Tugas Kecil I IF2211 Strategi Algoritma

Penyelesaian Cyberpunk 2077 Breach Protocol dengan Algroitma Brute Force

Disusun oleh :

13522157 – Muhammad Davis Adhipramana

Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung 2024

DAFTAR ISI

BAB I

DESKRIPSI ALGORITMA BRUTEFORCE

1.1. Algoritma Brute Force

Algoritma Brute force adalah suatu algoritma yang melakukan pendekatan yang lempang (straightforward) untuk memecahkan suatu persoalan. Algoritma brute force ini biasanya didasarkan pada pernyataan pada persoalan (problem statement) dan definisi atau konsep yang dilibatkan. Selain itu, algoritma brute force ini kerap digunakan untuk memecahkan persoalan karena penggunaan nya yang sangat sederhana, langsung, dan jelas caranya.

Algoritma bruteforce ini dalam penggunaanya biasanya dilakukan dengan melakukan enumerasi pada keseluruhan elemen atau melakukan perhitungan pada keseluruhan suatu elemen hingga akhirnya didapatkan suatu hasil yang diharapkan pada salah satu perhitungannya. Banyak sekali algoritma yang menggunakan teknik bruteforce ini diantaranya adalah mencari elemen terbesar pada suatu kumpulan elemen, *Sequential Search* atau pencarian beruntun, sorting pada buble sort, dan masih banyak lagi. Walaupun teknik brute force memiliki algoritma yang sederhana, tetapi penggunaan algoritma bruteforce seringkali tidak efektif dengan O(n) karena harus melakukan iterasi pada keseluruhan elemen.

1.2 Penjelasan Permainan Cyberpunk 2077 Breach Protocol

Cyberpunk 2077 Breach Protocol merupakan suatu minigame meretas yang ada pada permainan video game *Cyberpunk 2077*. Pada permainan ini terdapat beberapa komponen penting yang perlu diperhatikan diantaranya :

1. Token–terdiri dari dua karakter alfanumerik seperti E9, BD, dan 55.
2. Matriks– terdiri atas token-token yang akan dipilih untuk menyusun urutan kode.
3. Sekuens–sebuah rangkaian token (dua atau lebih) yang harus dicocokkan.
4. Buffer– jumlah maksimal token yang dapat disusun secara sekuensial.

terdapat beberapa aturan yang perlu diperhatikan antara lain:

1. Pemain bergerak dengan pola horizontal, vertikal, horizontal, vertikal (bergantian) hingga semua sekuens berhasil dicocokkan atau buffer penuh.
2. Pemain memulai dengan memilih satu token pada posisi baris paling atas dari matriks.
3. Sekuens dicocokkan pada token-token yang berada di buffer.
4. Satu token pada buffer dapat digunakan pada lebih dari satu sekuens.
5. Setiap sekuens memiliki bobot hadiah atau reward yang variatif.
6. Sekuens memiliki panjang minimal berupa dua token.

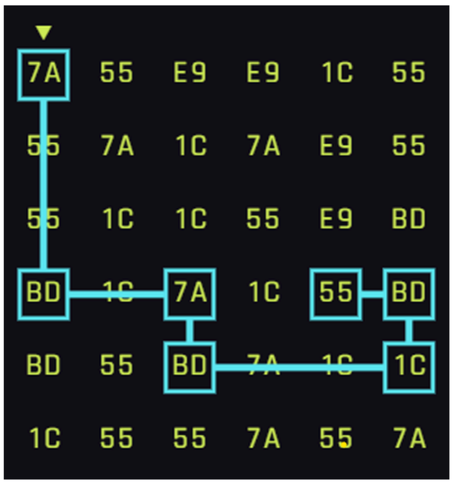
Ilustrasi kasus :
1. Diberikan data berupa buffer yang jumlahnya tujuh
2. Diberikan Contoh Matriks sebagai berikut

7A	55	E9	E9	1C	55
55	7A	1C	7A	E9	55
55	1C	1C	55	E9	BD
BD	1C	7A	1C	55	BD
BD	55	BD	7A	1C	1C
1C	55	55	7A	55	7A

3. Diberikan pula sekuens sebagai berikut :
    1. BD E9 1C dengan hadiah berbobot 15.
    2. BD 7A BD dengan hadiah berbobot 20.
    3. BD 1C BD 55 dengan hadiah berbobot 30.

Solusi optimal yang akan didapat yaitu :
- Bobot hadiah : 50
- Langkah : 6

  Ilustrasi Langkah :



1.3 Deskripsi Langkah Pengerjaan

Pada pencarian solusi untuk permainan Breach Protocol, akan digunakan algoritma brute force yang tujuannya melakukan iterasi untuk menemukan solusi paling optimal untuk tiap kombinasi matriks

1. Terdapat 2 Input yang dapat diterima oleh sistem. pemain atau user yang menggunakan program dapat memilih teknik Input yaitu melalui text file dalam format txt atau dengan melakukan input secara langsung
2. Pada input dengan text file dalam format txt akan diminta beberapa hal diantaranya : buffer_size, matrix_width matrix_height, matrix, number_of_sequences, dan serangkaian sequences dengan masing masing rewardnya. Penempatan data pada filetext harus berurutan dan sesuai dengan format yang diberikan.
3. Pada input dengan input secara langsung terdapat beberapa input yang diperlukan yaitu jumlah_token_unik, token, ukuran_buffer, ukuran_matriks, jumlah_sequens, dan ukuran_maksimal_sekuens. Setelah itu akan dihasilkan yaitu matriks, sekuens, dengan sekuens_rewardnya masing masing secara random dan automatis berdasarkan data input yang diberikan
4. Program akan mencari semua kemungkinan kombinasi yang selanjutnya akan dicocokan dengan sekuens yang ada
5. pada awalnya, program akan mencari seluruh token unique yang ada. selanjutnya, program juga akan mencari seluruh token yang dapat ditemui untuk masing masing unique token. hal ini bertujuan sebagai caching untuk mempercepat perkerjaan nantinya. hal ini selanjutnya akan dijadikan sebagai dictionary agar mudah digunakan
6. Setelah itu, program akan mencari seluruh kombinasi token yang dapat didapatkan. program akan melakukan iterasi untuk tiap elemen pada matriks baris ke-0. selanjutnya, secara recursive, akan dicari kombinasi yang mungkin untuk didapatkan. program ini juga akan menyimpan data lainnya seperti coordinat path yang dilalui.
7. Setiap kali program terulang secara recursive, program akan mencari token-token yang mungkin dilalui nantinya dengan index dari tiap token tersebut. setelah itu akan di iterasi kembali untuk tiap token yang mungkin di tiap jalur yang ditemui, tiap iterasi akan kembali dilakukan perhitungan recursive yang sama dengan jalur yang berbeda dengan aturan horizontal, vertical, horizontal, vertical.
8. Program akan berhenti pada sebuah basis dimana panjang list yang menyimpan token yang sudah dilalui memiliki panjang yang sama dengan jumlah buffer yang disediakan
9. Setelah semua kombinasi didapatkan, akan dicocokan tiap sequence pada combination yang ada. untuk sequence yang sudah cocok tidak dapat digunakan kembali dan total_reward nya akan dihitung
10. Ketika combinasi sudah di iterasi hingga habis, total_reward, token yang dilalui, kooridnat yang dilalui akan dikembalikan sebagai output dari program

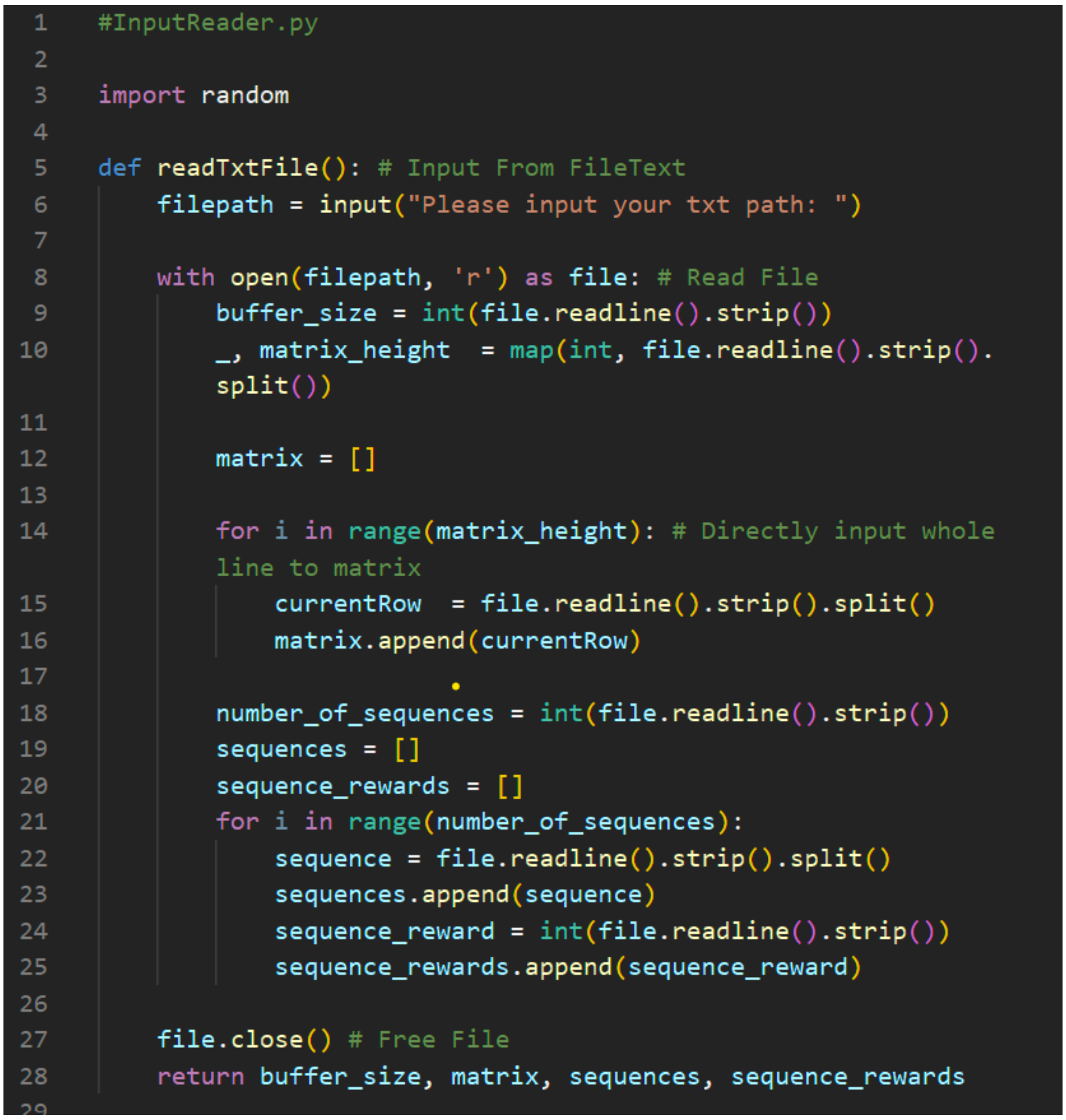
Inti dari program ini adalah program akan mendapatkan input yang nantinya akan dicari seluruh kombinasi yang ada dengan cara recursive untuk tiap kombinasi yang mungkin. setelah itu, untuk tiap kombinasi yang ada akan dilakukan pengecekan sekuens untuk tiap kombinasinya.

BAB II

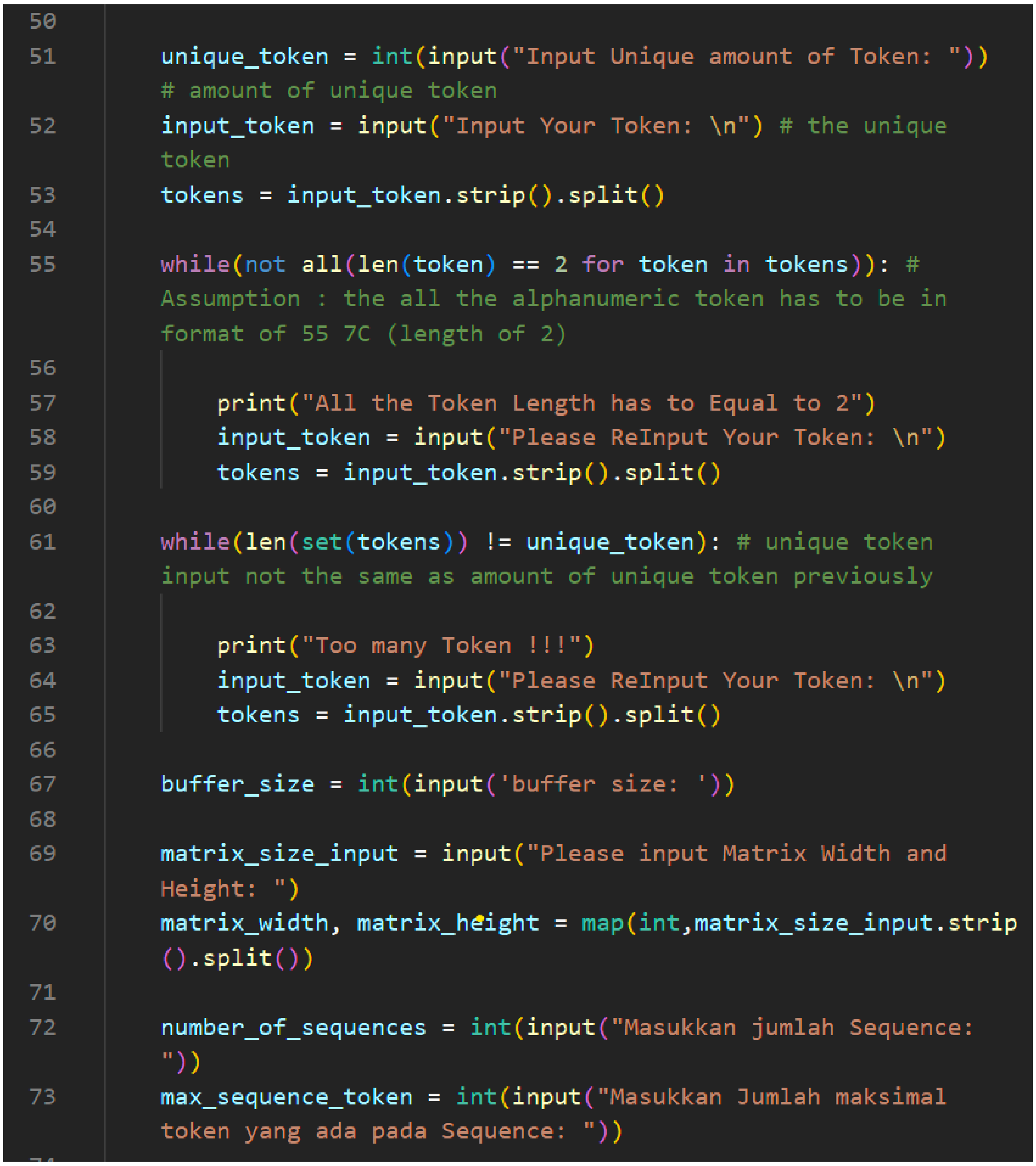
Source Program

Dalam pembuatan program ini, digunakan python sebagai bahasa pemrogramannya. Struktur dari program ini terbagi menjadi 5 file, yaitu **InputReader.py**, **CombinationGenerator.py**, **FileWriter.py**, **GUI.py**, dan **main.py**.

