DEVELOPING A PROGRAM THAT CAN ENCRYPT CHARACTERS AND DECRYPT MORSE CODE USING CORE PYTHON PROGRAMMING

Sai Rohit P

NARAYANA E-TECHNO SCHOOL
PALLAVARAM, CHENNAI – 600043

PROJECT REPORT DONE ON

DEVELOPING A PROGRAM THAT CAN ENCRYPT CHARACTERS AND DECRYPT MORSE CODE USING CORE PYTHON PROGRAMMING

THE DEPARTMENT OF COMPUTER SCIENCE
FOR THE YEAR 2020 – 21

SUBMITTED TO	**:**	**MR MANIVANNAN**
SUBMITTED BY	**:**	**SAI ROHIT P**
SUBJECT	**:**	**COMPUTER SCIENCE (083)**

NARAYANA E-TECHNO SCHOOL

CERTIFICATE

This is to certify that **Sai Rohit P** of class **XII - IIT A** of **Narayana E-techno School** has completed his project entitled **"Developing A Program That Can Encrypt Characters And Decrypt Morse Code Using Core Python Programming"** under the guidance of the subject teacher **Mr Manivannan** during the academic year 2020 – 21 in the partial fulfilment of the Computer Science practical examination in accordance to the laid down regulations by the **CBSE Board.**

Internal examiner Principal External examiner

ACKNOWLEDGEMENT

It is with pleasure that I acknowledge my sincere gratitude toward our teacher, Mr Manivannan, A guide, Mentor all the above a friend, who critically reviewed my project and helped in solving each and every problem that occurred during the implementation of the project.

I am specially indebted to our principal Ms Uma Venkatesh who has always been a source of encouragement and support and without whose inspiration this project wouldn't have been successful. I would like to place on record my heartfelt thanks and wishes toward him.

I would also thank my parents for their constant support and motivation they offered to keep me moving on with my studies and making this project possible amongst all. They are, and will remain to, stay an integral part of my life.

Finally, I would like to emphasize my sincere appreciation to all the other students in my batch who have made the experience all together more delightful and for their friendship, not to mention all the fine times we have shared together during the making of this project.

Sai Rohit P
Class XII – IIT A

INDEX

DEVELOPING A PROGRAM THAT CAN ENCRYPT CHARACTERS AND DECRYPT MORSE CODE USING CORE PYTHON PROGRAMMING

INTRODUCTION

Python is a very powerful object oriented programming language that can handle multiple tasks in raw and process them in a very easy and a simple way. It's user friendly and also highly scalable. Using this to my advantage, I have designed and developed a program that can translate back and forth between the entered morse code or the common english words and letters to yield an output that can be understood in the desired language selection apart from the one entered.

This main objective of the program is to be able to collect input from the user, asking him which language he / she would like to enter in, and the program automatically then converts it into the other language and displays the translated message on the screen. The process involved in converting from english to morse code will, from here on, be termed as encryption and for the process involved in converting morse code to human understandable english alphabets to be termed as decryption.

Morse code is a method of transmitting text information as a series of on-off tones, lights, or clicks that can be directly understood by a skilled listener or observer without special equipment. It has been named after Samuel F. B. Morse, an inventor of the telegraph.

OBJECTIVES OF THE PROJECT

The objective of this project is to allow ourselves to apply the programming knowledge into a real-world situation/problem and get exposed to how programming skills help in developing good software. They mainly include:

1. Writing programs & utilizing modern software tools.
2. Apply object oriented programming principles effectively when developing small to medium sized projects.
3. Write effective procedural code to solve small to medium sized problems.
4. Will allow us to demonstrate a breadth of knowledge in computer science, as exemplified in the areas of systems, theory and software development, and also to
5. Demonstrate the ability to conduct a research or applied Computer Science project, requiring writing and presentation skills which exemplify scholarly style in computer science.

Algorithm Involved

The algorithm is very simple. Every character in the English language is substituted by a series of 'dots' and 'dashes' or sometimes just singular 'dot' or 'dash' and vice versa.

Concept of Encryption

1. 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)
2. 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.
3. While encoding in morse code we need to add 1 space between every character and 2 consecutive spaces between every word.
4. If the character is a space then add another space to the variable containing the result. We repeat this process till we traverse the whole string

Concept of Decryption

1. In case of decryption, we start by adding a space at the end of the string to be decoded (this will be explained later).
2. Now we keep extracting characters from the string till we are not getting any space.
3. 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.

