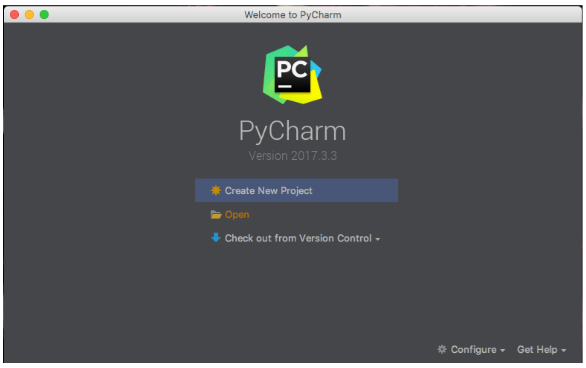
Table of Contents

Introduction:

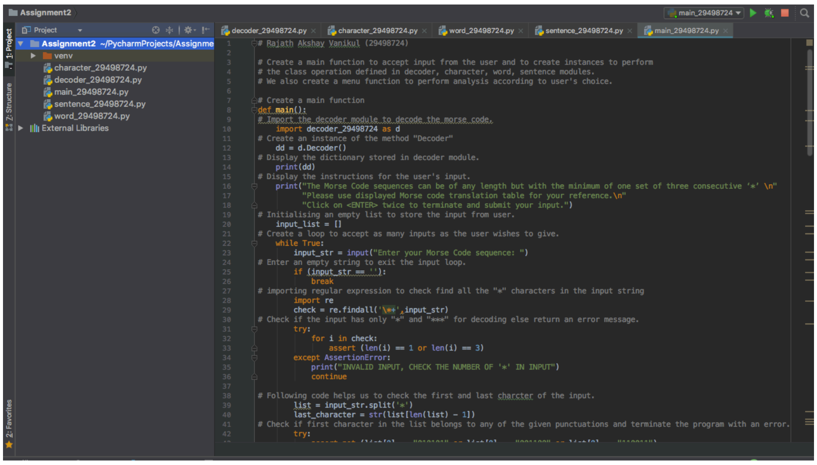
Morse code is a method of transmitting information through a series of patterned sound or light. In this assignment I will be representing the Morse code using binary digits and each character is separated with “*”. I will be decoding the code for the set of English alphabet (A to Z) and 10 numbers (0 to 9).

Instructions:

1. We now have 4 python files (Task1.py,Task2.py,Task3.py & Task4.py) in this folder. Each of these files perform tasks according to the explanation below.
2. Please launch PyCharm software on your desktop.
3. Click on “OPEN”.



4. Navigate to “29498724_A2” folder and click on open. You will be able to view multiple Task tabs once you click on the decoder_29498724.py/ character_29498724.py/ word_29498724.py/ sentence_29498724.py and main_29498724.py files. This can help you to view the program with code comments before every code line, comments will help you understand the program logic.