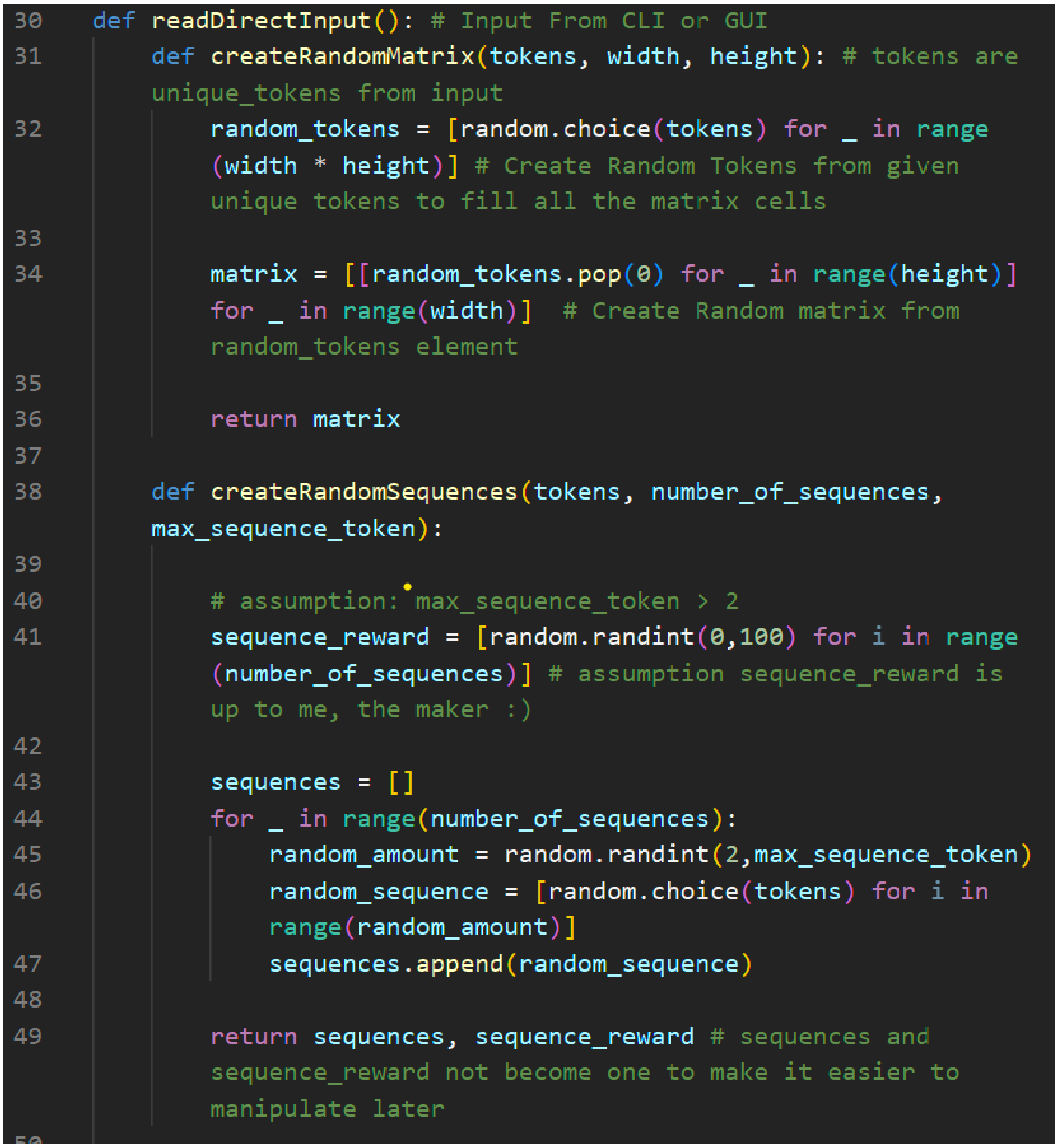
2.1 InputReader.py



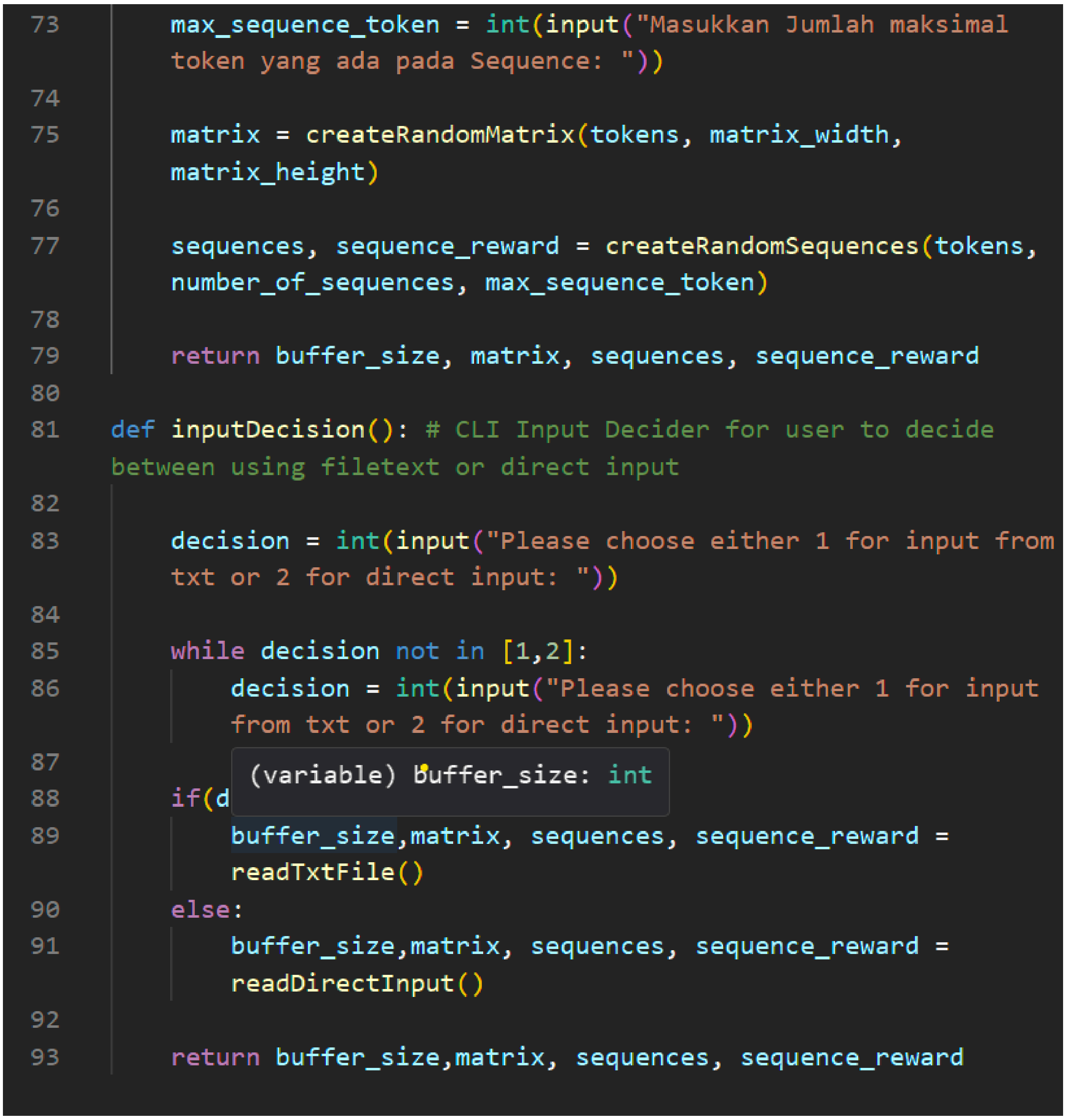
```python
#InputReader.py

import random

def readTxtFile(): # Input From FileText
    filepath = input("Please input your txt path: ")

    with open(filepath, 'r') as file: # Read File
        buffer_size = int(file.readline().strip())
        _, matrix_height  = map(int, file.readline().strip().split())

        matrix = []

        for i in range(matrix_height): # Directly input whole line to matrix
            currentRow  = file.readline().strip().split()
            matrix.append(currentRow)

        number_of_sequences = int(file.readline().strip())
        sequences = []
        sequence_rewards = []
        for i in range(number_of_sequences):
            sequence = file.readline().strip().split()
            sequences.append(sequence)
            sequence_reward = int(file.readline().strip())
            sequence_rewards.append(sequence_reward)

    file.close() # Free File
    return buffer_size, matrix, sequences, sequence_rewards
```



```python

    unique_token = int(input("Input Unique amount of Token: ")) # amount of unique token
    input_token = input("Input Your Token: \n") # the unique token
    tokens = input_token.strip().split()

    while(not all(len(token) == 2 for token in tokens)): # Assumption : the all the alphanumeric token has to be in format of 55 7C (length of 2)

        print("All the Token Length has to Equal to 2")
        input_token = input("Please ReInput Your Token: \n")
        tokens = input_token.strip().split()

    while(len(set(tokens)) != unique_token): # unique token input not the same as amount of unique token previously

        print("Too many Token !!!")
        input_token = input("Please ReInput Your Token: \n")
        tokens = input_token.strip().split()

    buffer_size = int(input('buffer size: '))

    matrix_size_input = input("Please input Matrix Width and Height: ")
    matrix_width, matrix_height = map(int,matrix_size_input.strip().split())

    number_of_sequences = int(input("Masukkan jumlah Sequence: "))
    max_sequence_token = int(input("Masukkan Jumlah maksimal token yang ada pada Sequence: "))
```

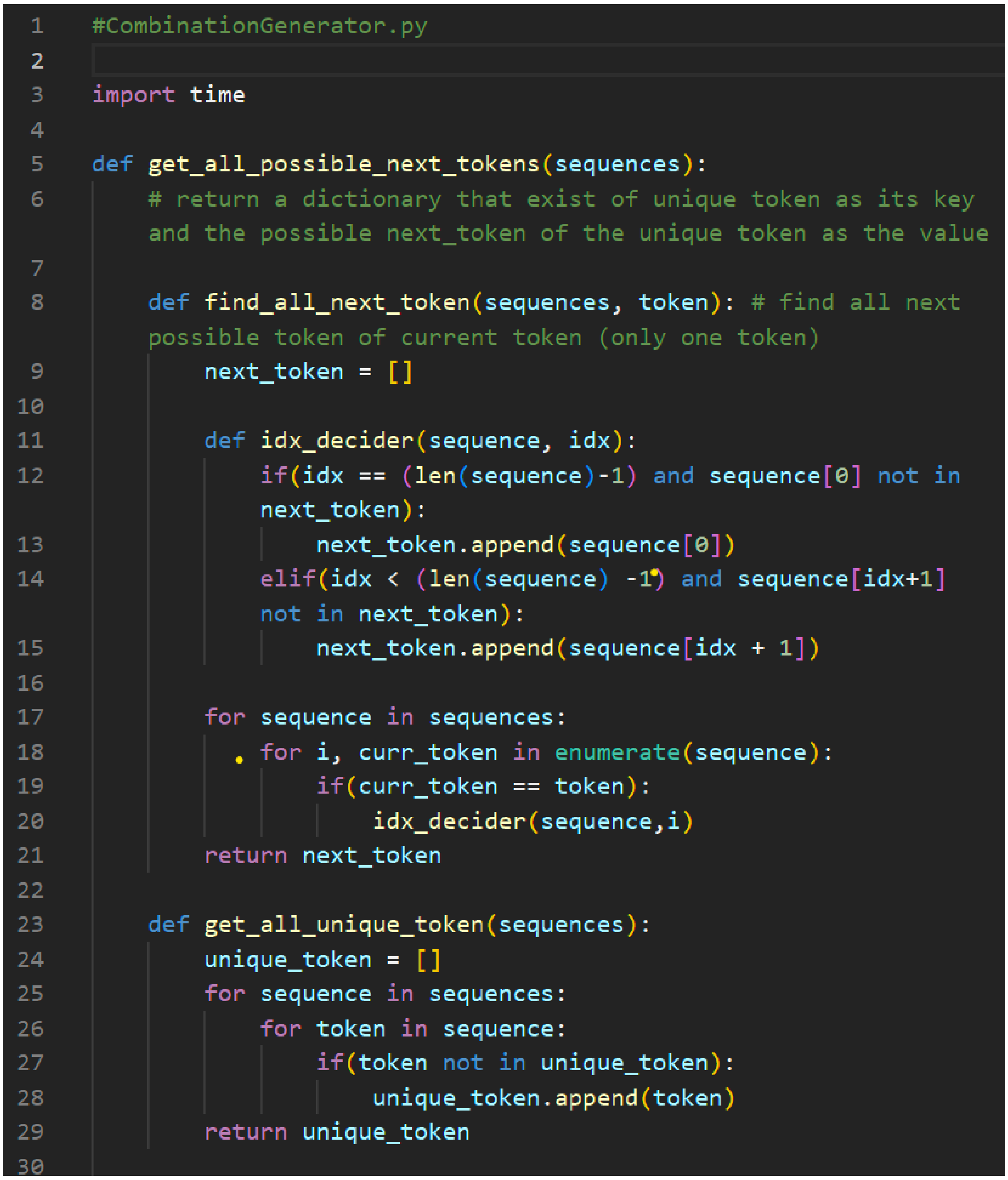


```python
def readDirectInput(): # Input From CLI or GUI
    def createRandomMatrix(tokens, width, height): # tokens are unique_tokens from input
        random_tokens = [random.choice(tokens) for _ in range(width * height)] # Create Random Tokens from given unique tokens to fill all the matrix cells

        matrix = [[random_tokens.pop(0) for _ in range(height)] for _ in range(width)]  # Create Random matrix from random_tokens element

        return matrix

    def createRandomSequences(tokens, number_of_sequences, max_sequence_token):

        # assumption: max_sequence_token > 2
        sequence_reward = [random.randint(0,100) for i in range(number_of_sequences)] # assumption sequence_reward is up to me, the maker :)

        sequences = []
        for _ in range(number_of_sequences):
            random_amount = random.randint(2,max_sequence_token)
            random_sequence = [random.choice(tokens) for i in range(random_amount)]
            sequences.append(random_sequence)

        return sequences, sequence_reward # sequences and sequence_reward not become one to make it easier to manipulate later
```

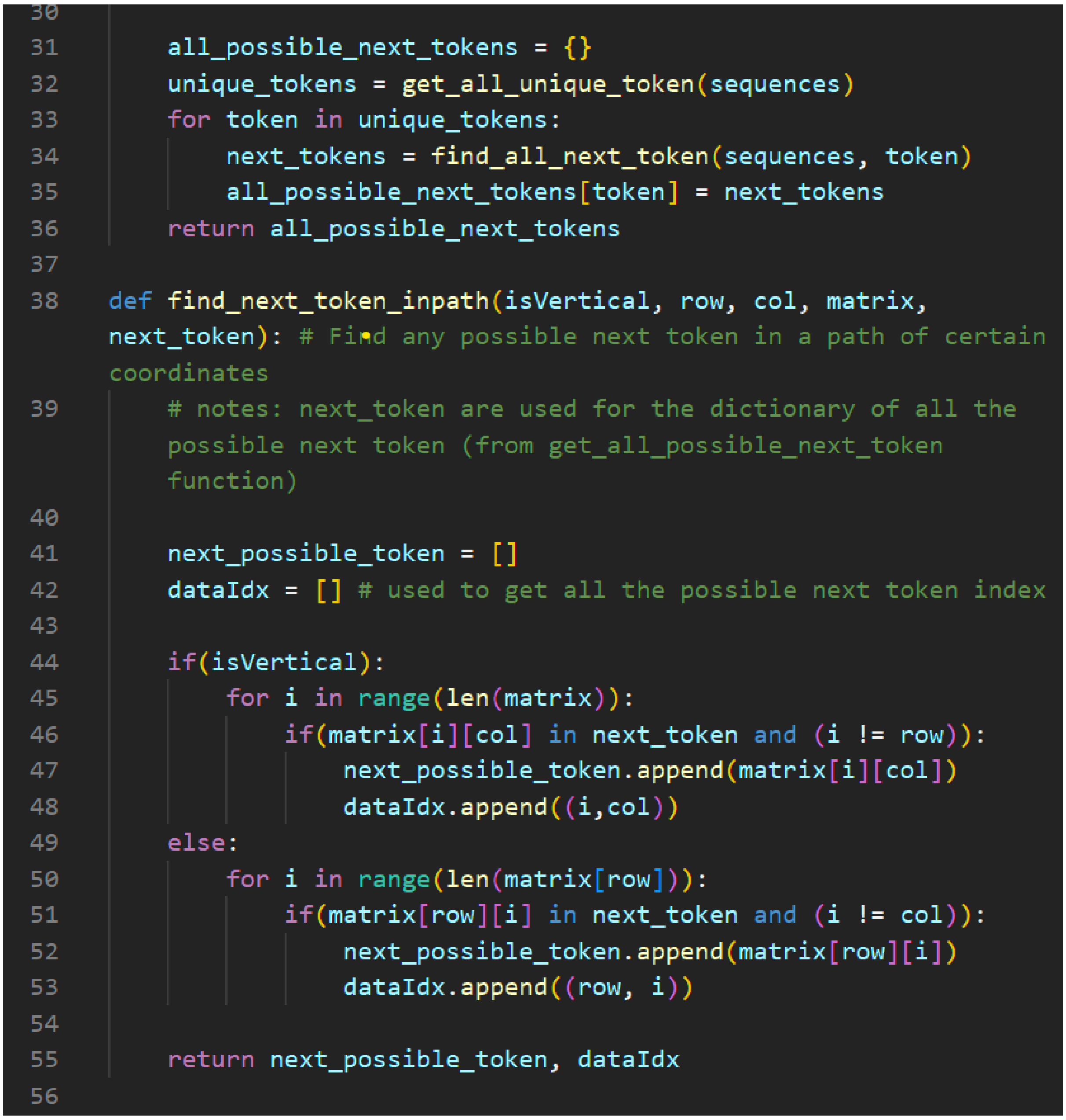


```
    max_sequence_token = int(input("Masukkan Jumlah maksimal token yang ada pada Sequence: "))

    matrix = createRandomMatrix(tokens, matrix_width, matrix_height)

    sequences, sequence_reward = createRandomSequences(tokens, number_of_sequences, max_sequence_token)

    return buffer_size, matrix, sequences, sequence_reward

def inputDecision(): # CLI Input Decider for user to decide between using filetext or direct input

    decision = int(input("Please choose either 1 for input from txt or 2 for direct input: "))

    while decision not in [1,2]:
        decision = int(input("Please choose either 1 for input from txt or 2 for direct input: "))

          (variable) buffer_size: int
    if(d
        buffer_size,matrix, sequences, sequence_reward = readTxtFile()
    else:
        buffer_size,matrix, sequences, sequence_reward = readDirectInput()

    return buffer_size,matrix, sequences, sequence_reward
```

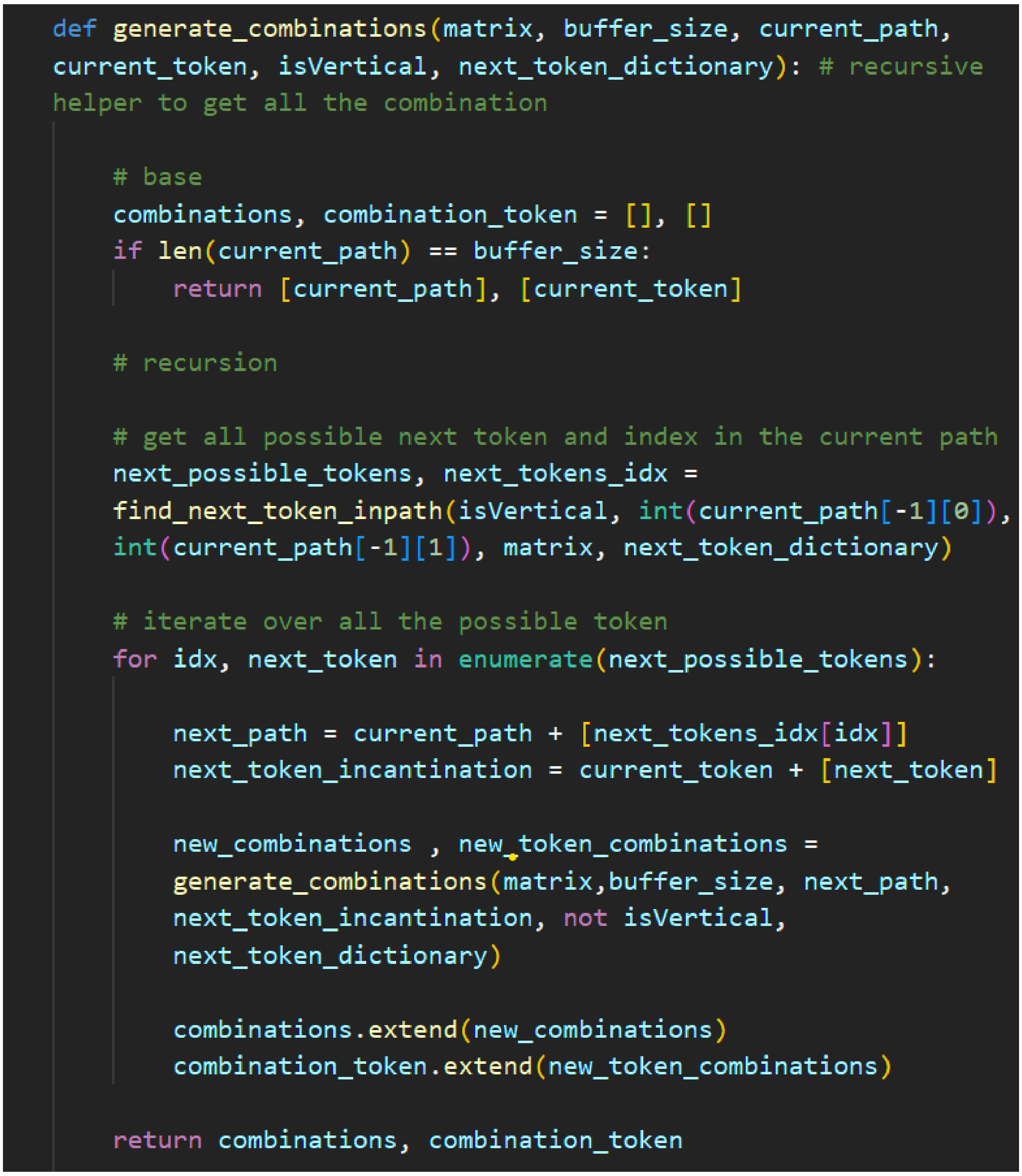
2.2 CombinationGenerator.py



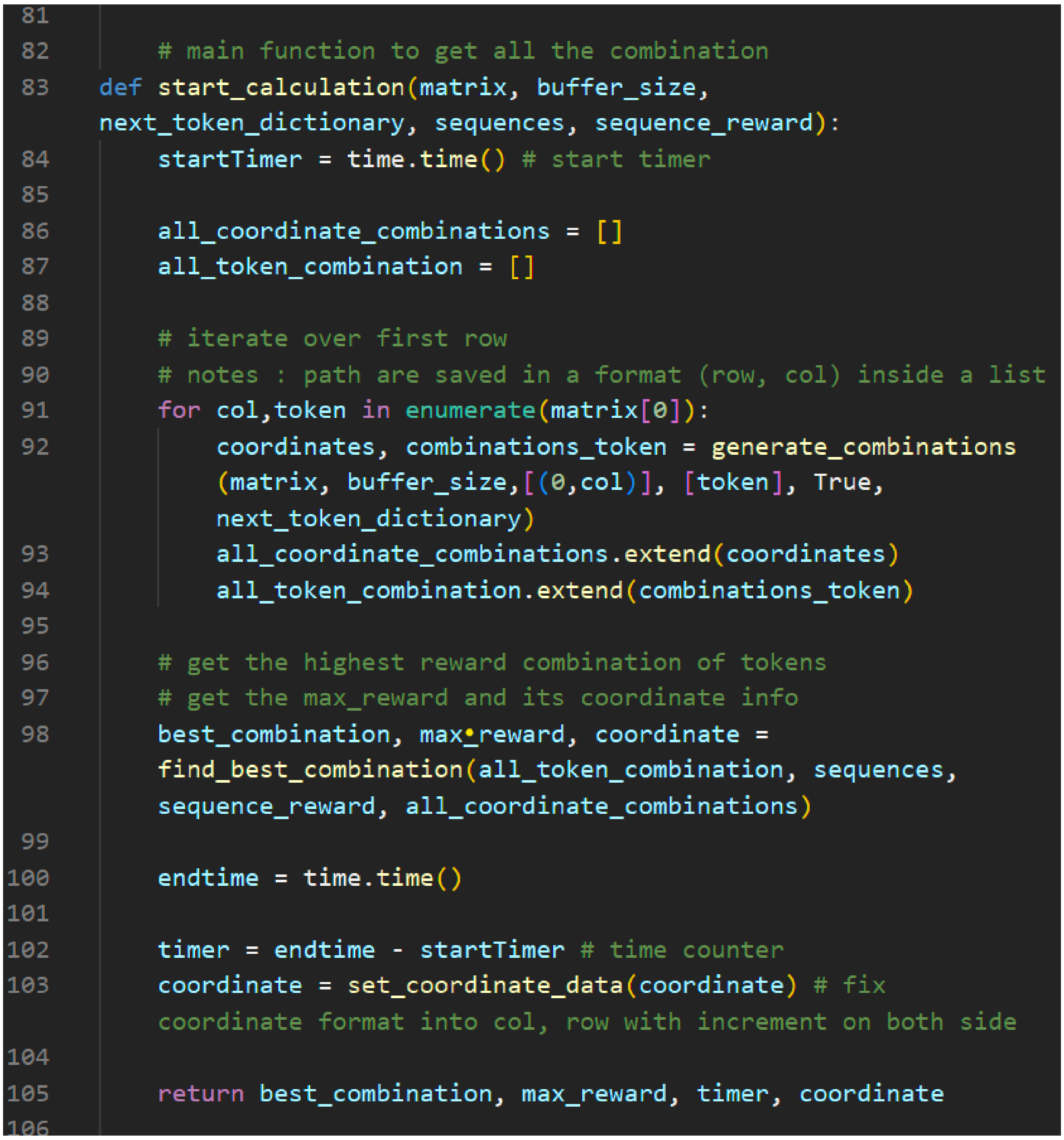
```python
#CombinationGenerator.py

import time

def get_all_possible_next_tokens(sequences):
    # return a dictionary that exist of unique token as its key and the possible next_token of the unique token as the value

    def find_all_next_token(sequences, token): # find all next possible token of current token (only one token)
        next_token = []

        def idx_decider(sequence, idx):
            if(idx == (len(sequence)-1) and sequence[0] not in next_token):
                next_token.append(sequence[0])
            elif(idx < (len(sequence) -1) and sequence[idx+1] not in next_token):
                next_token.append(sequence[idx + 1])

        for sequence in sequences:
            for i, curr_token in enumerate(sequence):
                if(curr_token == token):
                    idx_decider(sequence,i)
        return next_token

    def get_all_unique_token(sequences):
        unique_token = []
        for sequence in sequences:
            for token in sequence:
                if(token not in unique_token):
                    unique_token.append(token)
        return unique_token