4. 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.
5. The last space at the end of the string will help us identify the last sequence of morse code characters (since a space acts as a check for extracting characters and start decoding them).

Implementation

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 the morse code. We can save the morse code chart in a dictionary where (key-value pairs) => (English Characters-Morse Code). The 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.

Proposed System

Today one cannot afford to rely on the fallible human beings of be really wants to stand against today's merciless competition where not to wise saying "to err is human" no longer valid, it's outdated to rationalize your mistake. So, to keep pace with time, to bring about the best result without malfunctioning and greater efficiency so to replace the unending heaps of flies with a much sophisticated hard disk of the computer.

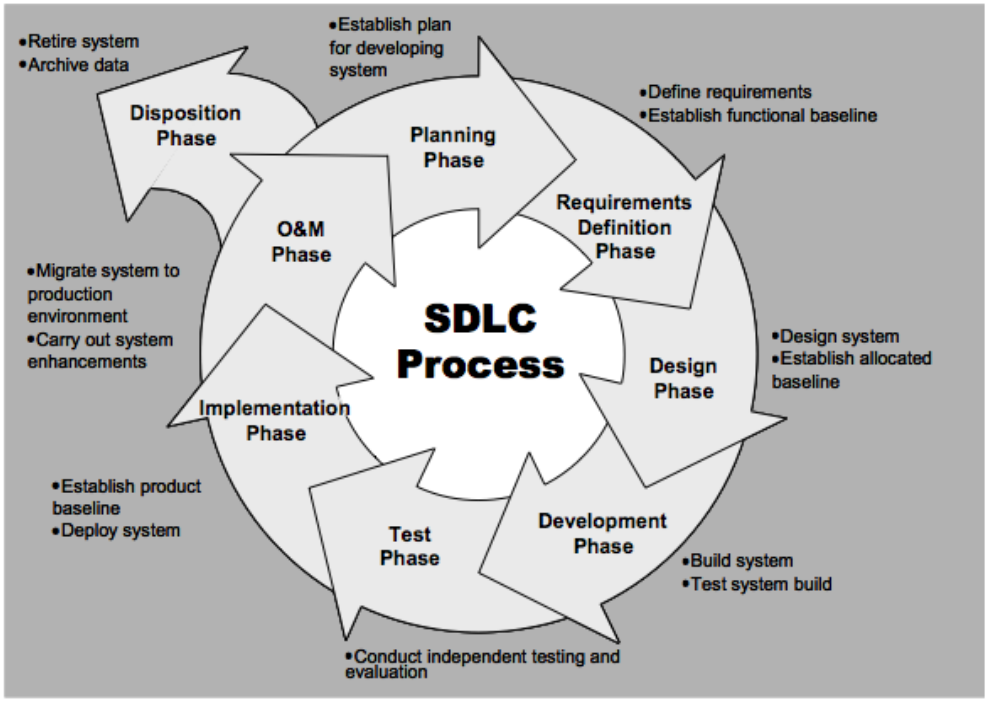
One has to use the data management software. Software has been an ascent in atomization various organisations. Many software products working are now in markets, which have helped in making the organizations work easier and efficiently. Data management initially had to maintain a lot of ledgers and a lot of paperwork has to be done but now software production this organization has made their work faster and easier. Now only this software has to be loaded on the computer and work can be done.

This prevents a lot of time and money. The work becomes fully automated and any information regarding the organization can be obtained by clicking the button. Moreover, now it's an age of computers of and automating such an organization gives the better look.

Scope

The purpose of this document is to summarize the design and operation of the Morse Code Translator. Requirements, dependencies, design, implementation, and testing of the Morse Code Translator are included in this document.

SYSTEM DEVELOPMENT LIFE CYCLE (SDLC)



The systems development life cycle is a project management technique that divides complex projects into smaller, more easily managed segments or phases. Segmenting projects allows managers to verify the successful completion of project phases before allocating resources to subsequent phases.

Software development projects typically include initiation, planning, design, development, testing, implementation, and maintenance phases. However, the phases may be divided differently depending on the organization involved.

