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).

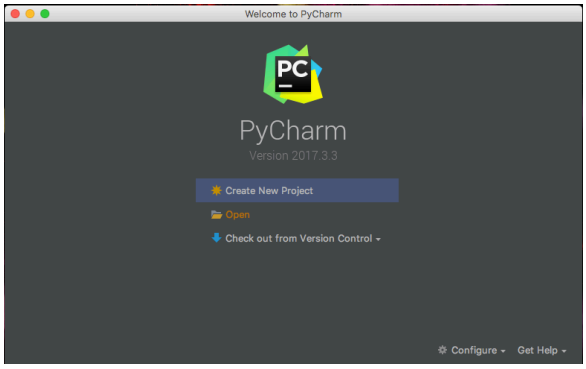
The table below shows the Morse code representation considered for this assignment.

Morse Code	Character	Morse Code	Character
01	A	11111	0
1000	B	01111	1
1010	C	00111	2
100	D	00011	3
0	E	00001	4
0010	F	00000	5
110	G	10000	6
0000	H	11000	7
00	I	11100	8
0111	J	11110	9
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		

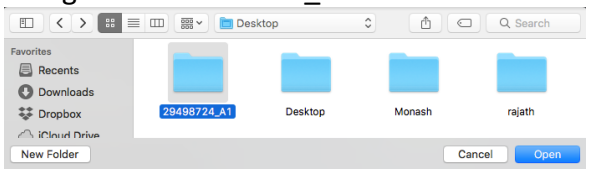
Instructions:

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.

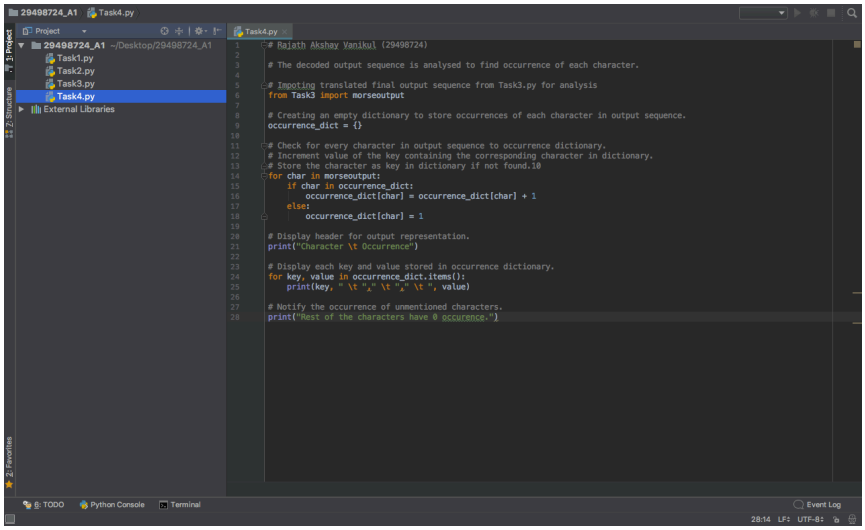
3. Click on “OPEN”.



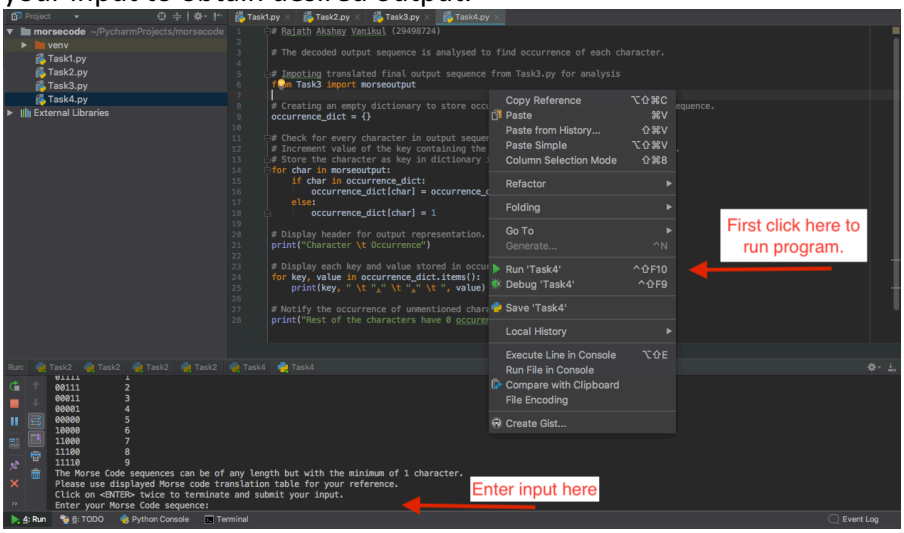
4. Navigate to “29498724_A1” folder and click on open.



5. Click on Task1.py/Task2.py/Task3.py/Task4.py accordingly to operate.
   You will be able to view multiple Task tabs once you click on these task.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.



6. 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.



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

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

Task1:

I start with creating and defining a dictionary for Morse Code which can be used in further programs as reference. This task will display the complete Morse code translation table to the user.

