Lists

Learning objectives

11.2.3.1 create a list
11.2.3.2 organize the output of a string using the split() and join() methods
11.2.2.1 perform access to the elements of strings, lists, tuples

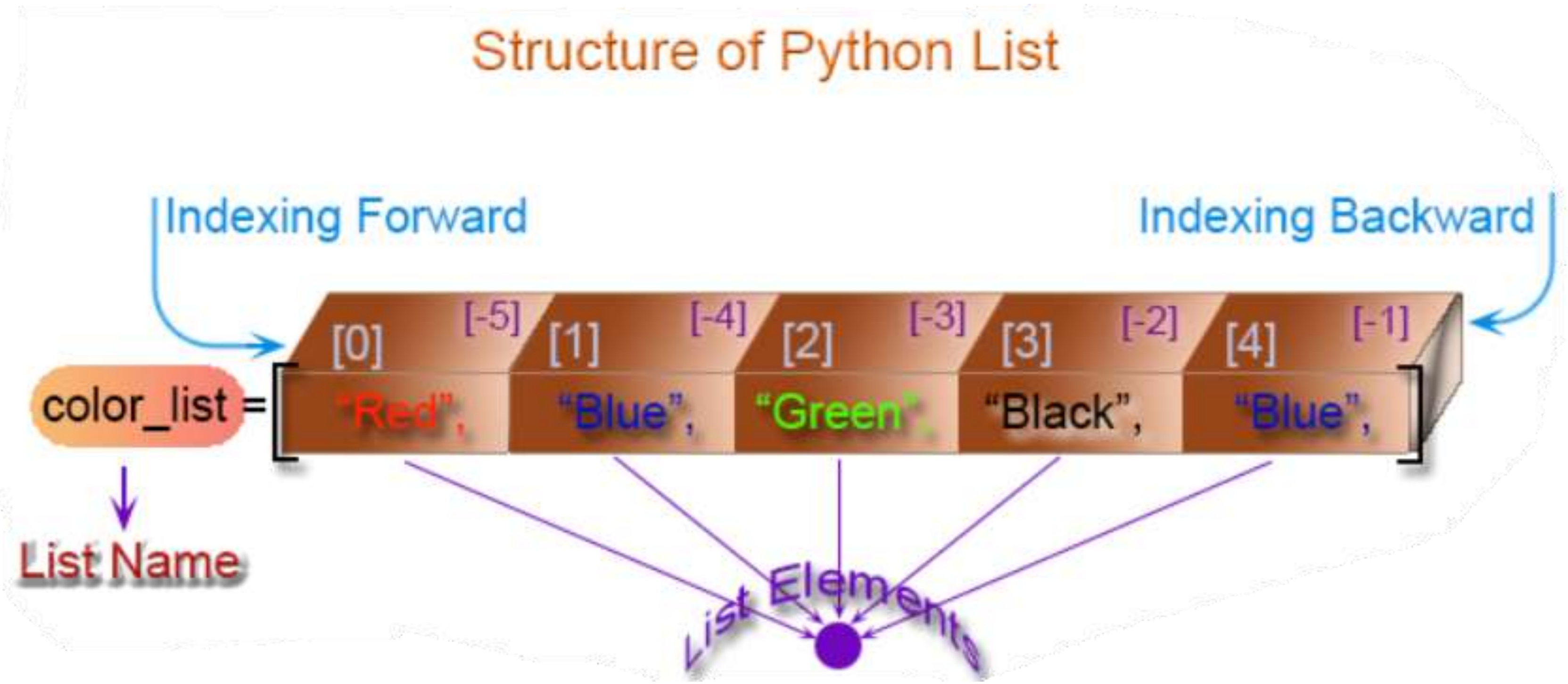
Assessment criteria:

- create a list
- add item to the list
- remove item from list
- perform access to list item by index

List

A list is a container which holds comma-separated values (items or elements) between square brackets where items or elements need not all have the same type.

In general, we can define a list as an object that contains multiple data items (elements). The contents of a list can be changed during program execution. The size of a list can also change during execution, as elements are added or removed from it.
Note: There are much programming languages which allow us to create arrays, which are objects similar to lists. Lists serve the same purpose as arrays and have many more built-in capabilities. Traditional arrays can not be created in Python.


Structure of Python List
Indexing Forward
Indexing Backward
[0]
[-5]
[1]
[-4]
[2]
[-3]
[3]
[-2]
[4]
[-1]
color_list = ["Red", "Blue", "Green", "Black", "Blue",]
List Name
List Elements

Examples of lists:

```
numbers = [10, 20, 30, 40, 50]
names = ["Sara", "David", "Warner", "Sandy"]
student_info = ["Sara", 1, "Chemistry"]
```

Create a Python list

Following list contains all integer values:

```
my_list1 = [5, 12, 13, 14] # the list contains all integer values
print(my_list1)
[5, 12, 13, 14]
```

Following list contains all string:

```
my_list2 = ['red', 'blue', 'black', 'white'] # the list contains
all string
values
print(my_list2)
['red', 'blue', 'black', 'white']
```

Create a Python list

Following list contains a string, an integer and a float values:

```
my_list3 = ['red', 12, 112.12] # the list contains a string, an
integer and
a float values
print(my_list3)
['red', 12, 112.12]
```

A list without any element is called an empty list. See the following statements.

```
my_list=[]
print(my_list)
[]
```