For example, initial project activities might be designated as request, requirements-definition, and planning phases, or initiation, concept-development, and planning phases. End users of the system under development should be involved in reviewing the output of each phase to ensure the system is being built to deliver the needed functionality.

PHASES INVOLVED IN SDCL

Initiation Phase

The Initiation Phase begins when a business sponsor identifies a need or an opportunity.

The purpose of the Initiation Phase is to:

- Identify and validate an opportunity to improve business accomplishments of the organization or a deficiency related to a business need.
- Identify significant assumptions and constraints on solutions to that need.
- Recommend the exploration of alternative concepts and methods to satisfy the need including questioning the need for technology, i.e., will a change in the business process offer a solution?
- Assure executive business and executive technical sponsorship. The Sponsor designates a Project Manager and the business need is documented in a Concept Proposal. The Concept Proposal includes information about the business process and the relationship to the Agency/Organization.
- Infrastructure and the Strategic Plan. A successful Concept Proposal results in a Project Management Charter which outlines the authority of the project manager to begin the project.

Careful oversight is required to ensure projects support strategic business objectives and resources are effectively implemented into an organization's enterprise architecture. The initiation phase begins when an opportunity to add, improve, or correct a system is identified and formally requested through the presentation of a business case.

The business case should, at a minimum, describe a proposal's purpose, identify expected benefits, and explain how the proposed system supports one of the organization's business strategies. The business case should also identify alternative solutions and detail as many informational, functional, and network requirements as possible.

System Concept Development Phase

The System Concept Development Phase begins after a business need or opportunity is validated by the Agency/Organization Program Leadership and the Agency/Organization CIO.

The purpose of the System Concept Development Phase is to:

- Determine the feasibility and appropriateness of the alternatives.
- Identify system interfaces.
- Identify basic functional and data requirements to satisfy the business need.
- Establish system boundaries; identify goals, objectives, critical success factors, and performance measures.

- Evaluate costs and benefits of alternative approaches to satisfy the basic functional requirements
- Assess project risks
- Identify and initiate risk mitigation actions, and Develop high-level technical architecture, process models, data models, and a concept of operations. This phase explores potential technical solutions within the context of the business need.
- It may include several trade-off decisions such as the decision to use COTS software products as opposed to developing custom software or reusing software components, or the decision to use an incremental delivery versus a complete, onetime deployment.
- Construction of executable prototypes is encouraged to evaluate technology to support the business process. The System Boundary Document serves as an important reference document to support the Information Technology Project Request (ITPR) process.
- The ITPR must be approved by the State CIO before the project can move forward.

Planning Phase

The planning phase is the most critical step in completing development, acquisition, and maintenance projects. Careful planning, particularly in the early stages of a project, is necessary to coordinate activities and manage project risks effectively. The depth and formality of project plans should be commensurate with the characteristics and risks of a given project. Project plans refine the information gathered during the initiation phase by further identifying the specific activities and resources required to complete a project.

A critical part of a project manager's job is to coordinate discussions between user, audit, security, design, development, and network personnel to identify and document as many functional, security, and network requirements as possible. During this phase, a plan is developed that documents the approach to be used and includes a discussion of methods, tools, tasks, resources, project schedules, and user input. Personnel assignments, costs, project schedule, and target dates are established.

A Project Management Plan is created with components related to acquisition planning, configuration management planning, quality assurance planning, concept of operations, system security, verification and validation, and systems engineering management planning.

Requirements & Analysis Phase

This phase formally defines the detailed functional user requirements using high-level requirements identified in the Initiation, System Concept, and Planning phases. It also delineates the requirements in terms of data, system performance, security, and maintainability requirements for the system. The requirements are defined in this phase to a level of detail sufficient for systems design to proceed. They need to be measurable, testable, and relate to the business need or opportunity identified in the Initiation Phase. The requirements that will be used to determine acceptance of the system are captured in the Test and Evaluation Masterplan.

The purposes of this phase are to:

- Further define and refine the functional and data requirements and document them in the Requirements Document,
- Complete business process reengineering of the functions to be supported (i.e., verify what information drives the business process, what information is generated, who generates it, where does the information go, and who processes it),
- Develop detailed data and process models (system inputs, outputs, and the process).
- Develop the test and evaluation requirements that will be used to determine acceptable system performance.

