Introduction:

Morse code, a historic communication method utilizing dots (.) and dashes (-), has found widespread use in telecommunications and cryptography. This article describes a Python-based Morse code encryption and decryption application, a useful tool for converting English sentences to Morse code and vice versa. Beyond its historical significance, Morse code remains relevant in a variety of circumstances, making this application useful in the digital age. This paper investigates the complexities of the Morse code application, its features, and the underlying mechanisms that make it a must-have tool for both enthusiasts and professionals.

Instructions for Using the Application:

Operating the Morse code encryption and decryption application is straightforward and user-friendly. Users can effortlessly interact with Morse code using the following steps:

•Running the Application: Execute the provided Python script in a Python environment.

•Choose Encryption or Decryption: Upon execution, the program prompts the user to choose between encryption and decryption. To indicate their choice, users enter 'E' for encryption or 'D' for decryption.

•Encryption (E Option): If encryption is selected, users input an English phrase. The application then encrypts the input into Morse code and displays the resulting Morse code representation.

•Decryption (D Option): If decryption is chosen, users input a Morse code string. The input should contain spaces between letters and slashes between words. The application decrypts the Morse code back into English text and displays the resulting phrase.

•Handling Invalid Input: In case of invalid input, the application provides an error message, guiding users to enter 'E' for encryption or 'D' for decryption.

Table of Unit Tests:

Below is a detailed summary of the unit tests conducted on the encryption and decryption functions:

Test Case	Function	Expected Output
"HELLO"	encrypt	'.... . .-.. .-.. ---'
"WORLD"	encrypt	'.-- --- .-. .-.. -..'
"SOS"	encrypt	'... --- ...'
"123"	encrypt	'.---- ..--- ...--'
Empty Input	encrypt	""
"TEST TEST"	encrypt	'- . ... -/- . ... -'
"@#$"	encrypt	""
"TEST123"	encrypt	'- . ... - .---- ..--- ...--'
"HELLO WORLD"	encrypt	'.... . .-.. .-.. ---/.-- --- .-. .-.. -..'
"A B C D E F G H I J K L M"	encrypt	'.-/-.../-.-./-.././..-./--./..../../.---/-.-/.-../--'
"N O P Q R S T U V W X Y Z"	encrypt	'-./---/.--./--.-/.-./.../-/..-/...-/.--/-..-/-.--/--..'
"TESTING Morse Code"	encrypt	'- . ... - .. -. --./-- --- .-. ... ./-.-. --- -.. .'
".... . .-.. .-.. ---"	decrypt	"HELLO"
".-- --- .-. .-.. -.."	decrypt	"WORLD"
"... --- ..."	decrypt	"SOS"
".---- ..--- ...--"	decrypt	"123"
Empty Input	decrypt	""
"- . ... - / - . ... -"	decrypt	"TEST TEST"
".---- ..--- ...--"	decrypt	"123"
".... . .-.. .-.. --- / .-- --- .-. .-.. -.."	decrypt	"HELLO WORLD"
".-/-.../-.-./-.././..-./--./..../../.---/-.-/.-../--"	decrypt	"A B C D E F G H I J K L M"
"-./---/.--./--.-/.-./.../-/..-/...-/.--/-..-/-.--/--.."	decrypt	"N O P Q R S T U V W X Y Z"
"- . ... - .. -. --./-- --- .-. ... ./-.-. --- -.. ."	decrypt	"TESTING MORSE CODE"

Link to Repository:

For access to the complete source code and project details, kindly visit the official Morse Code Repository at the following link: [Morse Code Repository.]

Description of Algorithms and Time Complexity:

The Morse code function, works by traversing the input phrase and transforming each character to its Morse code counterpart. This conversion makes use of a specified dictionary. This operation has a temporal complexity of O(N*M), where N is the length of the input phrase and M is the average length of Morse code representations in the dictionary. Within this level of complexity, each character in the input phrase is subjected to a constant-time dictionary lookup process.