```



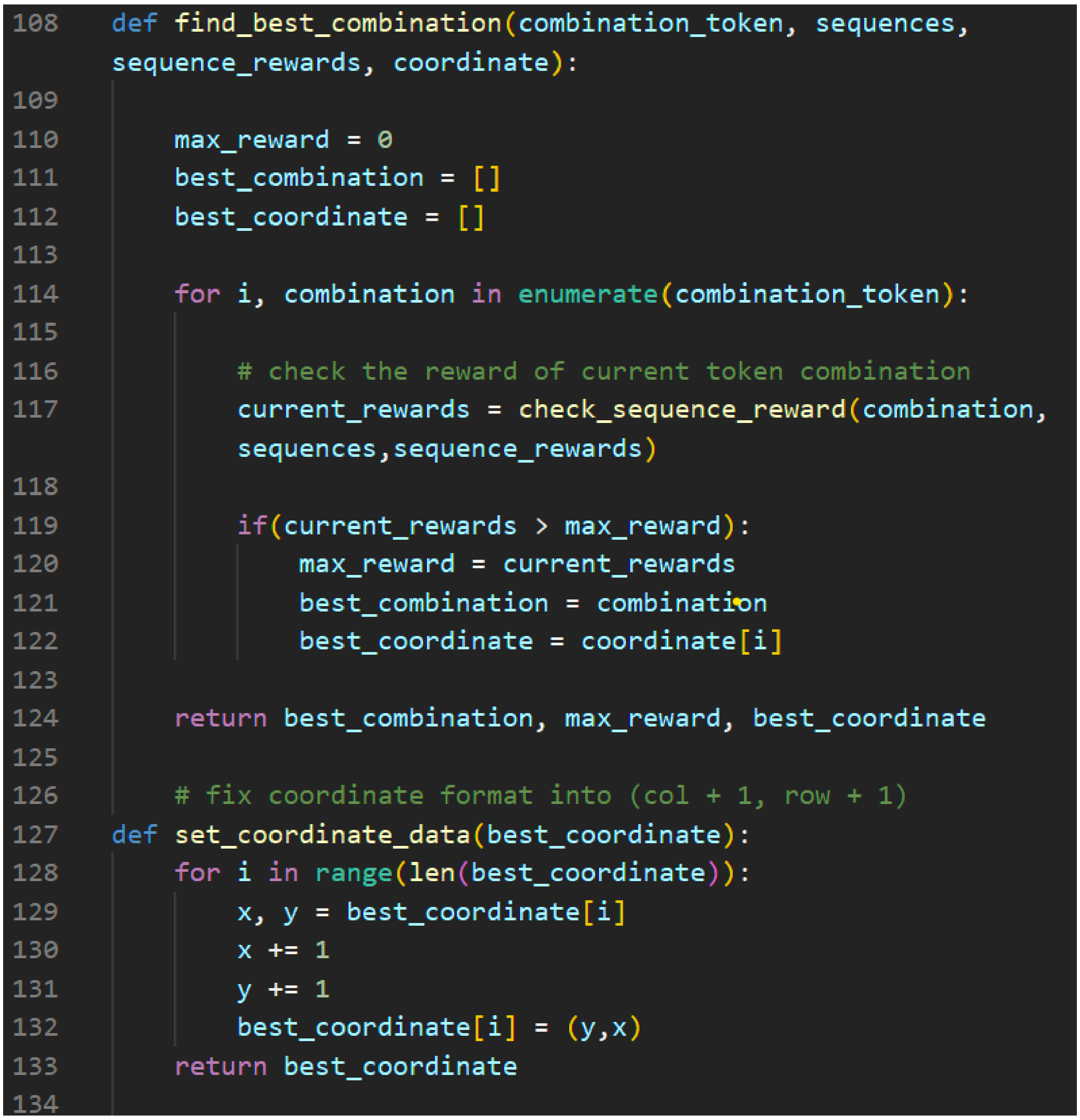
```python
    all_possible_next_tokens = {}
    unique_tokens = get_all_unique_token(sequences)
    for token in unique_tokens:
        next_tokens = find_all_next_token(sequences, token)
        all_possible_next_tokens[token] = next_tokens
    return all_possible_next_tokens

def find_next_token_inpath(isVertical, row, col, matrix, next_token): # Find any possible next token in a path of certain coordinates
    # notes: next_token are used for the dictionary of all the possible next token (from get_all_possible_next_token function)

    next_possible_token = []
    dataIdx = [] # used to get all the possible next token index

    if(isVertical):
        for i in range(len(matrix)):
            if(matrix[i][col] in next_token and (i != row)):
                next_possible_token.append(matrix[i][col])
                dataIdx.append((i,col))
    else:
        for i in range(len(matrix[row])):
            if(matrix[row][i] in next_token and (i != col)):
                next_possible_token.append(matrix[row][i])
                dataIdx.append((row, i))

    return next_possible_token, dataIdx

```



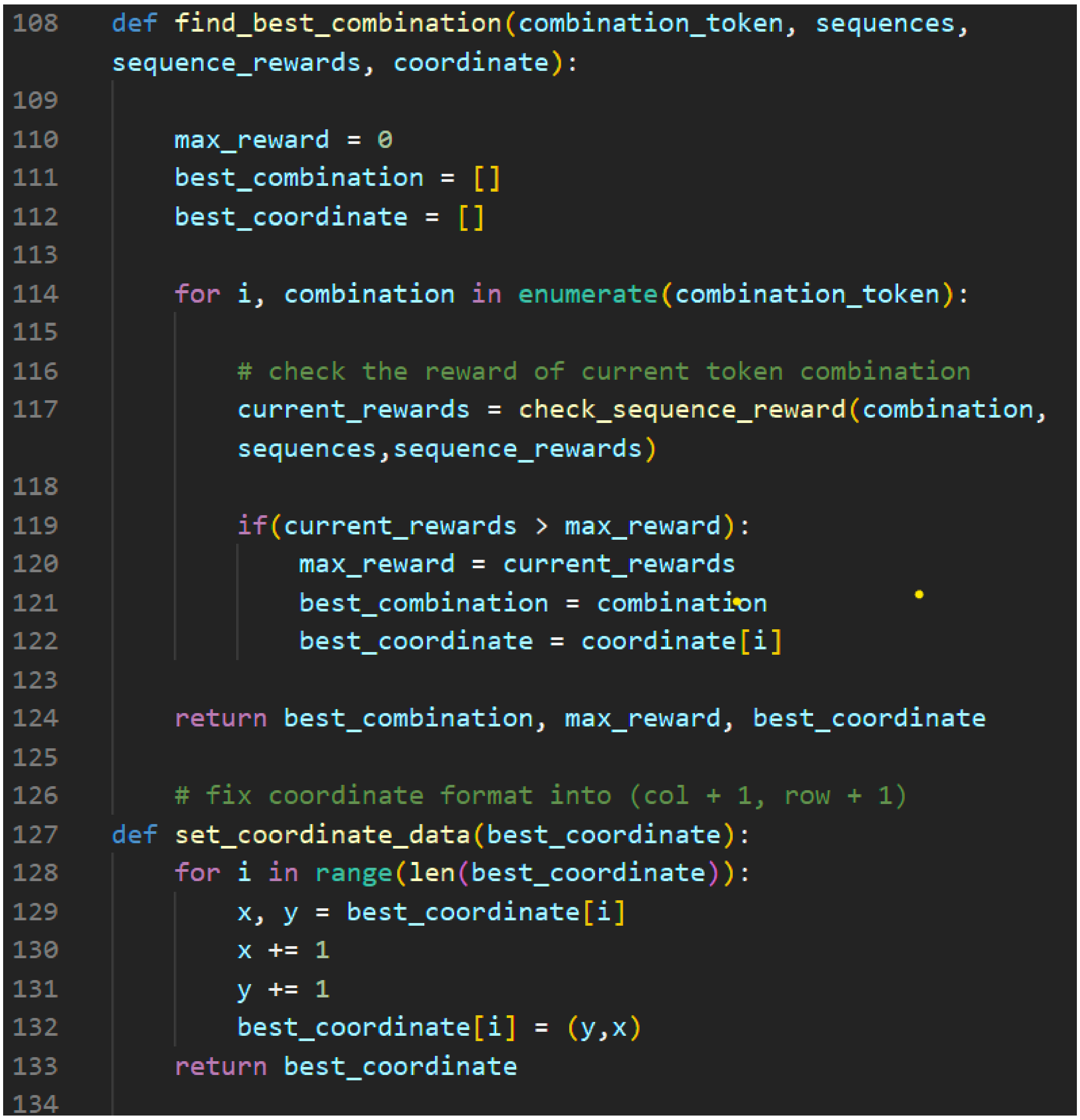
```python
def generate_combinations(matrix, buffer_size, current_path,
current_token, isVertical, next_token_dictionary): # recursive
helper to get all the combination

    # base
    combinations, combination_token = [], []
    if len(current_path) == buffer_size:
        return [current_path], [current_token]

    # recursion

    # get all possible next token and index in the current path
    next_possible_tokens, next_tokens_idx =
    find_next_token_inpath(isVertical, int(current_path[-1][0]),
    int(current_path[-1][1]), matrix, next_token_dictionary)

    # iterate over all the possible token
    for idx, next_token in enumerate(next_possible_tokens):

        next_path = current_path + [next_tokens_idx[idx]]
        next_token_incantination = current_token + [next_token]

        new_combinations , new_token_combinations =
        generate_combinations(matrix,buffer_size, next_path,
        next_token_incantination, not isVertical,
        next_token_dictionary)

        combinations.extend(new_combinations)
        combination_token.extend(new_token_combinations)

    return combinations, combination_token
```



```python

    # main function to get all the combination
def start_calculation(matrix, buffer_size, next_token_dictionary, sequences, sequence_reward):
    startTimer = time.time() # start timer

    all_coordinate_combinations = []
    all_token_combination = []

    # iterate over first row
    # notes : path are saved in a format (row, col) inside a list
    for col,token in enumerate(matrix[0]):
        coordinates, combinations_token = generate_combinations(matrix, buffer_size,[(0,col)], [token], True, next_token_dictionary)
        all_coordinate_combinations.extend(coordinates)
        all_token_combination.extend(combinations_token)

    # get the highest reward combination of tokens
    # get the max_reward and its coordinate info
    best_combination, max_reward, coordinate = find_best_combination(all_token_combination, sequences, sequence_reward, all_coordinate_combinations)

    endtime = time.time()

    timer = endtime - startTimer # time counter
    coordinate = set_coordinate_data(coordinate) # fix coordinate format into col, row with increment on both side

    return best_combination, max_reward, timer, coordinate

```

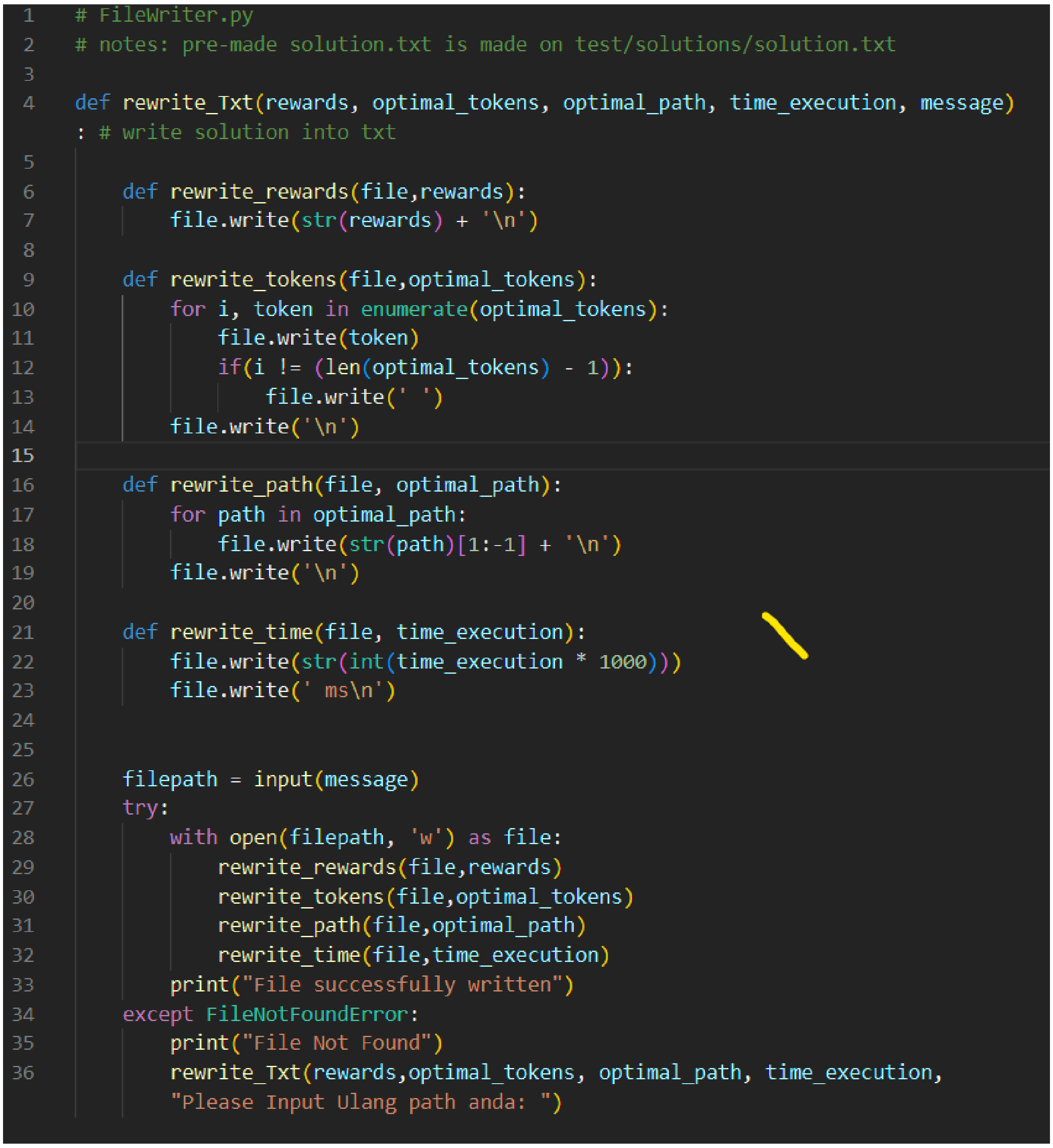


```python
def find_best_combination(combination_token, sequences, sequence_rewards, coordinate):

    max_reward = 0
    best_combination = []
    best_coordinate = []

    for i, combination in enumerate(combination_token):

        # check the reward of current token combination
        current_rewards = check_sequence_reward(combination, sequences,sequence_rewards)

        if(current_rewards > max_reward):
            max_reward = current_rewards
            best_combination = combination
            best_coordinate = coordinate[i]

    return best_combination, max_reward, best_coordinate

    # fix coordinate format into (col + 1, row + 1)
def set_coordinate_data(best_coordinate):
    for i in range(len(best_coordinate)):
        x, y = best_coordinate[i]
        x += 1
        y += 1
        best_coordinate[i] = (y,x)
    return best_coordinate

```



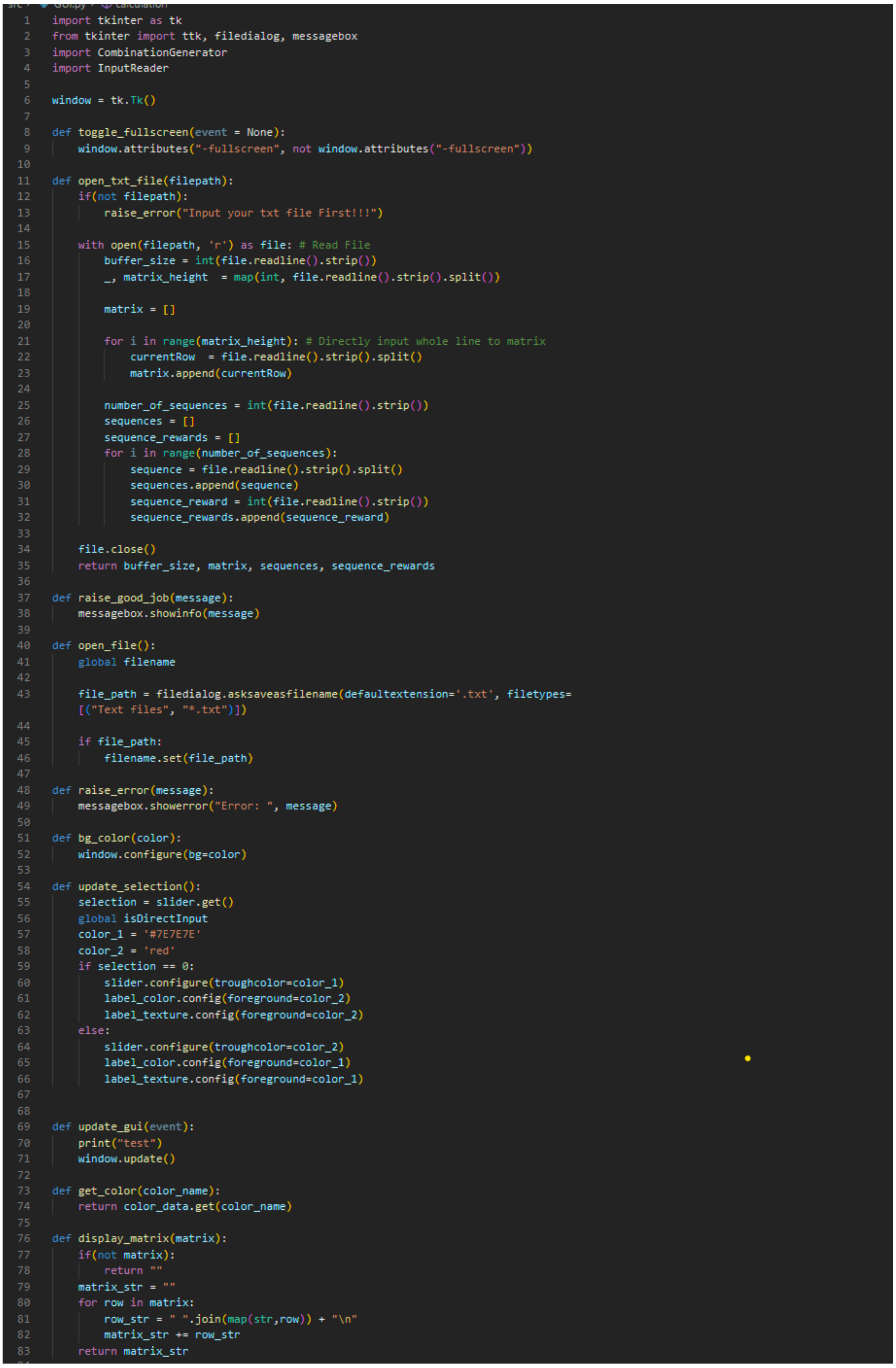
```python
def find_best_combination(combination_token, sequences, sequence_rewards, coordinate):

    max_reward = 0
    best_combination = []
    best_coordinate = []

    for i, combination in enumerate(combination_token):

        # check the reward of current token combination
        current_rewards = check_sequence_reward(combination, sequences,sequence_rewards)

        if(current_rewards > max_reward):
            max_reward = current_rewards
            best_combination = combination
            best_coordinate = coordinate[i]

    return best_combination, max_reward, best_coordinate

    # fix coordinate format into (col + 1, row + 1)
def set_coordinate_data(best_coordinate):
    for i in range(len(best_coordinate)):
        x, y = best_coordinate[i]
        x += 1
        y += 1
        best_coordinate[i] = (y,x)
    return best_coordinate