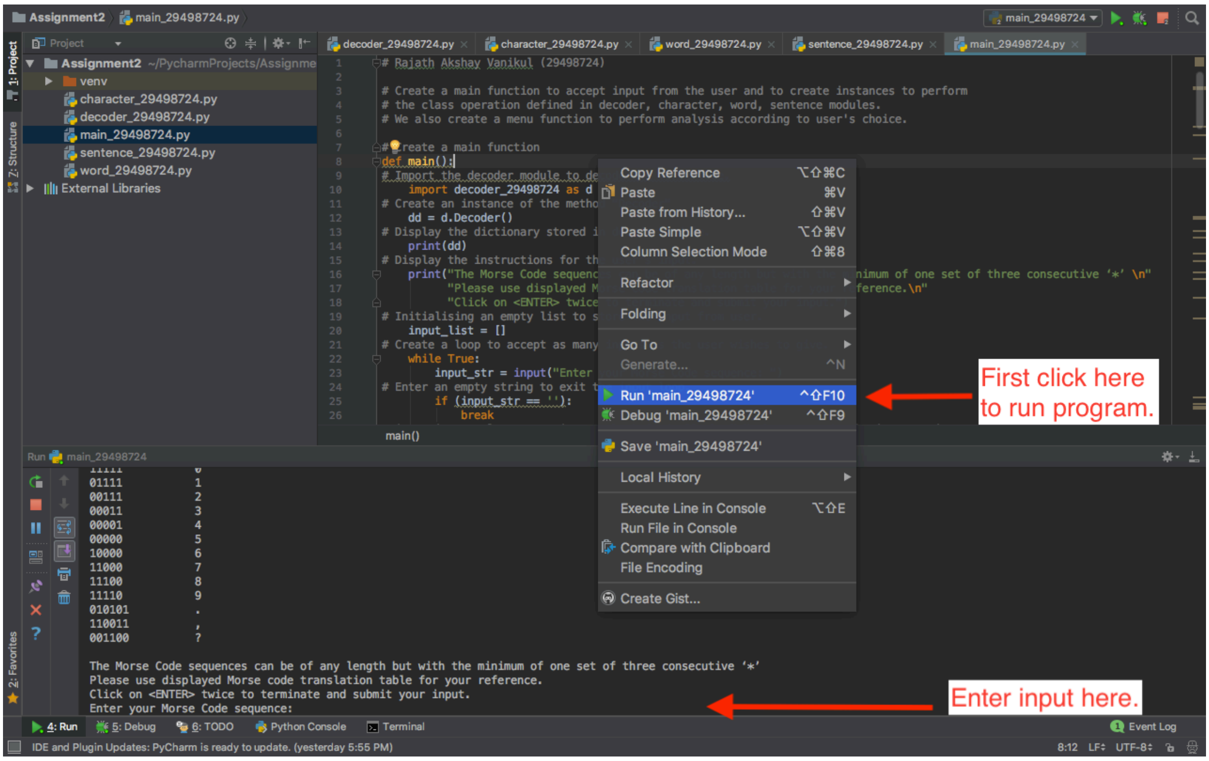
```python
# Rajath Akshay Vanikul (29498724)

# Create a main function to accept input from the user and to create instances to perform
# the class operation defined in decoder, character, word, sentence modules.
# We also create a menu function to perform analysis according to user's choice.

# Create a main function
def main():
# Import the decoder module to decode the morse code.
    import decoder_29498724 as d
# Create an instance of the method "Decoder"
    dd = d.Decoder()
# Display the dictionary stored in decoder module.
    print(dd)
# Display the instructions for the user's input.
    print("The Morse Code sequences can be of any length but with the minimum of one set of three consecutive ‘*’ \n"
          "Please use displayed Morse code translation table for your reference.\n"
          "Click on <ENTER> twice to terminate and submit your input.")
# Initialising an empty list to store the input from user.
    input_list = []
# Create a loop to accept as many inputs as the user wishes to give.
    while True:
        input_str = input("Enter your Morse Code sequence: ")
# Enter an empty string to exit the input loop.
        if (input_str == ''):
            break
# importing regular expression to check find all the "*" characters in the input string
        import re
        check = re.findall('\*+',input_str)
# Check if the input has only "*" and "***" for decoding else return an error message.
        try:
            for i in check:
                assert (len(i) == 1 or len(i) == 3)
        except AssertionError:
            print("INVALID INPUT, CHECK THE NUMBER OF '*' IN INPUT")
            continue

# Following code helps us to check the first and last charcter of the input.
        list = input_str.split('*')
        last_character = str(list[len(list) - 1])
# Check if first character in the list belongs to any of the given punctuations and terminate the program with an error.
        try:
```