This procedure is reversed by the Morse code decryption function, referred to as decrypt. It breaks down the Morse code string into words and characters, then uses a reverse dictionary to map each Morse code character back to its matching English character. Like the encryption function, the time complexity of this operation is O(N*M), where N signifies the number of Morse code characters in the input string and M denotes the average length of Morse code representations in the dictionary.

Conclusion:

The Morse code encryption and decryption program is built on the available Python source code. The codebase follows best practices, which ensures readability and maintainability. The unittest framework facilitates robust testing. This stringent testing procedure ensures the accuracy and dependability of the encryption and decryption processes over a wide range of input scenarios.

In conclusion, this Morse code encryption and decryption program not only provides a dependable solution for encoding and decoding messages, but it also demonstrates Python's adaptability in handling complicated textual jobs. The inclusion of rigorous unit tests, as well as the supply of clear, succinct instructions, considerably improves the application's reliability and usability. As a result, this application stands as an invaluable tool for Morse code enthusiasts and communication professionals, underscoring the enduring relevance of Morse code.

Source Code:

Main source code:

```python
"""
This module provides functions to encrypt and decrypt messages between English and Morse code.

The module contains the following functions:
    - encrypt: Encrypts a given phrase to Morse code.
    - decrypt: Decrypts a message from Morse code to English.
"""

dictionary = {
    'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..', 'E': '.', 'F': '..-.', 'G': '--.', 'H': '....',
    'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..', 'M': '--', 'N': '-.', 'O': '---', 'P': '.--.',
    'Q': '--.-', 'R': '.-.', 'S': '...', 'T': '-', 'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-',
    'Y': '-.--', 'Z': '--..',
    '0': '-----', '1': '.----', '2': '..---', '3': '...--', '4': '....-', '5': '.....', '6': '-....',
    '7': '--...', '8': '---..', '9': '----.', ' ': '/'
}

def encrypt(phrase):      You, 1 hour ago • semi_final
    """
    Encrypts a given phrase to Morse code.

    Args:
        phrase (str): The phrase to be encrypted.

    Returns:
        str: The encrypted Morse code string.
    """
    # Split the input phrase into words, convert each character to Morse code, and join with space.
    return '/'.join(' '.join(dictionary.get(char.upper(), '') for char in word)
                    for word in phrase.split() if all(char.upper() in dictionary for char in word))

def decrypt(morse_code):
    """
    Decrypts a message from Morse code to English.

    Args:
        morse_code (str): The Morse code message to decrypt.

    Returns:
        str: The decrypted message in English.
    """
    # Create a reverse dictionary mapping Morse code to characters
    reverse_morse_dict = {v: k for k, v in dictionary.items()}
    # Split the Morse code into words, then into characters, and map back to English characters.
    morse_words = morse_code.split('/')
    decrypted_message = ''
    for word in morse_words:
        morse_characters = word.strip().split()
        for char in morse_characters:
            if char in reverse_morse_dict:
                decrypted_message += reverse_morse_dict[char]
        decrypted_message += ' '
    return decrypted_message.strip()

if __name__ == "__main__":
    # Get input from the user
    choice = input("Enter 'E' for encryption or 'D' for decryption: ").upper()  # Convert input to uppercase
    if choice == 'E':
        input_phrase = input("Enter the English phrase to encrypt: ")
        # Encrypt the input phrase and print the Morse code
        encrypted_phrase = encrypt(input_phrase)
        print("Morse Code:", encrypted_phrase)
    elif choice == 'D':
        morse_code = input("Enter the Morse code to decrypt (use space between letters and '/' between words): ")
        # Decrypt the Morse code and print the English phrase
        decrypted_phrase = decrypt(morse_code)
        print("Decrypted Message:", decrypted_phrase)
    else:
        # Handle invalid input
        print("Invalid choice. Please enter 'E' for encryption or 'D' for decryption.")

