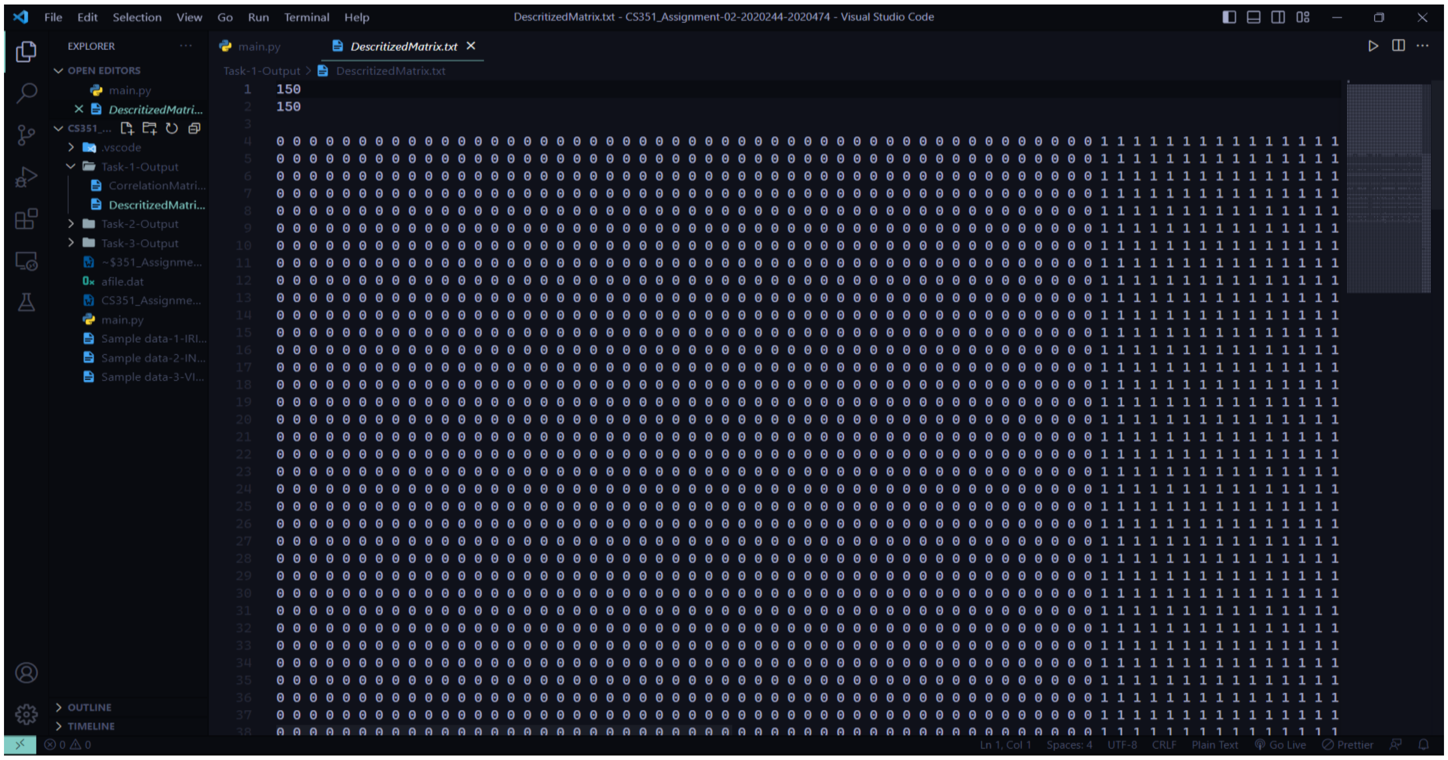
✓ Output Screenshots:

- *Correlation Matrix:*



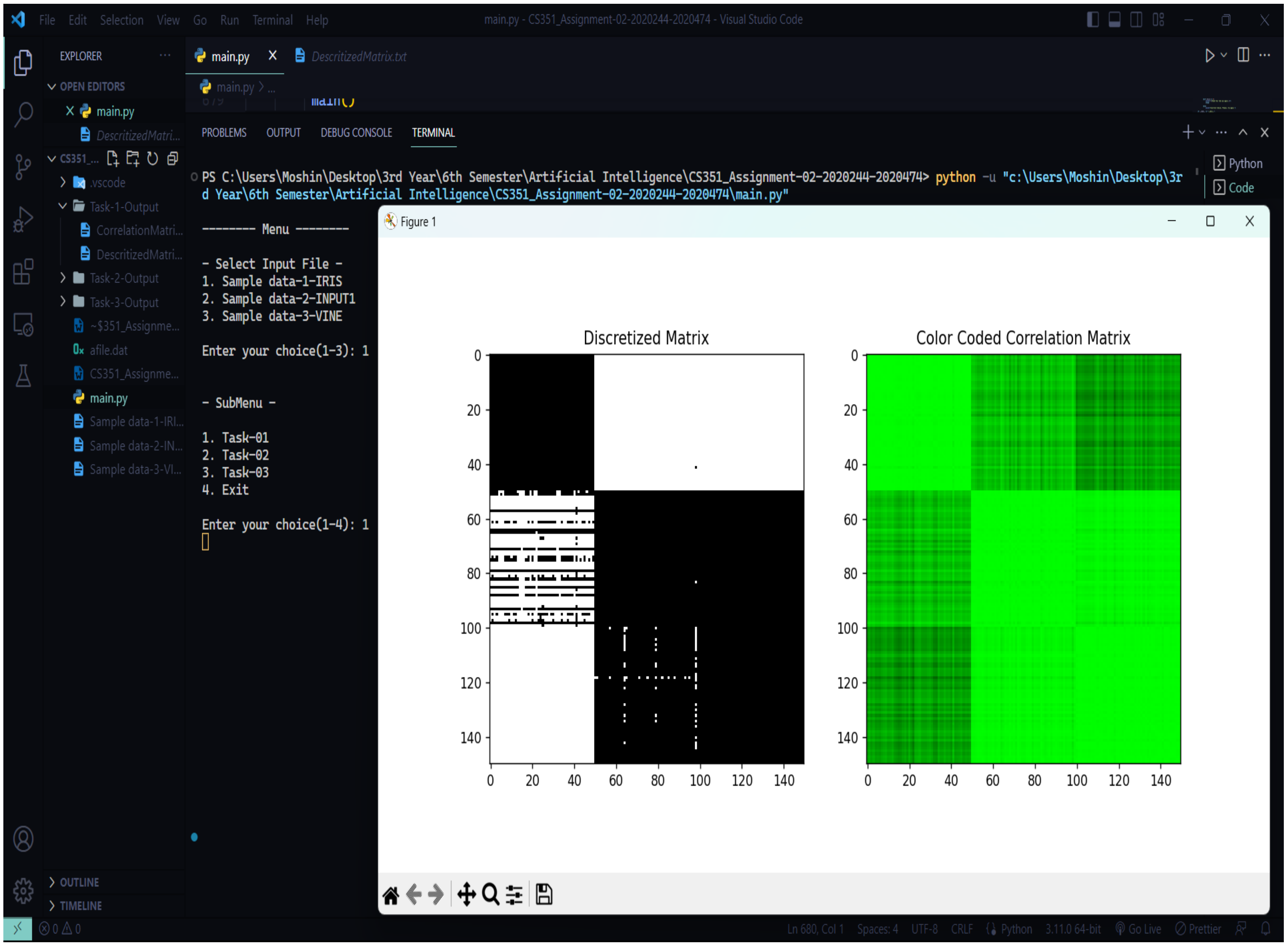
```
150
150

1.0 0.9959986612402597 0.9999739110463371 0.9981684517756669 0.9993473315003762 0.999586053147282 0.9988111930894706 0.9995381471
0.9959986612402597 1.0 0.9966070856700843 0.9973966302370952 0.9922326788092352 0.9935918900685261 0.9907205536232014 0.997118055
0.9999739110463371 0.9966070856700843 1.0 0.9983334792136432 0.9990611319525852 0.9993772965083523 0.9984376789289884 0.999604508
0.9981684517756669 0.9973966302370952 0.9983334792136432 1.0 0.9967188207718222 0.9978325793808983 0.996139366972495 0.9995455595
0.9993473315003762 0.9922326788092352 0.9990611319525852 0.9967188207718222 1.0 0.9998833196842202 0.999913955672672 0.9985031571
0.999586053147282 0.9935918900685261 0.9993772965083523 0.9978325793808983 0.9998833196842202 1.0 0.9997226058528015 0.999178825
0.9988111930894706 0.9907205536232014 0.9984376789289884 0.996139366972495 0.999913955672672 0.9997226058528015 1.0 0.99795212163
0.9995381471092247 0.9971180552830466 0.9996045085864702 0.9995455595116914 0.998503157108008 0.9991788253309201 0.9979521216387
0.9980766422884703 0.9985463413810004 0.9983560924306907 0.9998333094947469 0.9960308617752158 0.9972156538254117 0.9952139720312
0.9965519611817498 0.9990328554504463 0.9969855955816506 0.9993068476362708 0.9937611747584341 0.995260577355014 0.9927272216950
0.9999848493681355 0.9962007612829258 0.9999694161903563 0.998480882140341 0.9992941197236465 0.999606346154031 0.998767009425702
0.9980873281627499 0.9940657390512744 0.9979765056750847 0.999251030848576 0.998030848105655 0.9988035401685 0.9979449732126765 0
0.9969025266384053 0.9996002596935917 0.9973753033963687 0.998951366565395 0.9938720479683597 0.9952599371873011 0.9926933841793
0.9997770416664599 0.9939305264962917 0.9996101650823276 0.9969884811860896 0.999803390865976 0.9997447260446597 0.9994618942403
0.9975956606095019 0.9908695371600251 0.9974354839983591 0.9916446470335052 0.9974238725895811 0.9967365554271609 0.996775462287
0.994986568047197 0.9823440170933077 0.9943055714886065 0.9884711859631351 0.9974831612620926 0.9962896343332702 0.9978325112194
0.9969834430294691 0.9883561011108424 0.9966660945329476 0.9904827578847428 0.9975970032790247 0.996681180458878 0.9972143148524
0.9998685476102341 0.9959636206449571 0.9998716867199491 0.9972900269016551 0.9990257068235482 0.9991369738914083 0.9983612227146
0.9993924058088721 0.9983536291219081 0.9995869765385591 0.9992832238224868 0.9976877619004872 0.9984364589019239 0.9968676577088
0.9983327018611075 0.9892932614634885 0.9978898333164279 0.9950544451959891 0.9997631737417562 0.9993981049160341 0.9999326926187
0.9959963868284842 0.9993201552535591 0.9965027913694552 0.9988739604065847 0.992835691479755 0.9944140123976929 0.9916667933831
0.9993862480882908 0.9923154121954807 0.9991251729640674 0.9959886190024347 0.9998687135825688 0.9996171704910621 0.9996604579969
0.9914083460398294 0.9764834131110881 0.9905728885069145 0.9829588909596606 0.9946060922270124 0.992907436526865 0.9950880974346
0.996978773097599 0.9999004976627096 0.997491987468038 0.9982951563741039 0.993700709551656 0.9949726076135306 0.9923723824842227
0.9913076496851787 0.9896841186831626 0.9912957063509479 0.9967001694239169 0.9909646064580561 0.9926529840004826 0.9910759495847
0.992212476452579 0.9989178265068758 0.9929874630738981 0.9964160541580568 0.9877332592122264 0.9897763643604804 0.9861482944953
0.9994838709333603 0.998111205817221 0.9996491158499118 0.9993429124358386 0.9979293106640323 0.9986496536659432 0.9971652878006
0.9997033754910108 0.9975375325831075 0.9998054898111559 0.9992492658588037 0.9984591225612758 0.9990695046520252 0.9977930399228
0.9993759701744053 0.9984239464233481 0.9996042558093438 0.9983260206205423 0.9974477392706104 0.9980336791783782 0.9964721660399
0.9972721559535703 0.9963361132373705 0.9974013780718118 0.9998559512648284 0.9960477308079514 0.9972844096656366 0.9956107933708
0.9959489652513994 0.9983746336287396 0.9963597968048482 0.9993451057425999 0.9933100848165366 0.9949201777210779 0.9923820017914
0.9969972846313159 0.9992123748287343 0.9975099252479038 0.9960081440346743 0.9936019139471695 0.9944124048321638 0.9920601603286
0.993738385457887 0.9805129900383578 0.9929215327123014 0.9902321864197358 0.9970960693815727 0.9964012616379533 0.9979996884786
0.9958657788510079 0.9837881376668671 0.9952064850808355 0.990430150625687 0.9983057432298528 0.9973449849007845 0.9987114975556
0.9965519611817498 0.9990328554504463 0.9969855955816506 0.9993068476362708 0.9937611747584341 0.995260577355014 0.9927272216950
```