Program:
I have named Morse code dictionary as “morse_dict” and stored binary sequence as keys and the translated corresponding alphabets/numbers as their respective values.
I have used ‘for’ loop to display the contents on the dictionary in legible vertical fashion.

```
# Rajath Akshay Vanikul (29498724)

#  Building a Simple Morse Code Decoder


# Create a Morse dictionary and define morse code using binary digits for each English alphabet and number(0 to 9).
# Display the Morse code representation as output.


# Create and define the morse code dictionary
# Keys - binary representation of morse code, Values - English alphabets and numbers
morse_dict = {'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'}

# Display the header for representation.
print("Morse \t Character")

# Display each key and value stored in Morse code dictionary in legible vertical fashion.
for key,value in morse_dict.items():
    print(key,'  \t ',value)
```

Expected output:

```
/Users/Rajath/PycharmProjects/morsecode/venv/bin/python /Users/Rajath/PycharmProjects/morsecode/Task1.py
Morse    Character
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

Process finished with exit code 0
```

Task2:

This Task emphasizes on actively storing all the input sequences given by the user.
Firstly, Import Task1 for user's input reference. Which is shown in line 9.
I display the input options and directions for the users as follows:

- Please use displayed Morse code translation table for your reference.
- The Morse Code sequences can be of any length but with the minimum of 1 character.
- User must just click on <Enter> without and entry to submit the input.

Now I create an infinite loop to accept input from the user with following condition.

- If input string is empty, then break the loop
- If the set of input string is contained in the specified set defined (0,1,*) then append it to input list to store the value.
- Else notify the user for invalid entry and prompt the user to enter again.

Display the recent input and all the complete stored input by the user.
Program:

```
1   # Rajath Akshay Vanikul (29498724)
2
3   # Prompt the user to input any number of Morse code sequences until the user decides to stop.
4   # Display the Morse code for user's reference.
5   # To exit the input prompt, user must just click on <Enter> without an entry.
6
7
8   # Importing the result of Task1.py to display the Morse code dictionary.
9   import Task1
10
11  # Display the guidelines for user's input.
12  print("The Morse Code sequences can be of any length but with the minimum of 1 character. \n"
13        "Please use displayed Morse code translation table for your reference.\n"
14        "Click on <ENTER> twice to terminate and submit your input.")
15
16  # Create an empty list to store all the input sequences entered by the user.
17  input_list = []
18
19  # Creating a set of default characters to compare for valid translation.
20  default_str = set("01*")
21
22  # Define an infinite loop to prompt user's input.
23  # Check the input sequences to just contain '0's,'1's and '*'s and Store all to the list for further operations.
24  # break the loop if input is "".
25  while True:
26      input_str = input("Enter your Morse Code sequence: ")
27
28      if (input_str == ''):
29          break
30      if (set(input_str).issubset(default_str)):
31          input_list.append(input_str)
32          print("sequence: ", input_str)
33          print("total input sequence : ", input_list)
34      else:
35          print("INVALID INPUT, Please enter again.")
36
37  # Print the complete input sequence from the user with "*" between input sequences.
38  morsein_str = "*".join(input_list)
39  print("total input sequence : ",morsein_str)
```

Expected output:

```
The Morse Code sequences can be of any length but with the minimum of 1 character.
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: 01*1000*1010
sequence:  01*1000*1010
total input sequence :  ['01*1000*1010']
Enter your Morse Code sequence: 1*001*12*001
INVALID INPUT, Please enter again.
Enter your Morse Code sequence: 1*001*0001*r!
INVALID INPUT, Please enter again.
Enter your Morse Code sequence: 1011*11*10*11111
sequence:  1011*11*10*11111
total input sequence :  ['01*1000*1010', '1011*11*10*11111']
Enter your Morse Code sequence: 1
sequence:  1
total input sequence :  ['01*1000*1010', '1011*11*10*11111', '1']
Enter your Morse Code sequence: 11110
sequence:  11110
total input sequence :  ['01*1000*1010', '1011*11*10*11111', '1', '11110']
Enter your Morse Code sequence:
total input sequence :  01*1000*1010*1011*11*10*11111*1*11110

Process finished with exit code 0
```

Task3:

In this task, Complete input sequence is decoded and translated to English alphabet and numbers. This translated characters are displayed to user.

- I have imported both task1 and task2 in this task to help me with reference Morse code representation and to prompt the user to input a set of sequence for decoded result.
- Input list is now transferred as a string with "*" as a character between each element in the list.
- Now input string(morsein_str) will be split using "*" as a separator and stored as elements of a list (morsein_list).
- For all the nonempty elemets of the list, compare element of the list with each key in the dictionary (morse_dict) and append the corresponding value of the key to another list(morseout_list) if the item is found in dictionary
  If the item doesn't match any key in the dictionary then print the error item and display incorrect item.
- Translated stored output list is then transferred to a string in the same sequence for better readability.
- Display the translated output and the incorrect item in the input.