```

2.3 FileWriter.py

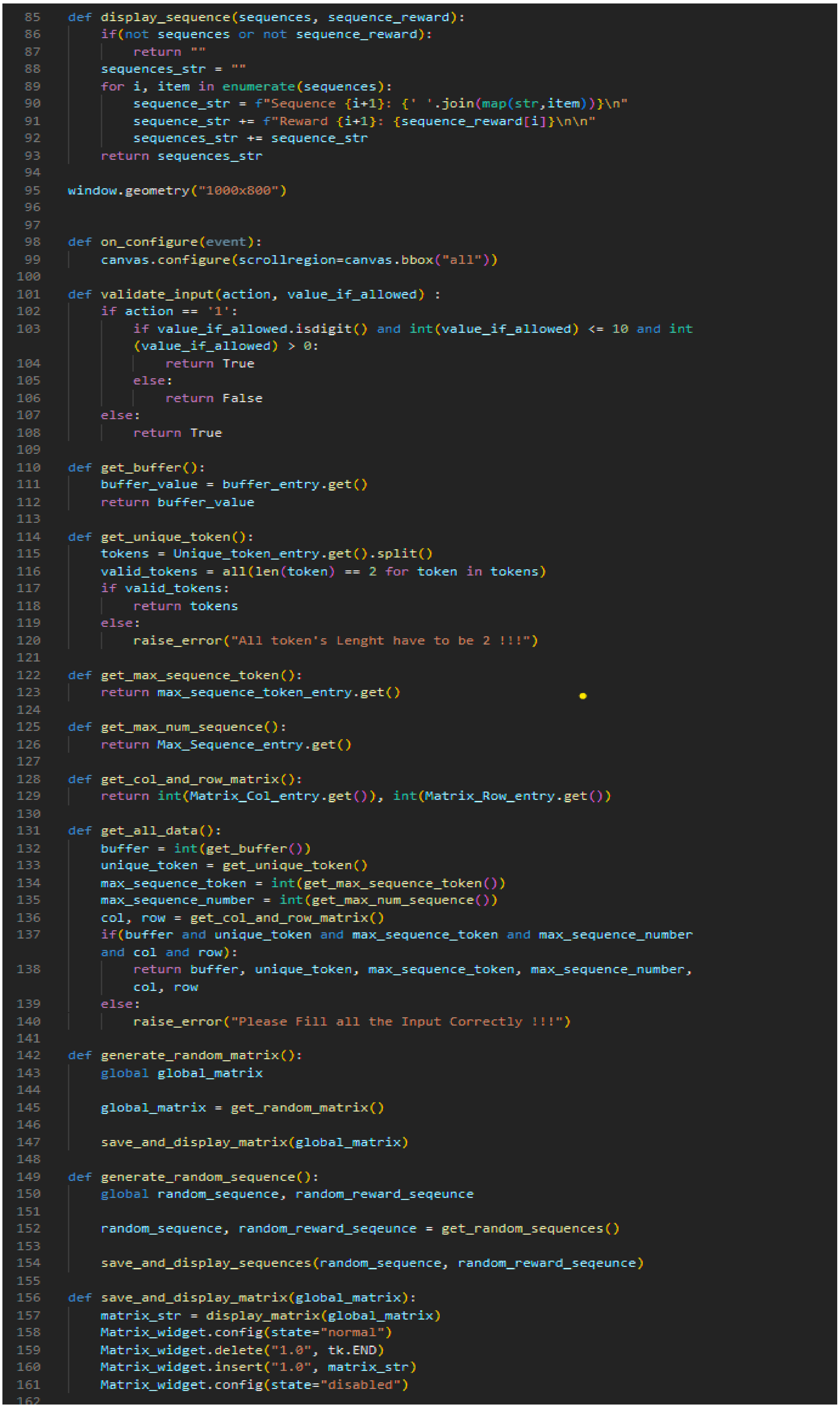


```python
# FileWriter.py
# notes: pre-made solution.txt is made on test/solutions/solution.txt

def rewrite_Txt(rewards, optimal_tokens, optimal_path, time_execution, message): # write solution into txt

    def rewrite_rewards(file,rewards):
        file.write(str(rewards) + '\n')

    def rewrite_tokens(file,optimal_tokens):
        for i, token in enumerate(optimal_tokens):
            file.write(token)
            if(i != (len(optimal_tokens) - 1)):
                file.write(' ')
        file.write('\n')

    def rewrite_path(file, optimal_path):
        for path in optimal_path:
            file.write(str(path)[1:-1] + '\n')
        file.write('\n')

    def rewrite_time(file, time_execution):
        file.write(str(int(time_execution * 1000)))
        file.write(' ms\n')


    filepath = input(message)
    try:
        with open(filepath, 'w') as file:
            rewrite_rewards(file,rewards)
            rewrite_tokens(file,optimal_tokens)
            rewrite_path(file,optimal_path)
            rewrite_time(file,time_execution)
        print("File successfully written")
    except FileNotFoundError:
        print("File Not Found")
        rewrite_Txt(rewards,optimal_tokens, optimal_path, time_execution, "Please Input Ulang path anda: ")
```

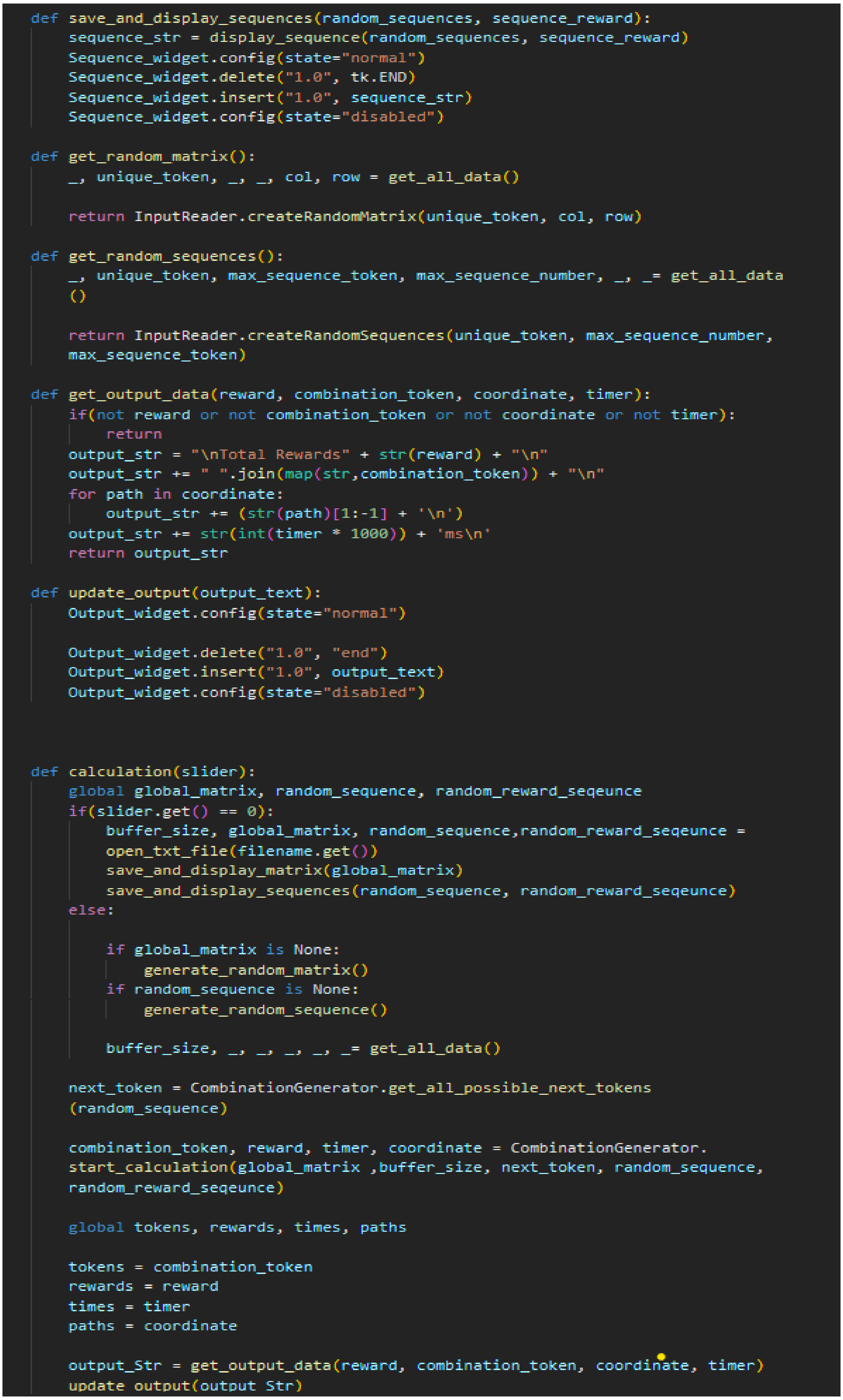
2.4 GUI.py



```python
import tkinter as tk
from tkinter import ttk, filedialog, messagebox
import CombinationGenerator
import InputReader

window = tk.Tk()

def toggle_fullscreen(event = None):
    window.attributes("-fullscreen", not window.attributes("-fullscreen"))

def open_txt_file(filepath):
    if(not filepath):
        raise_error("Input your txt file First!!!")

    with open(filepath, 'r') as file: # Read File
        buffer_size = int(file.readline().strip())
        _, matrix_height  = map(int, file.readline().strip().split())

        matrix = []

        for i in range(matrix_height): # Directly input whole line to matrix
            currentRow  = file.readline().strip().split()
            matrix.append(currentRow)

        number_of_sequences = int(file.readline().strip())
        sequences = []
        sequence_rewards = []
        for i in range(number_of_sequences):
            sequence = file.readline().strip().split()
            sequences.append(sequence)
            sequence_reward = int(file.readline().strip())
            sequence_rewards.append(sequence_reward)

    file.close()
    return buffer_size, matrix, sequences, sequence_rewards

def raise_good_job(message):
    messagebox.showinfo(message)

def open_file():
    global filename

    file_path = filedialog.asksaveasfilename(defaultextension='.txt', filetypes=
    [("Text files", "*.txt")])

    if file_path:
        filename.set(file_path)

def raise_error(message):
    messagebox.showerror("Error: ", message)

def bg_color(color):
    window.configure(bg=color)

def update_selection():
    selection = slider.get()
    global isDirectInput
    color_1 = '#7E7E7E'
    color_2 = 'red'
    if selection == 0:
        slider.configure(troughcolor=color_1)
        label_color.config(foreground=color_2)
        label_texture.config(foreground=color_2)
    else:
        slider.configure(troughcolor=color_2)
        label_color.config(foreground=color_1)
        label_texture.config(foreground=color_1)


def update_gui(event):
    print("test")
    window.update()

def get_color(color_name):
    return color_data.get(color_name)

def display_matrix(matrix):
    if(not matrix):
        return ""
    matrix_str = ""
    for row in matrix:
        row_str = " ".join(map(str,row)) + "\n"
        matrix_str += row_str
    return matrix_str
```

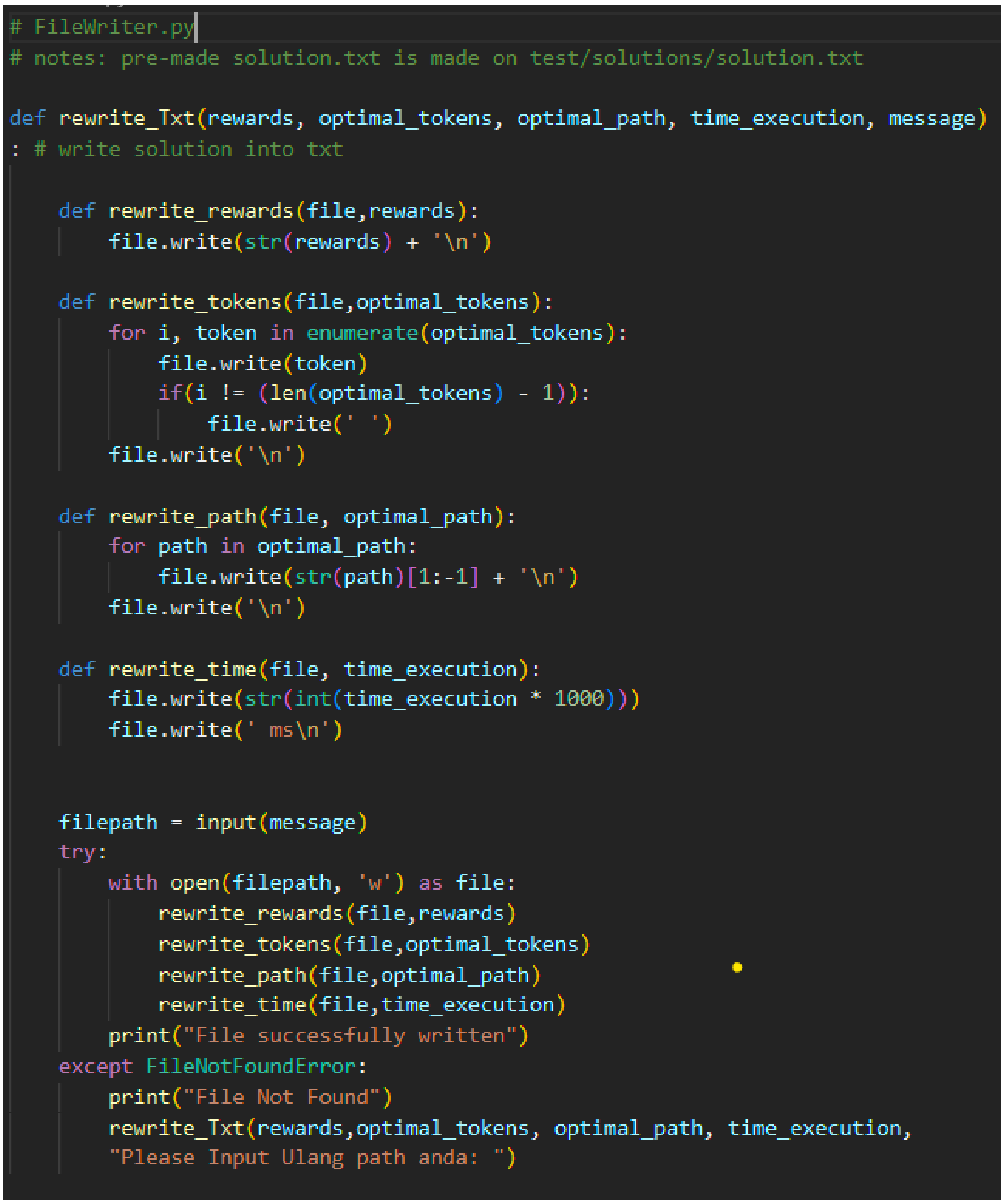


```python
def display_sequence(sequences, sequence_reward):
    if(not sequences or not sequence_reward):
        return ""
    sequences_str = ""
    for i, item in enumerate(sequences):
        sequence_str = f"Sequence {i+1}: {' '.join(map(str,item))}\n"
        sequence_str += f"Reward {i+1}: {sequence_reward[i]}\n\n"
        sequences_str += sequence_str
    return sequences_str

window.geometry("1000x800")


def on_configure(event):
    canvas.configure(scrollregion=canvas.bbox("all"))

def validate_input(action, value_if_allowed) :
    if action == '1':
        if value_if_allowed.isdigit() and int(value_if_allowed) <= 10 and int(value_if_allowed) > 0:
            return True
        else:
            return False
    else:
        return True

def get_buffer():
    buffer_value = buffer_entry.get()
    return buffer_value

def get_unique_token():
    tokens = Unique_token_entry.get().split()
    valid_tokens = all(len(token) == 2 for token in tokens)
    if valid_tokens:
        return tokens
    else:
        raise_error("All token's Lenght have to be 2 !!!")

def get_max_sequence_token():
    return max_sequence_token_entry.get()

def get_max_num_sequence():
    return Max_Sequence_entry.get()

def get_col_and_row_matrix():
    return int(Matrix_Col_entry.get()), int(Matrix_Row_entry.get())

def get_all_data():
    buffer = int(get_buffer())
    unique_token = get_unique_token()
    max_sequence_token = int(get_max_sequence_token())
    max_sequence_number = int(get_max_num_sequence())
    col, row = get_col_and_row_matrix()
    if(buffer and unique_token and max_sequence_token and max_sequence_number and col and row):
        return buffer, unique_token, max_sequence_token, max_sequence_number, col, row
    else:
        raise_error("Please Fill all the Input Correctly !!!")

def generate_random_matrix():
    global global_matrix

    global_matrix = get_random_matrix()

    save_and_display_matrix(global_matrix)

def generate_random_sequence():
    global random_sequence, random_reward_seqeunce

    random_sequence, random_reward_seqeunce = get_random_sequences()

    save_and_display_sequences(random_sequence, random_reward_seqeunce)

def save_and_display_matrix(global_matrix):
    matrix_str = display_matrix(global_matrix)
    Matrix_widget.config(state="normal")
    Matrix_widget.delete("1.0", tk.END)
    Matrix_widget.insert("1.0", matrix_str)
    Matrix_widget.config(state="disabled")
```

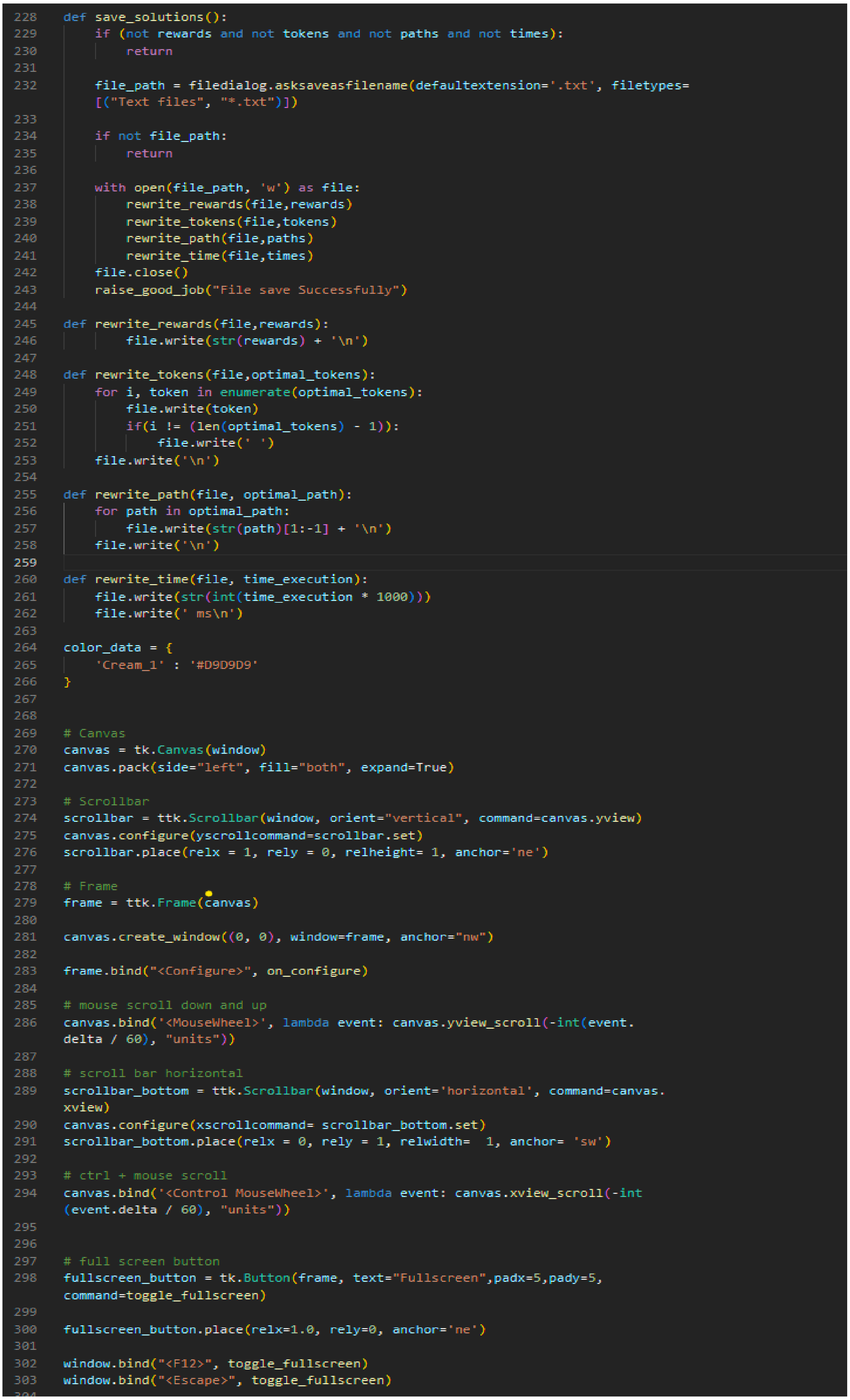


```
def save_and_display_sequences(random_sequences, sequence_reward):
    sequence_str = display_sequence(random_sequences, sequence_reward)
    Sequence_widget.config(state="normal")
    Sequence_widget.delete("1.0", tk.END)
    Sequence_widget.insert("1.0", sequence_str)
    Sequence_widget.config(state="disabled")

def get_random_matrix():
    _, unique_token, _, _, col, row = get_all_data()

    return InputReader.createRandomMatrix(unique_token, col, row)

def get_random_sequences():
    _, unique_token, max_sequence_token, max_sequence_number, _, _= get_all_data
    ()

    return InputReader.createRandomSequences(unique_token, max_sequence_number,
    max_sequence_token)

def get_output_data(reward, combination_token, coordinate, timer):
    if(not reward or not combination_token or not coordinate or not timer):
        return
    output_str = "\nTotal Rewards" + str(reward) + "\n"
    output_str += " ".join(map(str,combination_token)) + "\n"
    for path in coordinate:
        output_str += (str(path)[1:-1] + '\n')
    output_str += str(int(timer * 1000)) + 'ms\n'
    return output_str

def update_output(output_text):
    Output_widget.config(state="normal")

    Output_widget.delete("1.0", "end")
    Output_widget.insert("1.0", output_text)
    Output_widget.config(state="disabled")



def calculation(slider):
    global global_matrix, random_sequence, random_reward_seqeunce
    if(slider.get() == 0):
        buffer_size, global_matrix, random_sequence,random_reward_seqeunce =
        open_txt_file(filename.get())
        save_and_display_matrix(global_matrix)
        save_and_display_sequences(random_sequence, random_reward_seqeunce)
    else:

        if global_matrix is None:
            generate_random_matrix()
        if random_sequence is None:
            generate_random_sequence()

        buffer_size, _, _, _, _, _= get_all_data()

    next_token = CombinationGenerator.get_all_possible_next_tokens
    (random_sequence)

    combination_token, reward, timer, coordinate = CombinationGenerator.
    start_calculation(global_matrix ,buffer_size, next_token, random_sequence,
    random_reward_seqeunce)

    global tokens, rewards, times, paths

    tokens = combination_token
    rewards = reward
    times = timer
    paths = coordinate

    output_Str = get_output_data(reward, combination_token, coordinate, timer)
    update_output(output_Str)
```

