Faster Code:

1. **Tuples Vs Lists:**
   Create Tuple from 0-10 and list from 0-10 (not range) and time both operation, which is faster and why?
2. **List Generators:**
   a. Create a function power2(n) that return list of power 2(like 3**2) of list from 0-(n-1)?
   b. Then create a function power2_generator(n) that do the same thing as power2(n) but as ***generator***! Hint here
   c. Write needed code to print the list elements in both cases!
   d. Check timeit and memory profile for both functions power2(n) and power2_generator(n) for n=10000.
3. **Iter Object :**
   a. Converting data types like string to iter object is very useful. Read documentation about iter() and next() here and then answer these questions:
   b. Let's say my_string = "I'm not iterator object" try command next(my_string) what you get?
   c. Create function that convert a string to iterator and print each character in single line using next().
4. **Itertools:**
   a. Please Check the documentation of itertools here then:
   b. Create function string_perm_list(my_string) that return list of all possible order of the input string "ABCD" so the output like: (hint here)
   ```
   ['ABCD', 'ABDC', 'ACBD', 'ACDB',
   'ADBC', 'ADCB', 'BACD', 'BADC'…]
   ```
   c. Now create same function but it returns tuple string_prem_tuple(my_string).

   d. Use input_test_string= “0123456789” and check timing, and memory_profile for both the list and tuple version.
   e. What do think about results of both in d?

5. **List comprehensive:**
   a. Using list comprehensive create list av Fahrenheit from this Celsius = [0,10,20.1,34.5]
   b. Using Tuple comprehensive create tuple of T_Fahrenhite from the same Celsius list in a.
   c. Time both code and compare result , do the same using memory_profile , what do u think?

6. **List comprehensive with nested lists:**
   a. Using generator Let’s create my_nest_list that has 3 numbers of list from 0-4 ,5-9 and 10-14. output:
      ```
      [[0, 1, 2, 3, 4], [5, 6, 7, 8, 9], [10,
      11, 12, 13, 14]]
      ```
   b. Using list comprehension create a list of double value of even numbers in my_nest_list. output:
      ```
      [[0, 4, 8], [12, 16], [20, 24, 28]]
      ```
   c. Using list comprehension create a list of double value of even numbers in my_list also the number itself if it was odds in the same order. output :
      ```
      [[0, 1, 4, 3, 8], [5, 12, 7, 16, 9],
      [20, 11, 24, 13, 28]]
      ```
   d. Repeat a,b,c with tuples, then check if that faster and , if that save a memory?
   e. Using list comprehension flat the nested list my_nest_list
   f. Using itertools /chain to flat the same list.
   g. Compare timing of both operation using time it , what do u think ?