Name : Rejey Ezekiel

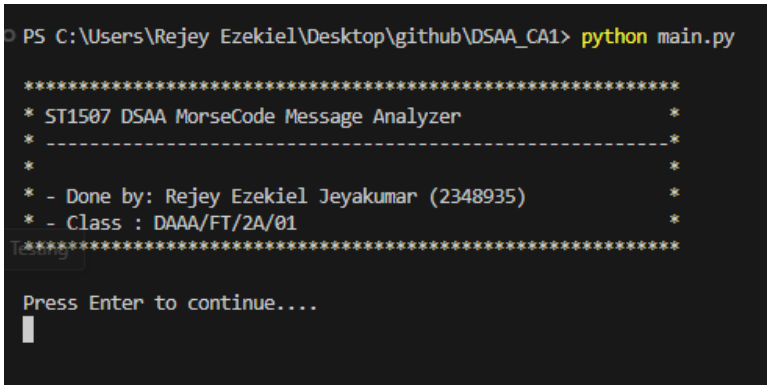
StudentID: 2348935

Class: DAAA/FT/2A/01

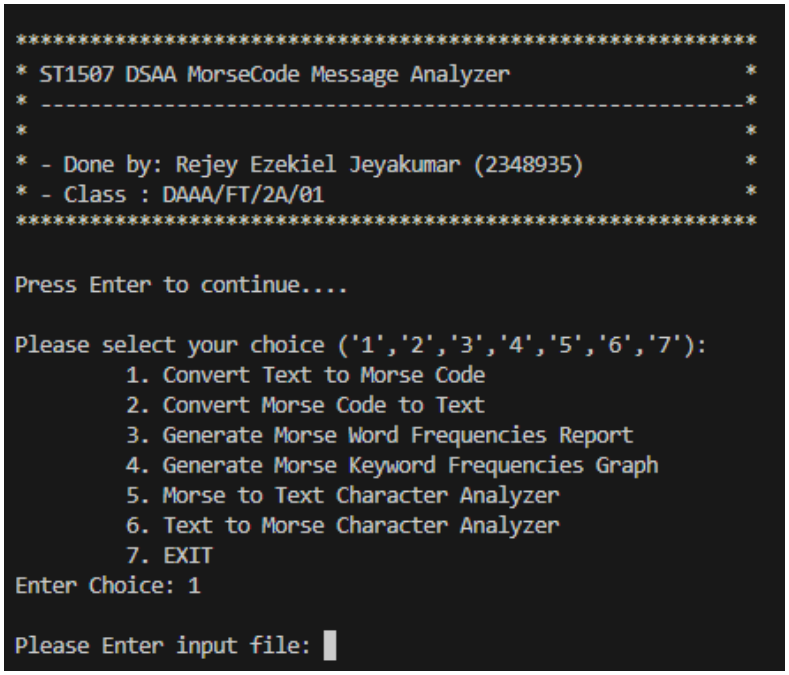
Assignment : ST1507_ASSIGNMENT_ONE_AY2425_SEM1

How to Operate my App:

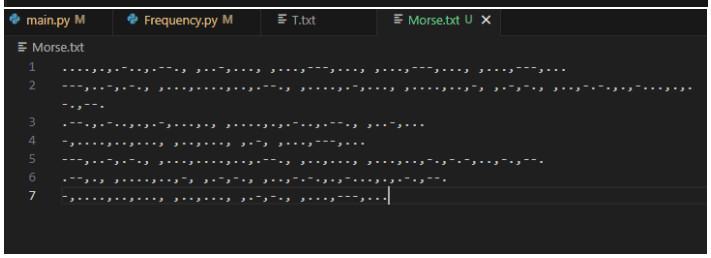
1. Run python main.py



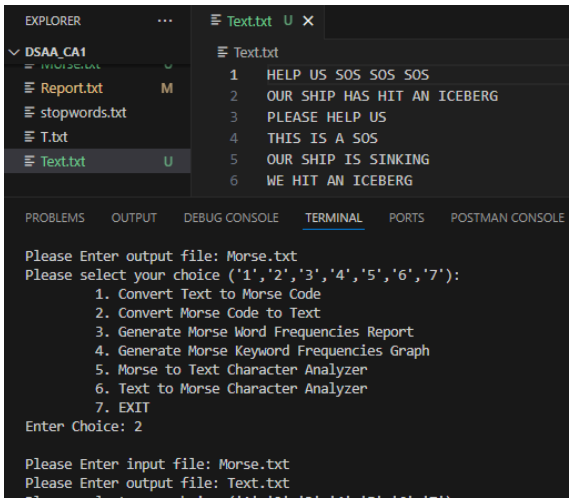
2. Press enter and select and option (Option: 1)



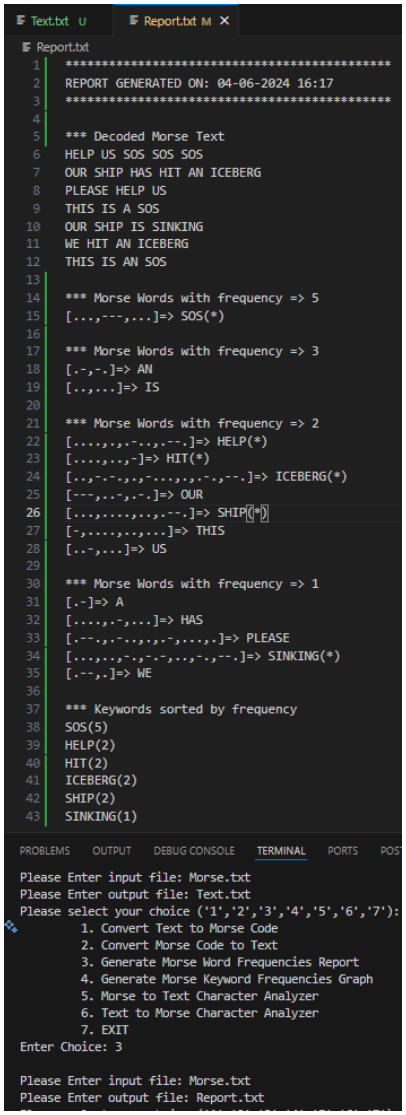
   a. Enter a Text file and a Output File :



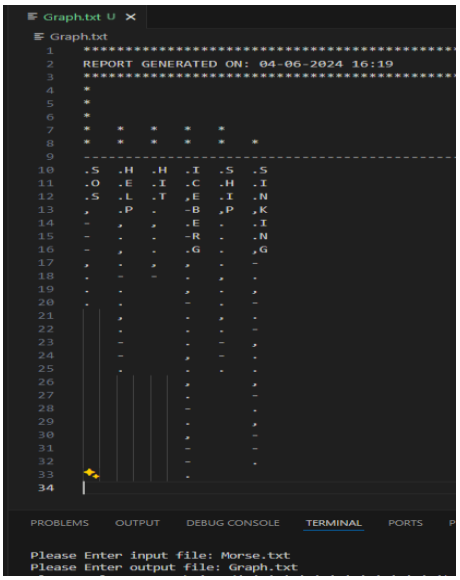
3. Select Option 2 Input the Morse file u just made and see it convert back to text.



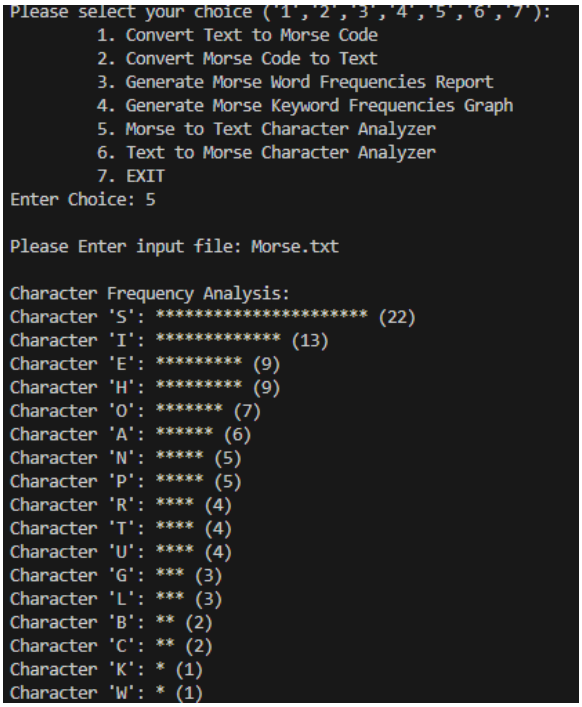
4. Select Option 3 to Generate morse Word Frequency Report