- *Discretized Matrix:*



```
150
150

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

- *Bitmap Visualization:*

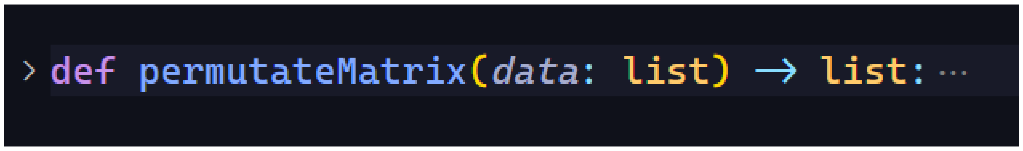


```
PS C:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474> python -u "c:\Users\Moshin\Desktop\3r
d Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474\main.py"

-------- Menu --------

- Select Input File -
1. Sample data-1-IRIS
2. Sample data-2-INPUT1
3. Sample data-3-VINE

Enter your choice(1-3): 1

- SubMenu -

1. Task-01
2. Task-02
3. Task-03
4. Exit

Enter your choice(1-4): 1
```

➢ Task 02:

✓ Function For Permutation/Shuffling:

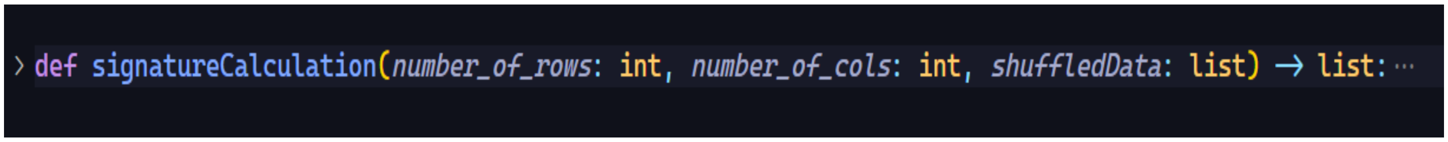
- Function permutateMatrix provides constant shuffle over a period. It shuffles the rows of inputData. A random seed (50) is used to ensure that results are reproducible.



```
› def permutateMatrix(data: list) -> list: ···
```

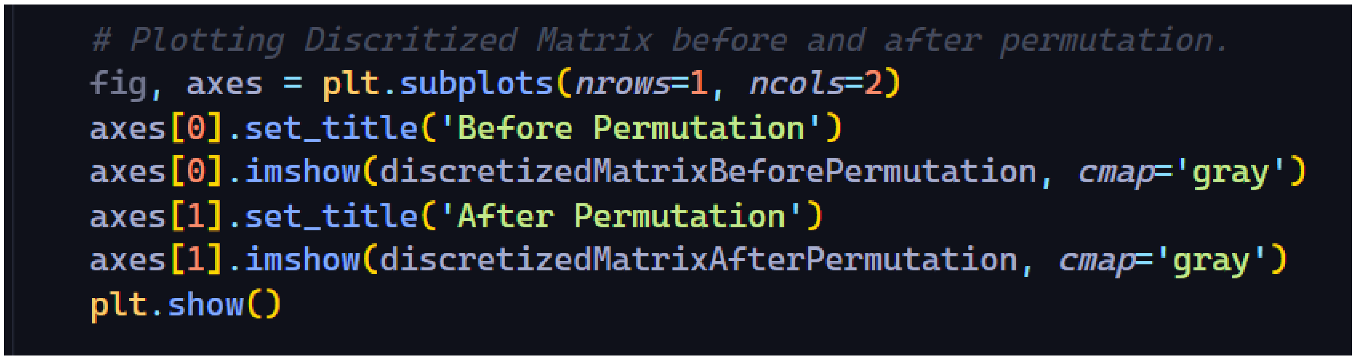
✓ Function For Calculating Signature Value and Sorting via Signature:

- Function signatureCalculation computes and returns sorted input data via their signature values. We passed the permutated data as input to this function, but I guess it really doesn't matter here if we pass the non-permutated data, in the end it's getting sorted via it's signature values.

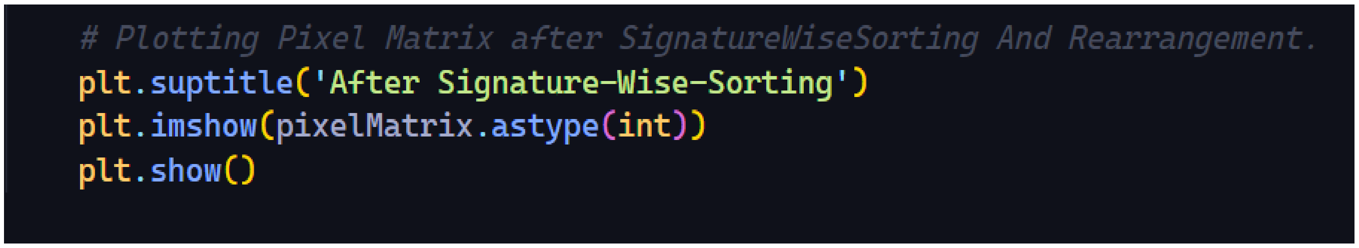


```
› def signatureCalculation(number_of_rows: int, number_of_cols: int, shuffledData: list) -> list: ···
```

✓ Using Matplotlib For Visualization:



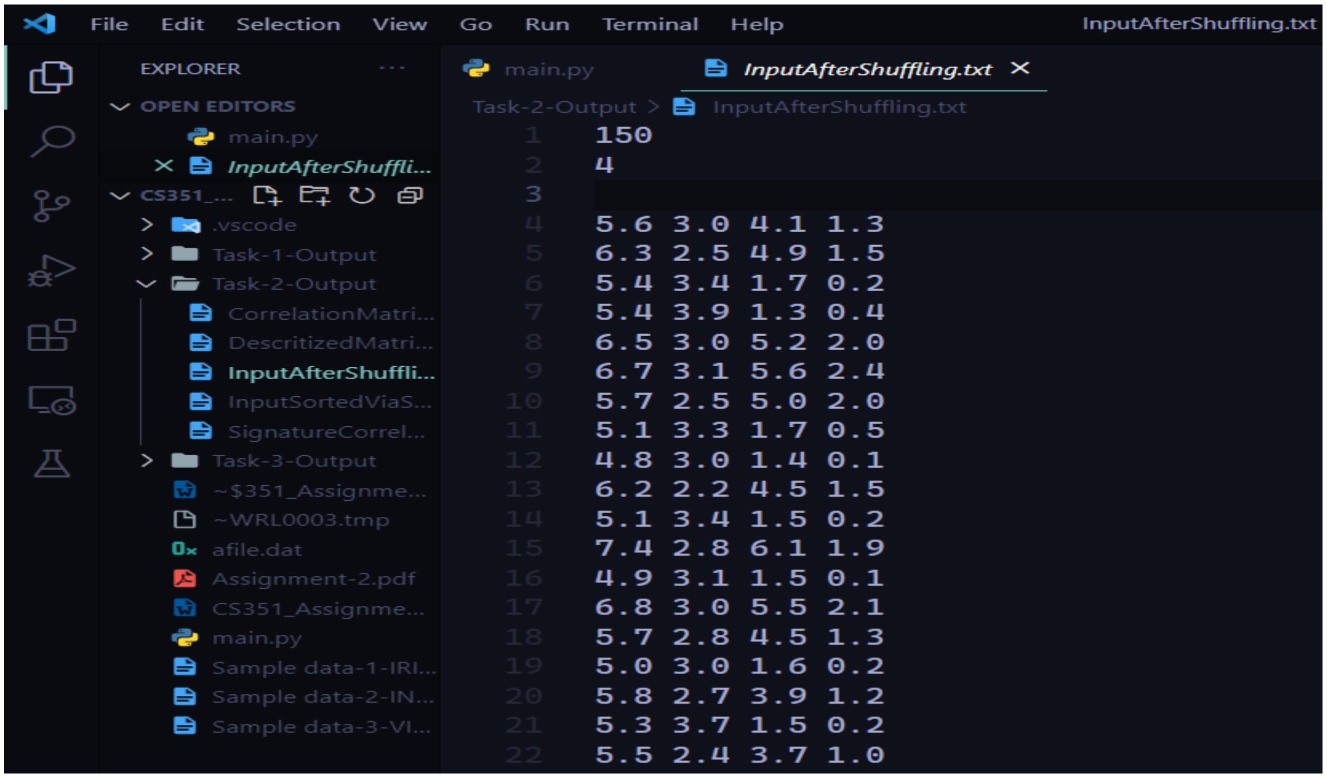
```
    # Plotting Discritized Matrix before and after permutation.
    fig, axes = plt.subplots(nrows=1, ncols=2)
    axes[0].set_title('Before Permutation')
    axes[0].imshow(discretizedMatrixBeforePermutation, cmap='gray')
    axes[1].set_title('After Permutation')
    axes[1].imshow(discretizedMatrixAfterPermutation, cmap='gray')
    plt.show()
```



