**4CS001 - Coding Challenge 2***Data Structures and Iteration*

**Due: Sunday 3rd November 2019 at 11:59am**   
**This assignment is worth 20% of the overall module grade**

Introduction: 2

Task Overview: 2

Getting Started: 2

Requirements: 3

General Hints: 3

Example Implementation (user input in red): 4

Challenge (Extra Credit): 5

Introduction: 5

Logical Breakdown: 5

Finding the Shortest Subsequence: 6

Example Implementation (user input in red): 7

Introduction:

This coding challenge will assess your knowledge of Python data structures and iteration. It also builds on many of the topics we have covered over the course of the module.

The marking scheme for this task is on Canvas, make sure you check back on a regular basis as you work through the assessment. **There is an additional challenge segment that can earn you extra credit, which you will need to complete to achieve the highest possible mark. Students who do not attempt this cannot achieve grades above 80% for this task.**

Your task is to develop a program that encodes/decodes Morse Code.

Task Overview:

**Morse code** is a character encoding scheme used in telecommunications. It is named after the inventor of the telegraph Samuel Morse and represents textual characters as combinations of long (dashes) and short (dots) electronic pulses, lights or sounds.

For example, the letter J is coded as .---, ! as -.-.-- and 1 as .−−−−

A **single space is used to separate characters within a word** and **3 spaces between words**. Morse is **case-insensitive**, so capital letters are used as a matter of course.

For example, the message **HELLO WORLD** in Morse code is:

.... . .-.. .-.. --- .-- --- .-. .-.. -..

In addition to letters, digits and punctuation, there are some special service codes, the most notorious of which is the international distress signal SOS, coded as ...−−−...  
These special codes are treated as single characters, but transmitted as separate words.

Getting Started:

Start by downloading the files morse.py and morse\_tests.txt from Canvas. Look through the Python file and read the included documentation. Add your name and student number to the top of the file. The text file contains various inputs and expected outputs you can use to test your program to make sure that it is working as intended.

Requirements:

You will develop a program capable of encoding and decoding Morse code.

1. Prompt users to select a mode (encode or decode).
2. Check if the mode the user entered is valid. If not, continue to prompt the user until a valid mode is selected.
3. Prompt the user for the message they would like to encode/decode.
4. Check if the message contains valid characters. If not, continue to prompt the user until a valid message is selected (dependent upon the mode selected).
5. Encode/decode the message as appropriate and print the output.
6. Prompt the user whether they would like to encode/decode another message.
7. Check if the user has entered a valid input (y/n) If not, continue to prompt the user until they enter a valid response. Depending upon the response you should either:
   1. End the program if the user selects no.
   2. Proceed directly to step 2 if the user says yes.

General Hints:

* You can make use of str.split() to generate a list of Morse words and characters by using the spaces between words and characters as a separator.

>>> morse = '.... . .-.. .-.. --- .-- --- .-. .-.. -..'

>>> words = morse.split(' ') # 3 spaces used to separate words

>>> words # List containing 2 words in Morse

['.... . .-.. .-.. ---', '.-- --- .-. .-.. -..']

>>> for word in words:

... characters = word.split(' ') # 1 space between letters

... characters

...

['....', '.', '.-..', '.-..', '---']

['.--', '---', '.-.', '.-..', '-..']

* You will also find str.join() useful for constructing a string from a list of strings.

>>> characters

['.--', '---', '.-.', '.-..', '-..', '-.-.--']

>>> ' '.join(characters)

'.-- --- .-. .-.. -.. -.-.--'

* You should use a loop to keep the programming running if the user says that would like to encode/decode more messages.
* Your program should handle both uppercase and lowercase inputs. You can use str.upper() and str.lower() to convert strings to a specific case.

Example Implementation (user input in red):

This program encodes and decodes Morse code.

Would you like to encode (e) or decode (d): g

Invalid mode.

Would you like to encode (e) or decode (d): d

What message would you like to decode: .... . .-.. .-.. --- .-- --- .-. .-.. -..

HELLO WORLD

Would you like to encode/decode another message? (y/n): t

Invalid response

Would you like to encode/decode another message? (y/n): y

Would you like to encode (e) or decode (d): e

What message would you like to encode: Hello WORld

.... . .-.. .-.. --- .-- --- .-. .-.. -..

Would you like to encode/decode another message? (y/n): n

Thanks for using the program, goodbye!