Design Phase

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to script programs during the development phase. Program designs a reconstructed in various ways. Using a top-down approach, designers first identify and link major program components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections.

Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections. Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design.

Audit, security, and quality assurance personnel should be involved in the review and approval process. During this phase, the system is designed to satisfy the functional requirements identified in the previous phase. Since problems in the design phase could be very expensive to solve in the later stage of the software development, a variety of elements are considered in the design to mitigate risk. These include:

- Identifying potential risks and defining mitigating design features.
- Performing a security risk assessment.
- Developing a conversion plan to migrate current data to the new system.
- Determining the operating environment.
- Defining major subsystems and their inputs and outputs.
- Allocating processes to resources.
- Preparing detailed logic specifications for each software module. The result is a draft System Design Document which captures the preliminary design for the system.

- Everything requiring user input or approval is documented and reviewed by the user. Once these documents have been approved by the Agency CIO and Business Sponsor, the final System Design Document is created to serve as the Critical/Detailed Design for the system.
- This document receives a rigorous review by Agency technical and functional representatives to ensure that it satisfies the business requirements. Concurrent with the development of the system design, the Agency Project Manager begins development of the Implementation Plan, Operations and Maintenance Manual, and the Training Plan.

Development Phase

The development phase involves converting design specifications into executable programs. Effective development standards include requirements that programmers and other project participants discuss design specifications before programming begins. The procedures help ensure programmers clearly understand program designs and functional requirements. Programmers use various techniques to develop computer programs.

The large transaction oriented programs associated with financial institutions have traditionally been developed using procedural programming techniques. Procedural programming involves the line-by-line scripting of logical instructions that are combined to form a program. Effective completion of the previous stages is a key factor in the success of the Development phase.

The Development phase consists of:

- Translating the detailed requirements and design into system components.
- Testing individual elements (units) for usability.
- Preparing for integration and testing of the IT system.

Integration And Test Phase

Subsystem integration, system, security, and user acceptance testing is conducted during the integration and test phase. The user, with those responsible for quality assurance, validates that the functional requirements, as defined in the functional requirements document, are satisfied by the developed or modified system. OIT Security staff assesses the system security and issue a security certification and accreditation prior to installation/implementation.

Multiple Levels Of Testing Are Performed, Including:

- Testing at the development facility by the contractor and possibly supported by end users
- Testing as a deployed system with end users working together with contract personnel
- Operational testing by the end user alone performing all functions. Requirements are traced throughout testing, a final Independent Verification & Validation evaluation is performed and all documentation is reviewed and accepted prior to acceptance of the system.

Implementation Phase

This phase is initiated after the system has been tested and accepted by the user. In this phase, the system is installed to support the intended business functions. System performance is compared to performance objectives established during the planning phase.

Implementation includes user notification, user training, installation of hardware, installation of software onto production computers, and integration of the system into daily work processes. This phase continues until the system is operating in production in accordance with the defined user requirements.

Operations And Maintenance Phase

The system operation is on-going. The system is monitored for continued performance in accordance with user requirements and needed system modifications are incorporated. Operations continue as long as the system can be effectively adapted to respond to the organization's needs. When modifications or changes are identified, the system may re-enter the planning phase.

The purpose of this phase is to:

- Operate, maintain, and enhance the system.
- Certify that the system can process sensitive information.
- Conduct periodic assessments of the system to ensure the functional requirements continue to be satisfied.
- Determine when the system needs to be modernized, replaced, or retired.