```
    # Plotting Pixel Matrix after SignatureWiseSorting And Rearrangement.
    plt.suptitle('After Signature-Wise-Sorting')
    plt.imshow(pixelMatrix.astype(int))
    plt.show()
```

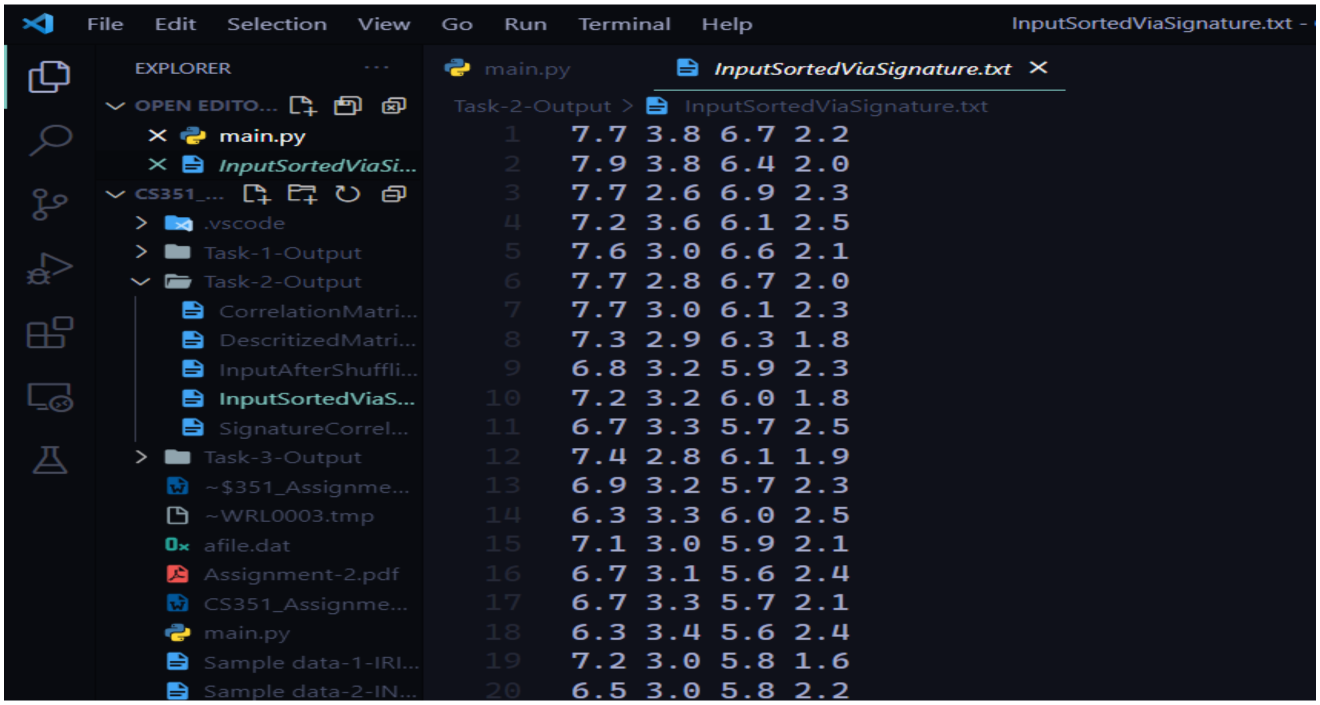
✓ **Output Screenshots:**

- *Input after permutation:*



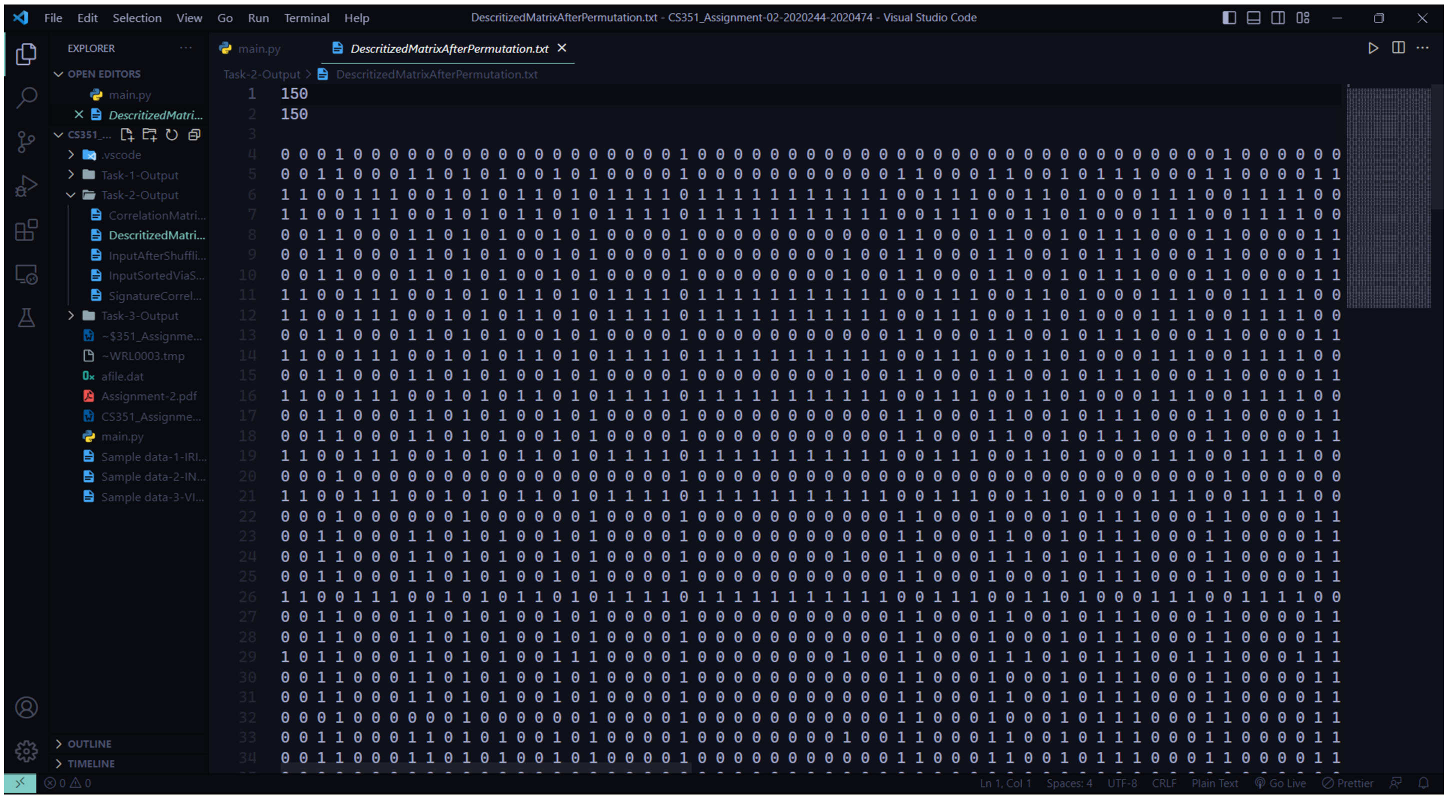
```
150
4