Challenge (Extra Credit):

Introduction:

Morse code was often transmitted using electric telegraphs. These devices were the first real electronic telecommunications devices and worked by sending electrical pulses down dedicated lines between two or more telegraph offices. These pulses were controlled by the operator depressing a key, allowing current to flow down the line and be detected at receiving stations. In Morse code the dots (short press) and dashes (long press) therefore represented periods during which current was flowing and the spaces between them periods when it was not. This means that timing and rhythm were important factors when transmitting Morse code. Therefore, the international standard specifies the following guidelines:

* A "Dot" should be 1 time unit long.
* A "Dash" should be 3 time units long.
* A Pause between dots and dashes in a character should be 1 time unit long.
* A Pause between characters inside a word should be 3 time units long.
* A Pause between words should be 7 time units long.

However, what the standard does not mandate is the length of a time unit. In fact, this varied greatly depending upon the skill of the operator. It might take an amateur a minute to transmit a single word, however, experienced operators have been known to achieve rates of around 60 words per minute. Robot operators can go even faster than this.

Your task is to improve your program so that it can decode remote telegraph signals in the form of 1’s (key is down) and 0’s (key is up). To simplify this task, we will assume that the underlying hardware receives the signal and records pulses in the correct format.

For example, the message: HELLO WORLD:   
.... . .-.. .-.. --- .-- --- .-. .-.. -..

might be received as:

110011001100110000001100000011001111110011001100000011001111110011001100000011111100111111001111110000000000000011001111110011111100000011111100111111001111110000001100111111001100000011001111110011001100000011111100110011

Here we can see that a dot (single time unit) is represented by two 1’s and a dash by 6 (3 time units), conforming to the international Morse standard described above. In this case, the hardware has sampled the line two times per dot, but this might not always be the case. You should implement additional logic that allows users to decode electric telegram messages in this form. To do this, you will need to determine the transmission rate of messages and convert them to dots and dashes before converting to plain text. Note that there may be additional 0’s at the beginning or end of messages, these should be ignored. If it is not possible to determine whether a signal is a dot or a dash, you should assume that it is a dot.

Logical Breakdown:

To determine the transmission rate of a message in 1’s and 0’s, you should search for the shortest subsequence of 1’s and 0’s. This should allow you to determine the time unit used to represent a dot and space between letters. It will then be possible to ascertain the sequences used to represent dashes, spaces between characters in a word and the spaces between words by following the rules defined in the international Morse standard.

1. Remove excess 0’s at the beginning and end of the message.
2. Find the shortest subsequence of 1’s and 0’s
3. Use this to determine the meaning of each sequence of 1’s and 0’s
4. Replace each of the subsequence’s in the string with the Morse equivalent.
5. Decode the Morse code to plain text as before.

The difficult part here is determining the shortest subsequence of 1’s and 0’s. You can use str.split() with ‘0’ and ‘1’ as delimiters to generate a list of subsequence’s with that character removed. You can then combine this with the min() function to find the shortest sequence of remaining characters in the list. The examples below demonstrate this method of finding the shortest subsequence.

**Note:** the same message is used in both examples, however, the time units differ as might be expected with varying transmission speeds in the real world.

Finding the Shortest Subsequence:

>>> code = '1010101000100010111010100010111010100011101110111'

>>> zeros = code.split('1') # List: subsequence’s of 0’s

>>> zeros = filter(None, zeros) # Remove blanks

>>> zeros

['0', '0', '0', '000', '000', '0', '0', '0', '000', '0', '0', '0', '000', '0', '0']

>>> len(min(zeros))

1 # The shortest subsequence of 0’s has length 1

Transmission speed is 1. Each dot is represented by 1 1. Each dash is represented by 3 1’s

>>> code = '11001100110011000000110000001100111111001100110000001100111111001100110000001111110011111100111111'

>>> zeros = code.split('1') # List: subsequence’s of 0’s

>>> zeros = filter(None, ones) # Remove blanks

>>> zeros

['00', '00', '00', '000000', '000000', '00', '00', '00', '000000', '00', '00', '00', '000000', '00', '00']

>>> len(min(zeros))

2 # The shortest subsequence of 0’s has length 2.

Transmission speed is 2. Each dot is represented by 2 1’s. Each dash is represented by 6 1’s

Example Implementation (user input in red):

This program encodes and decodes Morse code.

Would you like to encode (e), decode (d) or decode telegraph(dt): dt

What message would you like to decode: 1010101000100010111010100010111010100011101110111

.... . .-.. .-.. ---

HELLO

Would you like to encode/decode another message? (y/n): y

Would you like to encode (e), decode (d) or decode telegraph(dt): dt

What message would you like to decode: 11001100110011000000110000001100111111001100110000001100111111001100110000001111110011111100111111

.... . .-.. .-.. ---

HELLO

Would you like to encode/decode another message? (y/n): n

Thanks for using the program, goodbye!