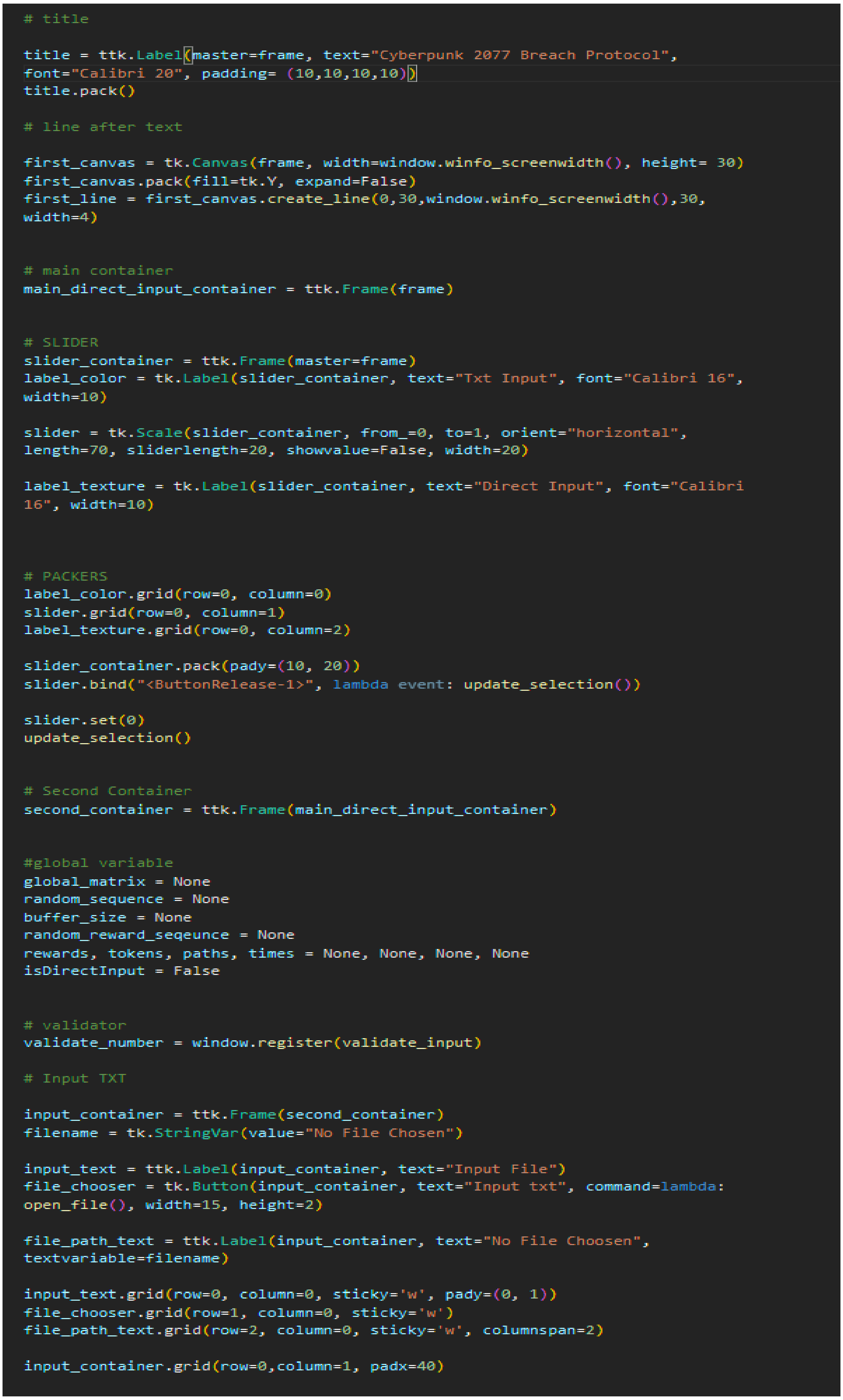
2.5 main.py



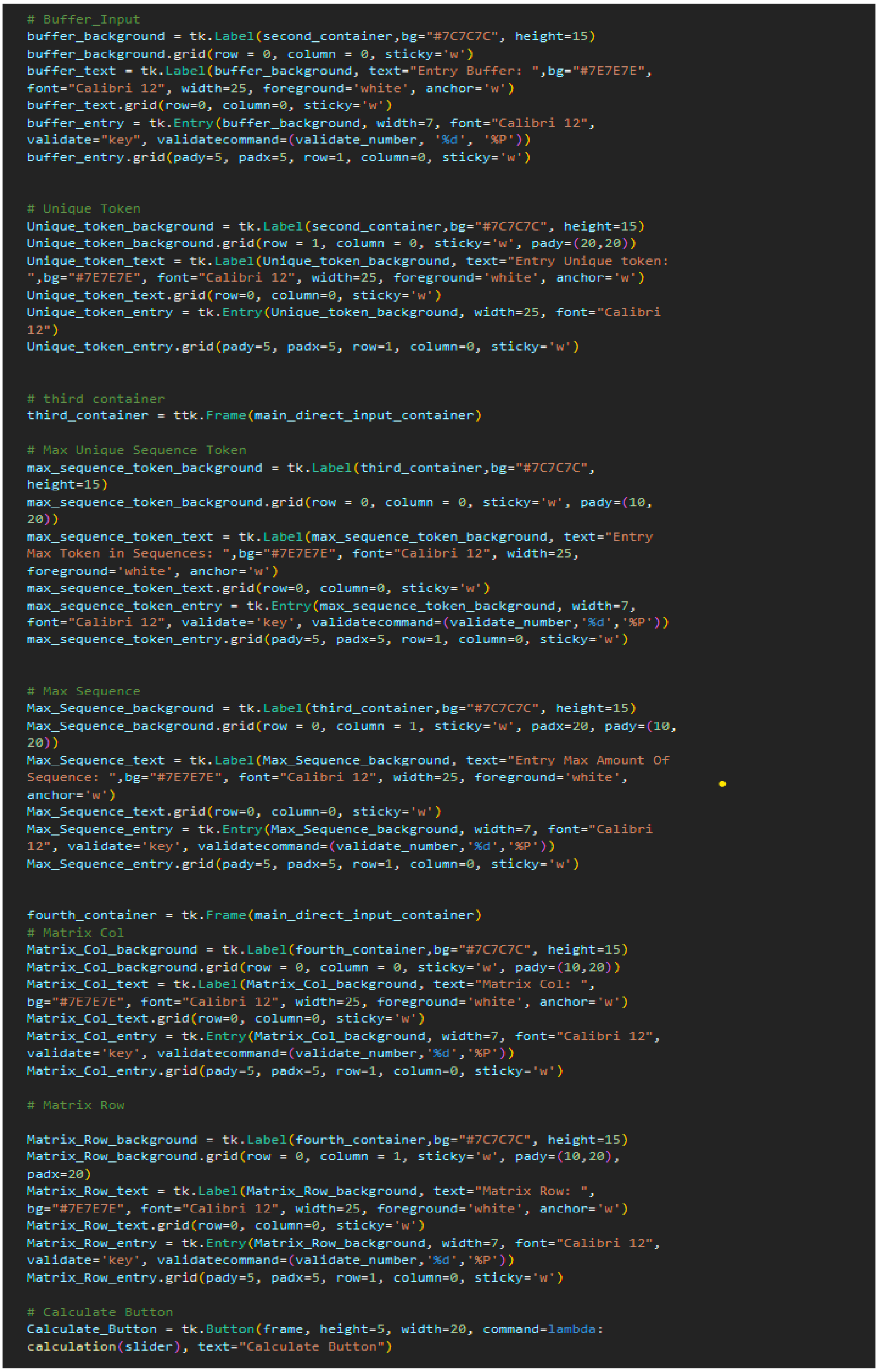
```python
# FileWriter.py
# notes: pre-made solution.txt is made on test/solutions/solution.txt

def rewrite_Txt(rewards, optimal_tokens, optimal_path, time_execution, message)
: # write solution into txt

    def rewrite_rewards(file,rewards):
        file.write(str(rewards) + '\n')

    def rewrite_tokens(file,optimal_tokens):
        for i, token in enumerate(optimal_tokens):
            file.write(token)
            if(i != (len(optimal_tokens) - 1)):
                file.write(' ')
        file.write('\n')

    def rewrite_path(file, optimal_path):
        for path in optimal_path:
            file.write(str(path)[1:-1] + '\n')
        file.write('\n')

    def rewrite_time(file, time_execution):
        file.write(str(int(time_execution * 1000)))
        file.write(' ms\n')


    filepath = input(message)
    try:
        with open(filepath, 'w') as file:
            rewrite_rewards(file,rewards)
            rewrite_tokens(file,optimal_tokens)
            rewrite_path(file,optimal_path)
            rewrite_time(file,time_execution)
        print("File successfully written")
    except FileNotFoundError:
        print("File Not Found")
        rewrite_Txt(rewards,optimal_tokens, optimal_path, time_execution,
        "Please Input Ulang path anda: ")
```



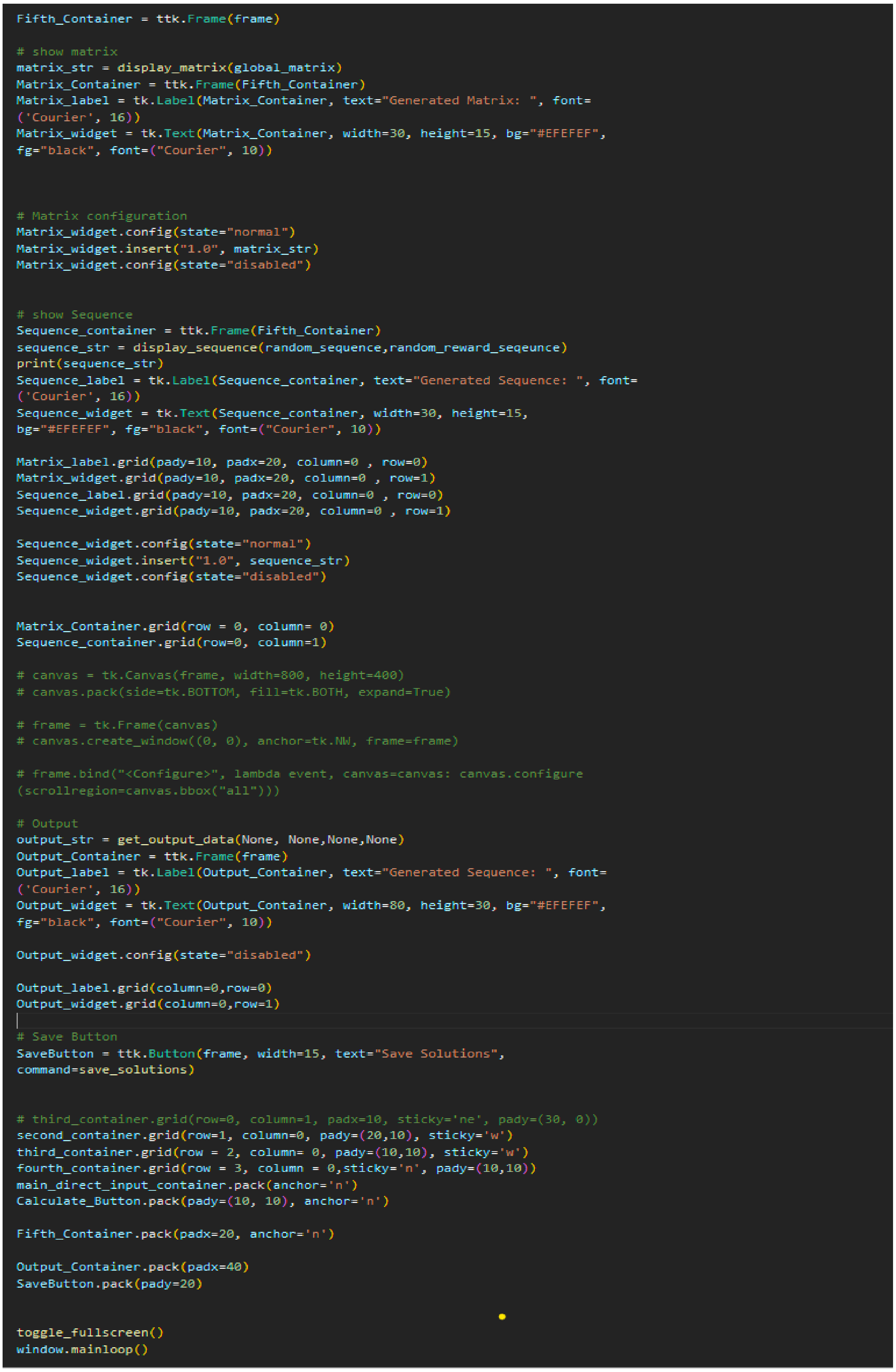
```python
def save_solutions():
    if (not rewards and not tokens and not paths and not times):
        return

    file_path = filedialog.asksaveasfilename(defaultextension='.txt', filetypes=[("Text files", "*.txt")])

    if not file_path:
        return

    with open(file_path, 'w') as file:
        rewrite_rewards(file,rewards)
        rewrite_tokens(file,tokens)
        rewrite_path(file,paths)
        rewrite_time(file,times)
    file.close()
    raise_good_job("File save Successfully")

def rewrite_rewards(file,rewards):
        file.write(str(rewards) + '\n')

def rewrite_tokens(file,optimal_tokens):
    for i, token in enumerate(optimal_tokens):
        file.write(token)
        if(i != (len(optimal_tokens) - 1)):
            file.write(' ')
    file.write('\n')

def rewrite_path(file, optimal_path):
    for path in optimal_path:
        file.write(str(path)[1:-1] + '\n')
    file.write('\n')

def rewrite_time(file, time_execution):
    file.write(str(int(time_execution * 1000)))
    file.write(' ms\n')

color_data = {
    'Cream_1' : '#D9D9D9'
}


# Canvas
canvas = tk.Canvas(window)
canvas.pack(side="left", fill="both", expand=True)

# Scrollbar
scrollbar = ttk.Scrollbar(window, orient="vertical", command=canvas.yview)
canvas.configure(yscrollcommand=scrollbar.set)
scrollbar.place(relx = 1, rely = 0, relheight= 1, anchor='ne')

# Frame
frame = ttk.Frame(canvas)

canvas.create_window((0, 0), window=frame, anchor="nw")

frame.bind("<Configure>", on_configure)

# mouse scroll down and up
canvas.bind('<MouseWheel>', lambda event: canvas.yview_scroll(-int(event.delta / 60), "units"))

# scroll bar horizontal
scrollbar_bottom = ttk.Scrollbar(window, orient='horizontal', command=canvas.xview)
canvas.configure(xscrollcommand= scrollbar_bottom.set)
scrollbar_bottom.place(relx = 0, rely = 1, relwidth=  1, anchor= 'sw')

# ctrl + mouse scroll
canvas.bind('<Control MouseWheel>', lambda event: canvas.xview_scroll(-int(event.delta / 60), "units"))


# full screen button
fullscreen_button = tk.Button(frame, text="Fullscreen",padx=5,pady=5, command=toggle_fullscreen)

fullscreen_button.place(relx=1.0, rely=0, anchor='ne')

window.bind("<F12>", toggle_fullscreen)
window.bind("<Escape>", toggle_fullscreen)
```



```
# title

title = ttk.Label(master=frame, text="Cyberpunk 2077 Breach Protocol",
font="Calibri 20", padding= (10,10,10,10))
title.pack()

# line after text

first_canvas = tk.Canvas(frame, width=window.winfo_screenwidth(), height= 30)
first_canvas.pack(fill=tk.Y, expand=False)
first_line = first_canvas.create_line(0,30,window.winfo_screenwidth(),30,
width=4)


# main container
main_direct_input_container = ttk.Frame(frame)


# SLIDER
slider_container = ttk.Frame(master=frame)
label_color = tk.Label(slider_container, text="Txt Input", font="Calibri 16",
width=10)

slider = tk.Scale(slider_container, from_=0, to=1, orient="horizontal",
length=70, sliderlength=20, showvalue=False, width=20)

label_texture = tk.Label(slider_container, text="Direct Input", font="Calibri
16", width=10)



# PACKERS
label_color.grid(row=0, column=0)
slider.grid(row=0, column=1)
label_texture.grid(row=0, column=2)

slider_container.pack(pady=(10, 20))
slider.bind("<ButtonRelease-1>", lambda event: update_selection())

slider.set(0)
update_selection()


# Second Container
second_container = ttk.Frame(main_direct_input_container)


#global variable
global_matrix = None
random_sequence = None
buffer_size = None
random_reward_seqeunce = None
rewards, tokens, paths, times = None, None, None, None
isDirectInput = False


# validator
validate_number = window.register(validate_input)

# Input TXT

input_container = ttk.Frame(second_container)
filename = tk.StringVar(value="No File Chosen")

input_text = ttk.Label(input_container, text="Input File")
file_chooser = tk.Button(input_container, text="Input txt", command=lambda:
open_file(), width=15, height=2)

file_path_text = ttk.Label(input_container, text="No File Choosen",
textvariable=filename)

input_text.grid(row=0, column=0, sticky='w', pady=(0, 1))
file_chooser.grid(row=1, column=0, sticky='w')
file_path_text.grid(row=2, column=0, sticky='w', columnspan=2)

input_container.grid(row=0,column=1, padx=40)
```



```
# Buffer_Input
buffer_background = tk.Label(second_container,bg="#7C7C7C", height=15)
buffer_background.grid(row = 0, column = 0, sticky='w')
buffer_text = tk.Label(buffer_background, text="Entry Buffer: ",bg="#7E7E7E",
font="Calibri 12", width=25, foreground='white', anchor='w')
buffer_text.grid(row=0, column=0, sticky='w')
buffer_entry = tk.Entry(buffer_background, width=7, font="Calibri 12",
validate="key", validatecommand=(validate_number, '%d', '%P'))
buffer_entry.grid(pady=5, padx=5, row=1, column=0, sticky='w')


# Unique Token
Unique_token_background = tk.Label(second_container,bg="#7C7C7C", height=15)
Unique_token_background.grid(row = 1, column = 0, sticky='w', pady=(20,20))
Unique_token_text = tk.Label(Unique_token_background, text="Entry Unique token:
",bg="#7E7E7E", font="Calibri 12", width=25, foreground='white', anchor='w')
Unique_token_text.grid(row=0, column=0, sticky='w')
Unique_token_entry = tk.Entry(Unique_token_background, width=25, font="Calibri
12")
Unique_token_entry.grid(pady=5, padx=5, row=1, column=0, sticky='w')


# third container
third_container = ttk.Frame(main_direct_input_container)

# Max Unique Sequence Token
max_sequence_token_background = tk.Label(third_container,bg="#7C7C7C",
height=15)
max_sequence_token_background.grid(row = 0, column = 0, sticky='w', pady=(10,
20))
max_sequence_token_text = tk.Label(max_sequence_token_background, text="Entry
Max Token in Sequences: ",bg="#7E7E7E", font="Calibri 12", width=25,
foreground='white', anchor='w')
max_sequence_token_text.grid(row=0, column=0, sticky='w')
max_sequence_token_entry = tk.Entry(max_sequence_token_background, width=7,
font="Calibri 12", validate='key', validatecommand=(validate_number,'%d','%P'))
max_sequence_token_entry.grid(pady=5, padx=5, row=1, column=0, sticky='w')


# Max Sequence
Max_Sequence_background = tk.Label(third_container,bg="#7C7C7C", height=15)
Max_Sequence_background.grid(row = 0, column = 1, sticky='w', padx=20, pady=(10,
20))
Max_Sequence_text = tk.Label(Max_Sequence_background, text="Entry Max Amount Of
Sequence: ",bg="#7E7E7E", font="Calibri 12", width=25, foreground='white',
anchor='w')
Max_Sequence_text.grid(row=0, column=0, sticky='w')
Max_Sequence_entry = tk.Entry(Max_Sequence_background, width=7, font="Calibri
12", validate='key', validatecommand=(validate_number,'%d','%P'))
Max_Sequence_entry.grid(pady=5, padx=5, row=1, column=0, sticky='w')


fourth_container = tk.Frame(main_direct_input_container)
# Matrix Col
Matrix_Col_background = tk.Label(fourth_container,bg="#7C7C7C", height=15)
Matrix_Col_background.grid(row = 0, column = 0, sticky='w', pady=(10,20))
Matrix_Col_text = tk.Label(Matrix_Col_background, text="Matrix Col: ",
bg="#7E7E7E", font="Calibri 12", width=25, foreground='white', anchor='w')
Matrix_Col_text.grid(row=0, column=0, sticky='w')
Matrix_Col_entry = tk.Entry(Matrix_Col_background, width=7, font="Calibri 12",
validate='key', validatecommand=(validate_number,'%d','%P'))
Matrix_Col_entry.grid(pady=5, padx=5, row=1, column=0, sticky='w')

# Matrix Row

Matrix_Row_background = tk.Label(fourth_container,bg="#7C7C7C", height=15)
Matrix_Row_background.grid(row = 0, column = 1, sticky='w', pady=(10,20),
padx=20)
Matrix_Row_text = tk.Label(Matrix_Row_background, text="Matrix Row: ",
bg="#7E7E7E", font="Calibri 12", width=25, foreground='white', anchor='w')
Matrix_Row_text.grid(row=0, column=0, sticky='w')
Matrix_Row_entry = tk.Entry(Matrix_Row_background, width=7, font="Calibri 12",
validate='key', validatecommand=(validate_number,'%d','%P'))
Matrix_Row_entry.grid(pady=5, padx=5, row=1, column=0, sticky='w')

# Calculate Button
Calculate_Button = tk.Button(frame, height=5, width=20, command=lambda:
calculation(slider), text="Calculate Button")
```



```
Fifth_Container = ttk.Frame(frame)

# show matrix
matrix_str = display_matrix(global_matrix)
Matrix_Container = ttk.Frame(Fifth_Container)
Matrix_label = tk.Label(Matrix_Container, text="Generated Matrix: ", font=
('Courier', 16))
Matrix_widget = tk.Text(Matrix_Container, width=30, height=15, bg="#EFEFEF",
fg="black", font=("Courier", 10))



# Matrix configuration
Matrix_widget.config(state="normal")
Matrix_widget.insert("1.0", matrix_str)
Matrix_widget.config(state="disabled")


# show Sequence
Sequence_container = ttk.Frame(Fifth_Container)
sequence_str = display_sequence(random_sequence,random_reward_seqeunce)
print(sequence_str)
Sequence_label = tk.Label(Sequence_container, text="Generated Sequence: ", font=
('Courier', 16))
Sequence_widget = tk.Text(Sequence_container, width=30, height=15,
bg="#EFEFEF", fg="black", font=("Courier", 10))

Matrix_label.grid(pady=10, padx=20, column=0 , row=0)
Matrix_widget.grid(pady=10, padx=20, column=0 , row=1)
Sequence_label.grid(pady=10, padx=20, column=0 , row=0)
Sequence_widget.grid(pady=10, padx=20, column=0 , row=1)

Sequence_widget.config(state="normal")
Sequence_widget.insert("1.0", sequence_str)
Sequence_widget.config(state="disabled")


Matrix_Container.grid(row = 0, column= 0)
Sequence_container.grid(row=0, column=1)

# canvas = tk.Canvas(frame, width=800, height=400)
# canvas.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True)

# frame = tk.Frame(canvas)
# canvas.create_window((0, 0), anchor=tk.NW, frame=frame)

# frame.bind("<Configure>", lambda event, canvas=canvas: canvas.configure
(scrollregion=canvas.bbox("all")))

# Output
output_str = get_output_data(None, None,None,None)
Output_Container = ttk.Frame(frame)
Output_label = tk.Label(Output_Container, text="Generated Sequence: ", font=
('Courier', 16))
Output_widget = tk.Text(Output_Container, width=80, height=30, bg="#EFEFEF",
fg="black", font=("Courier", 10))

Output_widget.config(state="disabled")

Output_label.grid(column=0,row=0)
Output_widget.grid(column=0,row=1)

# Save Button
SaveButton = ttk.Button(frame, width=15, text="Save Solutions",
command=save_solutions)


# third_container.grid(row=0, column=1, padx=10, sticky='ne', pady=(30, 0))
second_container.grid(row=1, column=0, pady=(20,10), sticky='w')
third_container.grid(row = 2, column= 0, pady=(10,10), sticky='w')
fourth_container.grid(row = 3, column = 0,sticky='n', pady=(10,10))
main_direct_input_container.pack(anchor='n')
Calculate_Button.pack(pady=(10, 10), anchor='n')

Fifth_Container.pack(padx=20, anchor='n')

Output_Container.pack(padx=40)
SaveButton.pack(pady=20)


toggle_fullscreen()
window.mainloop()
```