5.6 3.0 4.1 1.3
6.3 2.5 4.9 1.5
5.4 3.4 1.7 0.2
5.4 3.9 1.3 0.4
6.5 3.0 5.2 2.0
6.7 3.1 5.6 2.4
5.7 2.5 5.0 2.0
5.1 3.3 1.7 0.5
4.8 3.0 1.4 0.1
6.2 2.2 4.5 1.5
5.1 3.4 1.5 0.2
7.4 2.8 6.1 1.9
4.9 3.1 1.5 0.1
6.8 3.0 5.5 2.1
5.7 2.8 4.5 1.3
5.0 3.0 1.6 0.2
5.8 2.7 3.9 1.2
5.3 3.7 1.5 0.2
5.5 2.4 3.7 1.0
```

- *Input after sorting via signature:*



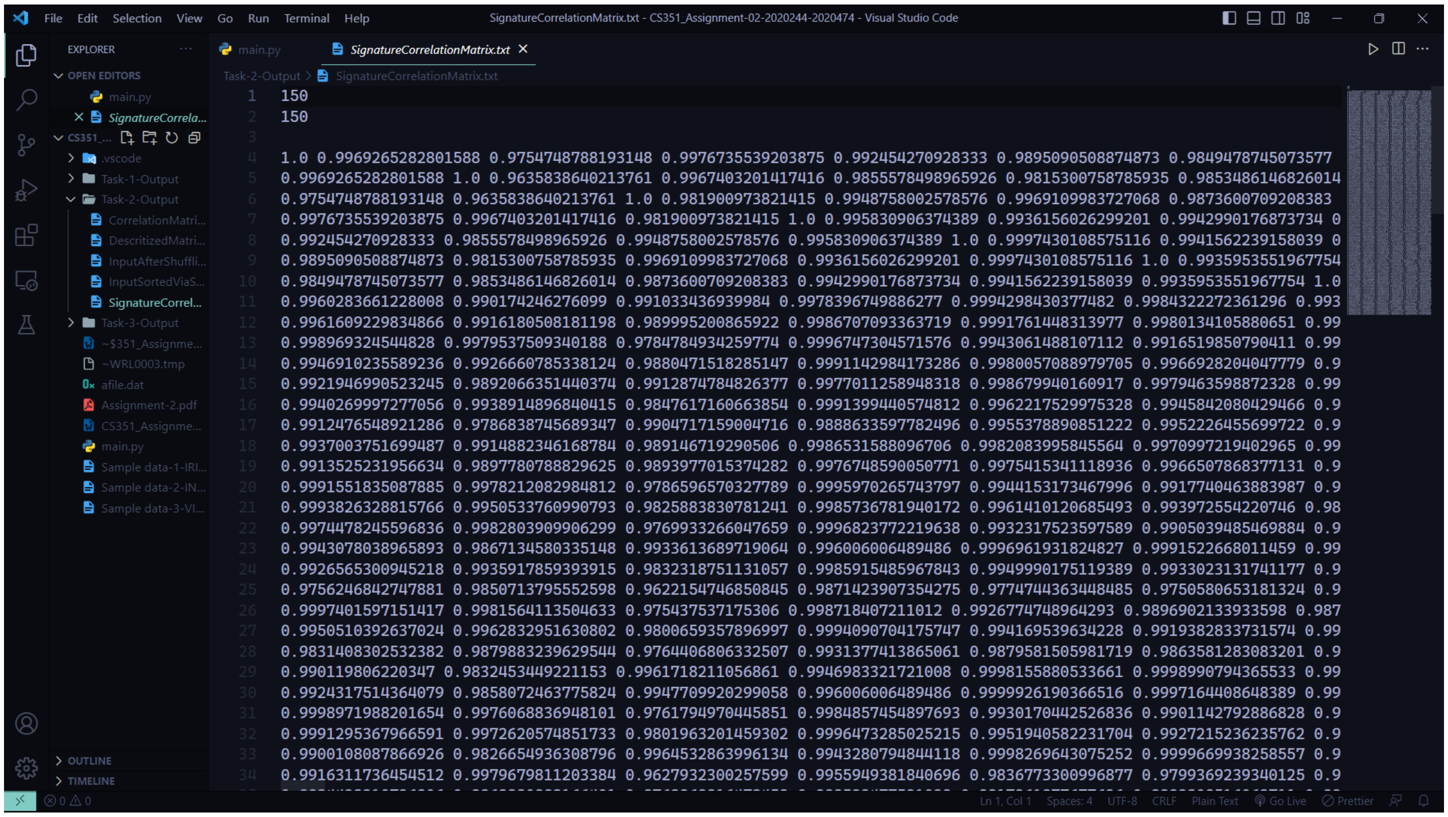
```
7.7 3.8 6.7 2.2
7.9 3.8 6.4 2.0
7.7 2.6 6.9 2.3
7.2 3.6 6.1 2.5
7.6 3.0 6.6 2.1
7.7 2.8 6.7 2.0
7.7 3.0 6.1 2.3
7.3 2.9 6.3 1.8
6.8 3.2 5.9 2.3
7.2 3.2 6.0 1.8
6.7 3.3 5.7 2.5
7.4 2.8 6.1 1.9
6.9 3.2 5.7 2.3
6.3 3.3 6.0 2.5
7.1 3.0 5.9 2.1
6.7 3.1 5.6 2.4
6.7 3.3 5.7 2.1
6.3 3.4 5.6 2.4
7.2 3.0 5.8 1.6
6.5 3.0 5.8 2.2
```

- *Discretized Matrix after permutation:*



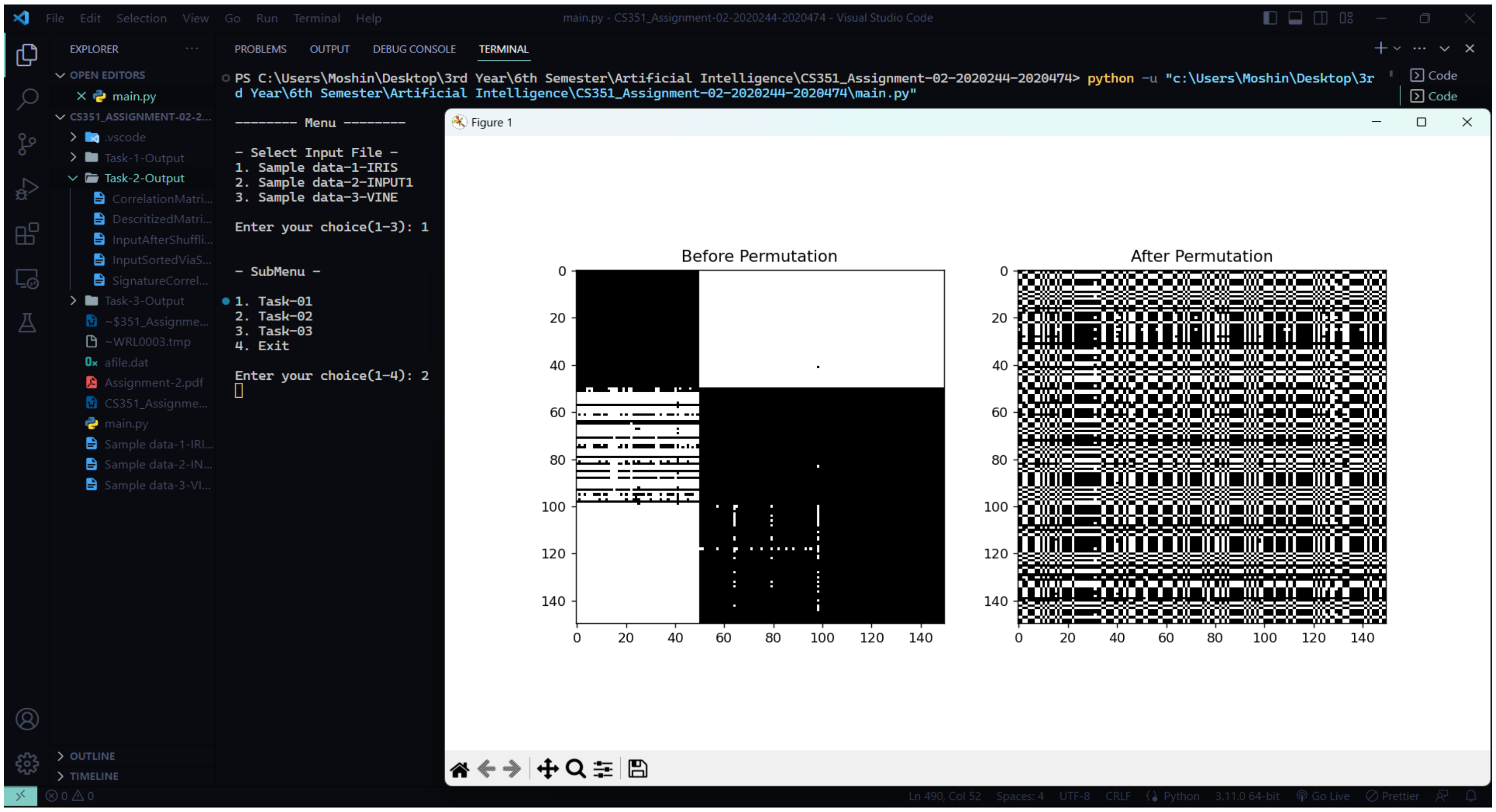
```
150
150

