SS24 HIT137

SOFTWARE NOW

Assignment II

Group Name SYD 02

Student Name	Student ID
Anuj Ojha	S385270
Rahul Sharma	S383483
Kishor Katuwal	S383928
Nirmal Subedi	S386743

Table of Contents

Introduction: 1

Requirements 2

Design and Implementation 2

- **Question 1: Text File Encryption and Decryption** 2
- **Question 2: Temperature Data Analysis** 3
- **Question 3: Recursive Tree Pattern with Turtle Graphics** 4

Testing 5

- **Question 1:** 5
- **Question 2:** 5
- **Question 3:** 6

Codes and Outputs: 7

Question 1: 7

Code 7

Output 8

Question 2: 9

Code: 9

Output: 11

Question 3: 12

Output: 13

Conclusion: 14

Group Discussion: 15

Introduction:

This document provides an in-depth overview of three programming tasks that aim to demonstrate the practical applications of Python programming in real-world scenarios. The tasks address diverse problem-solving objectives, including file encryption and decryption, temperature data analysis, and recursive graphics generation. By tackling these challenges, the assignment highlights Python's versatility, modularity, and capacity for both algorithmic and creative tasks.

The **first task** focuses on implementing an encryption and decryption system for text files. This task simulates secure data handling by leveraging a character-shifting algorithm based on specific rules. It demonstrates Python's capabilities in string manipulation, file handling, and modular function design, providing a robust method for encrypting and decrypting textual data.

The **second task** delves into the analysis of temperature data from multiple weather stations. By processing CSV files, the program calculates average temperatures for each month, showcasing Python's ability to handle structured data efficiently. This task emphasizes data aggregation, statistical computation, and file management, which are critical in data analysis workflows.

The **third task** explores recursion and graphical programming by generating a tree pattern using Python's Turtle graphics module. This creative application of recursion demonstrates Python's ability to create visually appealing patterns while reinforcing concepts like parameterized functions, depth control, and modularity in graphical programming.

Each solution is designed to be clear, reusable, and extendable, ensuring adaptability to broader use cases. Together, these tasks illustrate the powerful combination of Python's ease of use and its applicability across various domains, from security to data science and graphical applications.

Requirements

The tasks are divided as follows:

1. Encrypt and decrypt a text file using a simple encryption algorithm with rules based on user input parameters.
2. Analyse temperature data collected from multiple weather stations and calculated statistical outputs.
3. Generate a tree pattern using Python's turtle graphics, with parameters specified by the user.

Design and Implementation

Question 1: Text File Encryption and Decryption

Objective:

To implement an encryption and decryption system that processes a text file based on specific character manipulation rules.

Functional Requirements:

1. Read the input file raw_text.txt.
2. Encrypt the content based on:
    - **Lowercase letters:**
        - Shift forward by n * m if in a-m.
        - Shift backward by n + m if in n-z.
    - **Uppercase letters:**
        - Shift backward by n if in A-M.
        - Shift forward by m^2 if in N-Z.
    - **Special characters and numbers** remain unchanged.
3. Write the encrypted content to encrypted_text.txt.
4. Decrypt the content back to its original form.
5. Verify the decryption correctness.

Implementation:

- **Functions**:
    1. encrypt_text(input_file, output_file, n, m):
        - Encrypts the content and writes to the output file.
    2. decrypt_text(input_file, output_file, n, m):
        - Decrypts the encrypted file content.
    3. verify_decryption(original_file, decrypted_file):
        - Compares the decrypted file with the original to confirm accuracy.

Question 2: Temperature Data Analysis

Objective:

To process temperature data from multiple CSV files and compute average monthly temperatures.

Functional Requirements:

1. Read temperature data stored in multiple CSV files within a temperatures folder.
2. Calculate the average temperature for each month across all stations.
3. Save the output to average_temp.txt.

Optional Future Extensions:

1. Calculate seasonal averages.
2. Identify the station with the largest temperature range.
3. Find the warmest and coolest stations.

Implementation:

- **Functions**:
    1. calculate_monthly_averages(data_folder, output_file):
        - Computes monthly averages and saves results.

2. **(Optional)**:
    - calculate_seasonal_averages,find_largest_temp_range, find_extreme_stations.

Question 3: Recursive Tree Pattern with Turtle Graphics

Objective:

To generate a visually appealing tree pattern using recursion in Python's turtle graphics module.

Functional Requirements:

1. Take user inputs for:
    - Left and right branch angles.
    - Starting branch length.
    - Recursion depth.
    - Branch length reduction factor.
2. Recursively draw the tree pattern.

Implementation:

- **Function**:

1. draw_tree(t, branch_length, left_angle, right_angle, depth, reduction_factor):

    - Parameters:
        - t: Turtle object.
        - branch_length: Length of the current branch.
        - left_angle and right_angle: Branch angles.
        - depth: Recursion depth.
        - reduction_factor: Factor by which branch length is reduced.