SOURCE CODE

The code used in the skeleton of the program has been attached below

```
# Python program to implement Morse Code Encryption and Decryption
'''
VARIABLE KEY
'cipher' -> 'stores the morse translated form of the English string'
'decipher' -> 'stores the English translated form of the morse string'
'citext' -> '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'
'''
# Dictionary representing the morse code chart
MORSE_CODE_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': '-----', ', ': '--..--', '.': '.-.-.-',
                   '?': '..--..', '/': '-..-.', '-': '-....-',
                   '(': '-.--.', ')': '-.--.-'}
# Function to encrypt the string
# according to the morse code chart
def encrypt(message):
    cipher = ''
    for letter in message:
        if letter != ' ':
            # Looks up the dictionary and adds the
            # correspponding morse code
            # along with a space to separate
            # morse codes for different characters
            cipher += MORSE_CODE_DICT[letter] + ' '
        else:
            # 1 space indicates different characters
            # and 2 indicates different words
            cipher += ' '
```

```
    return cipher
# Function to decrypt the string
# from morse to english
def decrypt(message):
    # extra space added at the end to access the
    # last morse code
    message += ' '
    decipher = ''
    citext = ''
    for letter in message:
        if (letter != ' '):
            # counter to keep track of space
            i = 0
            # storing morse code of a single character
            citext += letter
            # in case of space
        else:
            # if i = 1 that indicates a new character
            i += 1
            # if i = 2 that indicates a new word
            if i == 2:
                # adding space to separate words
                decipher += ' '
            else:
                # accessing the keys using their values
                decipher +=
list(MORSE_CODE_DICT.keys())[list(MORSE_CODE_DICT.values()).index(citext)]
                citext = ''
    return decipher
def reAsk():
    ask = input("\nDo you want to continue again? (y/n): ")
    if ask == "y":
        menu()
    elif ask == "n":
        print("Thank you & a good day!")
    else:
        print("That's an oops. Try again by giving a valid input.\n")
        reAsk()
def menu():
    print("Welcome To EncryptDecrypt")
    print("Select A Facility\n1.Encrypt A Sentence\n2.Decrypt Morse")
    choice = input("Enter a choice: ")
    if choice == "1":
        message = input("Enter the sentence you want to encrypt: ")
        print("Your message has been encrypted.")
```

```
        print(encrypt(message.upper()))
        reAsk()
    elif choice == "2":
        message = input("Enter the morse you want to decrypt: ")
        print("Your message has been decrypted.")
        print(decrypt(message.upper()))
        reAsk()
    else:
        print("That's an oops. Try again by giving a valid input.\n")
        menu()

menu()
```

OUTPUT

Given below is a sample of the output thus obtained upon the execution of the code above.

```
Welcome To EncryptDecrypt
Select A Facility
1.Encrypt A Sentence
2.Decrypt Morse
Enter a choice: 1
Enter the sentence you want to encrypt: Hello please encrypt me
Your message has been encrypted.
.... . .-.. .-.. ---  .--. .-.. . .- ... .  . -. -.-. .-. -.-- .--. -  -- .

Do you want to continue again? (y/n): y

Welcome To EncryptDecrypt
Select A Facility
1.Encrypt A Sentence
2.Decrypt Morse
Enter a choice: 2
Enter the morse you want to decrypt: .... . .-.. .-.. ---  .--. .-.. . .- ... .  . -. -.-. .-.
-.-- .--. -  -- .
Your message has been decrypted.
HELLO PLEASE ENCRYPT ME

Do you want to continue again? (y/n): n
Thank you & a good day!

Process finished with exit code 0
```

Thus it has been verified that the desired output is being obtained in compliance to the aim of this project. The functions, code and the __main__ execution is happening seamlessly without any errors.

SCREENSHOTS

Here I attach screenshots of the original work as done and viewed in the IDE. They are in order, corresponding to the above source code.

Source Code Screenshots

```
# Python program to implement Morse Code Encryption and Decryption

'''
VARIABLE KEY
'cipher' -> 'stores the morse translated form of the english string'
'decipher' -> 'stores the english translated form of the morse string'
'citext' -> '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'
'''

# Dictionary representing the morse code chart
MORSE_CODE_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': '-----', ', ': '--..--', '.': '.-.-.-',
                   '?': '..--..', '/': '-..-.', '-': '-....-',
                   '(': '-.--.', ')': '-.--.-'}
```

```
# Function to encrypt the string
# according to the morse code chart
def encrypt(message):
    cipher = ''
    for letter in message:
        if letter != ' ':
            # Looks up the dictionary and adds the
            # corresponding morse code
            # along with a space to separate
            # morse codes for different characters
            cipher += MORSE_CODE_DICT[letter] + ' '
        else:
            # 1 space indicates different characters
            # and 2 indicates different words
            cipher += ' '
    return cipher
```

```
# Function to decrypt the string
# from morse to english
def decrypt(message):
    # extra space added at the end to access the
    # last morse code
    message += ' '
    decipher = ''
    citext = ''
    for letter in message:
        if (letter != ' '):
            # counter to keep track of space
            i = 0
            # storing morse code of a single character
            citext += letter
            # in case of space
        else:
            # if i = 1 that indicates a new character
            i += 1
            # if i = 2 that indicates a new word
            if i == 2:
                # adding space to separate words
                decipher += ' '
            else:
                # accessing the keys using their values (reverse of encryption)
                decipher += list(MORSE_CODE_DICT.keys())[list(MORSE_CODE_DICT.values()).index(citext)]
                citext = ''
    return decipher
```

```
def reAsk():
    ask = input("\nDo you want to continue again? (y/n): ")
    if ask == "y":
        menu()
    elif ask == "n":
        print("Thank you & a good day!")
    else:
        print("That's an oops. Try again by giving a valid input.\n")
        reAsk()
```