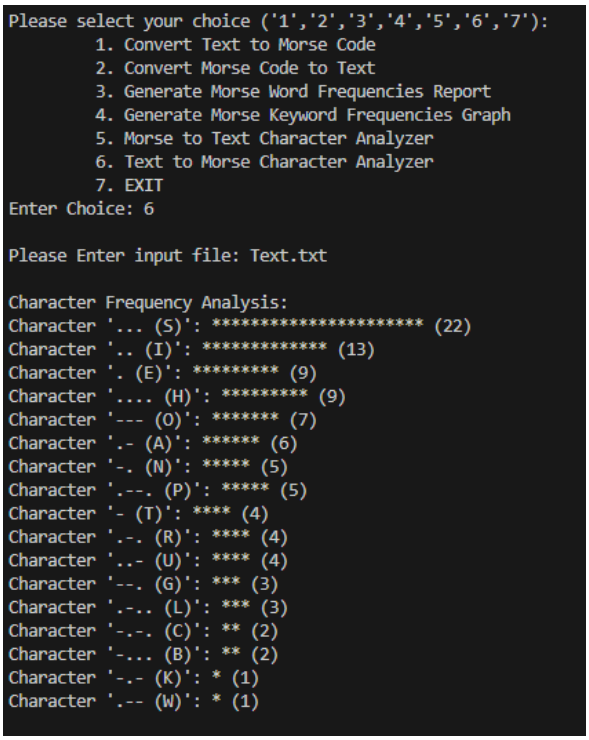
5. Select Option 4 to Generate Morse Keyword Frequencies Graph and input created Morse file.



6. Select Option 5 to view Text Character Analyzer



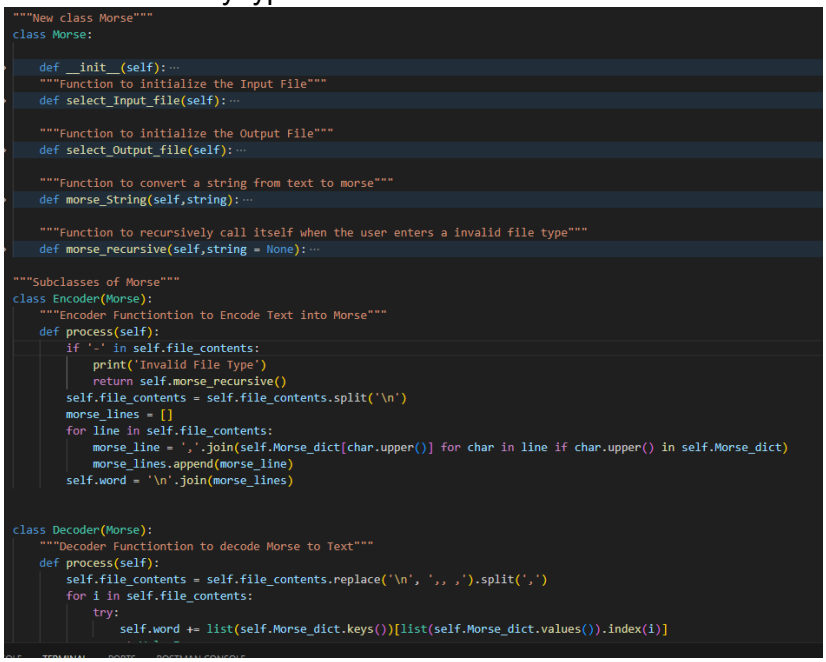
7. Select Option 6 to view Morse Character Analyzer