Program:

```
# Rajath Akshay Vanikul (29498724)

# Each Morse code sequence will be processed to translate the sequence into corresponding English alphabets and numbers.
# Display corresponding translated elements in the same sequence.
# Display the invalid sequences entered by the user.


# Display the result from Task1.py for user's reference and import the dictionary.
from Task1 import morse_dict

# Importing the input sequence list from Task2.py for reference.
from Task2 import input_list

# Store the list as a string with "*" between the sequences.
morsein_str = "*".join(input_list)

# Complete input string will be split using "*" as a separator and stored as a list.
morsein_list = morsein_str.split("*")

# Create an empty list to store result.
morseout_list= []

# For all the non-empty elements in the list check if item in list to each key in dictionary,
# Store the corresponding value in sequence.
# Display invalid input if not found in the dictionary.

for item in morsein_list:
    if item != "":
        if item in morse_dict:
            morseout_list.append(morse_dict[item])
        else:
            print(item,"- incorrect item")

# Display the translated output list as a string in sequence.
morseoutput_str= "".join(morseout_list)
print("Translated sequence is : ",morseoutput_str)
```

Expected output:

```
The Morse Code sequences can be of any length but with the minimum of 1 character.
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: 100*01*1*01
sequence:  100*01*1*01
total input sequence :  ['100*01*1*01']
Enter your Morse Code sequence: 01*1000*12*1010*100
INVALID INPUT, Please enter again.
Enter your Morse Code sequence: 000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1
sequence:  000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1
total input sequence :  ['100*01*1*01', '000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1']
Enter your Morse Code sequence: 1000*10*r!*100
INVALID INPUT, Please enter again.
Enter your Morse Code sequence: 1010101010*01111*00011*10001100*01111
sequence:  1010101010*01111*00011*10001100*01111
total input sequence :  ['100*01*1*01', '000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1', '1010101010*01111*00011*10001100*01111']
Enter your Morse Code sequence:
total input sequence :  100*01*1*01*000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1*1010101010*01111*00011*10001100*01111
0000000000000000000 - incorrect item
11111111 - incorrect item
1010101010 - incorrect item
10001100 - incorrect item
DATASCIENTIST131

Process finished with exit code 0
```

Task4:

Finally, We analyze the translated output sequence which we have received to find occurrences of each character and display the the result.

- I first import task3 which in turn calls task1 and task2 to assist run all three tasks in order.
- Once we get the translated output result from task3, We use an empty dictionary to compare each character of the string to dictionary (occurrence_dict).
- If the character in string is found as a key in dictionary, then increase the value by 1. Else, store the character as key and assign the value as 1.

This logic will help me in counting the occurrence of each character in the given string. Display each element of the dictionary with occurrence of each character in vertical fashion.

Program:

```
# Rajath Akshay Vanikul (29498724)

# The decoded output sequence is analysed to find occurrence of each character.

# Impoting translated final output sequence from Task3.py for analysis
from Task3 import morseoutput_str

# Creating an empty dictionary to store occurrences of each character in output sequence.
occurrence_dict = {}

# Check for every character in output sequence to occurrence dictionary.
# Increment value of the key containing the corresponding character in dictionary.
# Store the character as key in dictionary if not found.10
for char in morseoutput_str:
    if char in occurrence_dict:
        occurrence_dict[char] = occurrence_dict[char] + 1
    else:
        occurrence_dict[char] = 1

# Display header for output representation.
print("Character \t Occurrence")

# Display each key and value stored in occurrence dictionary.
for key, value in occurrence_dict.items():
    print(key, " \t ", " \t ", value)

# Notify the occurrence of unmentioned characters.
print("Rest of the characters have 0 occurence.")
```

Expected output:

```
The Morse Code sequences can be of any length but with the minimum of 1 character.
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: 100*01*1*01
sequence:  100*01*1*01
total input sequence :  ['100*01*1*01']
Enter your Morse Code sequence: 01*1000*12*1010*100
INVALID INPUT, Please enter again.
Enter your Morse Code sequence: 000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1
sequence:  000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1
total input sequence :  ['100*01*1*01', '000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1']
Enter your Morse Code sequence: 1000*10*r!*100
INVALID INPUT, Please enter again.
Enter your Morse Code sequence: 1010101010*01111*00011*10001100*01111
sequence:  1010101010*01111*00011*10001100*01111
total input sequence :  ['100*01*1*01', '000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1', '1010101010*01111*00011*10001100*01111']
Enter your Morse Code sequence:
total input sequence :  100*01*1*01*000*1010*0000000000000000000*00*0*10*1*00*000*11111111*1*1010101010*01111*00011*10001100*01111
0000000000000000000 - incorrect item
11111111 - incorrect item
1010101010 - incorrect item
10001100 - incorrect item
DATASCIENTIST131
Character    Occurrence
D            1
A            2
T            3
S            2
C            1
I            2
E            1
N            1
1            2
3            1
Rest of the characters have 0 occurence.

Process finished with exit code 0
```