1. Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers.
   *Sample data : 3, 5, 7, 23*
   *Output :*
   *List : ['3', ' 5', ' 7', ' 23']*
   *Tuple : ('3', ' 5', ' 7', ' 23')*

2. Write a function translate(text) that will translate a text that doubles every consonant and place an occurrence of "o" in between. For example, translate("this is fun") should return the string "tothohisos isos fofunon"

3. Write a Python program to print the calendar of a given month and year.
   *Note :* Use 'calendar' module.
   Python calendar.month(theyear, themonth, w=0, l=0):
   The function returns a month's calendar in a multi-line string using the formatmonth() of the TextCalendar class. 'l' specifies the number of lines that each week will use.

4. Write a function is_member(val) that takes a value (i.e. a number, string, etc) x and a list of values a, and returns True if x is a member of a, False otherwise. (Note that this is exactly what the in operator does, but for the sake of the exercise you should pretend Python did not have this operator.)

5. Define a function overlapping(lst1, lst2) that takes two lists and returns True if they have at least one member in common, False otherwise. You may use your is_member(val) function, or the in operator, but for the sake of the exercise, you should (also) write it using two nested for-loops.

6. Define a procedure histogram(lst) that takes a list of integers and prints a histogram to the screen. For example, histogram([4, 9, 7]) should print the following:
   ****
   *********
   *******

7. Write a program that maps a list of words into a list of integers representing the lengths of the correponding words.

8. Write a function find_longest_word(lwords) that takes a list of words and returns the length of the longest one.

9. Write a function filter_long_words(lwords) that takes a list of words and an integer n and returns the list of words that are longer than n.

10. A pangram is a sentence that contains all the letters of the English alphabet at least once, for example: The quick brown fox jumps over the lazy dog. Your task here is to write a function to check a sentence to see if it is a pangram or not.

11. Write a function char_freq(string) that takes a string and builds a frequency listing of the characters contained in it.

12. In cryptography, a Caesar cipher is a very simple encryption techniques in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet. For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to

communicate with his generals. ROT-13 ("rotate by 13 places") is a widely used example of a Caesar cipher where the shift is 13. Your task in this exercise is to implement an encoder/decoder of ROT-13.

13. The third person singular verb form in English is distinguished by the suffix -s, which is added to the stem of the infinitive form: run -> runs. A simple set of rules can be given as follows:
    a) If the verb ends in y, remove it and add ies
    b) If the verb ends in o, ch, s, sh, x or z, add es
    c) By default just add s

    Your task in this exercise is to define a function makeForms(verb) which given a verb in infinitive form returns its third person singular form. Test your function with words like try, brush, run and fix. Note however that the rules must be regarded as heuristic, in the sense that you must not expect them to work for all cases. Tip: Check out the string method endswith().

14. In English, the present participle is formed by adding the suffix -ing to the infinite form: go -> going. A simple set of heuristic rules can be given as follows:
    a) If the verb ends in e, drop the e and add ing (if not exception: be, see, flee, knee, etc.)
    b) If the verb ends in ie, change ie to y and add ing
    c) For words consisting of consonant-vowel-consonant, double the final letter before adding ing
    d) By default just add ing

    Your task in this exercise is to define a function makeForming() which given a verb in infinitive form returns its present participle form. Test your function with words such as lie, see, move and hug. However, you must not expect such simple rules to work for all cases.