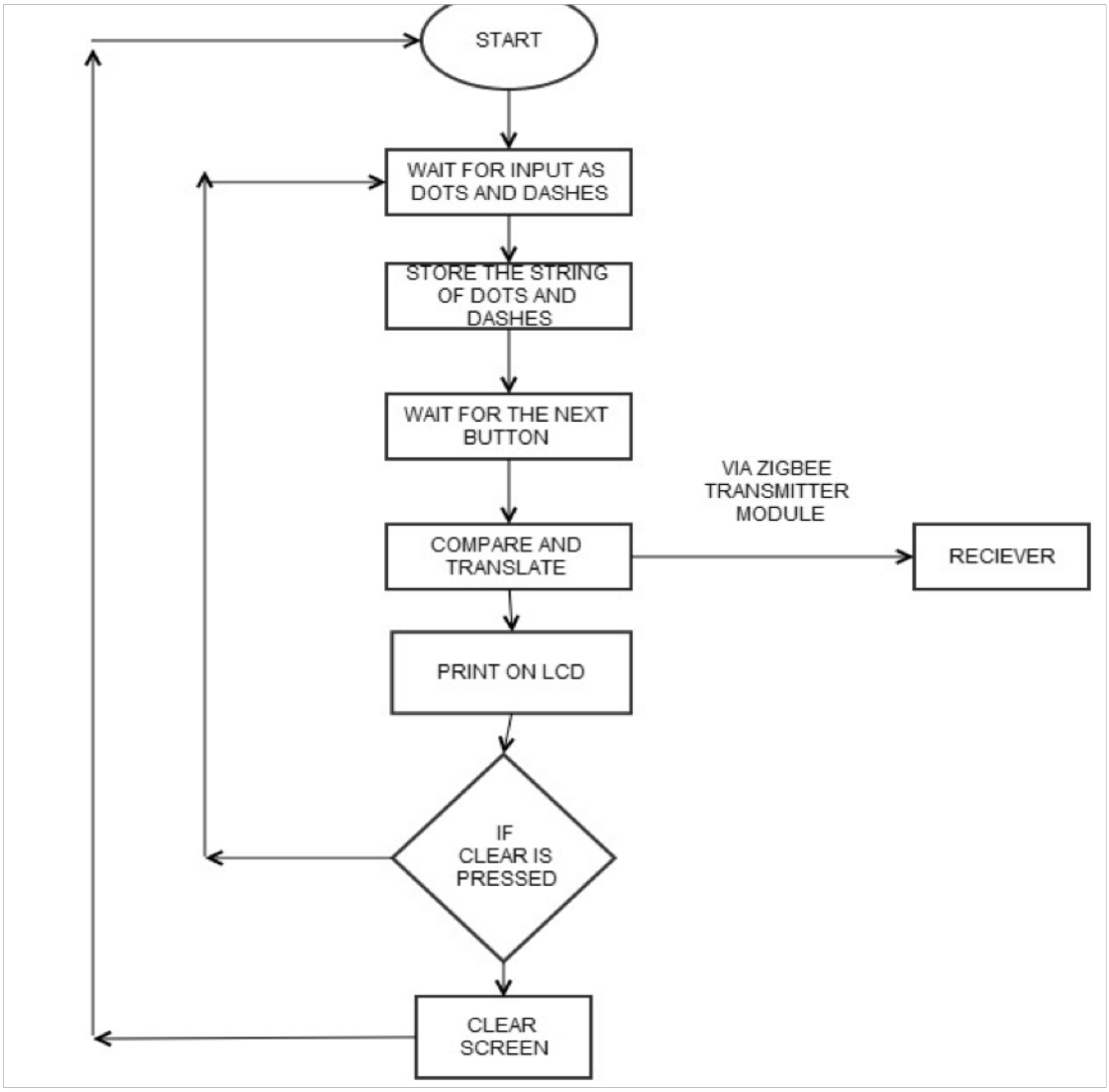
```
def menu():
    print("Welcome To EncryptDecrypt")
    print("Select A Facility\n1.Encrypt A Sentence\n2.Decrypt Morse")
    choice = input("Enter a choice: ")
    if choice == "1":
        message = input("Enter the sentence you want to encrypt: ")
        print("Your message has been encrypted.")
        print(encrypt(message.upper()))
        reAsk()
    elif choice == "2":
        message = input("Enter the morse you want to decrypt: ")
        print("Your message has been decrypted.")
        print(decrypt(message.upper()))
        reAsk()
    else:
        print("That's an oops. Try again by giving a valid input.\n")
        menu()

menu()
```