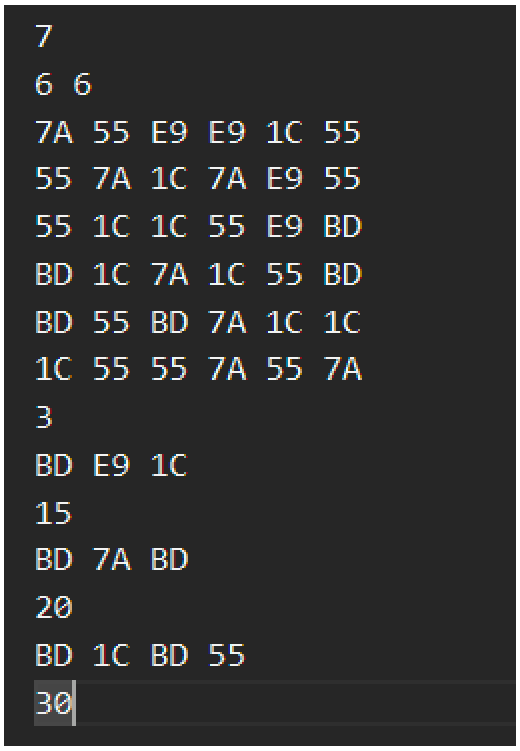
BAB III

PROGRAM TESTCASE

Pada 3.1-3.3, akan terdapat 1 input dengan 2 output sedangkan pada sisanya akan memiliki 2 input dengan 2 output.

3.1 Input From TextFile I

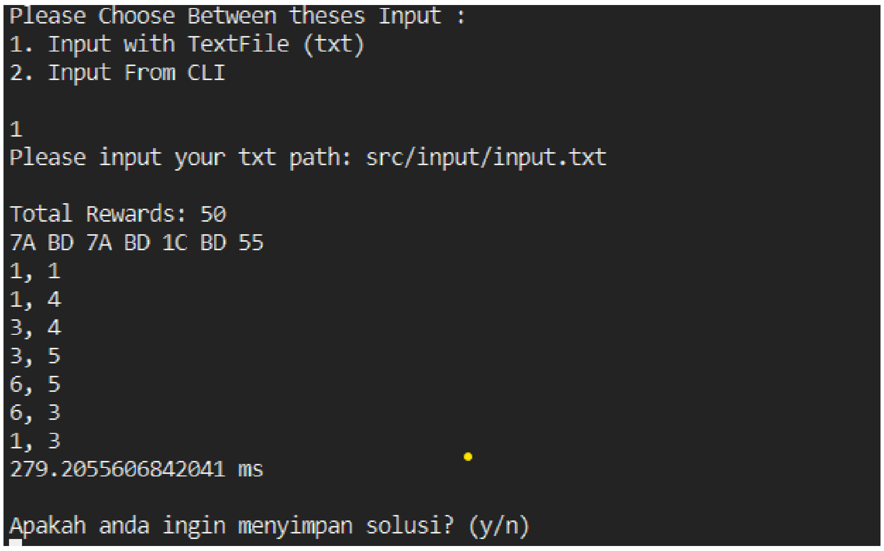
input :



```
7
6 6
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
3
BD E9 1C
15
BD 7A BD
20
BD 1C BD 55
30
```

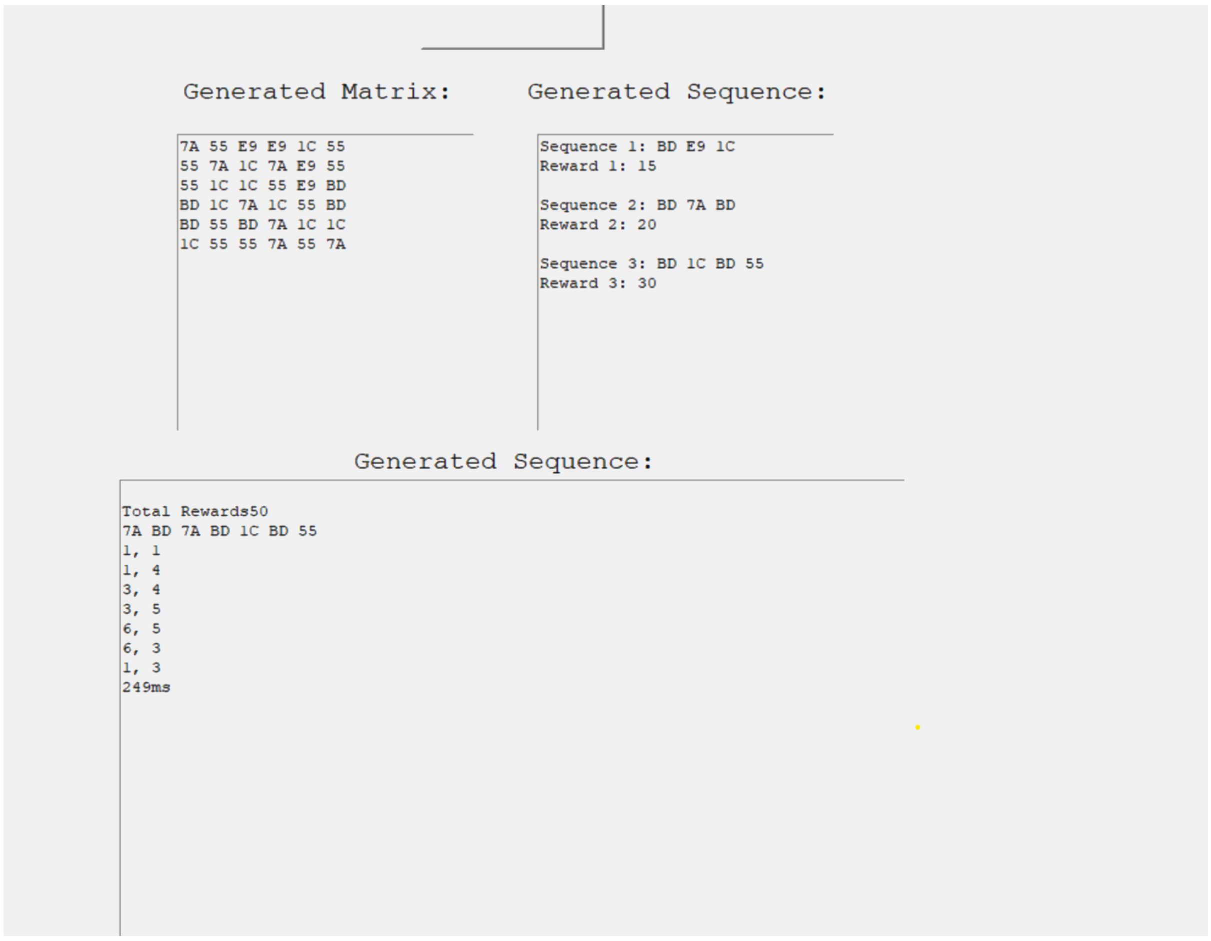
output :

1. CLI



```
Please Choose Between theses Input :
1. Input with TextFile (txt)
2. Input From CLI

1
Please input your txt path: src/input/input.txt

Total Rewards: 50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3
279.2055606842041 ms

Apakah anda ingin menyimpan solusi? (y/n)
```

2. GUI



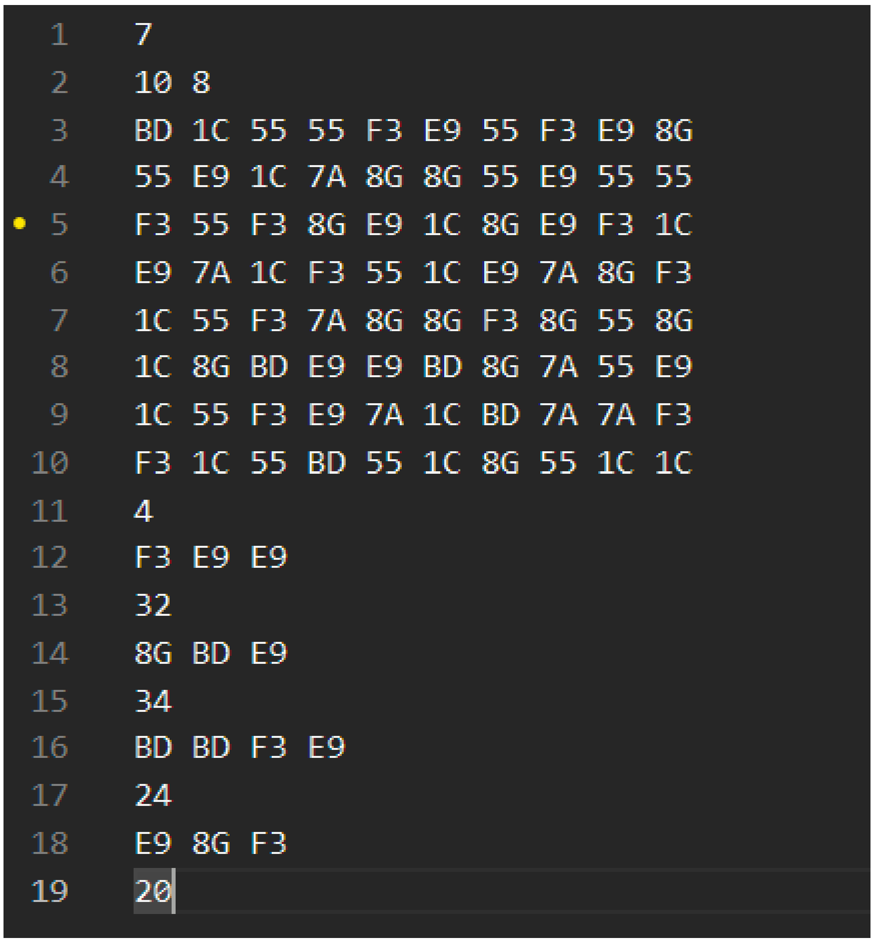
```
7A 55 E9 E9 1C 55
55 7A 1C 7A E9 55
55 1C 1C 55 E9 BD
BD 1C 7A 1C 55 BD
BD 55 BD 7A 1C 1C
1C 55 55 7A 55 7A
```

```
Sequence 1: BD E9 1C
Reward 1: 15

Sequence 2: BD 7A BD
Reward 2: 20

Sequence 3: BD 1C BD 55
Reward 3: 30
```

```
Total Rewards50
7A BD 7A BD 1C BD 55
1, 1
1, 4
3, 4
3, 5
6, 5
6, 3
1, 3
249ms
```

3.2 Input From TextFile II

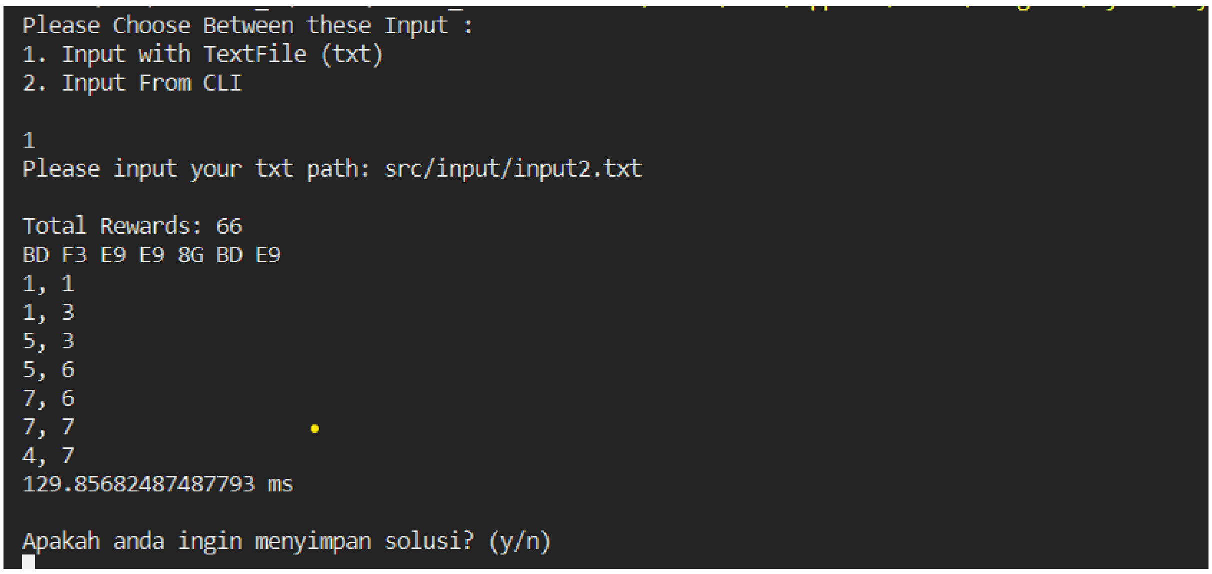
input :



```
7
10 8
BD 1C 55 55 F3 E9 55 F3 E9 8G
55 E9 1C 7A 8G 8G 55 E9 55 55
F3 55 F3 8G E9 1C 8G E9 F3 1C
E9 7A 1C F3 55 1C E9 7A 8G F3
1C 55 F3 7A 8G 8G F3 8G 55 8G
1C 8G BD E9 E9 BD 8G 7A 55 E9
1C 55 F3 E9 7A 1C BD 7A 7A F3
F3 1C 55 BD 55 1C 8G 55 1C 1C
4
F3 E9 E9
32
8G BD E9
34
BD BD F3 E9
24
E9 8G F3
20
```

