# CHAPTER-1

## INTRODUCTION

This is the implementation of a morse code translator using Python. The translator makes use of a programming language as a key and a simple algorithm of changing characters to dots and dash and vice versa. This can be used in different areas like morse code for ships at sea to communicate over long distance using large lights. Morse code is mainly used in world war because it greatly improved the speed of communication. Naval ships were able to communicate with their bases and provide critical information to each other. War planes also used morse code to detail locations for enemy ships, bases, and troops and relay them back to headquarters. The rapid development of wireless technologies over the last decade has added great convenience to our lives. To accommodate different application requirements for wireless performance (e.g., range, throughput, reliability, timeliness, and energy), a wide range of wireless technologies have been proposed. Many of these technologies such as Wi-Fi and zig bee, operate in the various ways of the public, operational way of the. A possible solution is to obtain data by synthetic instead of natural means. Synthetic data are generated using computer algorithms instead of being collected from real-world scenarios. The public spectrum. where devices much compete with each other to resources. The uncoordinated competition among incompatible wireless technologies leads to considerable low reliability. To the best of our knowledge. Morse is the first work that leverages all existing traffic to achieve the technology communication between to Heterogeneous. The training stage is data-hungry and typically requires thousands of labelled examples. It is therefore often a challenge to obtain adequate amounts of high quality and accurate data required to sufficiently train a NN. The use of the hash maps is most predominant in python for accessing the data structure .so every key of the character has its value in the database we need to access through a function called itertools. Cipher stores the morse translated the form of the English string. Decipher stores the English translated form of the morse string. Context stores morse code of a single character. i keeps count of the spaces between morse characters. Message 'stores the string to be encoded or decoded. The revolutionary is knowledgeable of all of the special unwritten Morse code symbols for the standard Prosing’s for Morse code and the meanings of these special procedural signals in standard Morse code communications protocol.

# CHAPTER-2

## DESIGN FRAMEWORK

During this process of translation from the below diagram, we can say that when a character is given as input by translating it we get a morse code from the hashmaps as output. When Morse code is given as input then by translating it we get a character as output. The use of the hashmaps is most predominant in python for accessing the data structure .so every key of the character has its value in the database. Fig.2: System architecture Because we choose python as it is efficient in both fast and accessing a database. Cipher stores the morse translated the form of the English string. Decipher stores the English translated form of the morse string. Context stores morse code of a single character. It keeps count of the spaces between morse characters

.

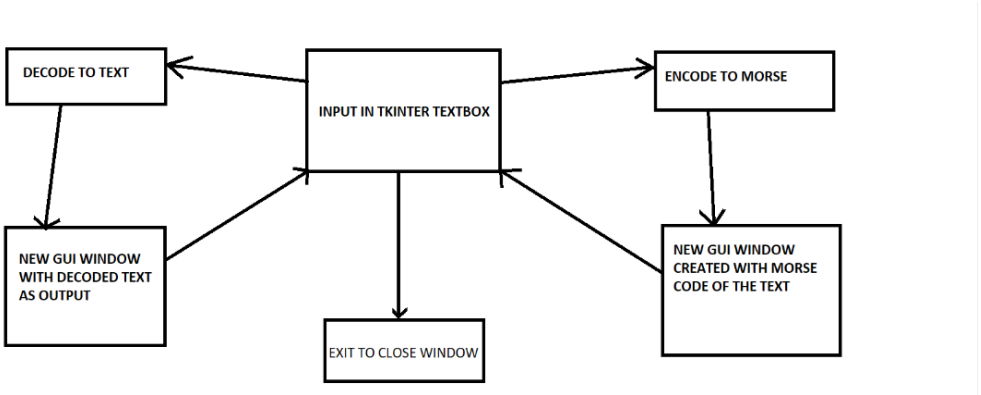


Fig. 2: System architecture

### 2.1 INTERNATIONAL MORSE CODE

- Short sign dot or dit (•): 1
- longer sign, dash or dah (–): 111
- Intra character gap between the dots and dashes within a character: 0
- The short interval between letters: 000
- The medium interval between words: 0000000
- One dash = Three dots
- The space between parts of the same letter = One dot

- The position between letters = Three dots
- The position between words = Seven dot