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 0 0
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
1 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 0 0 1 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
0 0 1 1 0 0 0 1 1 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1
```

- *Correlation Matrix after sorting via signature.*

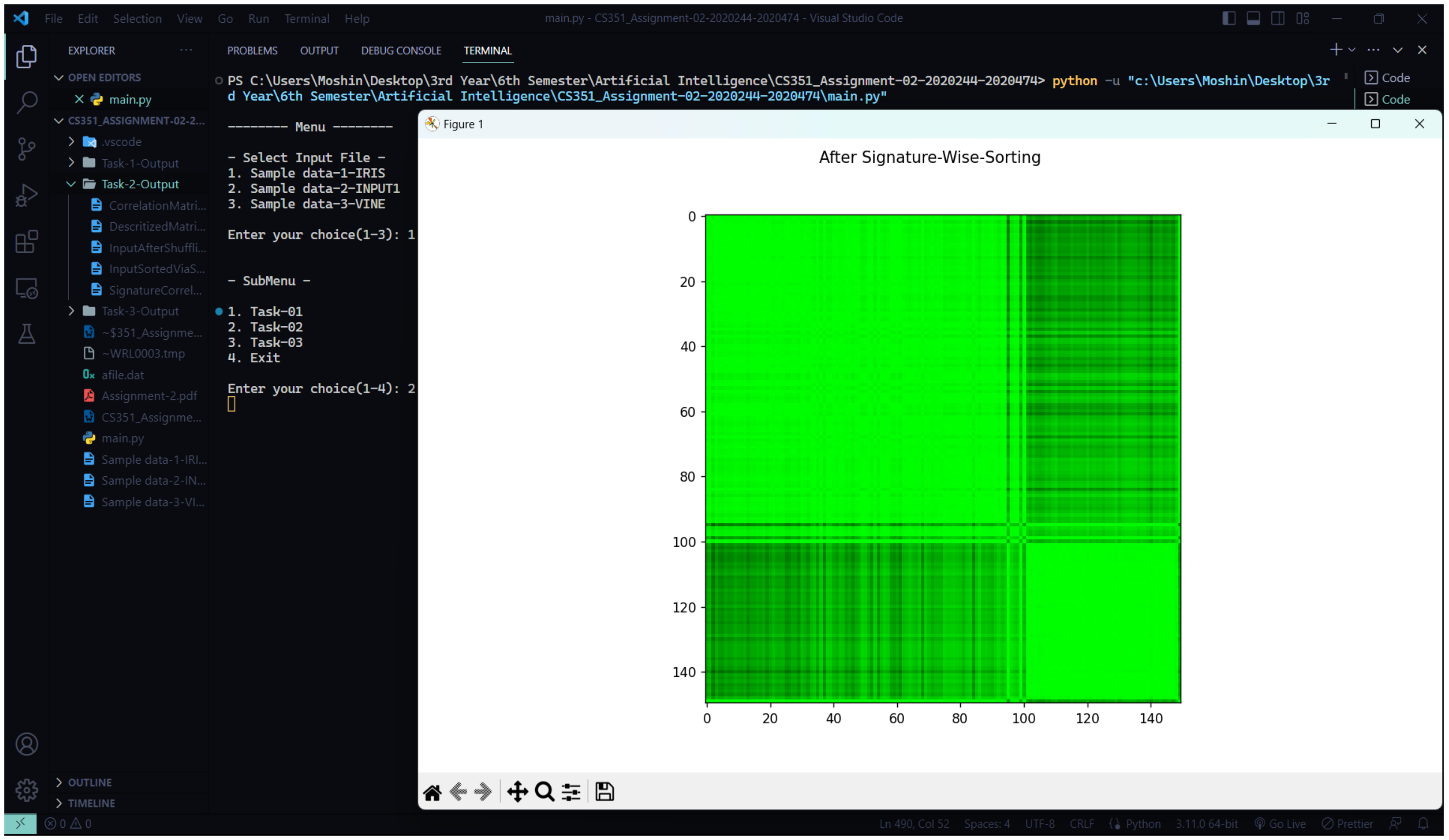


```
150
150

1.0 0.9969265282801588 0.9754748788193148 0.9976735539203875 0.992454270928333 0.9895090508874873 0.9849478745073577
0.9969265282801588 1.0 0.9635838640213761 0.9967403201417416 0.9855578498965926 0.9815300758785935 0.9853486146826014
0.9754748788193148 0.9635838640213761 1.0 0.981900973821415 0.9948758002578576 0.9969109983727068 0.9873600709208383
0.9976735539203875 0.9967403201417416 0.981900973821415 1.0 0.995830906374389 0.9936156026299201 0.9942990176873734 0
0.992454270928333 0.9855578498965926 0.9948758002578576 0.995830906374389 1.0 0.9997430108575116 0.9941562239158039 0
0.9895090508874873 0.9815300758785935 0.9969109983727068 0.9936156026299201 0.9997430108575116 1.0 0.9935953551967754
0.9849478745073577 0.9853486146826014 0.9873600709208383 0.9942990176873734 0.9941562239158039 0.9935953551967754 1.0
0.9960283661228008 0.990174246276099 0.991033436939984 0.9978396749886277 0.9994298430377482 0.9984322272361296 0.993
0.9961609229834866 0.9916180508181198 0.989995200865922 0.9986707093363719 0.9991761448313977 0.9980134105880651 0.99
0.998969324544828 0.9979537509340188 0.9784784934259774 0.9996747304571576 0.9943061488107112 0.9916519850790411 0.99
0.9946910235589236 0.9926660785338124 0.9880471518285147 0.9991142984173286 0.9980057088979705 0.9966928204047779 0.9
0.9921946990523245 0.9892066351440374 0.9912874784826377 0.9977011258948318 0.9986799401600917 0.9979463598872328 0.99
0.9940269997277056 0.9938914896840415 0.9847617160663854 0.9991399440574812 0.9962217529975328 0.9945842080429466 0.9
0.9912476548921286 0.9786838745689347 0.9904717159004716 0.9888633597782496 0.9955378890851222 0.9952226455699722 0.9
0.9937003751699487 0.9914882346168784 0.989146719290506 0.9986531588096706 0.9982083995845564 0.9970997219402965 0.99
0.9913525231956634 0.9897780788829625 0.9893977015374282 0.9976748590050771 0.9975415341118936 0.9966507868377131 0.9
0.9991551835087885 0.9978212082984812 0.9786596570327789 0.9995970265743797 0.9944153173467996 0.9917740463883987 0.9
0.9993826328815766 0.9950533760990793 0.9825883830781241 0.9985736781940172 0.9961410120685493 0.9939725542207460 0.98
0.9974478245596836 0.9982803909906299 0.9769933266047659 0.9996823772219638 0.9932317523597589 0.9905039485469884 0.9
0.9943078038965893 0.9867134580335148 0.9933613689719064 0.996006006489486 0.9996961931824827 0.9991522668011459 0.99
0.9926565300945218 0.9935917859393915 0.9832318751131057 0.9985915485967843 0.9949990175119389 0.9933023131741177 0.9
0.9756246842747881 0.9850713795552598 0.9622154746850845 0.9871423907354275 0.9774744363448485 0.9750580653181324 0.9
0.9997401597151417 0.9981564113504633 0.975437537175306 0.998718407211012 0.9926774748964293 0.9896902133933598 0.987
0.9950510392637024 0.9962832951630802 0.9800659357896997 0.9994090704175747 0.994169539634228 0.9919382833731574 0.99
0.9831408302532382 0.9879883239629544 0.9764406806332507 0.9931377413865061 0.9879581505981719 0.9863581283083201 0.9
0.990119806220347 0.9832453449221153 0.9961718211056861 0.9946983321721008 0.9998155880533661 0.9998990794365533 0.99
0.9924317514364079 0.9858072463775824 0.9947709920299058 0.996006006489486 0.9999926190366516 0.9997164408648389 0.99
0.9998971988201654 0.9976068836948101 0.9761794970445851 0.9984857454897693 0.9930170442526836 0.9901142792886828 0.9
0.9991295367966591 0.9972620574851733 0.9801963201459302 0.9996473285025215 0.9951940582231704 0.9927215236235762 0.9
0.9900108087866926 0.9826654936308796 0.9964532863996134 0.9943280794844118 0.9998269643075252 0.9999669938258557 0.9
0.9916311736454512 0.9979679811203384 0.9627932300257599 0.9955949381840696 0.9836773300996877 0.9799369239340125 0.9
```

- *Bitmap Visualization:*



```
PS C:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474> python -u "c:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474\main.py"

-------- Menu --------

- Select Input File -
1. Sample data-1-IRIS
2. Sample data-2-INPUT1
3. Sample data-3-VINE

Enter your choice(1-3): 1

- SubMenu -

1. Task-01
2. Task-02
3. Task-03
4. Exit

Enter your choice(1-4): 2
```

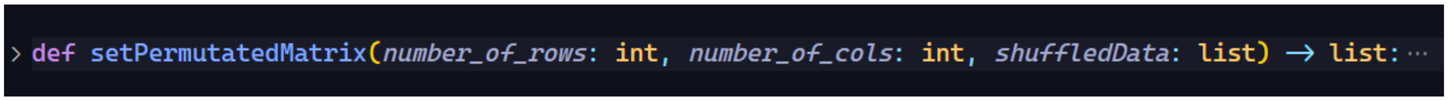


```
PS C:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474> python -u "c:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474\main.py"

-------- Menu --------

- Select Input File -
1. Sample data-1-IRIS
2. Sample data-2-INPUT1
3. Sample data-3-VINE

Enter your choice(1-3): 1

- SubMenu -

1. Task-01
2. Task-02
3. Task-03
4. Exit

Enter your choice(1-4): 2
```

➢ Task 03:

✓ Function For Setup of Permutated Correlation Matrix:

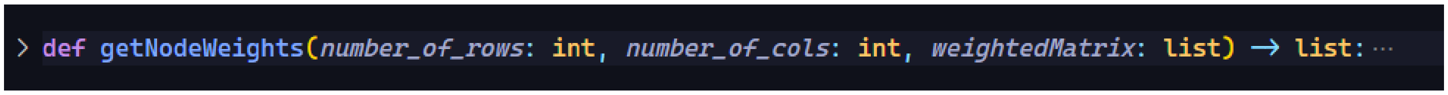
- Function setPermutatedMatrix takes permutated data set's correlation matrix and based on a threshold, it updates that matrix. It iterates through each value in the matrix and if the value is less than the threshold it makes it 0, else it is **not** updated.



```
> def setPermutatedMatrix(number_of_rows: int, number_of_cols: int, shuffledData: list) → list:···
```

✓ Functions For Computing Node Weights of the updated matrix with respect to Threshold:

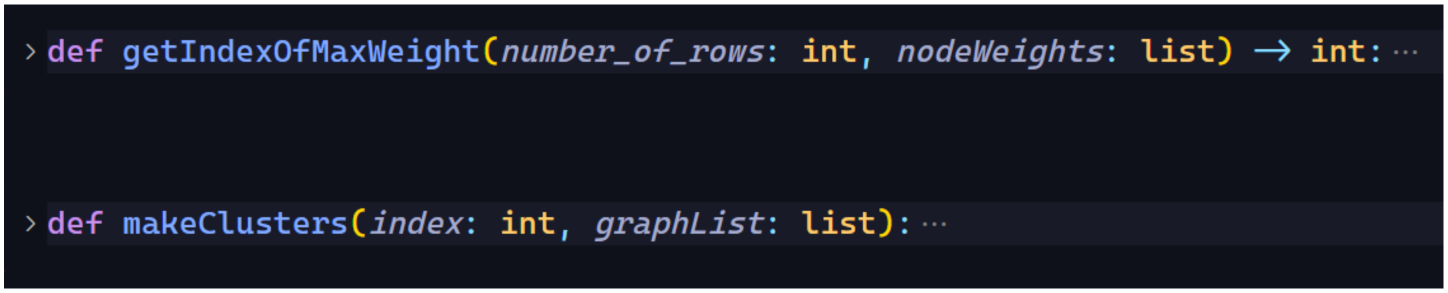
- Function getNodeWeights computes and returns the nodeWeights matrix by calculating node weights of each row. It is calculated as the sum of all columns in a respective row.