Testing

Question 1:

Testing for text encryption and decryption involves validating the correctness of the implementation under a variety of scenarios. The following test cases were conducted:

1. **Basic Functionality:**
    - Input: A sample raw_text.txt file containing uppercase and lowercase letters, numbers, and special characters.
    - Method: Encrypt the file using different values of n and m. Decrypt the encrypted file.
    - Validation: Compare the decrypted file with the original to ensure correctness.
2. **Edge Cases:**
    - Empty file: Verify that the program gracefully handles an empty file without errors.
    - Special characters-only file: Ensure no modifications occur during encryption or decryption.
    - Extreme n and m values: Validate behaviour with very large or small inputs for n and m.
3. **Stress Testing:**
    - Large input files with extensive content were tested to ensure efficiency and correctness.

Question 2:

The testing for temperature data analysis involved verifying the calculated averages and the program's robustness with the following scenarios:

1. **Sample Data Verification:**
    - Input: Create small sample CSV files with known temperature data for each month.
    - Method: Manually compute the monthly averages and compare with the program output.

2. **Data Integrity Tests:**
    - Missing data: Include files with incomplete temperature entries and verify proper handling.
    - Improper file formatting: Test files with incorrect formatting to ensure error handling.
3. **Performance Testing:**
    - Large Datasets: Provide large sets of temperature data files and validate both performance and accuracy of the computed averages.
4. **Output Verification:**
    - Ensure that the average_temp.txt file is correctly formatted and contains accurate data for each month.

Question 3:

The recursive tree pattern generation was tested for functionality, aesthetics, and robustness:

1. **Parameter Testing:**
    - Input: Different values for left and right branch angles, starting branch length, recursion depth, and reduction factors.
    - Validation: Visually inspect the tree patterns for symmetry, proper scaling, and adherence to user inputs.
2. **Edge Cases:**
    - Zero recursion depth: Confirm that only the base branch is drawn.
    - Extreme angles: Test with very small or large angles to observe the visual impact.
    - Reduction factors close to 0 or 1: Validate the effect on branch lengths and overall tree size.
3. **Error Handling:**
    - Non-integer or invalid inputs: Test how the program responds to invalid user inputs for angles, depths, and reduction factors.
4. **Performance Testing:**
    - High recursion depth: Test the program's ability to handle deep recursion levels without crashes or excessive delays.

Codes and Outputs:

Question 1:

Code

```python
def encrypt_text(n, m):
    def shift_char(char):
        if 'a' <= char <= 'm':
            return chr(((ord(char) - ord('a') + (n * m)) % 13) + ord('a'))
        elif 'n' <= char <= 'z':
            return chr(((ord(char) - ord('n') - (n + m)) % 13) + ord('n'))
        elif 'A' <= char <= 'M':
            return chr(((ord(char) - ord('A') - n) % 13) + ord('A'))
        elif 'N' <= char <= 'Z':
            return chr(((ord(char) - ord('N') + (m ** 2)) % 13) + ord('N'))
        else:
            return char

    try:
        with open("raw_text.txt", "r") as file:
            raw_text1 = file.read()

        encrypted_text = "".join(shift_char(char) for char in raw_text1)

        with open("encrypted_text.txt", "w") as file:
            file.write(encrypted_text)
            print("Encrypted Text:", encrypted_text)

        print("Encryption complete. Encrypted text saved to 'encrypted_text.txt'.")
    except FileNotFoundError:
        print("File 'raw_text.txt' not found.")


def decrypt_text(n, m):
    def reverse_shift_char(char):
        if 'a' <= char <= 'm':
            return chr(((ord(char) - ord('a') - (n * m)) % 13) + ord('a'))
        elif 'n' <= char <= 'z':
            return chr(((ord(char) - ord('n') + (n + m)) % 13) + ord('n'))
        elif 'A' <= char <= 'M':
            return chr(((ord(char) - ord('A') + n) % 13) + ord('A'))
        elif 'N' <= char <= 'Z':
            return chr(((ord(char) - ord('N') - (m ** 2)) % 13) + ord('N'))
        else:
            return char

    try:
        with open("encrypted_text.txt", "r") as file:
            encrypted_text = file.read()


        decrypted_text = "".join(reverse_shift_char(char) for char in encrypted_text)
        print("Decryption Text: ", decrypted_text)
        print("Decryption complete.")
        return decrypted_text
    except FileNotFoundError:
        print("File 'encrypted_text.txt' not found.")
        return ""
```

```python
def verify_decryption(raw_text1, decrypted_text):
    if raw_text1 == decrypted_text:
        print("Decryption verified: The original and decrypted texts match.")
    else:
        print("Decryption failed: The original and decrypted texts do not match.")


# Example usage
if __name__ == "__main__":
    n = int(input("Enter value for n: "))
    m = int(input("Enter value for m: "))

    encrypt_text(n, m)

    try:
        with open("raw_text.txt", "r") as file:
            original_text = file.read()

        decrypted = decrypt_text(n, m)
        verify_decryption(original_text, decrypted)
    except FileNotFoundError:
        print("File 'raw_text.txt' not found. Cannot verify decryption.")