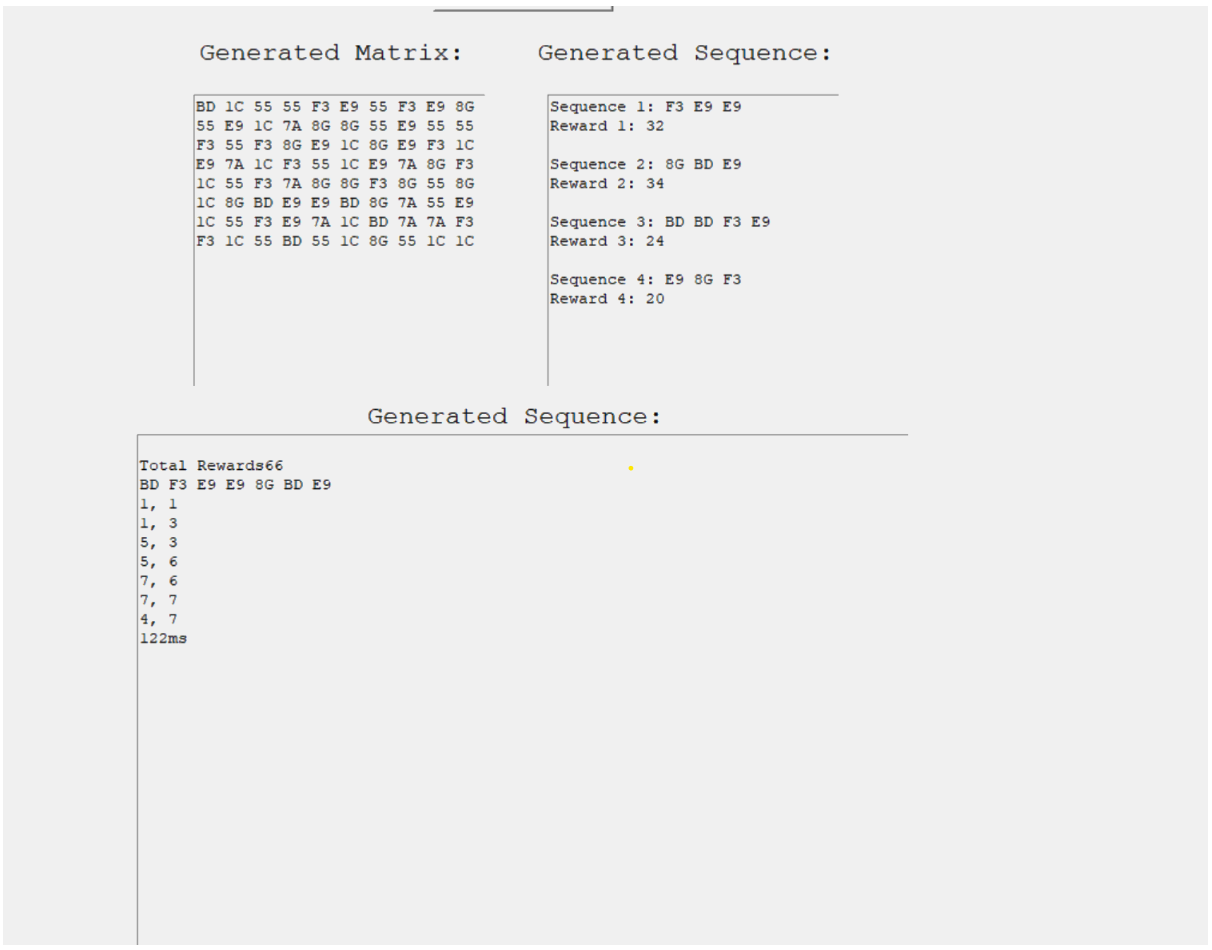
output:

1. CLI



```
Please Choose Between these Input :
1. Input with TextFile (txt)
2. Input From CLI

1
Please input your txt path: src/input/input2.txt

Total Rewards: 66
BD F3 E9 E9 8G BD E9
1, 1
1, 3
5, 3
5, 6
7, 6
7, 7
4, 7
129.85682487487793 ms

Apakah anda ingin menyimpan solusi? (y/n)
```

2. GUI



Generated Matrix:

```
BD 1C 55 55 F3 E9 55 F3 E9 8G
55 E9 1C 7A 8G 8G 55 E9 55 55
F3 55 F3 8G E9 1C 8G E9 F3 1C
E9 7A 1C F3 55 1C E9 7A 8G F3
1C 55 F3 7A 8G 8G F3 8G 55 8G
1C 8G BD E9 E9 BD 8G 7A 55 E9
1C 55 F3 E9 7A 1C BD 7A 7A F3
F3 1C 55 BD 55 1C 8G 55 1C 1C
```

Generated Sequence:

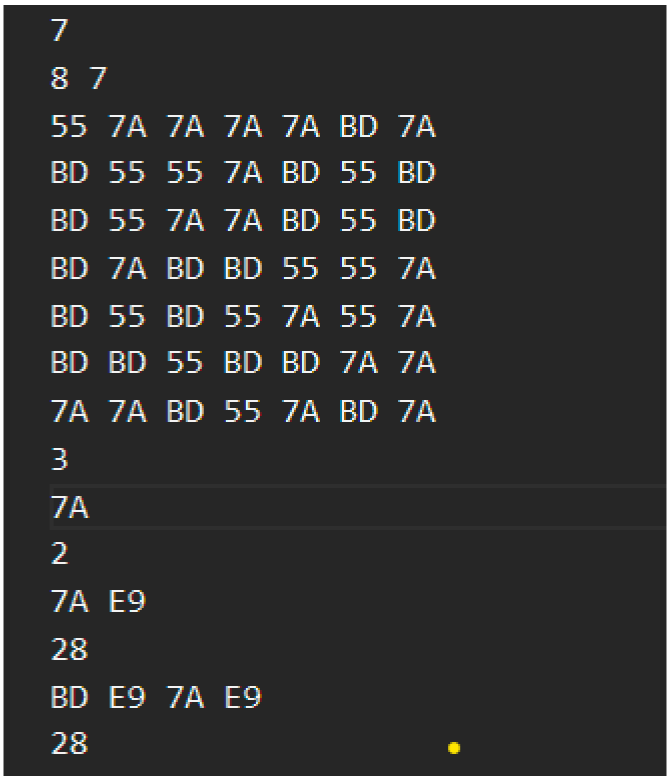
```
Sequence 1: F3 E9 E9
Reward 1: 32

Sequence 2: 8G BD E9
Reward 2: 34

Sequence 3: BD BD F3 E9
Reward 3: 24

Sequence 4: E9 8G F3
Reward 4: 20
```

Generated Sequence:

```
Total Rewards66
BD F3 E9 E9 8G BD E9
1, 1
1, 3
5, 3
5, 6
7, 6
7, 7
4, 7
122ms
```

3.3 Input From TextFile III

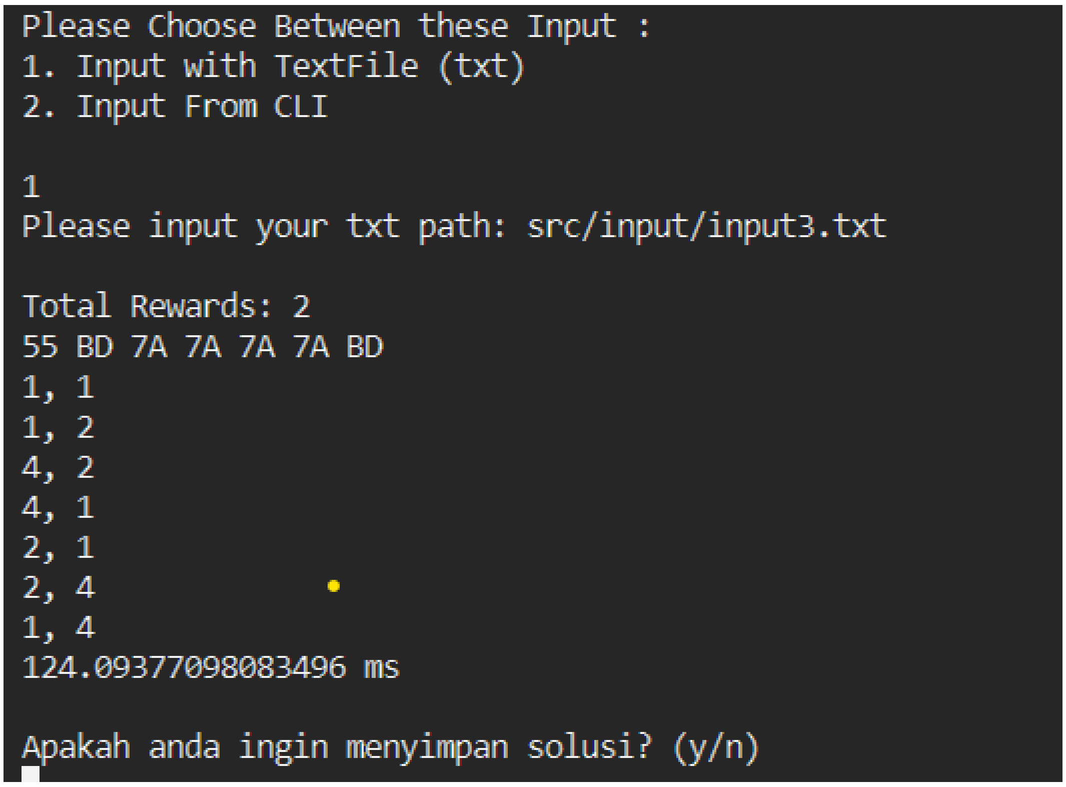
input :



```
7
8 7
55 7A 7A 7A 7A BD 7A
BD 55 55 7A BD 55 BD
BD 55 7A 7A BD 55 BD
BD 7A BD BD 55 55 7A
BD 55 BD 55 7A 55 7A
BD BD 55 BD BD 7A 7A
7A 7A BD 55 7A BD 7A
3
7A
2
7A E9
28
BD E9 7A E9
28
```

output :

1. CLI



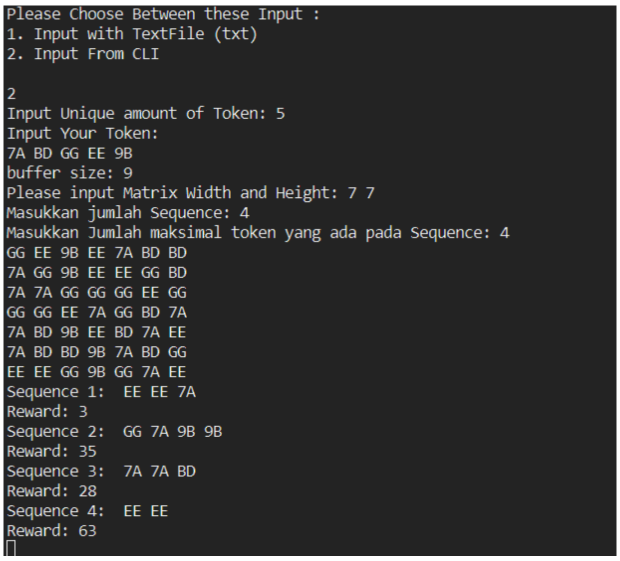
```
Please Choose Between these Input :
1. Input with TextFile (txt)
2. Input From CLI

1
Please input your txt path: src/input/input3.txt

Total Rewards: 2
55 BD 7A 7A 7A 7A BD
1, 1
1, 2
4, 2
4, 1
2, 1
2, 4
1, 4
124.09377098083496 ms

Apakah anda ingin menyimpan solusi? (y/n)
```

2. GUI

3.4 Input From Direct Input I

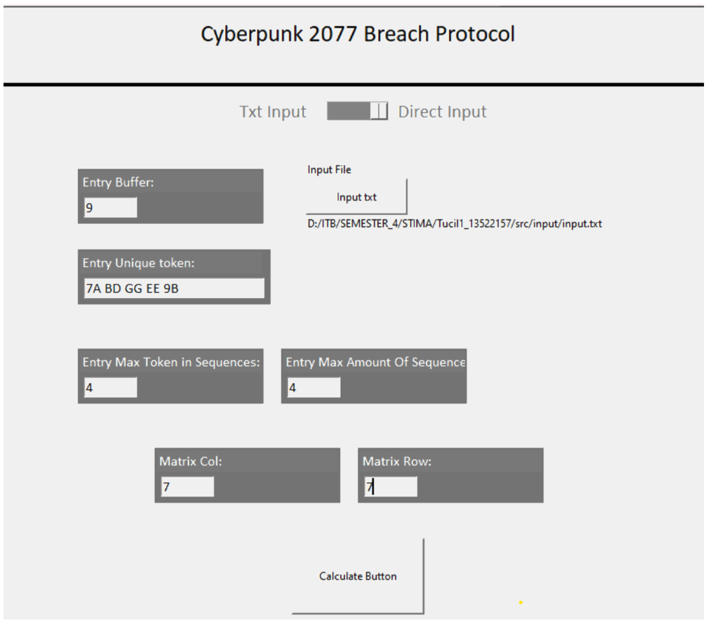
input :

1. CLI



```
Please Choose Between these Input :
1. Input with TextFile (txt)
2. Input From CLI

2
Input Unique amount of Token: 5
Input Your Token:
7A BD GG EE 9B
buffer size: 9
Please input Matrix Width and Height: 7 7
Masukkan jumlah Sequence: 4
Masukkan Jumlah maksimal token yang ada pada Sequence: 4
GG EE 9B EE 7A BD BD
7A GG 9B EE EE GG BD
7A 7A GG GG GG EE GG
GG GG EE 7A GG BD 7A
7A BD 9B EE BD 7A EE
7A BD BD 9B 7A BD GG
EE EE GG 9B GG 7A EE
Sequence 1:  EE EE 7A
Reward: 3
Sequence 2:  GG 7A 9B 9B
Reward: 35
Sequence 3:  7A 7A BD
Reward: 28
Sequence 4:  EE EE
Reward: 63
```

2. GUI



Cyberpunk 2077 Breach Protocol

D:/ITB/SEMESTER_4/STIMA/Tucil1_13522157/src/input/input.txt

Entry Buffer: 9

Entry Unique token: 7A BD GG EE 9B

Entry Max Token in Sequences: 4

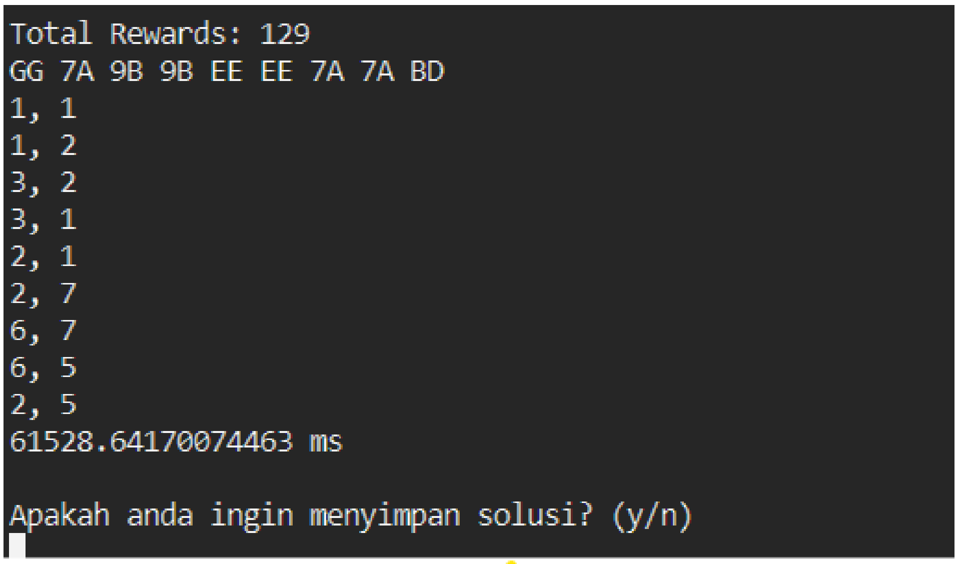
Entry Max Amount Of Sequence: 4

Matrix Col: 7

Matrix Row: 7

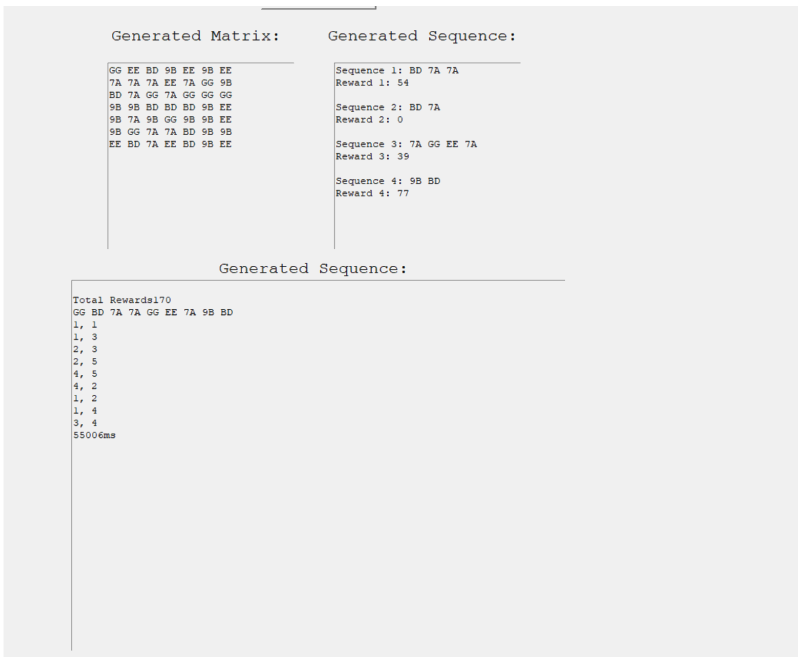
output:

1. CLI



```
Total Rewards: 129
GG 7A 9B 9B EE EE 7A 7A BD
1, 1
1, 2
3, 2
3, 1
2, 1
2, 7
6, 7
6, 5
2, 5
61528.64170074463 ms

Apakah anda ingin menyimpan solusi? (y/n)
```

2. GUI



```
Generated Matrix:

GG EE BD 9B EE 9B EE
7A 7A 7A EE 7A GG 9B
BD 7A GG 7A GG GG GG
9B 9B BD BD BD 9B EE
9B 7A 9B GG 9B 9B EE
9B GG 7A 7A BD 9B 9B
EE BD 7A EE BD 9B EE
```

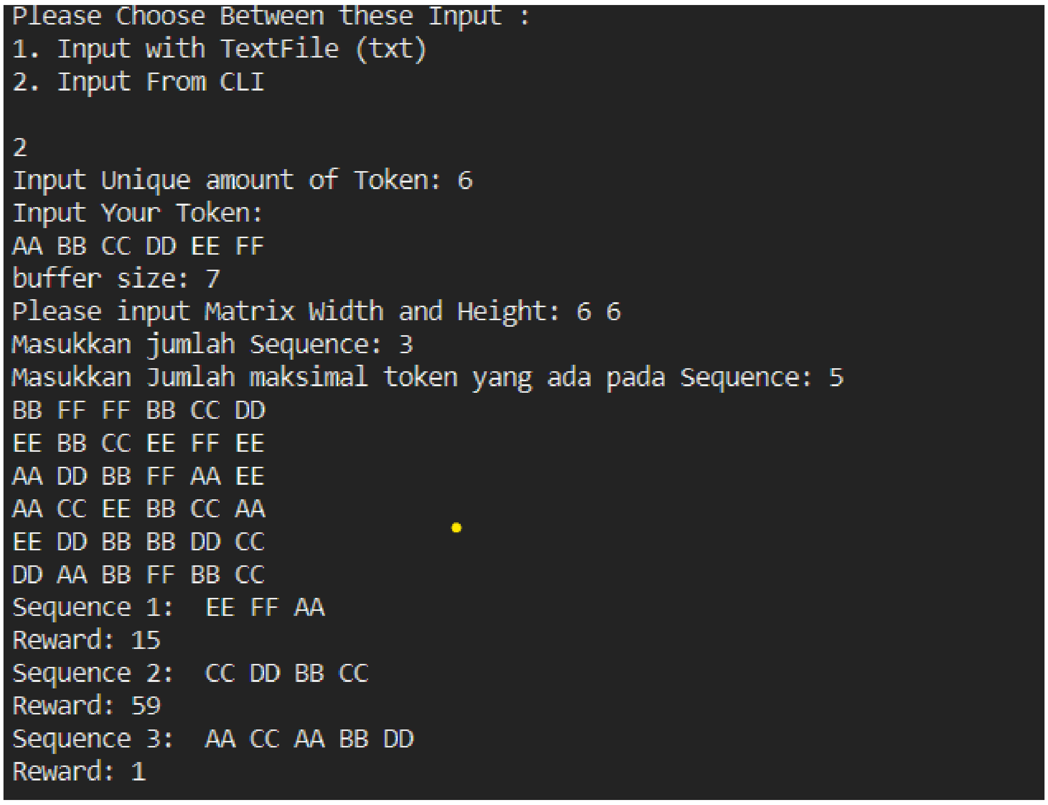
```
Generated Sequence:

Sequence 1: BD 7A 7A
Reward 1: 54

Sequence 2: BD 7A
Reward 2: 0

Sequence 3: 7A GG EE 7A
Reward 3: 39

Sequence 4: 9B BD
Reward 4: 77
```

```
Generated Sequence:

Total Rewards170
GG BD 7A 7A GG EE 7A 9B BD
1, 1
1, 3
2, 3
2, 5
4, 5
4, 2
1, 2
1, 4
3, 4
55006ms
```