5. You can Execute the program by right click on the respective task tab and click on RUN'****'. You will notice the console open to the bottom of the screen and you can now enter your input to obtain desired output.



```
# Rajath Akshay Vanikul (29498724)

# Create a main function to accept input from the user and to create instances to perform
# the class operation defined in decoder, character, word, sentence modules.
# We also create a menu function to perform analysis according to user's choice.

#Create a main function
def main():
# Import the decoder module to de
    import decoder_29498724 as d
# Create an instance of the metho
    dd = d.Decoder()
# Display the dictionary stored i
    print(dd)
# Display the instructions for th
    print("The Morse Code sequenc                                  inimum of one set of three consecutive '*' \n"
          "Please use displayed M                                   ference.\n"
          "Click on <ENTER> twice
# Initialising an empty list to s
    input_list = []
# Create a loop to accept as many
    while True:
        input_str = input("Enter
# Enter an empty string to exit t
        if (input_str == ''):
            break
```

First click here to run program.

```
11111          0
01111          1
00111          2
00011          3
00001          4
00000          5
10000          6
11000          7
11100          8
11110          9
010101         .
110011         ,
001100         ?

The Morse Code sequences can be of any length but with the minimum of one set of three consecutive '*'
Please use displayed Morse code translation table for your reference.
Click on <ENTER> twice to terminate and submit your input.
Enter your Morse Code sequence:
```

Enter input here.

6. Program instruction and output will be displayed on the same console.

7. If you would like to re-execute a program, you can do so by repeating step 6.

Task 1:

I start with creating a class 'Decoder' which contains methods to decode the morse code inserted. This task will display the complete Morse code translation table to the user and decode the input sequence by the user to a readable English format.

Program:

- Initiated the reference morse code dictionary under the constructor for this class.
- Overload the print function to display the morse code dictionary in readable format.
- Define a method called 'decode' which accepts the input string to produce a decoded sentences.
- Splitting the input sequence where all '***' occur.
- Initialise a list to store decoded sequence.
- Check for each character in dictionary and append the corresponding decoded value. return a space if you can't find the sequence in dictionary with an error output.
- Return the decoded string.

Task 2:

This Task emphasizes on counting the number of occurrences for each of the letters (‘A’ to ‘Z) and numerals (‘0’ to ‘9’) from the decoded sequences. This does not include the count of punctuations.

Program:

- Create a class 'CharacterAnalyser' which contains methods to count each character in decoded sequence.
- Initiate the reference dictionary to store the output of character analysis.
- Overload the print function to display the occurrence dictionary in readable format.
- Define a method to analyse the count of each character in the decoded sequence and return the occurrence of each character as a dictionary.
- Return a dictionary of occurrences.

Task 3:

This Task performs counting the number of occurrences for each of the English words from the decoded sequences.

Program:

- Create a class 'WordAnalyser' which contains methods to count occurrence each word in decoded sequence.
- Initiate the reference dictionary to store the output of word analysis.
- Overload the print function to display the occurrence dictionary in readable format.
- Define a method to analyse the count of occurrence of each word in the decoded sequence.
  Return the occurrence as a dictionary.

Task 4:

This Task deals with counting the number of clauses, a full sentence and a question in the decoded sequences.

Program:

- Create a class 'SentenceAnalyser' which contains methods to count occurrence each sentence in decoded sequence.
- Initiate the reference dictionary to store the output of sentence analysis.
- Overload the print function to display the occurrence dictionary in readable format.
- Define a method to analyse the count of occurrence of each sentence in the decoded sequence and
  Return the occurrence as a dictionary.

Task 5:

This task focuses on accepting as many inputs as the user wishes and to channelize the analysis process as requested by the user.

Program:

1. Create a main function to accept input from the user and to create instances to perform the class operation defined in decoder, character, word, sentence modules. We also create a menu function to perform analysis according to user's choice.
2. Create a main function
3. Import the decoder module to decode the morse code.
4. Create an instance of the method "Decoder"
5. Display the dictionary stored in decoder module for input reference.
6. Display the instructions for the user's input.
7. Create a loop to accept as many inputs as the user wishes to give.
   - Check if the input has only "*" and "***" for decoding else return an error message.
   - Check if first character in the list belongs to any of the given punctuations and display the error.
   - Check if last character in the list belongs to any of the given punctuations and display the error.
   - Check for two punctuations in series and display the error.
   - Check the input sequences to just contain '0's,'1's and '*'s with a minimum of one "***".
   - Check the decoded sequence and run the whole function if the previous input is incorrect. Else transfer the sequence to character/word/sentence analysis.