```

Output

```
tion1/Python-assignment-2/Question1/Q1.py
Enter value for n: 5
Enter value for m: 6
Encrypted Text: Qli swmgb ftqyp jqz awdru qxit vli ceon hqk fipievl vli ulehn ymccqyu. Qli hq
k, uvetvcih jtqd lmu riegijwc ejvitpqqp per, swmgbcn tmuiu eph gleuiu ejvit vli dmuglmixqwu j
qz. Qltqwkl xmftepv diehqyu eph reuv fwoompk fiilmxiu vlin tegi, hmuvwtfmpk e jcqgb qj swemcu
 vlev ugevvit mpvq vli gtmur ewvwdp ubn. Qli jqz, swmvi rcieuih ymvl lmu gcixit rtepb, heuliu
 mpvq lmu gqon wphitktqwph hip ylmci vli hqk, pqy izlewuvih jtqd vli oiecqwu rwtuwmv, tivwtpu
 vq lmu jexqtmvi urqv wphit vli ylmuritmpk ftepgliu vq tiuwdi lmu swmiv ucwdfit.
Encryption complete. Encrypted text saved to 'encrypted_text.txt'.
Decryption Text:  The quick brown fox jumps over the lazy dog beneath the shady willows. The
dog, startled from his peaceful afternoon nap, quickly rises and chases after the mischievous
 fox. Through vibrant meadows and past buzzing beehives they race, disturbing a flock of quai
ls that scatter into the crisp autumn sky. The fox, quite pleased with his clever prank, dash
es into his cozy underground den while the dog, now exhausted from the zealous pursuit, retur
ns to his favorite spot under the whispering branches to resume his quiet slumber.
Decryption complete.
Decryption verified: The original and decrypted texts match.
PS C:\Users\KishorKatuwal\Desktop\Question1\Python-assignment-2\Question1>
```

encrypted_text.txt

```
Qli swmgb ftqyp jqz awdru qxit vli ceon hqk fipievl vli ulehn ymccqyu. Qli hqk, uvetvcih jtqd lmu riegijwc ejvitpqqp per, swmgbcn tmuiu eph gleuiu ejvit vli dmuglmixqwu jqz. Qltqwkl xmftepv diehqyu eph reuv fwoompk fiilmxiu vlin tegi, hmuvwtfmpk e jcqgb qj swemcu vlev ugevvit mpvq vli gtmur ewvwdp ubn. Qli jqz, swmvi rcieuih ymvl lmu gcixit rtepb, heuliu mpvq lmu gqon wphitktqwph hip ylmci vli hqk, pqy izlewuvih jtqd vli oiecqwu rwtuwmv, tivwtpu vq lmu jexqtmvi urqv wphit vli ylmuritmpk ftepgliu vq tiuwdi lmu swmiv ucwdfit.
```

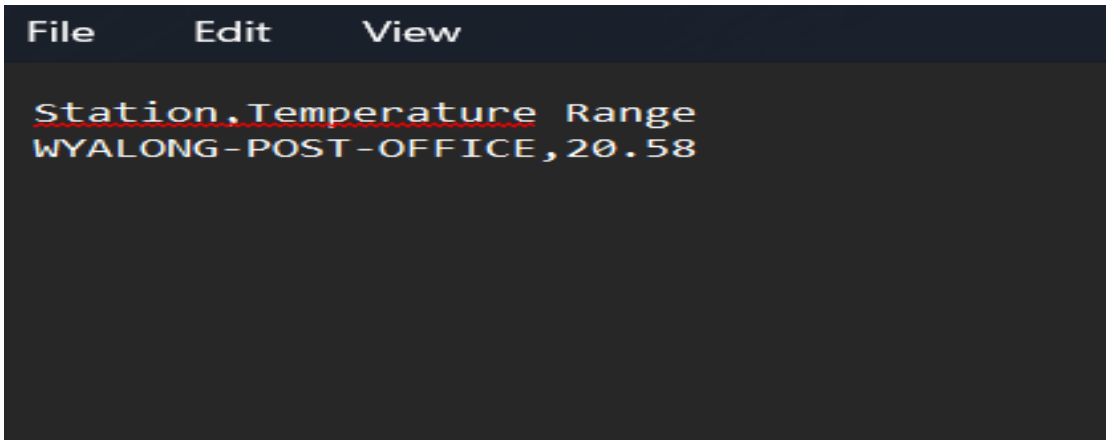
Question 2:

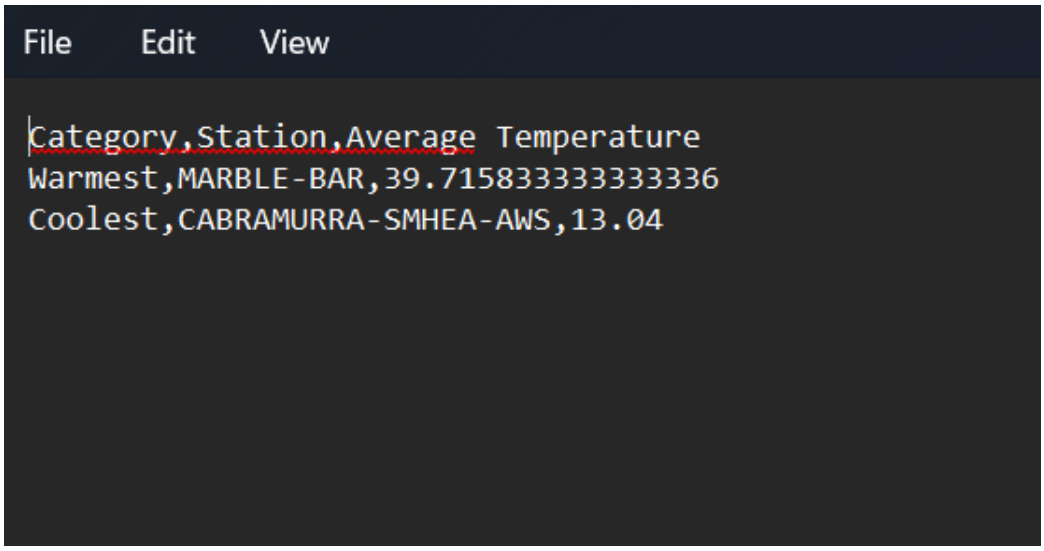
Code:

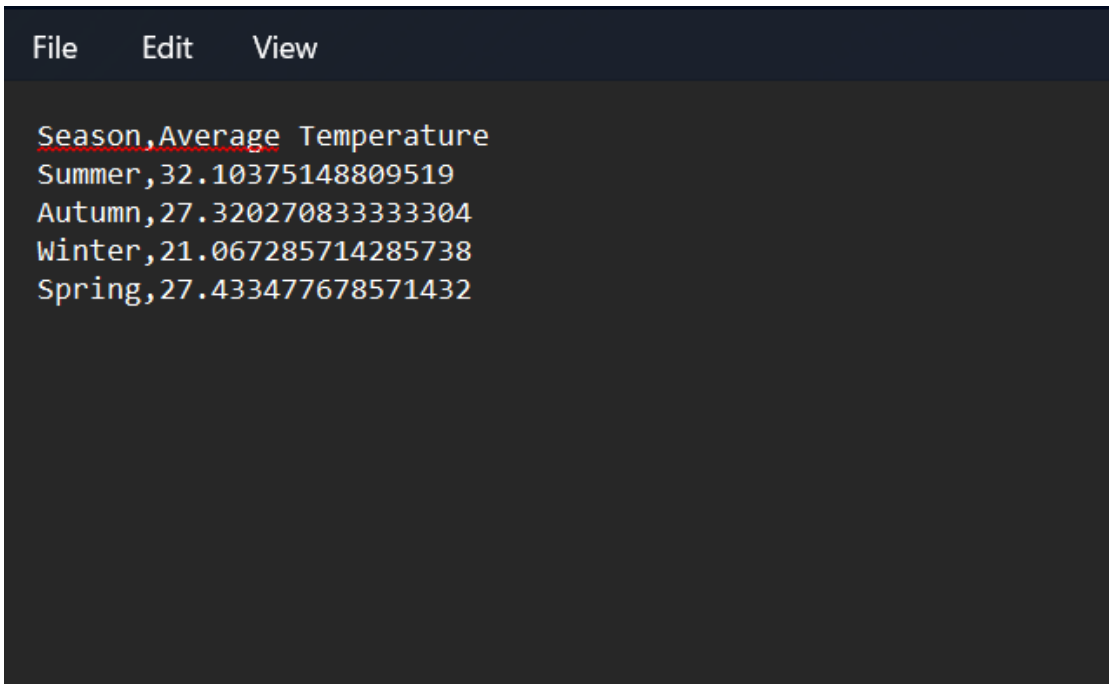
```python
import os
import pandas as pd

# Month names mapped to their respective numbers
month_names = ["January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]

def put_file):
    (variable) seasons: dict[str, list[int]]
    seasons = {
        "Summer": [12, 1, 2],
        "Autumn": [3, 4, 5],
        "Winter": [6, 7, 8],
        "Spring": [9, 10, 11]
    }
    seasonal_data = {season: [] for season in seasons}


    # Iterate through CSV files in the folder
    for file_name in os.listdir(temp_folder):
        if file_name.endswith(".csv"):
            file_path = os.path.join(temp_folder, file_name)
            data = pd.read_csv(file_path)

            # Ensure the CSV file contains the expected columns
            if all(month in data.columns for month in month_names):
                for season, months in seasons.items():
                    for month in months:
                        # Get all temperatures for the current month
                        month_name = month_names[month - 1]
                        monthly_temps = data[month_name]

                        # Add to the seasonal data
                        seasonal_data[season].extend(monthly_temps.dropna().tolist())

    # Calculate average for each season
    seasonal_averages = {}
    for season, temps in seasonal_data.items():
        if temps:
            seasonal_averages[season] = sum(temps) / len(temps)
        else:
            seasonal_averages[season] = None

    # Save the results to the output file
    with open(output_file, "w") as f:
        f.write("Season,Average Temperature\n")
        for season, avg_temp in seasonal_averages.items():
            f.write(f"{season},{avg_temp if avg_temp is not None else 'No Data'}\n")

def find_largest_temp_range(temp_folder, output_file):
    station_ranges = {}

    # Iterate through CSV files in the folder
    for file_name in os.listdir(temp_folder):
        if file_name.endswith(".csv"):
            file_path = os.path.join(temp_folder, file_name)
            data = pd.read_csv(file_path)

            # Ensure the CSV file contains the expected columns
            if "STATION_NAME" in data.columns and all(month in data.columns for month in month_names):
                for _, row in data.iterrows():
                    station_name = row["STATION_NAME"]
                    temperatures = row[month_names].dropna().values

                    if len(temperatures) > 0:
                        temp_range = max(temperatures) - min(temperatures)
                        station_ranges[station_name] = temp_range

    # Find the station(s) with the largest range
    max_range = max(station_ranges.values())
    largest_range_stations = [station for station, temp_range in station_ranges.items() if temp_range == max_range]