Output Screenshot

```
Welcome To EncryptDecrypt
Select A Facility
1.Encrypt A Sentence
2.Decrypt Morse
Enter a choice: 1
Enter the sentence you want to encrypt: Hello please encrypt me
Your message has been encrypted.
.... . .-.. .-.. ---  .--. .-.. . .- ... .  . -. -.-. .-. -.-- .--. -  -- .

Do you want to continue again? (y/n): y


Welcome To EncryptDecrypt
Select A Facility
1.Encrypt A Sentence
2.Decrypt Morse
Enter a choice: 2
Enter the morse you want to decrypt: .... . .-.. .-.. ---  .--. .-.. . .- ... .  . -. -.-. .-. -.-- .--. -  -- .
Your message has been decrypted.
HELLO PLEASE ENCRYPT ME

Do you want to continue again? (y/n): n
Thank you & a good day!

Process finished with exit code 0
```

FLOWCHARTS

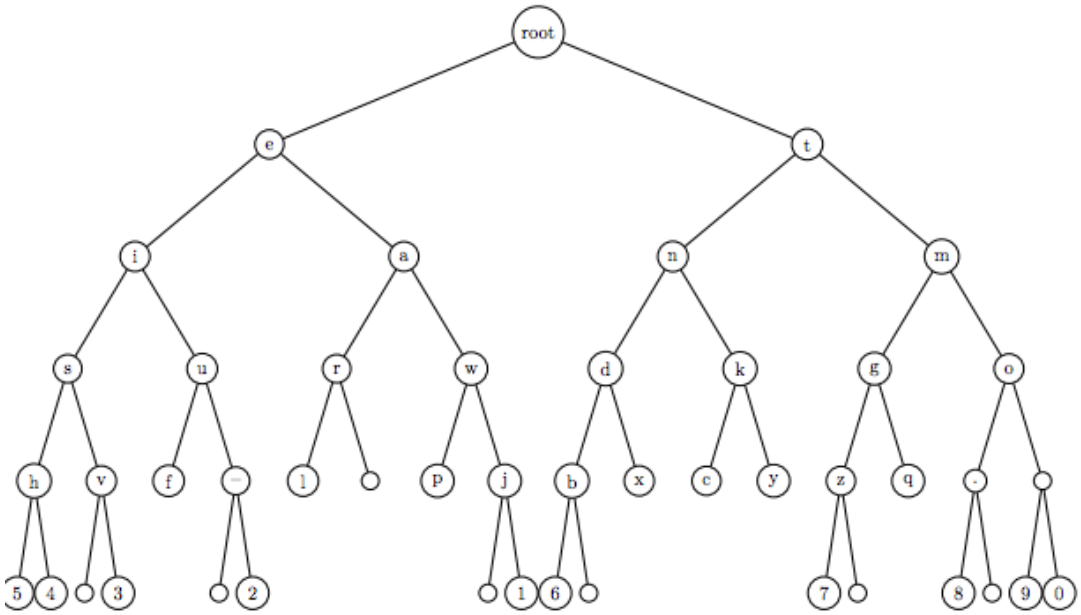
Program Highlight



Morse Lookup

A ●−	J ●−−−	S ●●●
B −●●●	K −●−	T −
C −●−●	L ●−●●	U ●●−
D −●●	M−−	V ●●●−
E ●	N −●	W ●−−
F ●●−●	O −−−	X −●●−
G −−●	P●−−●	Y−●−−
H●●●●	Q−−●−	Z −−●●
I●●	R ●−●	

Binary Tree for Morse Code



TESTING

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test[1] , with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs.