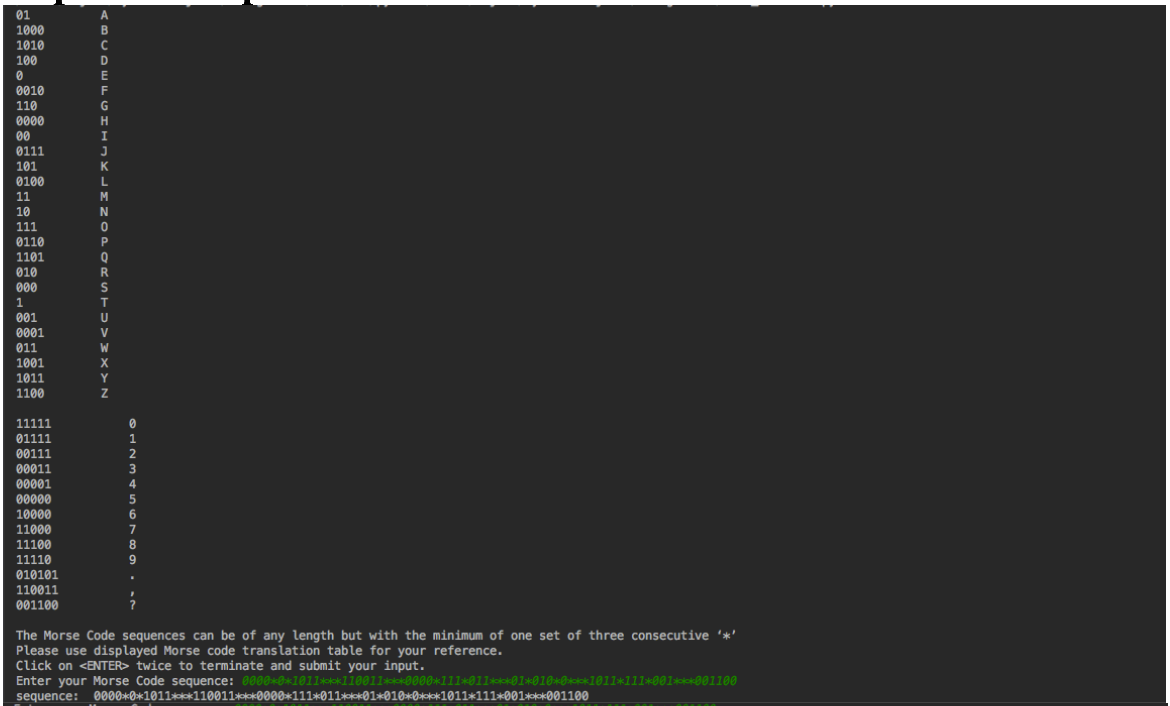
8. Define a function to access character analysis module and display the result.
9. Define a function to access word character analysis module and display the result.
10. Define a function to access character analysis module and display the result.
11. Create a function named "Menu" to request for user's choice for analysis
    - Enter 1 to check for character analysis on your input
    - Enter 2 to check for word analysis on your input
    - Enter 3 to check for sentence analysis on your input
    - Enter 4 to exit
    - Enter 0 to input another sequence

The complete program can run continuously by accepting inputs from the user and without a a program to exit unless you choose for option 4 at the menu.

Assumptions:

1. For accurate word analysis, all the punctuations must be entered after a space(***).
2. Input must start with morse code character (‘0’s and ‘1’s) and not with ‘*’s.