3.5 Input From Direct Input II

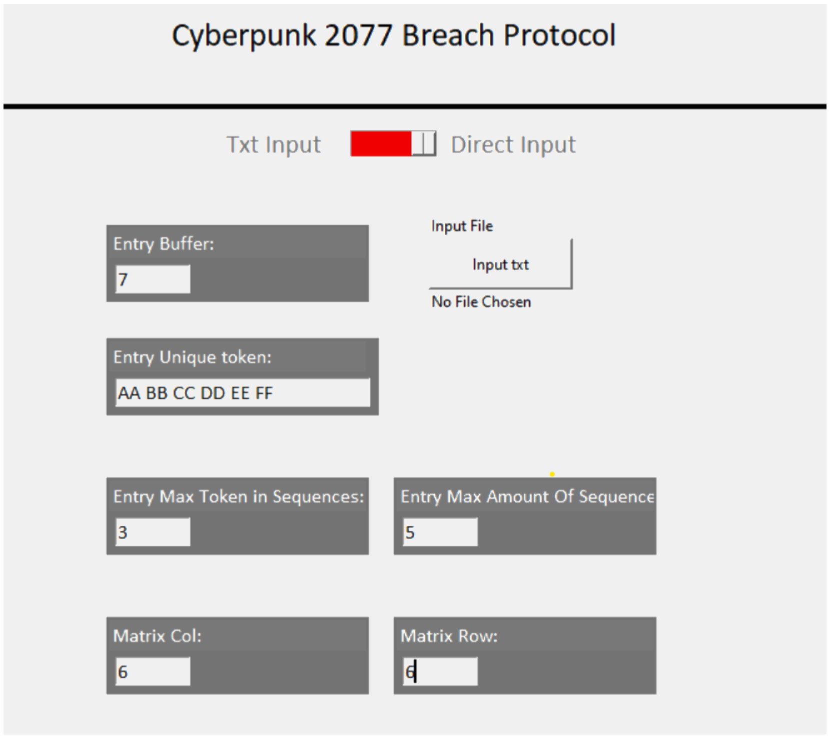
input :

1. CLI



```
Please Choose Between these Input :
1. Input with TextFile (txt)
2. Input From CLI

2
Input Unique amount of Token: 6
Input Your Token:
AA BB CC DD EE FF
buffer size: 7
Please input Matrix Width and Height: 6 6
Masukkan jumlah Sequence: 3
Masukkan Jumlah maksimal token yang ada pada Sequence: 5
BB FF FF BB CC DD
EE BB CC EE FF EE
AA DD BB FF AA EE
AA CC EE BB CC AA
EE DD BB BB DD CC
DD AA BB FF BB CC
Sequence 1:  EE FF AA
Reward: 15
Sequence 2:  CC DD BB CC
Reward: 59
Sequence 3:  AA CC AA BB DD
Reward: 1
```

2. GUI



Cyberpunk 2077 Breach Protocol

Entry Buffer: 7

Entry Unique token: AA BB CC DD EE FF

Entry Max Token in Sequences: 3

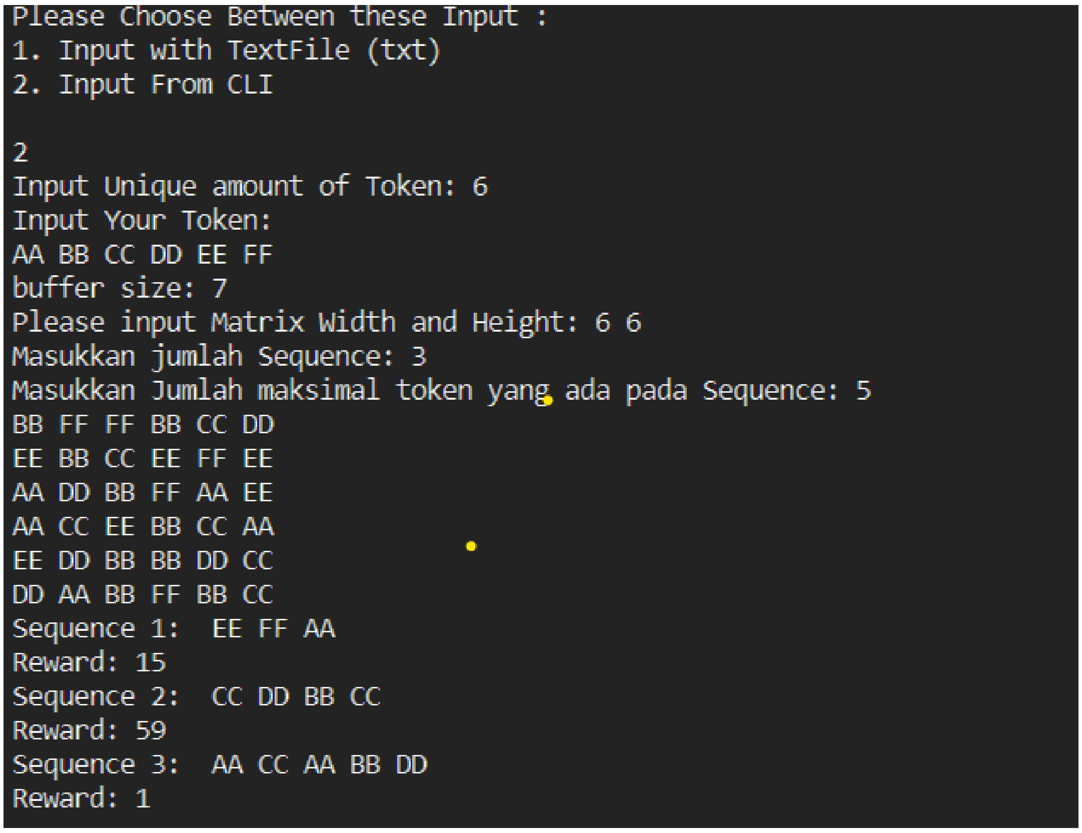
Entry Max Amount Of Sequence: 5

Matrix Col: 6

Matrix Row: 6

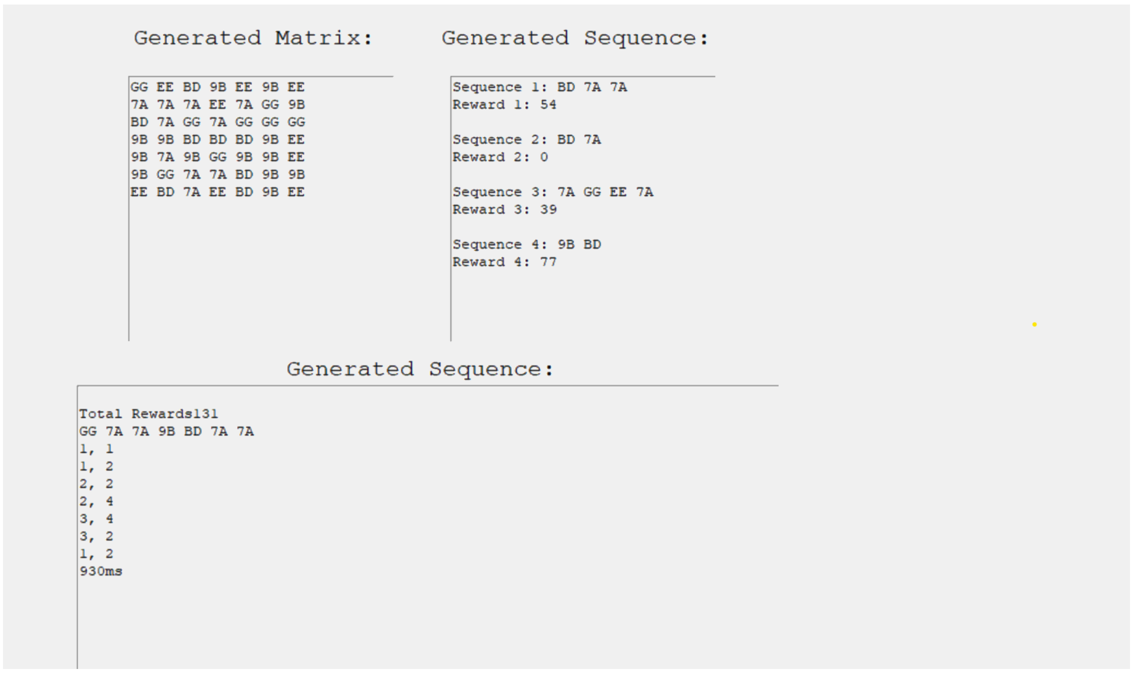
output:

1. CLI



```
Please Choose Between these Input :
1. Input with TextFile (txt)
2. Input From CLI

2
Input Unique amount of Token: 6
Input Your Token:
AA BB CC DD EE FF
buffer size: 7
Please input Matrix Width and Height: 6 6
Masukkan jumlah Sequence: 3
Masukkan Jumlah maksimal token yang ada pada Sequence: 5
BB FF FF BB CC DD
EE BB CC EE FF EE
AA DD BB FF AA EE
AA CC EE BB CC AA
EE DD BB BB DD CC
DD AA BB FF BB CC
Sequence 1:  EE FF AA
Reward: 15
Sequence 2:  CC DD BB CC
Reward: 59
Sequence 3:  AA CC AA BB DD
Reward: 1
```

2. GUI



```
Generated Matrix:

GG EE BD 9B EE 9B EE
7A 7A 7A EE 7A GG 9B
BD 7A GG 7A GG GG GG
9B 9B BD BD BD 9B EE
9B 7A 9B GG 9B 9B EE
9B GG 7A 7A BD 9B 9B
EE BD 7A EE BD 9B EE
```

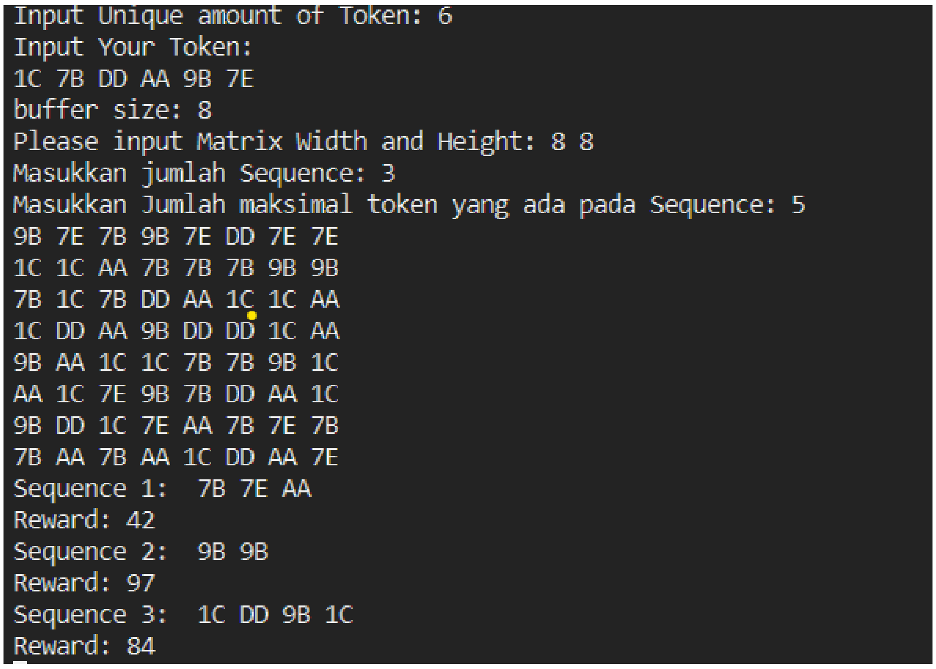
```
Generated Sequence:

Sequence 1: BD 7A 7A
Reward 1: 54

Sequence 2: BD 7A
Reward 2: 0

Sequence 3: 7A GG EE 7A
Reward 3: 39

Sequence 4: 9B BD
Reward 4: 77
```

```
Generated Sequence:

Total Rewards131
GG 7A 7A 9B BD 7A 7A
1, 1
1, 2
2, 2
2, 4
3, 4
3, 2
1, 2
930ms
```

3.6 Input From Direct Input III

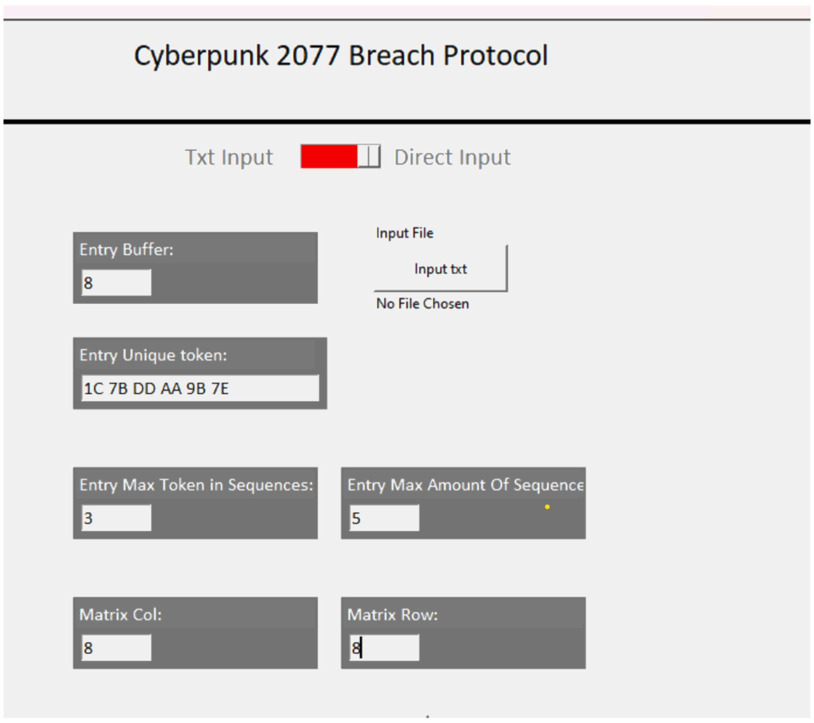
input:

1. CLI



```
Input Unique amount of Token: 6
Input Your Token:
1C 7B DD AA 9B 7E
buffer size: 8
Please input Matrix Width and Height: 8 8
Masukkan jumlah Sequence: 3
Masukkan Jumlah maksimal token yang ada pada Sequence: 5
9B 7E 7B 9B 7E DD 7E 7E
1C 1C AA 7B 7B 7B 9B 9B
7B 1C 7B DD AA 1C 1C AA
1C DD AA 9B DD DD 1C AA
9B AA 1C 1C 7B 7B 9B 1C
AA 1C 7E 9B 7B DD AA 1C
9B DD 1C 7E AA 7B 7E 7B
7B AA 7B AA 1C DD AA 7E
Sequence 1:  7B 7E AA
Reward: 42
Sequence 2:  9B 9B
Reward: 97
Sequence 3:  1C DD 9B 1C
Reward: 84
```

2. GUI



Cyberpunk 2077 Breach Protocol

Entry Buffer: 8

Entry Unique token: 1C 7B DD AA 9B 7E

Entry Max Token in Sequences: 3

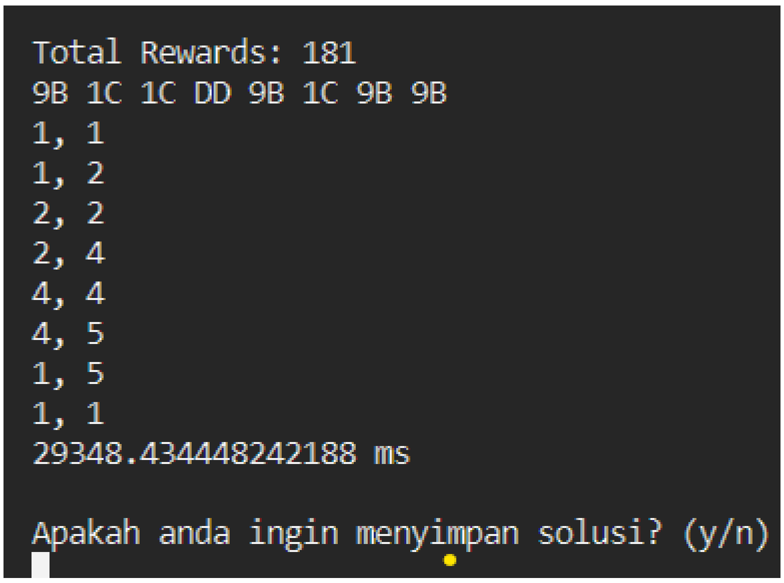
Entry Max Amount Of Sequence: 5

Matrix Col: 8

Matrix Row: 8

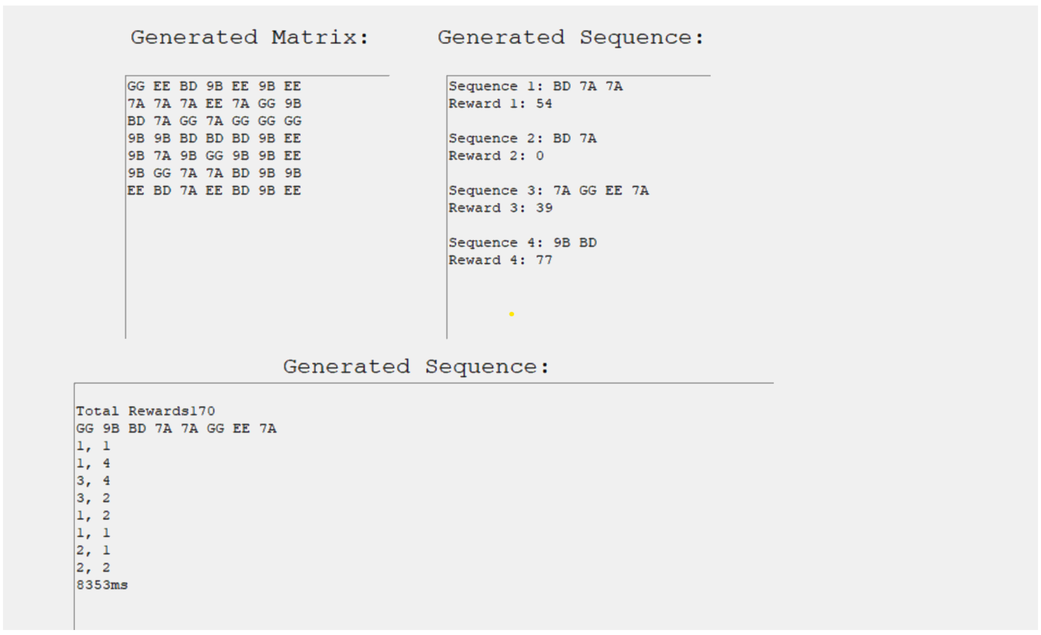
output :

1. CLI



```
Total Rewards: 181
9B 1C 1C DD 9B 1C 9B 9B
1, 1
1, 2
2, 2
2, 4
4, 4
4, 5
1, 5
1, 1
29348.434448242188 ms

Apakah anda ingin menyimpan solusi? (y/n)
```

2. GUI



```
Generated Matrix:

GG EE BD 9B EE 9B EE
7A 7A 7A EE 7A GG 9B
BD 7A GG 7A GG GG GG
9B 9B BD BD BD 9B EE
9B 7A 9B GG 9B 9B EE
9B GG 7A 7A BD 9B 9B
EE BD 7A EE BD 9B EE
```

```
Generated Sequence:

Sequence 1: BD 7A 7A
Reward 1: 54

Sequence 2: BD 7A
Reward 2: 0

Sequence 3: 7A GG EE 7A
Reward 3: 39

Sequence 4: 9B BD
Reward 4: 77
```

```
Generated Sequence:

Total Rewards170
GG 9B BD 7A 7A GG EE 7A
1, 1
1, 4
3, 4
3, 2
1, 2
1, 1
2, 1
2, 2
8353ms
```

BAB IV

REPOSITORY

Repository Link :

Poin	**Ya**	**Tidak**
1. Program berhasil dikompilasi tanpa kesalahan	✓	
2. Program berhasil dijalankan	✓	
3. Program dapat membaca masukan berkas .txt	✓	
4. Program dapat menghasilkan masukan secara acak	✓	
5. Solusi yang diberikan program optimal	✓	
6. Program dapat menyimpan solusi dalam berkas .txt	✓	
7. Program memiliki GUI	✓	