## 2.2 INTERNATIONAL MORSE CODE CHART

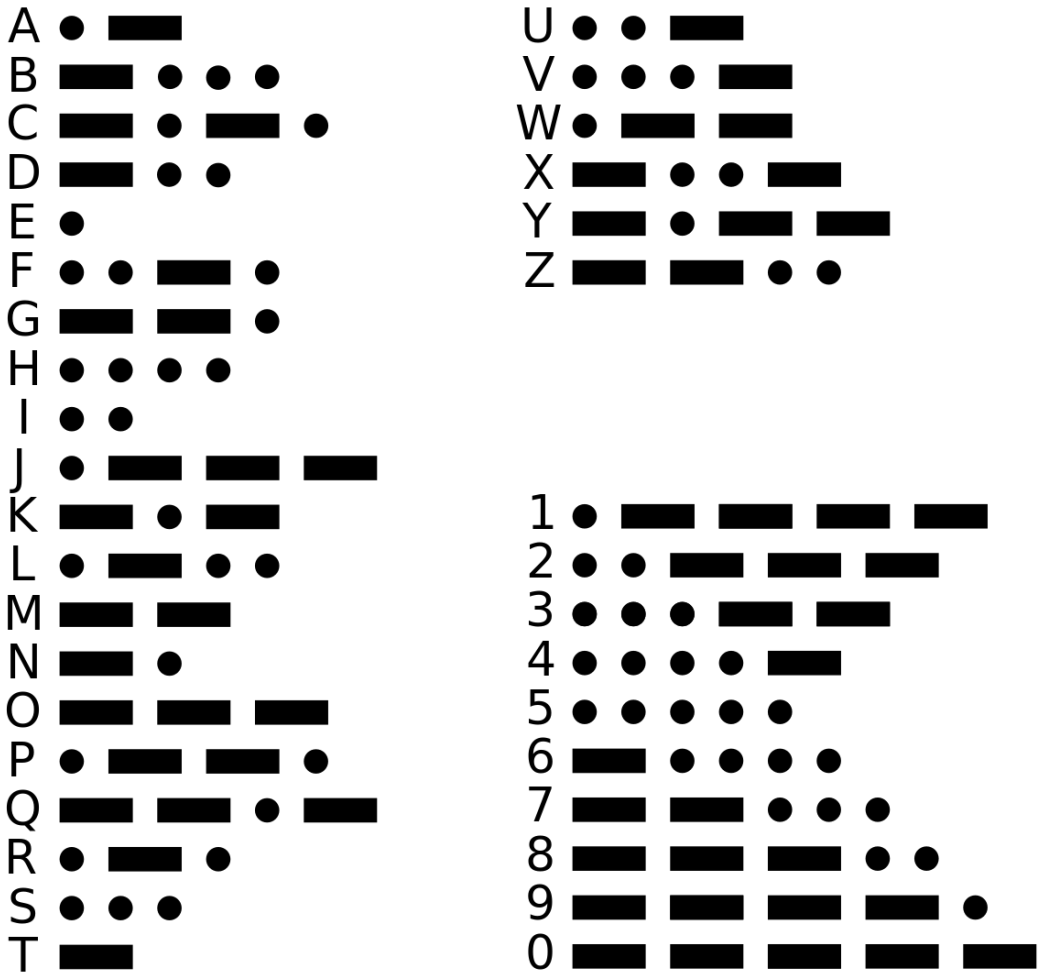


Fig. 4.1.3: Morse Code Chart

# CHAPTER-3

## METHODOLOGY

**3.1 ENCRYPTION**:- In case of encryption we extract each character (if not a space) from a word one at a time and match it with its corresponding morse code stored in whichever data structure we have chosen (if you are coding in python, dictionaries can turn out to be very useful in this case) Store the morse code in a variable which will contain our encoded string and then we add a space to our string which will contain the result. While encoding in morse code we need to add 1 space between every character and 2 consecutive spaces between every word. If the character is a space then add another space to the variable containing the result. We repeat this process until we traverse the whole string**.**

**3.2 DECRYPTION: -** In the case of decryption, we start by adding a space at the end of the string to be decoded (this will be explained later). Now we keep extracting characters from the string till we are not getting any space. As soon as we get a space we look up the corresponding English language character to the extracted sequence of characters (or our morse code) and add it to a variable which will store the result. Remember keeping track of the space is the most important part of this decryption process. As soon as we get 2 consecutive spaces, we will add another space to our variable containing the decoded string. The last space at the end of the string will help us identify the last sequence of Morse code characters (since space acts as a plaintext (English characters) take the place of keys and the ciphertext (Morse code) form the values of the corresponding keys. The values of keys can be accessed from the dictionary in the same way we access the values of an array through their index and vice versa