```
> def getNodeWeights(number_of_rows: int, number_of_cols: int, weightedMatrix: list) → list:···
```

✓ Function For Making Cluster Of the Row in Correlation Matrix with maximum node weights:

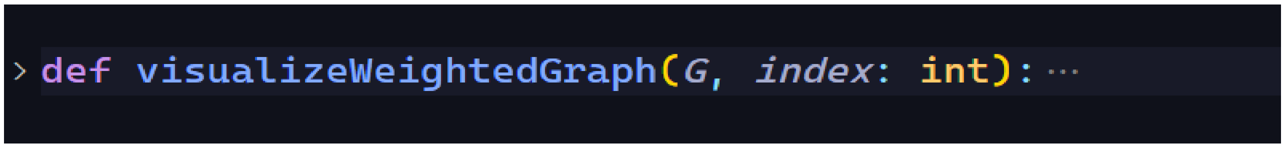
- Function getIndexOfMaxWeight computes and returns the index of the node with maximum weightage. Once it is visited, matrix of node weights at that particular index is placed with -1 so that that node is excluded from our to-cluster list.
- Function makeCluster makes the graph of element with maximum node weight, it connects the edges to the elements in graphList that have survived the threshold test and then it connects to them and forms a cluster. Adding nodes and edges in between is done in this function.



```
> def getIndexOfMaxWeight(number_of_rows: int, nodeWeights: list) → int:···


> def makeClusters(index: int, graphList: list):···
```

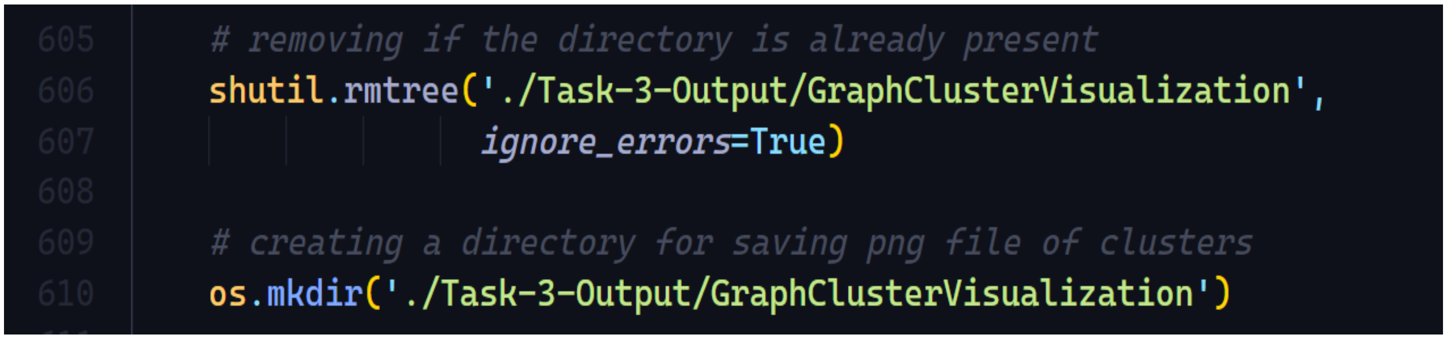
✓ Function For Visualization of weighted graph in form of clusters:

- Function visualizeWeightedGraph takes input a weighted graph and index with maximum weight, it then used matplotlib to create a figure. Further it uses networkx library to draw the nodes and edges between source and destination. It can also add weights in between edges but it doesn't look good when we are visualizing it. It saves the figure in png format.



```
> def visualizeWeightedGraph(G, index: int):···
```

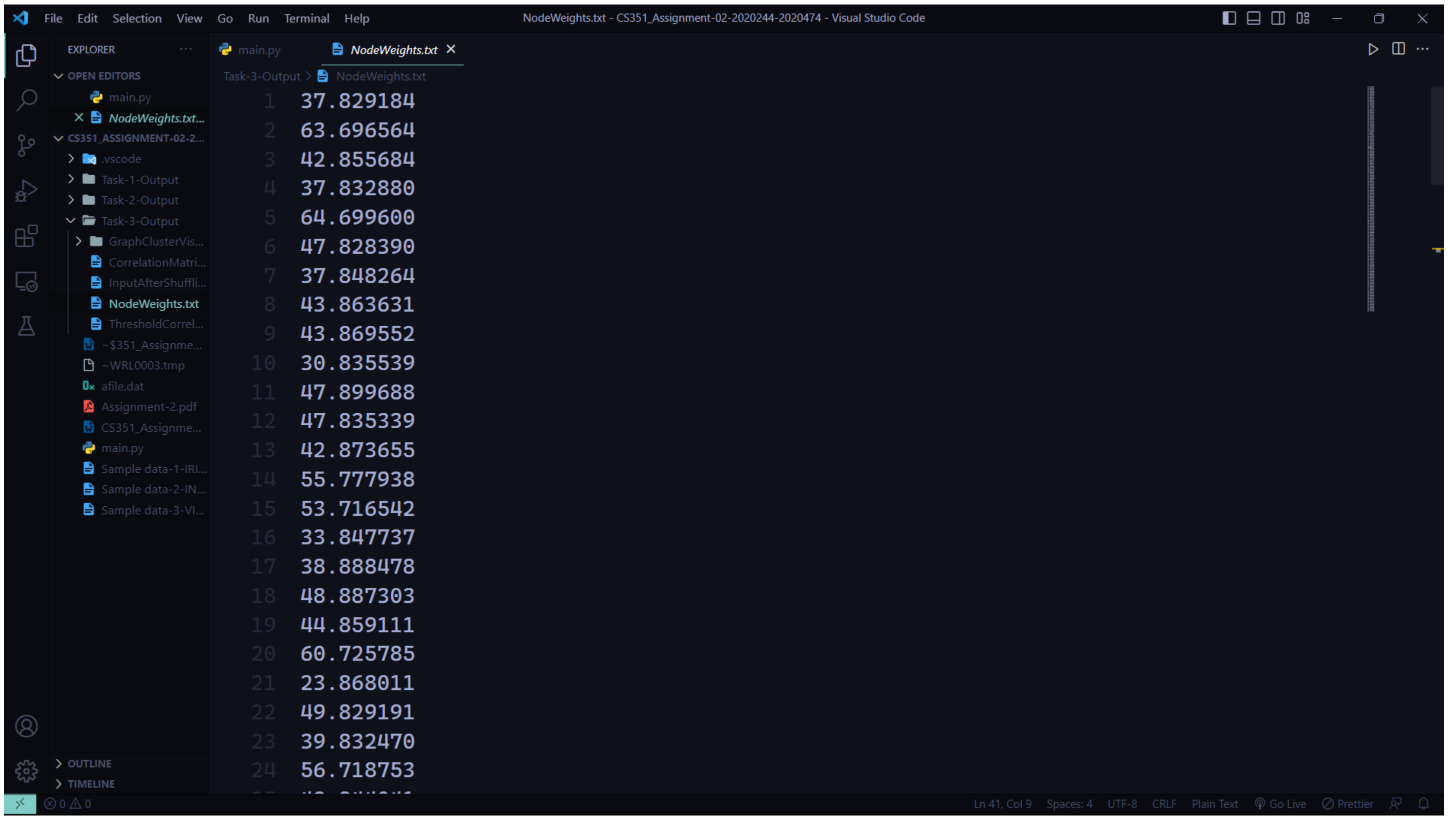
✓ **Creating a Directory for our Cluster images:**



```python
# removing if the directory is already present
shutil.rmtree('./Task-3-Output/GraphClusterVisualization',
                    ignore_errors=True)