It can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, so that it works as expected and can be implemented with the same characteristics. Software Testing, depending on the testing method employed, can be implemented at any time in the development process, however the most test effort is employed after the requirements have been defined and coding process has been completed.

TESTING METHODS

Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

Black Box Testing

Black box testing treats the software as a "black box," without any knowledge of internal implementation. Black box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, fuzz testing, model-based testing, traceability matrix, exploratory testing and specification-based testing.

Specification - Based Testing

Specification-based testing aims to test the functionality of software according to the applicable requirements. Thus, the tester inputs data into, and only sees the output from, the test object. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behaviour), either "is" or "is not" the same as the expected value specified in the test case. Specification-based testing is necessary, but it is insufficient to guard against certain risks

Advantages And Disadvantages

The black box tester has no "bonds" with the code, and a tester's perception is very simple: a code must have bugs. Using the principle, "Ask and you shall receive," black box testers find bugs where programmers don't. But, on the other hand, black box testing has been said to be "like a walk in a dark labyrinth without a flashlight," because the tester doesn't know how the software being tested was actually constructed.

That's why there are situations when

- a black box tester writes many test cases to check something that can be tested by only one test case, and / or
- some parts of the back end are not tested at all.

Therefore, black box testing has the advantage of "an unaffiliated opinion," on the one hand, and the disadvantage of "blind exploring," on the other.

White Box Testing

White box testing, by contrast to black box testing, is when the tester has access to the internal data structures and algorithms (and the code that implement these)

Types of White Box Testing:-

The following types of white box testing exist:

- API testing - Testing of the application using Public and Private APIs.
- Code coverage - creating tests to satisfy some criteria of code coverage.

For example, the test designer can create tests to cause all statements in the program to be executed at least once.

- fault injection methods.
- mutation testing methods.
- static testing - White box testing includes all static testing.

CODE COMPLETENESS EVALUATION

White box testing methods can also be used to evaluate the completeness of a test suite that was created with black box testing methods. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested.

Two Common Forms Of Code Coverage Are:

- Function Coverage: Which reports on functions executed and
- Statement Coverage: Which reports on the number of lines executed to complete the test.

They both return coverage metric, measured as a percentage

CONCLUSION

The test results demonstrate the functionality of the Morse Code Translator. The Morse Code Translator successfully outputs all alphanumeric characters, as shown. The Morse Code Translator alerts the user when it is initialized by outputting a space. The test demonstrated a correct input debounce and the output was as expected. Referenced the video accompanying this document for verification of the speaker's buzzing when the button is pressed.

I have also attached photos of a successful output IDE screen of PyCharm The Morse Code Translator met each requirement. The wide error range available to the user makes input simple, even for one learning Morse code. The binary tree in the code allows the Morse Code Translator to move quickly and efficiently through each character in the tree. The Morse Code Translator is a powerful and simple tool that performs everything it was designed to do.

SYSTEM REQUIREMENTS

Hardware Requirements

1.	Operating System	:	macOS or Windows (7 and above)
2.	Processor	:	Pentium (any) or AMD Athalon (3800+ to 4200+ dual core)
3.	RAM	:	512mb+
4.	Hard Disk	:	SATA or SSD type, with 40 GB or above
5.	Network	:	A good Wi-Fi for installation of packages

Software Requirements:

1. Latest version of Python & it's packages.
2. An IDE, Preferably PyCharm.

BIBLIOGRAPHY

1. **Computer with Python:** Sumita Aurora, Class XII CBSE Textbook
2. **The Most Insightful Stories About Python:** Nik Piepenbreier, medium.com
3. **A View to an Interpreter:** Arnet Jansen, The Daily Catalogue
4. **Encryption & Decryption Technology:** Courtesy of wikipedia.com
5. **Mobile Networking Technology:** Courtesy of forbes.com
6. **Morse Code, Samuel Morse & The Telegraph:** Courtesy of wikipedia.com
7. **About Python:** Courtesy of geeksforgeeks.org, An article by Palash Nigam
8. **File Editing Tool:** by convertcase.net