数据容器练习

```
# （1）编写一个函数
# sum_list，返回一个列表中所有元素的和。[1, 2, 3, 4]


def sum_list(l):
    sum = 0
    for i in (l):
        sum = sum + i
    return sum
n = [1,2,3,4]
print(sum_list(n))

def sum_list(l):
    s = sum(l)
    return s
n = [1,2,3,4]
print(sum_list(n))


# （2）编写一个函数
# find_max，返回列表中的最大值。[1, 5, 3, 9, 2]

def find_max(l):
    m = max(l)
    return m

n = [1,5,3,9,2]
print(find_max(n))




# （3）编写一个函数
# factorial，使用
# while 循环计算一个数的阶乘。    5

def factorial(n):
    sum = 1
    while(n > 0):
```

```
        sum = sum * n
        n -= 1
    return sum

print(factorial(int(input("请输入一个数:"))))


# （4）使用
# for 循环遍历一个列表并将每个元素乘以2。[1, 2, 3, 4, 5]

n = [1,2,3,4,5]
l = []
for i in n:
    l.append(2 * i)
print(l)




# （5）编写一个函数
# fibonacci，返回前n个斐波那契数。  10

def f(n):
        if (n <= 2 and n > 0):
            return 1
        else:
            return f(n - 1) + f(n - 2)

def fibonacci(n):
    while(n > 0):
        l.append(f(n))
        n -= 1
    return l

n = int(input("输入n:"))
l = []
print(fibonacci(n))
```

或者

```
def fibonacci(n):
    sum1 = 1
    l_first = [1,1] #初始数列
```

```
    if (n <= 2):
        sum1 = 1
    while (n > 2):
        if (n > 2):
            l_first.append(l_first[-1] + l_first[-2])#相当于f(n-1)+f(n-2)
            sum1 += l_first[-1]
        n -= 1
    print(l_first)
    return sum(l_first)
print(fibonacci(5))



# （6）编写一个函数
# remove_duplicates，去除列表中的重复元素。[1, 2, 2, 3, 4, 4, 5]

def remove_duplicates(l):
    index = []
    for i in range(len(l) - 1):
        if l[i] == l[i + 1]:
            index.append(i)
    for j in index:
        l.pop(j)
    return l

list = [1,2,2,3,4,4,5]
print(remove_duplicates(list))

# （7）编写一个函数
# flatten_list，将嵌套列表展开成一个单层列表。 [1, [2, 3], [4, [5, 6]], 7]

new = []#没有嵌套的
old = []#嵌套的
def flatten_list(l):
    for i in range(len(l)):
        if type(l[i]) != type(l):
            new.append(l[i])
        else:
            flatten_list(l[i])
    return new

lst = [1, [2, 3], [4, [5, 6]], 7]
print(flatten_list(lst))
```

```python
# （8）编写一个函数
# remove_duplicates_preserve_order，从列表中删除重复项并保持原来的顺序。[1, 2, 2, 3, 4, 4, 5]
def remove_duplicates_preserve_order(l):
    index = []
    for i in range(len(l) - 1):
        if l[i] == l[i + 1]:
            index.append(i)
    for j in index:
        l.pop(j)
    return l

list = [1,2,2,3,4,4,5]
print(remove_duplicates(list))
# （9）编写一个函数
# find_pairs_with_sum，在一个列表中找到所有和为指定值的数对。[1, 2, 3, 4, 5, 6]；target = 7
#
def find_pairs_with_sum(l,n):
    r = []#返回的列表
    kv = []#数对
    for i in range(len(l)):
        for j in range(i+1,len(l)):
            if (l[i] + l[j] == 7):
                kv.append(l[i])
                kv.append(l[j])
                r.append(kv)
                kv = []
    l.reverse()
    for i in range(len(l)):
        for j in range(i + 1, len(l)):
            if (l[i] + l[j] == 7):
                kv.append(l[i])
                kv.append(l[j])
                r.append(kv)
                kv = []
    return r

l = [1,2,3,4,5,6]
print(find_pairs_with_sum(l,7))
```

或者

```
def find_pairs_with_sum(lst,target):
    lst = sorted(set(lst)) #将列表去重然后排序，然后返回排序后的列表
    seen = set() #记录补数,并去重
    result = []
    for number in lst: #遍历字典
        compelement = target - number #求数字的补数
        if compelement in seen:
            result.append((number, compelement)) #将(数字,补数)添加到列表中
        seen.add(number) #不管是否找到都将其加入集合中，方便后面的查找
    lst.reverse()
    # print(result)
    # print(lst)
    seen1 = set()
    result_reverse = []
    for number1 in lst:
        compelement = target - number1
        if compelement in seen1:
            result_reverse.append((number1, compelement))
        seen1.add(number1)
    # print(result_reverse)
    result.extend(result_reverse)
    result.sort()
    return result

lst1 = [1,2,3,4,5,6]
print(find_pairs_with_sum(lst1,7))

# （10）编写一个函数
# longest_consecutive_sequence，找到一个列表中的最长连续子序列的长度。[100, 4, 200, 1, 3, 2]

# s = []#子序列
longest = []#最长子序列

def longest_consecutive_sequence(l):
    # s = []
    l.sort()
    len1 = 1 #连续子序列的长度
    longest_len = 1 #最长的连续子序列的长度
    for i in range(len(l) - 1):
        if (l[i] == l[i+1]):
            continue
```

```
        if ((l[i+1] - l[i]) == 1):
            # print("进循环了")
            if (longest_len == len1):
                len1 += 1
        else:
            len1 = 1 #当序列不连续时,长度重置
        longest_len = max(len1,longest_len)#当longest最大时,为最长的序列
    return longest_len
l = [100, 4, 200, 1, 3, 2]
a = longest_consecutive_sequence(l)
print(f'最长的子序列长度为为: {a}')

##解法仅限此题，其他例子不行
```