8. Lastly to exit the app press 7

```
Please select your choice ('1','2','3','4','5','6','7'):
         1. Convert Text to Morse Code
         2. Convert Morse Code to Text
         3. Generate Morse Word Frequencies Report
         4. Generate Morse Keyword Frequencies Graph
         5. Morse to Text Character Analyzer
         6. Text to Morse Character Analyzer
         7. EXIT
Enter Choice: 7
Bye, thanks for using ST1057 DSAA: MorseCode Message Analyzer
PS C:\Users\Rejey Ezekiel\Desktop\github\DSAA_CA1>
```

I have used many types of Inheritance like below :



```
"""New class Morse"""
class Morse:

    def __init__(self):...
    """Function to initialize the Input File"""
    def select_Input_file(self):...

    """Function to initialize the Output File"""
    def select_Output_file(self):...

    """Function to convert a string from text to morse"""
    def morse_String(self,string):...

    """Function to recursively call itself when the user enters a invalid file type"""
    def morse_recursive(self,string = None):...

"""Subclasses of Morse"""
class Encoder(Morse):
    """Encoder Functiontion to Encode Text into Morse"""
    def process(self):
        if '-' in self.file_contents:
            print('Invalid File Type')
            return self.morse_recursive()
        self.file_contents = self.file_contents.split('\n')
        morse_lines = []
        for line in self.file_contents:
            morse_line = ','.join(self.Morse_dict[char.upper()] for char in line if char.upper() in self.Morse_dict)
            morse_lines.append(morse_line)
        self.word = '\n'.join(morse_lines)


class Decoder(Morse):
    """Decoder Functiontion to decode Morse to Text"""
    def process(self):
        self.file_contents = self.file_contents.replace('\n', ',, ,').split(',')
        for i in self.file_contents:
            try:
                self.word += list(self.Morse_dict.keys())[list(self.Morse_dict.values()).index(i)]
```

I have also used Algorithms like Recursion to call itself and OOP approaches like Polymorphism:

```
    """Function to recursively call itself when the user enters a invalid file type"""
    def morse_recursive(self,string = None):
        self.select_Input_file()
        if Utility.CheckFileType(self.Input_File):
            self.file_contents = Utility.OpenTextFile(self.Input_File)
            self.process()
        else:
            self.morse_recursive()
        if string == None:
            state = True
            if state:
                self.select_Output_file()
                if Utility.CheckFileType(self.Output_File,1):
                    Utility.WriteFile(self.Output_File, self.word)
                    state = False
        else:
            return self.word
```

Operation overloading for my word class:

```
class Word(Node):
    def __init__(self, name, frequency=1):
        self.original_name = name  # Store original case
        self.name = name.lower().strip()  # Normalize to lower case for comparison
        self.frequency = frequency
        super().__init__()

    def size(self):
        return len(self.name)

    def __eq__(self, otherNode):
        if otherNode is None:
            return False
        return self.name == otherNode.name

    def __lt__(self, otherNode):
        if otherNode is None:
            raise TypeError(
                "'<' not supported between instances of 'Word' and 'NoneType'")
        if self.frequency == otherNode.frequency:
            return self.name < otherNode.name
        return self.frequency > otherNode.frequency

    def __str__(self):
        return f'{self.original_name}: {self.frequency}'
```

Private Function :

```
def __appendToHead(self, newNode):
    oldHeadNode = self.headNode
    self.headNode = newNode
    self.headNode.nextNode = oldHeadNode
"""
```

Initializing class with its super class :

```
class Report(Frequencies):

    """Initializes the Report class and its attributes."""
    def init(self):
        super()._init_()
        self.x = []
        self.frequency = None
        self.Default = None
```

For this assignment I have used 2 main Algorithms which are SortedList and Recursion.

Challenges I faced :
I felt that I didn’t have a lot of time to implement more extravagant things as I had many issues where code wouldn’t work here and would work at some other point therefore I had to rewrite my code many times to also follow the OOP standards.

Hierarchy of Classes :