**MET CS 521 TERM PROJECT REPORT**

Table of Contents

[ABOUT THIS PROJECT 2](#_Toc102849750)

[SET UP 2](#_Toc102849751)

[INTERACTING WITH THE APPLICATION 2](#_Toc102849752)

[PROJECT REQUIREMENTS & FULFILLMENT 3](#_Toc102849753)

[SAMPLE I/O SCREENSHOTS 8](#_Toc102849754)

# ABOUT THIS PROJECT

This document and code included in this folder/directory serves as the deliverables for term project for MET CS 521. This project shows usage of various python structures and operations and fulfills the requirements as stated in the Project Guideline document. This project implements two types of lossless data compression. Lossless compression is a class of data compression algorithms that allows the original data to be perfectly reconstructed from the compressed data. The algorithms used in this application are Huffman algorithm and Run-Length algorithm. The idea behind Huffman algorithm is to assign variable-length codes to input characters where lengths of the assigned codes are based on the frequencies of corresponding characters. The most frequent character gets the smallest code, and the least frequent character gets the largest code. The variable-length codes assigned to input characters are [Prefix Codes](http://en.wikipedia.org/wiki/Prefix_code), means the codes (bit sequences) are assigned in such a way that the code assigned to one character is not the prefix of code assigned to any other character. This is how Huffman Coding makes sure that there is no ambiguity when decoding the generated bitstream. As for the Run-length encoding (RLE), it is a form of lossless data compression in which runs of data (sequences in which the same data value occurs in many consecutive data elements) are stored as a single data value and count, rather than as the original run.

# SET UP

This project requires Python 3.10 or above to execute.

# INTERACTING WITH THE APPLICATION

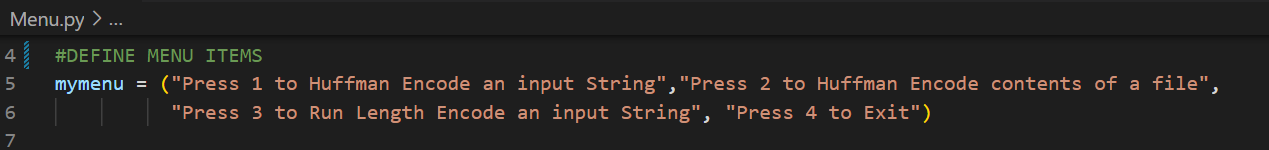
Menu.py serves as the entry point for our project/application. This file displays a variety of options. Option 1 allows user to encode data using Huffman algorithm by obtaining the user input via console. Option 2 allows user to encode data using Huffman algorithm for the data present in file HuffmanFile.txt. Currently, the file includes the nerdy poem which was provided to us by professor for the implementation of string editor. Option 3 allows user to enter the input and encode the data using run-length algorithm. Option 4 terminates the program. After execution of each option, user is prompted to continue with the execution of program to perform any of the options mentioned above. User can enter y or Y to continue. Any other key press will terminate the program.

# PROJECT REQUIREMENTS & FULFILLMENT

The locations of code snippets that implement each project requirement are shown in the screenshots below. Each screenshot has the file name and the location (line number) of occurrence of said implementation.

1. CONTAINER TYPE

TUPLE



LIST

Text

Description automatically generated

DICTIONARY

Graphical user interface, text, application

Description automatically generated

1. ITERATION TYPE

WHILE LOOP

A screenshot of a computer

Description automatically generated with medium confidence

FOR LOOP

A screenshot of a computer

Description automatically generated with medium confidence

1. CONDITIONAL (IF)

A screenshot of a computer

Description automatically generated with medium confidence

1. TRY BLOCKS

A screenshot of a computer

Description automatically generated with medium confidence

1. USER DEFINED FUNCTIONS

Text

Description automatically generated

1. INPUT AND/OR OUTPUT FILE (submit input data)

Text

Description automatically generated

1. USER-DEFINED CLASS. THE CLASS MUST BE IMPORTED BY YOUR MAIN PROGRAM AND HAVE THE FOLLOWING REQUIRED STRUCTURES.

Text

Description automatically generated

* 1. AT LEAST 1 PRIVATE AND 2 PUBLIC SELF ATTRIBUTES

A screenshot of a computer

Description automatically generated with medium confidence

* 1. AT LEAST 1 PRIVATE AND 1 PUBLIC METHOD THAT TAKE ARGUMENTS, RETURN VALUES AND ARE USED BY YOUR PROGRAM

Text

Description automatically generated

* 1. AN INIT() METHOD THAT TAKES AT LEAST 1 ARGUMENT

A screenshot of a computer

Description automatically generated with medium confidence

* 1. A REPR() METHOD

Graphical user interface, text

Description automatically generated

1. PROVIDE UNIT TESTS THAT PROVE THAT YOUR CLASS METHODS WORK AS EXPECTED. THE TESTS SHOULD EVALUATE RESULTS USING ASSERT STATEMENTS.

Text

Description automatically generated

# SAMPLE I/O SCREENSHOTS

Main menu

Graphical user interface, text

Description automatically generated

Selecting 1 and running Huffman Algorithm on an input string

Text

Description automatically generated

Selecting 2 and running Huffman algorithm on contents of a file

Text

Description automatically generated

Selecting 3 and running run length algorithm

Text

Description automatically generated