# creating a directory for saving png file of clusters
os.mkdir('./Task-3-Output/GraphClusterVisualization')
```

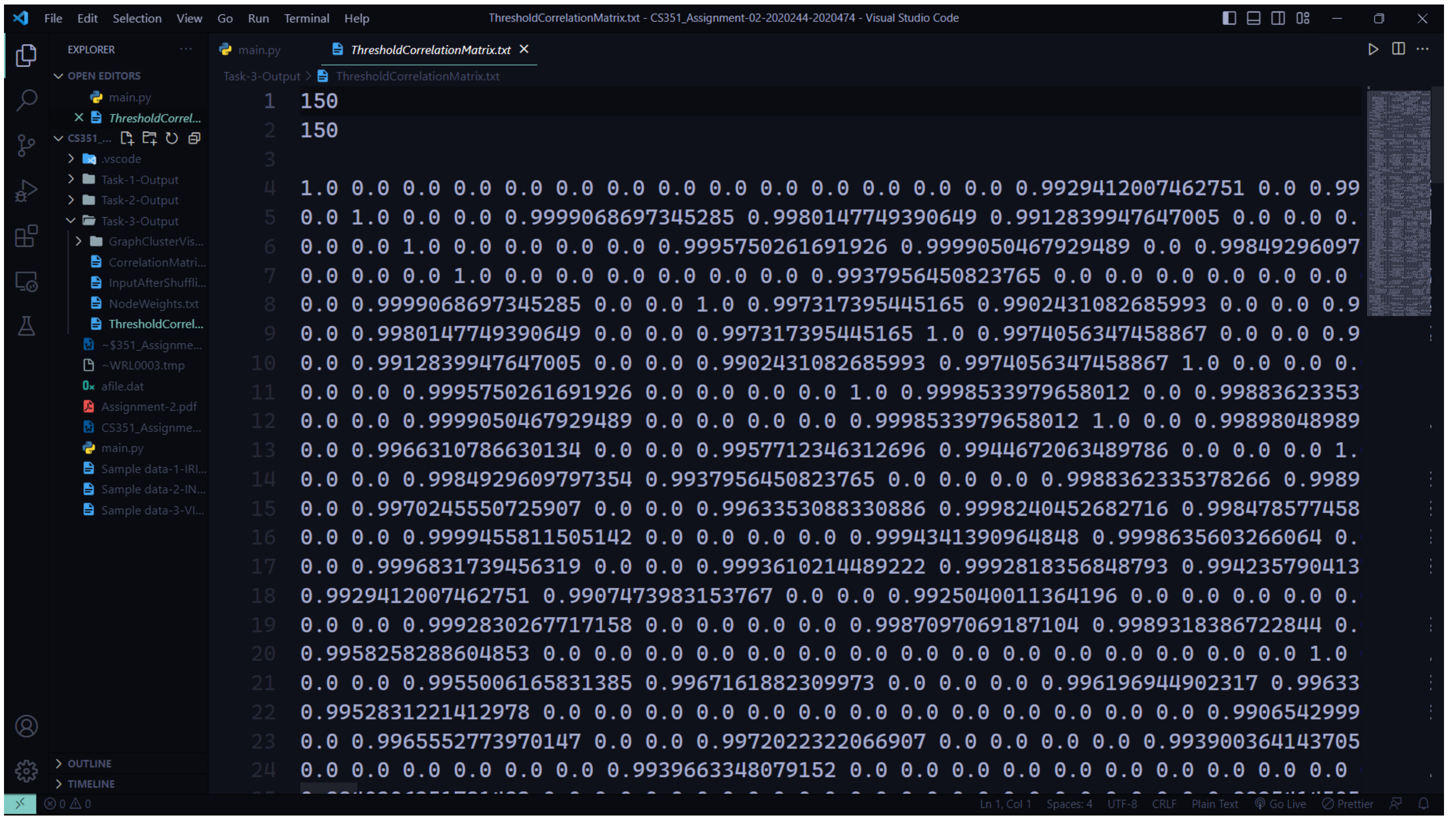
✓ **Output Screenshots:**

- *Node Weights Matrix:*
    - *The threshold is set to 0.99.*



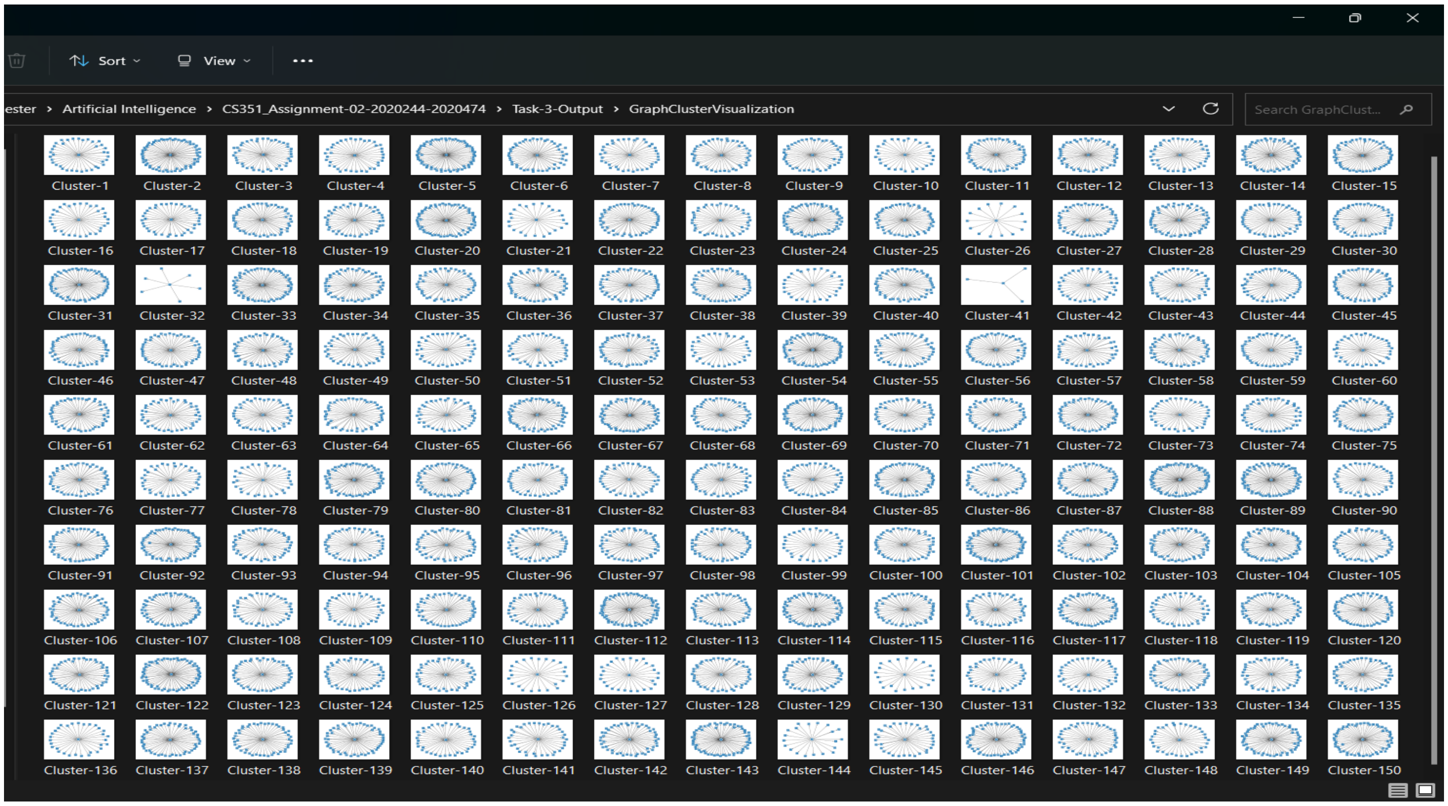
```
37.829184
63.696564
42.855684
37.832880
64.699600
47.828390
37.848264
43.863631
43.869552
30.835539
47.899688
47.835339
42.873655
55.777938
53.716542
33.847737
38.888478
48.887303
44.859111
60.725785
23.868011
49.829191
39.832470
56.718753
```

- *Threshold Correlation Matrix:*

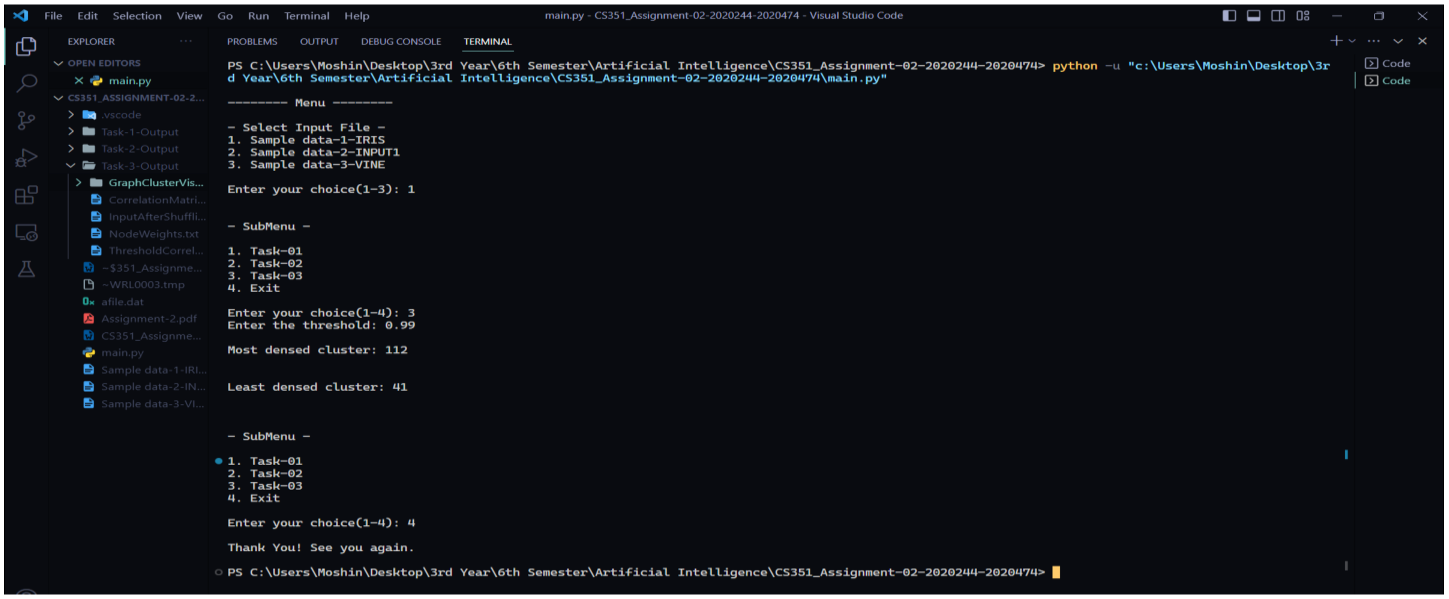