```

```
    # Save the results to the output file
    with open(output_file, "w") as f:
        f.write("Station,Temperature Range\n")
        for station in largest_range_stations:
            f.write(f"{station},{max_range}\n")

def find_warmest_and_coolest_stations(temp_folder, output_file):
    station_averages = {}

    # Iterate through CSV files in the folder
    for file_name in os.listdir(temp_folder):
        if file_name.endswith(".csv"):
            file_path = os.path.join(temp_folder, file_name)
            data = pd.read_csv(file_path)

            # Ensure the CSV file contains the expected columns
            if "STATION_NAME" in data.columns and all(month in data.columns for month in month_names):
                for _, row in data.iterrows():
                    station_name = row["STATION_NAME"]
                    temperatures = row[month_names].dropna().values

                    if len(temperatures) > 0:
                        avg_temp = sum(temperatures) / len(temperatures)
                        station_averages[station_name] = avg_temp

    # Find the warmest and coolest stations
    max_avg_temp = max(station_averages.values())
    min_avg_temp = min(station_averages.values())

    warmest_stations = [station for station, avg_temp in station_averages.items() if avg_temp == max_avg_temp]
    coolest_stations = [station for station, avg_temp in station_averages.items() if avg_temp == min_avg_temp]

    # Save the results to the output file
    with open(output_file, "w") as f:
            f.write("Category,Station,Average Temperature\n")
            for station in warmest_stations:
                f.write(f"Warmest,{station},{max_avg_temp}\n")
            for station in coolest_stations:
                f.write(f"Coolest,{station},{min_avg_temp}\n")

# Specify the folder containing temperature data and the output files
temperature_folder = "temperature_data"
seasonal_output_file = "average_temp.txt"
largest_range_output_file = "largest_temp_range_station.txt"
warmest_coolest_output_file = "warmest_and_coolest_station.txt"

# Run the functions
calculate_seasonal_average(temperature_folder, seasonal_output_file)
find_largest_temp_range(temperature_folder, largest_range_output_file)
find_warmest_and_coolest_stations(temperature_folder, warmest_coolest_output_file)

print(f"Seasonal average temperatures saved to {seasonal_output_file}.")
print(f"Largest temperature range station(s) saved to {largest_range_output_file}.")
print(f"Warmest and coolest stations saved to {warmest_coolest_output_file}.")
```

Output:


File Edit View
Station,Temperature Range
WYALONG-POST-OFFICE,20.58


File Edit View
Category,Station,Average Temperature
Warmest,MARBLE-BAR,39.715833333333336
Coolest,CABRAMURRA-SMHEA-AWS,13.04