### 3.3 ALGORITHM FOR DECODING

- Creating a Dictionary in Python which contain Morse codes as Key and Decoded text as Values of respective keys
- Get Input from User as Morse Code and store it in a List after splitting it in list1
- Create another List as list2
- Use "For i in list1" append all the Elements in list2 as MORSE_DICT[i]

- Use. join(list2) to create a String and return it.

```
MORSE_DICT = { '.-':'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',  '.----':'1', '..---':'2', '...--':'3',   '....-':'4',
'.....':'5', '-....':'6',   '--...':'7', '---..':'8', '----.':'9',  '-----':'0', '--..--':', ', '.-.-.-':'.',  '..--..':'?', '-..-
.':'/', '-....-':'-',   '-.--.':'(', '-.--.-':')',":" "}
i = input()
list1 = []
list1= i.split(" ")
print(list1)
Input : .--- ..- .. -  ... --- .-.. .- -.
Output: ['.---', '..-', '..', '-', '', '...', '---', '.-..', '.-', '-.']
list2 = []
for i in list1:
    list2.append(MORSE_DICT[i])
str1 = ""
x = str1.join(list2)
print(x)
Output: JUIT SOLAN
```

### 3.4 ALGORITHM FOR ENCODING

- Get the Input from User and convert it to upper letters to make it compatible with Dictionary.
- Append Each letter in list1
- Using "Double For loop" to find key form Dictionary and Append it in another list2
- Join it with spaces to convert in string to form Morse code.

```
i = input()
```

```
list1= []
i = i.upper()
for j in i:
   list1.append(j)
list2= []
for i in list1:
   for key, value in MORSE_DICT.items():
      if i == value:
         list2.append(key)
print(list2)
str1 = " "
x = str1.join(list2)
print(x)
```

### 3.5 ALGORITHM FOR TKINTER

- Create an empty Tkinter GUI
- Add a Text Box and a Button in the GUI
- Create a command to run when button is pressed.
- Command creates a new GUI Window and Displays the result.

