第十四周

本周目标

1. 掌握递归

1. Write a Python program to calculate the sum of a list of numbers.

2. Write a Python program to converting an integer to a string in any base.

3. Write a Python program of recursion list sum.
   Suppose the following input is supplied to the program:
   `[1, 2, [3, 4], [5, 6]]`
   Then, the output should be:
   `21`

4. Write a Python program to get the factorial of a non-negative integer.

5. Write a Python program to get the sum of digitals of a non-negative integer.
   Suppose the following input is supplied to the program:
   `345`
   Then, the output should be:
   `12`

6. Write a Python program to calculate the value of `'a'` to the power `'b'`.
   Suppose the following input is supplied to the program:
   `3, 4`
   Then, the output should be:
   `81`

7. Binary Search:
   Given a sorted list `L` in strictly ascending order, for example `L = [2, 5, 6, 8, 10]`. Write a function to see whether a given element `x` is in `L` for not.
   Since `L[0, …, n]` is in ascending order, we can have an efficient way to find `x`: We divide `L` into two part by its middle element `L[n/2]`: `L[0: n/2]`, `L[n/2: n]`, if `x > L[n/2]`, then we search in the right part, otherwise we search in the left part.

   Suppose the following input is supplied to the program:
   `L = [2, 5, 6, 8, 10], x = 6`
   Then, the output should be:
   `2`

8. Merge:
   Given two lists `a`, `b` in ascending order, implement a function `merge()` to merge these two lists to generate a new list with ascending order.

   Suppose the following input is supplied to the program:
   ```
   L1 = [1, 2, 5, 6]
   L2 = [-1, -2, 3, 4]
   ```
   Then, the output should be:
   ```
   [-1, -2, 1, 2, 3, 4, 5, 6]
   ```

9. Merge Sort:
   Given a list `L`, arrange the elements in `L` in ascending order.
   We divide `L` into two parts `L1 = L[0: n/2]` and `L2 = L[n/2: n]`, then we sort them respectively. After that, we obtain two sorted list `L1`, `L2`. Finally, we need to merge `L1` and `L2` to obtain a new sorted list `L`.

   Suppose the following input is supplied to the program:
   ```
   L = [6, 2, 3, 4, 3, 3, 3, 3, 3, 1, 5]
   ```
   Then, the output should be:
   ```
   [1, 2, 3, 3, 3, 3, 3, 3, 4, 5, 6]
   ```

10. The special syntax, `*args` in function definitions is used to pass a variable number of arguments to a function. It is used to pass a non-keyworded, variable-length argument list, and the double asterisk form is used to pass a keyworded, variable-length argument list.

    Here is an example of how to use the non-keyworded form. This example passes one formal (positional) argument, and two more variable length arguments.
    ```
    def test_var_args(farg, *args):
       print ("formal arg: ", farg)
       for arg in args:
             print("another arg: ", arg)
    test_var_args(1, "two", 3)
    ```
    Results:
    ```
    formal arg: 1
    another arg: two
    another arg: 3
    ```

    Try to implement `sum()` on `*args`.

    Suppose the following input is supplied to the program:
    ```
    sum(1, 3, 5)
    ```
    Then, the output should be:
    ```
    9
    ```