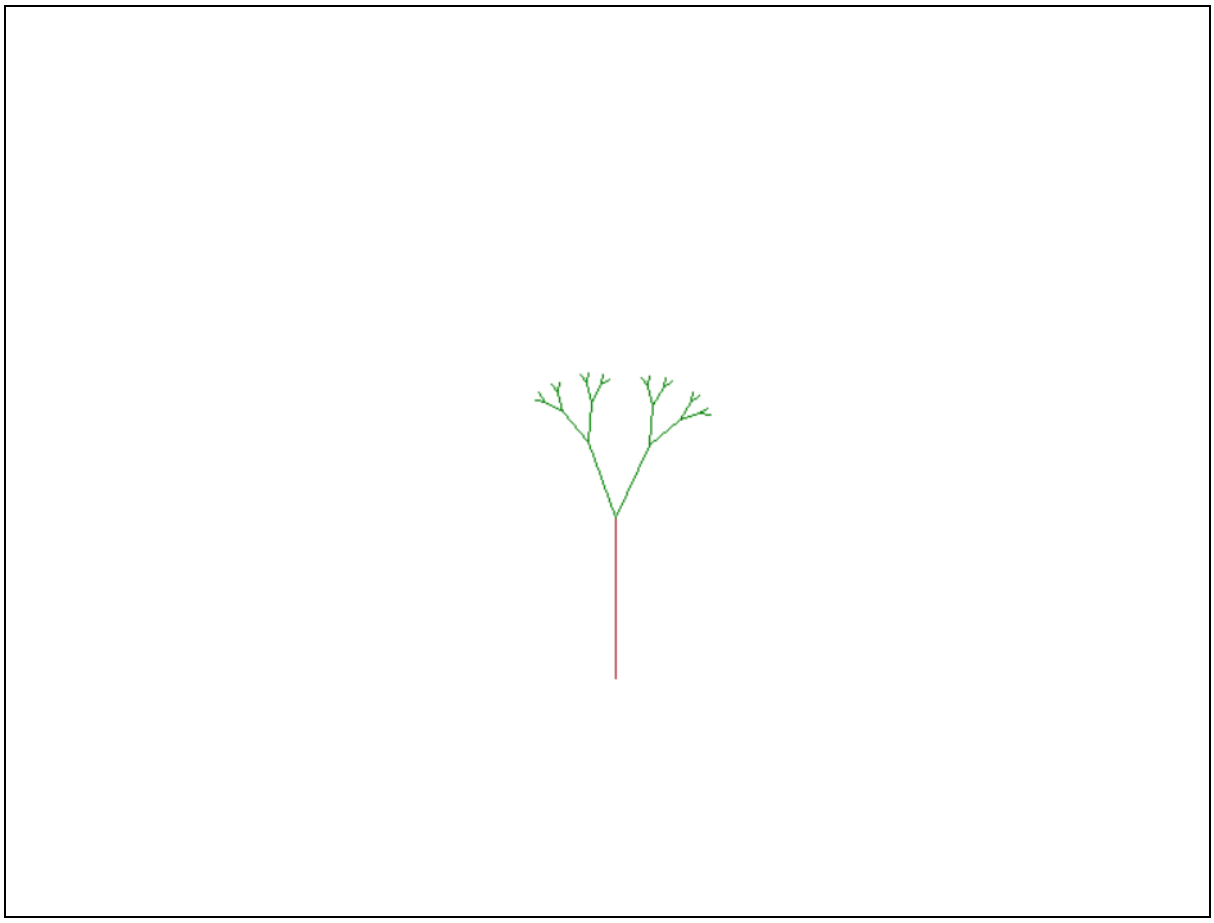

File Edit View
Season,Average Temperature
Summer,32.10375148809519
Autumn,27.320270833333304
Winter,21.067285714285738
Spring,27.433477678571432

Question 3:

C: > Users > Rahul > Desktop > tree.py > ...

```python
import turtle

def draw_branch(t, branch_length, left_angle, right_angle, depth, reduction_factor):
    if depth > 0:
        # Draw the current branch
        t.forward(branch_length)
        t.pencolor("brown")
        # Draw the left branch
        t.left(left_angle)
        t.pencolor("green")
        draw_branch(t, branch_length * reduction_factor, left_angle, right_angle, depth - 1, reduction_factor)
        # Return to the main branch
        t.right(left_angle + right_angle)
        # Draw the right branch
        draw_branch(t, branch_length * reduction_factor, left_angle, right_angle, depth - 1, reduction_factor)
        # Return to the main branch
        t.left(right_angle)
        t.backward(branch_length)



def main():
    # Get user input for parameters
    left_angle = int(input("Enter left branch angle: "))
    right_angle = int(input("Enter right branch angle: "))
    starting_branch_length = int(input("Enter starting branch length: "))
    depth = int(input("Enter recursion depth: "))
    reduction_factor = float(input("Enter branch length reduction factor: "))

    # Set up turtle graphics
    t = turtle.Turtle()
    screen = turtle.Screen()
    screen.bgcolor("white")
    t.speed("fast")

    # Position the turtle
    t.left(90)
    t.up()
    t.backward(100)
    t.down()

    # Draw the tree
    draw_branch(t, starting_branch_length, left_angle, right_angle, depth, reduction_factor)

    # Hide the turtle and display the result
    t.color("brown")
    t.forward(100)

    t.hideturtle()
    turtle.done()

if __name__ == "__main__":
    main()
```

Output:

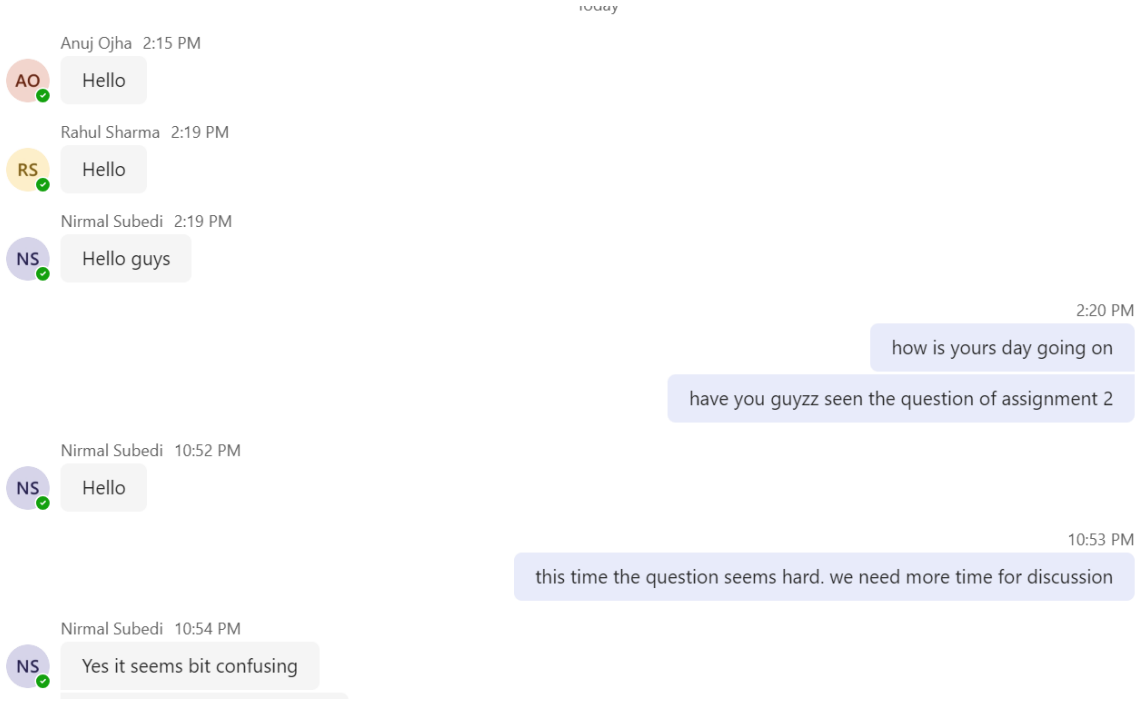


Conclusion:

This assignment effectively demonstrates Python's capability in handling diverse programming challenges. The encryption and decryption task showcases text processing and algorithm design, providing a secure and efficient approach to file manipulation. The temperature data analysis task highlights the utility of Python in data processing and statistical computation, presenting insights derived from real-world datasets. Finally, the recursive tree generation task emphasizes the creative use of recursion and graphical modules, producing visually appealing results.

The solutions not only meet the requirements but also incorporate modular designs, ensuring code reusability and extensibility. Future improvements and extensions can be easily integrated into the current implementation, making these solutions scalable for broader applications.

Group Discussion:



I've been getting more errors

Can you look at this code I have forwarded you in what's

Whatsapp

10:55 PM

lets try to solve 1 question each so it will be easy for us

Nirmal Subedi 10:55 PM

NS Yea. I've been doing question number 3

10:56 PM

i wii be doing question 1 and rahul will be doing question 2

anuj will help us in managing resources needed for it

Anuj Ojha 11:03 PM

AO Hello guys

How are you

11:03 PM

all good dear friend

Nirmal Subedi 11:03 PM

NS Ok we will do that

At first we will try to run our code and then will push to git

I'm facing errors so I'm trying to resolve it

Anuj Ojha 11:05 PM

AO Rahul had send me the codes for the question number 2 and i have checked the output and codes has given thw right output and the codes run well enough there's no error in the code

11:06 PM

rahul have already made the repository

Nirmal Subedi 11:06 PM

NS That's great I will also send you after getting proper output

Rahul Sharma 11:07 PM

RS

> Nirmal Subedi 1/16/2025 10:54 PM
> Can you look at this code I have forwarded you in what's

As Dicussed in the classroom yesterday, i have explained you logic and if you have any doubt pls let me know

11:07 PM

he has sent an email and we need to accept the invitation

Rahul Sharma 11:07 PM

RS Any errors in the code if we do it together we will be able complete the faster

Nirmal Subedi 11:08 PM

NS

> Rahul Sharma 1/16/2025 11:07 PM
> As Dicussed in the classroom yesterday, i have explained you logic and if you have any doubt pls let me know

Yes I'm using that logic and solving
I will try and also if I get confused I will tell you

11:08 PM

if you guys complete your portion the push in the github

Nirmal Subedi 11:09 PM

NS Sure mate we will do after completing it

Anuj Ojha 11:09 PM

AO Guys can you push your codes in the github as rahul has already pushed his code and me and rahul will looking the output and if there is any error we will let you know

11:09 PM

if you guys have any difficulty in cloning the githhub you can contact me 👍

Nirmal Subedi 11:09 PM

NS Can you guys look at my code by joining video call on messenger

11:10 PM

```python
def encrypt_text(n, m):
    def shift_char(char):
        if 'a' <= char <= 'm':
            return chr(((ord(char) - ord('a') + (n * m)) % 13) + ord('a'))
        elif 'n' <= char <= 'z':
            return chr(((ord(char) - ord('n') - (n + m)) % 13) + ord('n'))
        elif 'A' <= char <= 'M':
            return chr(((ord(char) - ord('A') - n) % 13) + ord('A'))
        elif 'N' <= char <= 'Z':
            return chr(((ord(char) - ord('N') + (m ** 2)) % 13) + ord('N'))
        else:
            return char

```

the is the first portion of the code

Rahul Sharma 11:10 PM

RS

> Kishor Katuwal 1/16/2025 11:08 PM
> if you guys complete your portion the push in the github

guys before pushing the code make user yu include all the necessary files required for the code to run smoothly without any bugs and errors

11:10 PM

and it is working well

i will push in the github

Anuj Ojha 11:11 PM

AO Let me check the output and i will send a screenshot then

Rahul Sharma 11:11 PM

RS

> Nirmal Subedi 1/16/2025 11:09 PM
> Can you guys look at my code by joining video call on messenger

lets stick to msteams and please all the communication on the ms teams only

11:13 PM

this is the best and secure platfrom for doing assignments

Anuj Ojha 11:14 PM

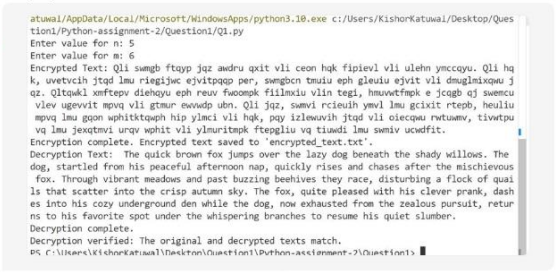
AO I have check the codes for the question one that you have push in github and the output is correct and there is no error

11:14 PM

if you guyzz and any progress or problem share here

Anuj Ojha 11:14 PM

AO



This is the output of the code

11:15 PM

this one is working perfectly

Anuj Ojha 11:15 PM

AO Rahul have you pushed your code ls for question number 2 in github?