```
150
150

1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.9929412007462751 0.0 0.99
0.0 1.0 0.0 0.0 0.9999068697345285 0.9980147749390649 0.9912839947647005 0.0 0.0 0.
0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.9995750261691926 0.9999050467929489 0.0 0.99849296097
0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.9937956450823765 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.9999068697345285 0.0 0.0 1.0 0.997317395445165 0.9902431082685993 0.0 0.0 0.9
0.0 0.9980147749390649 0.0 0.0 0.997317395445165 1.0 0.9974056347458867 0.0 0.0 0.9
0.0 0.9912839947647005 0.0 0.0 0.9902431082685993 0.9974056347458867 1.0 0.0 0.0 0.
0.0 0.0 0.9995750261691926 0.0 0.0 0.0 0.0 1.0 0.9998533979658012 0.0 0.99883623353
0.0 0.0 0.9999050467929489 0.0 0.0 0.0 0.0 0.9998533979658012 1.0 0.0 0.99898048989
0.0 0.9966310786630134 0.0 0.0 0.9957712346312696 0.9944672063489786 0.0 0.0 0.0 1.
0.0 0.0 0.9984929609797354 0.9937956450823765 0.0 0.0 0.0 0.9988362335378266 0.9989
0.0 0.9970245550725907 0.0 0.0 0.9963353088330886 0.9998240452682716 0.998478577458
0.0 0.0 0.9999455811505142 0.0 0.0 0.0 0.0 0.9994341390964848 0.9998635603266064 0.
0.0 0.9996831739456319 0.0 0.0 0.9993610214489222 0.9992818356848793 0.994235790413
0.9929412007462751 0.9907473983153767 0.0 0.0 0.9925040011364196 0.0 0.0 0.0 0.0 0.
0.0 0.0 0.9992830267717158 0.0 0.0 0.0 0.0 0.9987097069187104 0.9989318386722844 0.
0.9958258288604853 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
0.0 0.0 0.9955006165831385 0.9967161882309973 0.0 0.0 0.0 0.996196944902317 0.99633
0.9952831221412978 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.9906542999
0.0 0.9965552773970147 0.0 0.0 0.9972022322066907 0.0 0.0 0.0 0.0 0.993900364143705
0.0 0.0 0.0 0.0 0.0 0.0 0.9939663348079152 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

- *Directory of png's overview/Clusters saved:*

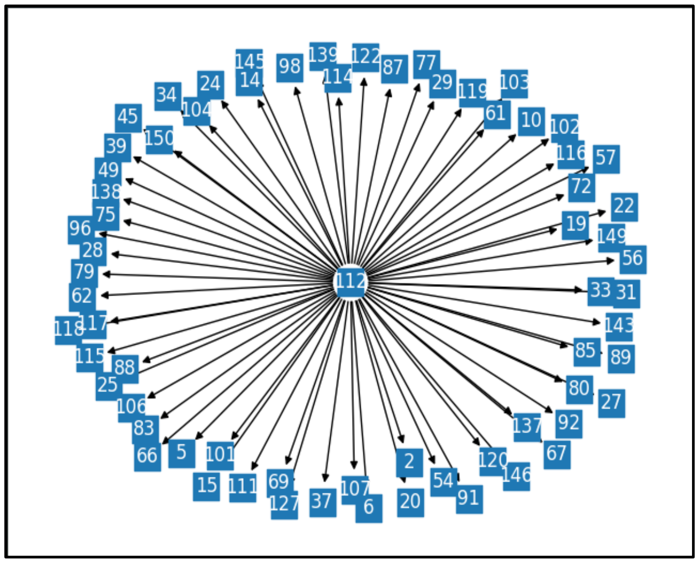

Sort
View
ester > Artificial Intelligence > CS351_Assignment-02-2020244-2020474 > Task-3-Output > GraphClusterVisualization
Search GraphClust...
Cluster-1 Cluster-2 Cluster-3 Cluster-4 Cluster-5 Cluster-6 Cluster-7 Cluster-8 Cluster-9 Cluster-10 Cluster-11 Cluster-12 Cluster-13 Cluster-14 Cluster-15
Cluster-16 Cluster-17 Cluster-18 Cluster-19 Cluster-20 Cluster-21 Cluster-22 Cluster-23 Cluster-24 Cluster-25 Cluster-26 Cluster-27 Cluster-28 Cluster-29 Cluster-30
Cluster-31 Cluster-32 Cluster-33 Cluster-34 Cluster-35 Cluster-36 Cluster-37 Cluster-38 Cluster-39 Cluster-40 Cluster-41 Cluster-42 Cluster-43 Cluster-44 Cluster-45
Cluster-46 Cluster-47 Cluster-48 Cluster-49 Cluster-50 Cluster-51 Cluster-52 Cluster-53 Cluster-54 Cluster-55 Cluster-56 Cluster-57 Cluster-58 Cluster-59 Cluster-60
Cluster-61 Cluster-62 Cluster-63 Cluster-64 Cluster-65 Cluster-66 Cluster-67 Cluster-68 Cluster-69 Cluster-70 Cluster-71 Cluster-72 Cluster-73 Cluster-74 Cluster-75
Cluster-76 Cluster-77 Cluster-78 Cluster-79 Cluster-80 Cluster-81 Cluster-82 Cluster-83 Cluster-84 Cluster-85 Cluster-86 Cluster-87 Cluster-88 Cluster-89 Cluster-90
Cluster-91 Cluster-92 Cluster-93 Cluster-94 Cluster-95 Cluster-96 Cluster-97 Cluster-98 Cluster-99 Cluster-100 Cluster-101 Cluster-102 Cluster-103 Cluster-104 Cluster-105
Cluster-106 Cluster-107 Cluster-108 Cluster-109 Cluster-110 Cluster-111 Cluster-112 Cluster-113 Cluster-114 Cluster-115 Cluster-116 Cluster-117 Cluster-118 Cluster-119 Cluster-120
Cluster-121 Cluster-122 Cluster-123 Cluster-124 Cluster-125 Cluster-126 Cluster-127 Cluster-128 Cluster-129 Cluster-130 Cluster-131 Cluster-132 Cluster-133 Cluster-134 Cluster-135
Cluster-136 Cluster-137 Cluster-138 Cluster-139 Cluster-140 Cluster-141 Cluster-142 Cluster-143 Cluster-144 Cluster-145 Cluster-146 Cluster-147 Cluster-148 Cluster-149 Cluster-150

- *Most dense and least dense clusters formed:*

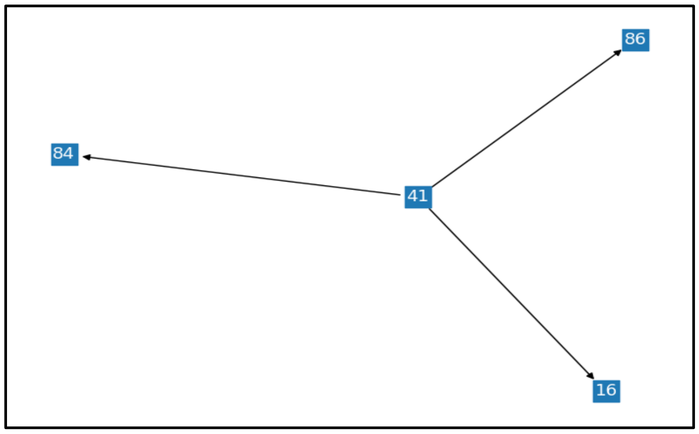


```
PS C:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474> python -u "c:\Users\Moshin\Desktop\3r
d Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474\main.py"
-------- Menu --------

- Select Input File -
1. Sample data-1-IRIS
2. Sample data-2-INPUT1
3. Sample data-3-VINE

Enter your choice(1-3): 1


- SubMenu -

1. Task-01
2. Task-02
3. Task-03
4. Exit

Enter your choice(1-4): 3
Enter the threshold: 0.99

Most densed cluster: 112


Least densed cluster: 41



- SubMenu -

1. Task-01
2. Task-02
3. Task-03
4. Exit

Enter your choice(1-4): 4

Thank You! See you again.

PS C:\Users\Moshin\Desktop\3rd Year\6th Semester\Artificial Intelligence\CS351_Assignment-02-2020244-2020474>
```

    - *Most dense cluster:*



    - *Least dense cluster:*



➢ Comparison Of Task 2 and 3:

✓ Task-2:

- The primary goal of task 2 is to demonstrate the dissimilarity in bitmap by carrying out permutations on the data and presenting the bitmap of the discretized iris data before and after the permutation process. Subsequently, the permutated data undergoes sorting based on its signature values, and a color-coded image of the correlation data is generated and displayed.

✓ Task-3:

- The primary objective of task 3 is to identify clusters within the correlation matrix subsequent to the permutation of data. To accomplish this, a threshold value is established, which in our case is 0.99, and the correlation matrix is updated accordingly. Following this, we pinpoint the node with the highest weight and construct a cluster including its surviving neighbors from the threshold process. This process is repeated iteratively until clusters are generated for each row in our graph list. Ultimately, this technique enables us to effectively determine the clusters present within the correlation matrix.

➢ Work distribution among group members:

Throughout the completion of this assignment, it is evident that we have made substantial contributions towards its successful completion. From the initiation of the project to the final submission, we both have actively participated in the task allocation, research, and implementation phases. Additionally, the collaborative nature of the project enabled both of us to offer valuable insights and suggestions, leading to the development of a comprehensive and well-structured final submission. We both have played an integral role throughout the assignment, and combined efforts have resulted in a successful outcome.

➢ Insights:

During the completion of this assignment, we have gained several valuable insights. Firstly, we discovered that the Pearson correlation matrix, although correlation is always symmetric process, didn’t produce symmetric bitmap for discretized data. While initially considering that we may have made an error in our formula, using the predefined pd.DataFrame.corr() function produced results that were consistent with our output. Despite this, doubts remained about the accuracy of the sample image produced by us. It should also be noted that discretized data is calculated by taking the mean of the column and updating each value in that column. I think that it’s the reason why the bitmap is not looking symmetric. Another observation made was that sorting via signature sorts both the input data and shuffled data to the same matrix, rendering the shuffling process is unnecessary here. Finally, we recognized the immense power of the Python language. Tasks that would have taken a significant amount of time in other programming languages such as C++, such as bitmap visualization (I first tried visualization using QT after completing these tasks in C++ and it was becoming too complex ☹), were easily accomplished using Matplotlib in just two lines of code in Python.

The End .