

1 Instruction

This program contains 4 files(.py) named as Task1.py, Task2.py, Task3.py and Task4.py. Each python file gives answer to one single task. You can have a clear view of the 4 files by following steps:

- 1) Open .py files from Task1 to Task4 one by one.
- 2) Pick a single file that you wish to check first.
- 3) Click on 'run'.
- 4) Read the Morse Code dictionary or import your own Morse Sequences as suggested.
- 5) Check answers.

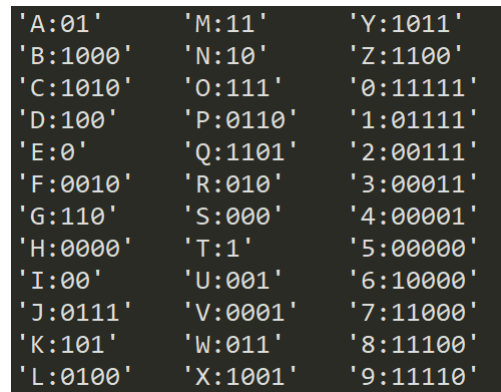
It is necessary to mention that from Task2 to Task 4, you are allowed to input multiple Morse Code sequences not until you would like to terminate (by input nothing and clicking 'Enter'). Also, each sequence is performed after input. This means every time you type your sequence and click on 'Enter', the program gives you the answer of that sequence and ask you to type the next one, until you type nothing and click on 'Enter' directly.

2 Assumption and Description

In order to further illustrate how to run my programs, here are some assumptions and descriptions given by tasks.

2.1 Task 1

In this task, I made a dictionary to store Morse Code table. The keys of the dictionary are the Morse Code in binary digits format. And the values of the keys are the standard 26 letters and the 10 numerals. Then I used repr() in print() to make the output dictionary readable.



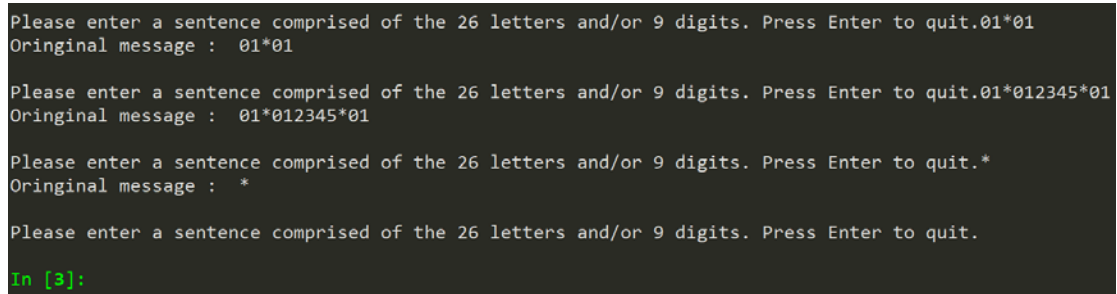
'A:01'	'M:11'	'Y:1011'
'B:1000'	'N:10'	'Z:1100'
'C:1010'	'O:111'	'0:11111'
'D:100'	'P:0110'	'1:01111'
'E:0'	'Q:1101'	'2:00111'
'F:0010'	'R:010'	'3:00011'
'G:110'	'S:000'	'4:00001'
'H:0000'	'T:1'	'5:00000'
'I:00'	'U:001'	'6:10000'
'J:0111'	'V:0001'	'7:11000'
'K:101'	'W:011'	'8:11100'
'L:0100'	'X:1001'	'9:11110'

FIGURE 2.1 Output of Task1

2.2 Task 2

Assumption: If the user would like to terminate, type nothing and press 'Enter'.

In this task, I designed a while loop for user to input multiple Morse Code sequences. If the user wish to terminate, press 'Enter' then the program will stop automatically. According to the assignment requirements, the program prints every input originally.



```
Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*01
Original message : 01*01

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*012345*01
Original message : 01*012345*01

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.*
Original message : *

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.

In [3]:
```

FIGURE 2.2 Example and Output of Task 2

2.3 Task 3

Assumption: Each ‘Enter’ indicates a sequence, characters between two ‘*’ means subsequence.

Extending from task 2, Task3.py displays an error message indicating which subsequence is invalid and decodes the valid subsequences into the corresponding characters.

Especially, if the input contains chain of ‘*’, the subsequence between two ‘*’ is nothing thus there will be no error message printed.

```
Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*01
Original message : 01*01
Translated message : AA

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*012345*01
Original message : 01*012345*01
012345 is invalid!
Translated message : AA

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*****01
Original message : 01*****01
Translated message : AA

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.****
Original message : ****

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.

In [5]: |
```

FIGURE 2.3 Example and Output of Task 3

2.4 Task 4

Extending from task 2 and task 3, Task4.py find out number of the occurrences for each of the characters and print them in a readable format.

```
Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*01
Original message : 01*01
Translated message : AA
Occurrence for each character :
'A:2' 'M:0' 'Y:0'
'B:0' 'N:0' 'Z:0'
'C:0' 'O:0' '0:0'
'D:0' 'P:0' '1:0'
'E:0' 'Q:0' '2:0'
'F:0' 'R:0' '3:0'
'G:0' 'S:0' '4:0'
'H:0' 'T:0' '5:0'
'I:0' 'U:0' '6:0'
'J:0' 'V:0' '7:0'
'K:0' 'W:0' '8:0'
'L:0' 'X:0' '9:0'

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.01*012345
Original message : 01*012345
012345 is invalid!
Translated message : A
Occurrence for each character :
'A:1' 'M:0' 'Y:0'
'B:0' 'N:0' 'Z:0'
'C:0' 'O:0' '0:0'
'D:0' 'P:0' '1:0'
'E:0' 'Q:0' '2:0'
'F:0' 'R:0' '3:0'
'G:0' 'S:0' '4:0'
'H:0' 'T:0' '5:0'
'I:0' 'U:0' '6:0'
'J:0' 'V:0' '7:0'
'K:0' 'W:0' '8:0'
'L:0' 'X:0' '9:0'

Please enter a sentence comprised of the 26 letters and/or 9 digits. Press Enter to quit.
Occurrence for each character :
'A:3' 'M:0' 'Y:0'
'B:0' 'N:0' 'Z:0'
'C:0' 'O:0' '0:0'
'D:0' 'P:0' '1:0'
'E:0' 'Q:0' '2:0'
'F:0' 'R:0' '3:0'
'G:0' 'S:0' '4:0'
'H:0' 'T:0' '5:0'
'I:0' 'U:0' '6:0'
'J:0' 'V:0' '7:0'
'K:0' 'W:0' '8:0'
'L:0' 'X:0' '9:0'
```

FIGURE 2.4 Example and Output of Task 4