Rahul Sharma 11:16 PM

RS yep, wait let me send you the code snippet and verify the output and its working

Anuj Ojha 11:16 PM

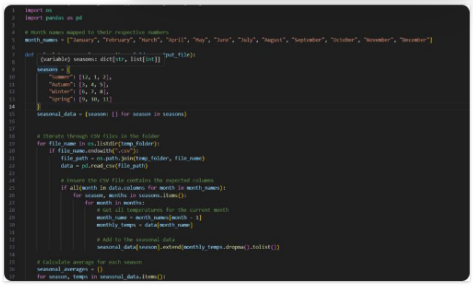
AO Oh okay let me have a look

Nirmal Subedi 11:17 PM

NS Thank you guys finally my code has also worked

Rahul Sharma 11

RS

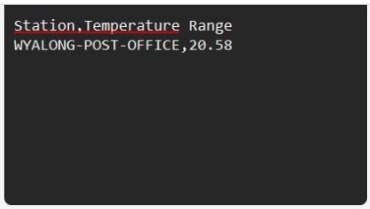


Nirmal Subedi 11:17 PM

NS Let me push to git

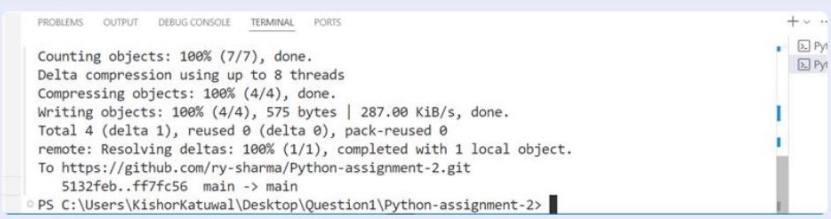
Rahul Sharma 11:17 PM

RS



The output from sloving the dataset manually and the output generated from the code both are accurate

11:20 PM



i have push question 1 in the github

Anuj Ojha 11:20 PM

AO The 2nd question is done and is already pushed on the github

11:21 PM

Anuj Ojha 11:21 PM

AO Nirmal Subedi could you send me the code snippet for 3rd question and make sure to push in github for all of us acces it

Rahul Sharma 11:22 PM

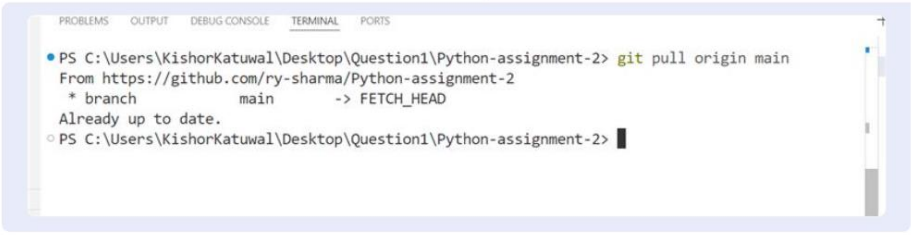
RS

> Kishor Katuwal 1/16/2025 11:21 PM
> you guyzz can also try running it un your system

yep done and commands run perfectly

11:23 PM

dont forget to pull the code form the github



Anuj Ojha 11:25 PM

AO I am about to finish the documentation part that was allocated for me i will finish it shortly and let you

AO I am about to finish the documentation part that was allocated for me i will finish it shortly and let you guys know

11:26 PM

don't forget to follow the formatting guidelines

Nirmal Subedi 11:27 PM

NS

Anuj Ojha 11:28 PM

AO Nirmal Subedi can you please push the codes in the github i have to check the output and if there is any error in the code as Rahul and kishor have already push there codes in the github

Nirmal Subedi 11:28 PM

NS Codes of questions 3

Yea

I've pushed recently

Can you have a look at that

Anuj Ojha 11:28 PM

AO Okay thank you i will have a look and inform you shortly

Nirmal Subedi 11:29 PM

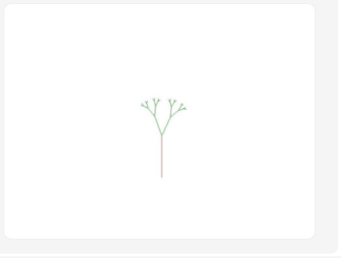
NS Ok friend

Anuj Ojha 11:31 PM

AO I have run the codes of question number 3 and there is no error and the output is also correct

Anuj Ojha 11:31 PM

AO I have run the codes of question number 3 and there is no error and the output is also correct



Thank guys for the codes i will be finishing my part and i will send a pdf file there about the full documentation