### 3.6 FLOW CHART

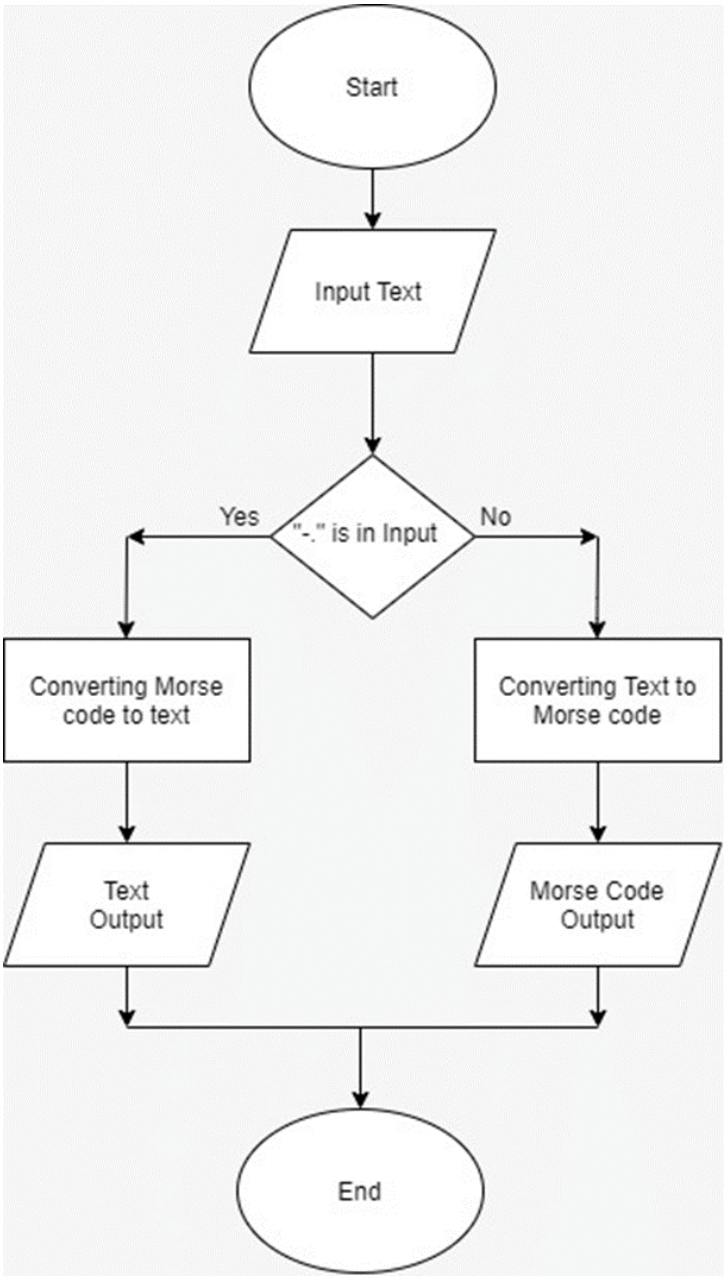


Fig. 3.6: Flow Chart

# CHAPTER 4

## CODE ANALYSIS

### 4.1 CODE: -

### 4.1.1) Morse Dictionary

```
MORSE_DICT = { '.-':'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',
                '.----':'1', '..---':'2', '...--':'3',
                '....-':'4', '.....':'5', '-....':'6',
                '--...':'7', '---..':'8', '----.':'9',
                '-----':'0', '--..--':', ', '.-.-.-':'.',
                '..--..':'?', '-..-.':'/', '-....-':'-',
                '-.--.':'(', '-.--.-':')',":" "}
```

### 4.1.2) Actual Code

```
from tkinter import *
#Decode to TExt Button fnx
def input():
    try:
        #Code to Convert Morse to Text
        i=textBox.get("1.0","end-1c")#excess value from root text box
        list1 = []
        list1 = i.split(" ")#Spliting Data in a list
        list2 = []
        for i in list1:
            list2.append(MORSE_DICT[i])
```

```
        str1 = ""
        x = str1.join(list2)
    except:
        #if input is not in Dict return same value
        x = textBox.get("1.0","end-1c")

    #create a new Window to present Output
    newWindow = Toplevel(root)
    newWindow.title("Output WIndow")
    newWindow.geometry("200x200")
    T = Text(newWindow, height=20, width=50)
    T.pack()
    T.insert(END, str(x))

#Encode to Morse Button fnx
def input2():
    try:
        #Code Text to Morse COde
        i = textBox.get("1.0","end-1c")#excess value from root text box
        list1 = []
        i = i.upper()#Convert to Upper case letters
        #Appending Each Letter in list
        for j in i:
            list1.append(j)
        list2 = []
        #using 2 for loops to find value to find value of the key from dict()
        for i in list1:
            for key, value in MORSE_DICT.items():
                if i == value:
                    list2.append(key)
        str1 = " "
        x = str1.join(list2)
    except:
        #if input is not in Dict return same value
```

```
        x = textBox.get("1.0","end-1c")

    #create a new Window to present Output
    newWindow2 = Toplevel(root)
    newWindow2.title("Output WIndow")
    newWindow2.geometry("200x200")
    T = Text(newWindow2, height=20, width=50)
    T.pack()
    T.insert(END, str(x))

root=Tk()#creating the main window
root.geometry("400x280")#defing  its size
root.title("INPUT Window")#Giving Title to Window
root.configure(bg='#add8e6')#Added Cyan color to backgrouund


textBox=Text(root, height=10, width=40)#Create text box to put value
textBox.pack()
buttonCommit=Button(root, height=2, width=15, text="Encode To Morse",command=lambda:
input2(),bg = 'red')#Encode to Morse Code Btn
buttonCommit.pack()
buttonCommit=Button(root, height=2, width=15, text="Decode To Text",command=lambda:
input(),bg = 'yellow')#Decode to Text Btn
buttonCommit.pack()
Label(root,text= "Created by Mukund and Shivam ",width=100,font=("Times",14)).pack()
mainloop()
```

# CHAPTER 5

# RESULT

Python provides a data structure called dictionary which stores information in the form of key-value pairs which is very convenient for implementing a cipher such as a Morse code. We can save the morse code chart in a dictionary where (key -value pairs) => (English character – morse code). The plaintext (English characters) takes the place of keys and the ciphertext (Morse code) form the values of the corresponding keys. The values of keys can be accessed from the dictionary in the same way we access the values of an array through their index and vice versa. We know that we have hashmaps in python which is a form of data used to find the frequency of characters in a string. () Basically a dictionary is like a list instead of integer index it can be of any type.to define a dictionary we use a key value pair with a colon between them iter () method returns an iterator for the given object .in case of sentinel provided, it returns the iterator object that calls the callable object until the sentinel character isn't found. We need to import the dictionary by calling the function MORSE_CODE_DICT (). We can use any of the python compilers like python 3.7(32bit). or any other online python compilers .so for fast output we can use online python compilers.