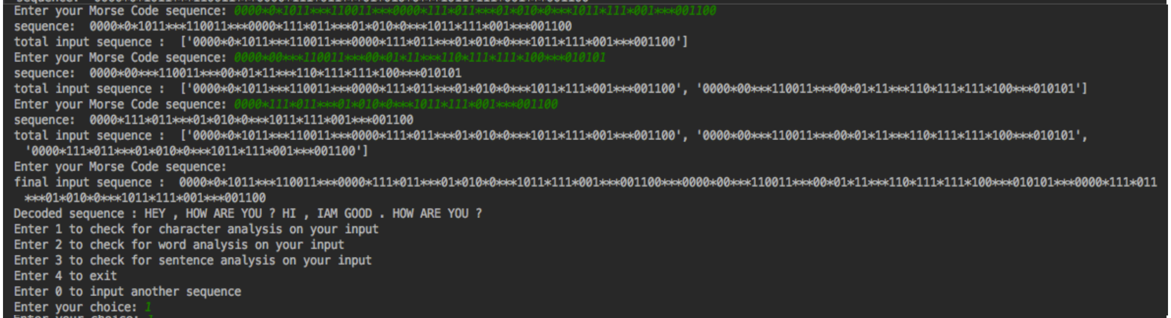
Expected output:



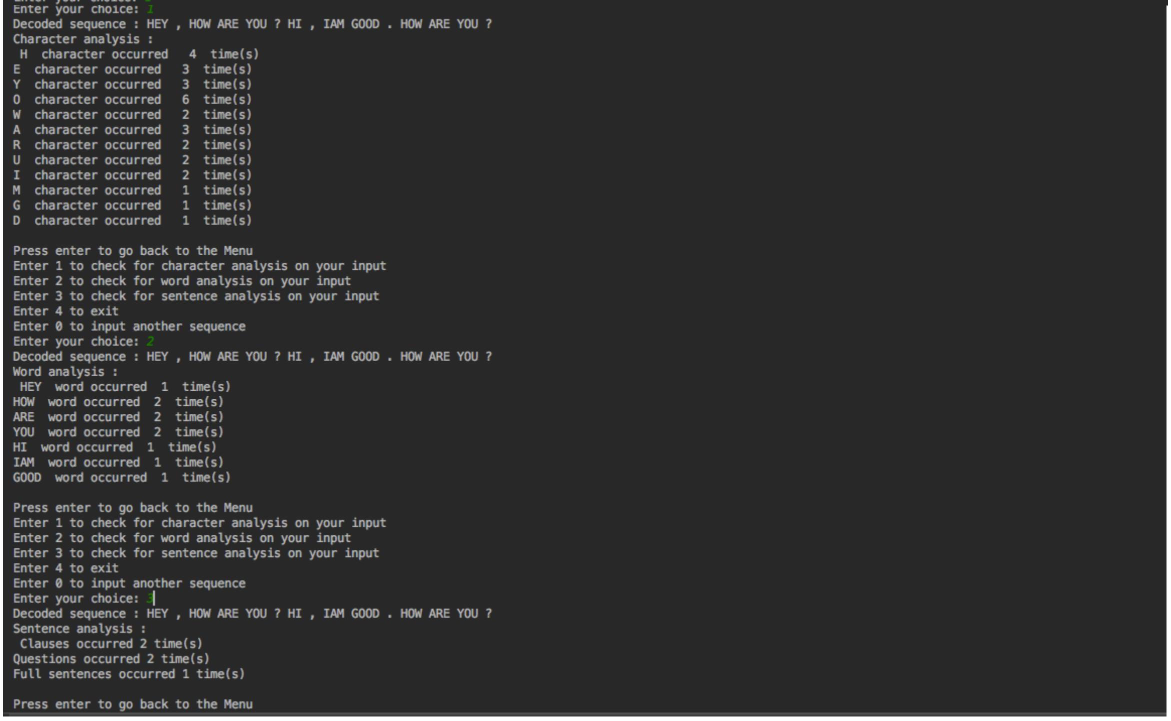
```
01        A
1000      B
1010      C
100       D
0         E
0010      F
110       G
0000      H
00        I
0111      J
101       K
0100      L
11        M
10        N
111       O
0110      P
1101      Q
010       R
000       S
1         T
001       U
0001      V
011       W
1001      X
1011      Y
1100      Z

11111          0
01111          1
00111          2
00011          3
00001          4
00000          5
10000          6
11000          7
11100          8
11110          9
010101         .
110011         ,
001100         ?

The Morse Code sequences can be of any length but with the minimum of one set of three consecutive ‘*’
Please use displayed Morse code translation table for your reference.
Click on <ENTER> twice to terminate and submit your input.
Enter your Morse Code sequence: 0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100
sequence:  0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100
```



```
Enter your Morse Code sequence: 0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100
sequence:  0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100
total input sequence :  ['0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100']
Enter your Morse Code sequence: 0000*00***110011***00*01*11***110*111*111*100***010101
sequence:  0000*00***110011***00*01*11***110*111*111*100***010101
total input sequence :  ['0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100', '0000*00***110011***00*01*11***110*111*111*100***010101']
Enter your Morse Code sequence: 0000*111*011***01*010*0***1011*111*001***001100
sequence:  0000*111*011***01*010*0***1011*111*001***001100
total input sequence :  ['0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100', '0000*00***110011***00*01*11***110*111*111*100***010101',
 '0000*111*011***01*010*0***1011*111*001***001100']
Enter your Morse Code sequence:
final input sequence :  0000*0*1011***110011***0000*111*011***01*010*0***1011*111*001***001100***0000*00***110011***00*01*11***110*111*111*100***010101***0000*111*011
 ***01*010*0***1011*111*001***001100
Decoded sequence : HEY , HOW ARE YOU ? HI , IAM GOOD . HOW ARE YOU ?
Enter 1 to check for character analysis on your input
Enter 2 to check for word analysis on your input
Enter 3 to check for sentence analysis on your input
Enter 4 to exit
Enter 0 to input another sequence
Enter your choice: 1
```



```
Enter your choice: 1
Decoded sequence : HEY , HOW ARE YOU ? HI , IAM GOOD . HOW ARE YOU ?
Character analysis :
 H  character occurred   4  time(s)
E  character occurred   3  time(s)
Y  character occurred   3  time(s)
O  character occurred   6  time(s)
W  character occurred   2  time(s)
A  character occurred   3  time(s)
R  character occurred   2  time(s)
U  character occurred   2  time(s)
I  character occurred   2  time(s)
M  character occurred   1  time(s)
G  character occurred   1  time(s)
D  character occurred   1  time(s)

Press enter to go back to the Menu
Enter 1 to check for character analysis on your input
Enter 2 to check for word analysis on your input
Enter 3 to check for sentence analysis on your input
Enter 4 to exit
Enter 0 to input another sequence
Enter your choice: 2
Decoded sequence : HEY , HOW ARE YOU ? HI , IAM GOOD . HOW ARE YOU ?
Word analysis :
 HEY  word occurred  1  time(s)
HOW  word occurred  2  time(s)
ARE  word occurred  2  time(s)
YOU  word occurred  2  time(s)
HI  word occurred  1  time(s)
IAM  word occurred  1  time(s)
GOOD  word occurred  1  time(s)

Press enter to go back to the Menu
Enter 1 to check for character analysis on your input
Enter 2 to check for word analysis on your input
Enter 3 to check for sentence analysis on your input
Enter 4 to exit
Enter 0 to input another sequence
Enter your choice: 3
Decoded sequence : HEY , HOW ARE YOU ? HI , IAM GOOD . HOW ARE YOU ?
Sentence analysis :
 Clauses occurred 2 time(s)
Questions occurred 2 time(s)
Full sentences occurred 1 time(s)

Press enter to go back to the Menu
```

Input’s choice of ‘4’ will again let the user input his morse code for decode. This program will terminate only if the user’s choice is ‘0’.