```

Test units source code:

```python
import unittest
from Morse_code import encrypt, decrypt

import unittest

You, 1 hour ago | 1 author (You)
class TestMorseCode(unittest.TestCase):
    def test_encrypt(self):
        """
        Test the encrypt function with various inputs.
        You, 1 hour ago • semi_final
        Standard inputs:
        - Test with "HELLO", "WORLD", "SOS", and "123".

        Edge cases and special characters:
        - Test with empty input, multiple words, special characters not in the Morse code dictionary, and alphanumeric characters.

        Additional test cases:
        - Test with multiple words, all letters, all letters contd., and mix of uppercase, lowercase, and spaces.
        """
        # Standard inputs
        self.assertEqual(encrypt("HELLO"), '.... . .-.. .-.. ---')
        self.assertEqual(encrypt("WORLD"), '.-- --- .-. .-.. -..')
        self.assertEqual(encrypt("SOS"), '... --- ...')
        self.assertEqual(encrypt("123"), '.---- ..--- ...--')

        # Edge cases and special characters
        self.assertEqual(encrypt(""), "")  # Empty input
        self.assertEqual(encrypt("TEST TEST"), '- . ... -/- . ... -')  # Multiple words
        self.assertEqual(encrypt("@#$"), "")  # Special characters not in the Morse code dictionary
        self.assertEqual(encrypt("TEST123"), '- . ... - .---- ..--- ...--')  # Alphanumeric characters

        # Additional test cases
        self.assertEqual(encrypt("HELLO WORLD"), '.... . .-.. .-.. ---/.-- --- .-. .-.. -..')  # Multiple words
        self.assertEqual(encrypt("A B C D E F G H I J K L M"), '.-/-.../-.-./-.././..-./--./..../../.---/-.-/.-../--')  # All letters
        self.assertEqual(encrypt("N O P Q R S T U V W X Y Z"), '-./---/.--./--.-/.-./.../-/../.--/.-./--./-.--/--..')  # All letters contd.
        self.assertEqual(encrypt("TESTING Morse Code"), '- . ... - .. -. --./-- --- .-. ... ./-.-. --- -.. .')  # Mix of uppercase, lowercase, and spaces

    def test_decrypt(self):
        """
        Test the decrypt function with various inputs.

        Standard inputs:
        - Test with ".... . .-.. .-.. ---", ".-- --- .-. .-.. -..", "... --- ...", and ".---- ..--- ...--".

        Edge cases and special characters:
        - Test with empty input, multiple words, and alphanumeric characters.

        Additional test cases:
        - Test with multiple words, all letters, all letters contd., and mix of uppercase, lowercase, and spaces.
        """
        # Standard inputs
        self.assertEqual(decrypt('.... . .-.. .-.. ---'), "HELLO")
        self.assertEqual(decrypt('.-- --- .-. .-.. -..'), "WORLD")
        self.assertEqual(decrypt('... --- ...'), "SOS")
        self.assertEqual(decrypt('.---- ..--- ...--'), "123")

        # Edge cases and special characters
        self.assertEqual(decrypt(''), "")  # Empty input
        self.assertEqual(decrypt('- . ... - / - . ... -'), "TEST TEST")  # Multiple words
        self.assertEqual(decrypt('.---- ..--- ...--'), "123")  # Alphanumeric characters

        # Additional test cases
        self.assertEqual(decrypt('.... . .-.. .-.. --- / .-- --- .-. .-.. -..'), "HELLO WORLD")  # Multiple words
        self.assertEqual(decrypt('.-/-.../-.-./-.././..-./--./..../../.---/-.-/.-../--'), "A B C D E F G H I J K L M")  # All letters
        self.assertEqual(decrypt('-./---/.--./--.-/.-./.../-/../.--/.-./--./-.--/--..'), "N O P Q R S T U V W X Y Z")  # All letters contd.
        self.assertEqual(decrypt('- . ... - .. -. --./-- --- .-. ... ./-.-. --- -.. .'), "TESTING MORSE CODE")  # Mix of uppercase, lowercase, and spaces

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