## 5.1 PROJECT SCREENSHOTS

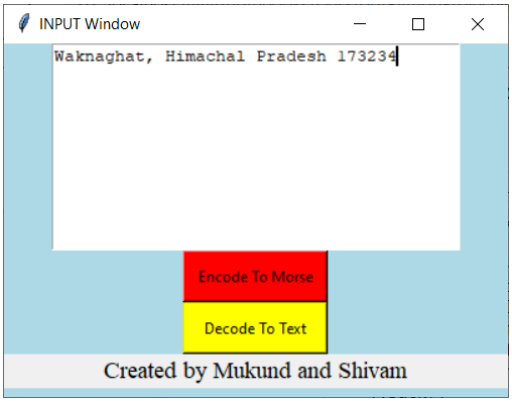


Fig. 5.1.1: Text to Morse Code (Text Input)

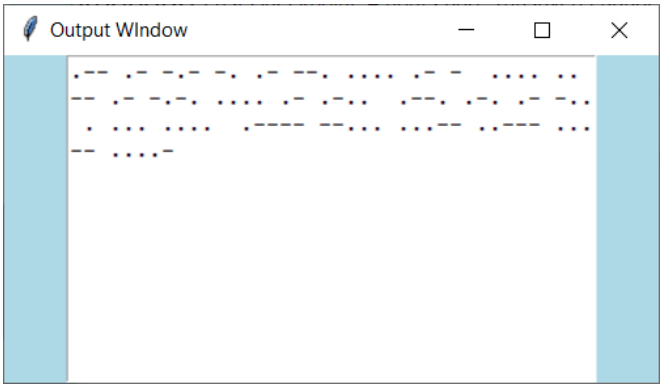


Fig. 5.1.2: Text to Morse Code (Morse Output)

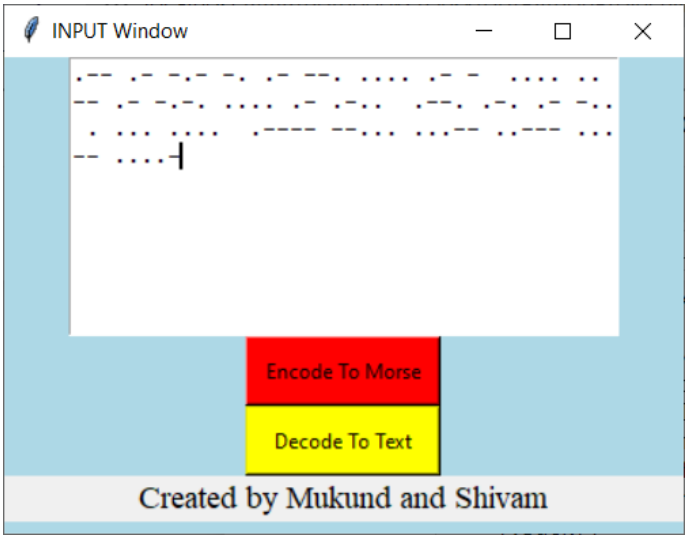


Fig. 5.1.3: Morse Code to Text (Morse input)

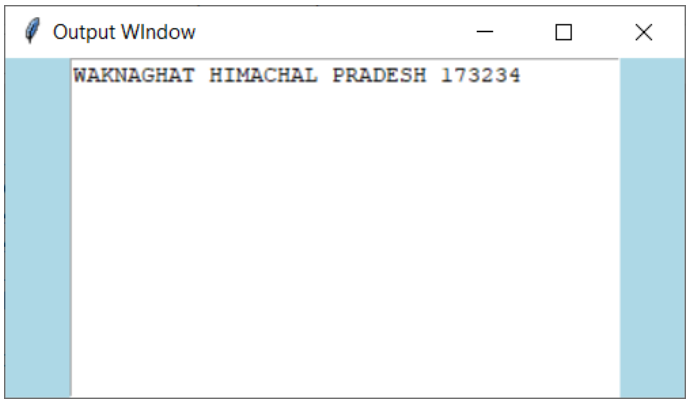


Fig. 5.1.4: Morse Code to Text (Text Output)