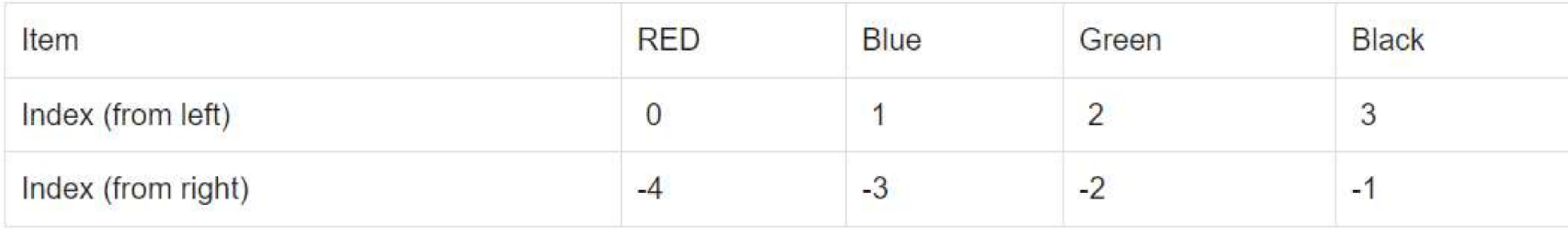
Create a Python list

Use + operator to create a new list that is a concatenation of two lists and use * operator to repeat a list. See the following statements.

```
color_list1 = ["White", "Yellow"]
color_list2 = ["Red", "Blue"]
color_list3 = ["Green", "Black"]
color_list = color_list1 + color_list2 + color_list3
print(color_list)
['White', 'Yellow', 'Red', 'Blue', 'Green', 'Black']
number = [1,2,3]
print(number[0]*4)
4
print(number*4)
[1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]
```

List indices

List indices work the same way as string indices, list indices start at 0. If an index has a positive value it counts from the beginning and similarly it counts backward if the index has a negative value. As positive integers are used to index from the left end and negative integers are used to index from the right end, so every item of a list gives two alternatives indices. Let create a list called color_list with four items.
color_list=["RED", "Blue", "Green", "Black"]



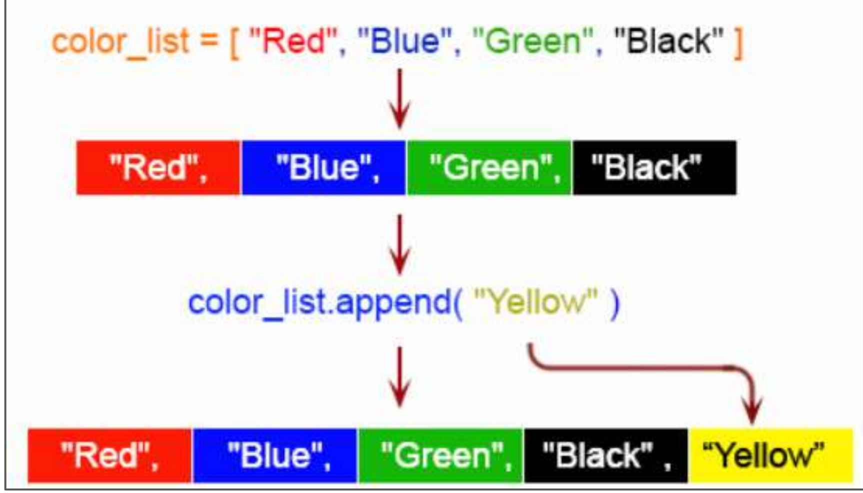
Item	RED	Blue	Green	Black
Index (from left)	0	1	2	3
Index (from right)	-4	-3	-2	-1

List indices

If you give any index value which is out of range then interpreter creates an error message. See the following statements.

```
>>> color_list=["Red", "Blue", "Green", "Black"] # The list have four elements
indices start at 0 and end at 3
>>> color_list[0] # Return the First Element
'Red'
>>> print(color_list[0],color_list[3]) # Print First and Last Elements
Red Black
>>> color_list[-1] # Return Last Element
'Black'
>>> print(color_list[4]) # Creates Error as the indices is out of range
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

Add an item to the end of the list

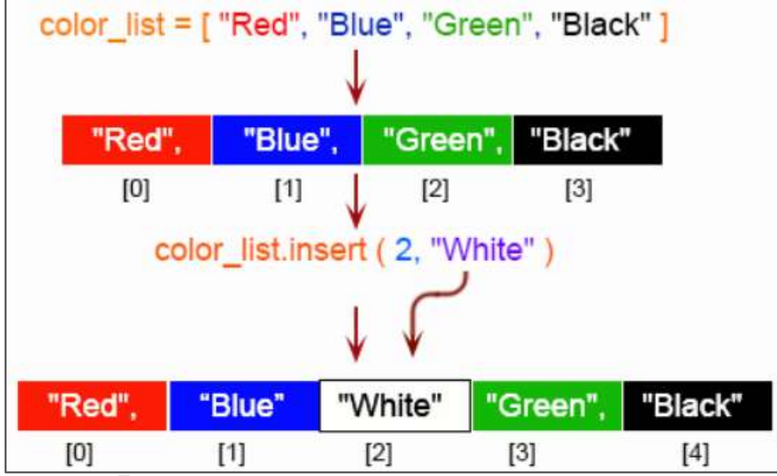

color_list = ["Red", "Blue", "Green", "Black"]
"Red",
"Blue",
"Green",
"Black"
color_list.append("Yellow")
"Red",
"Blue",
"Green",
"Black" ,
“Yellow”

See the following statements:

```
1. color_list=["Red", "Blue", "Green", "Black"]
2. print(color_list)

   ['Red', 'Blue', 'Green', 'Black']
1. color_list.append("Yellow")
2. print(color_list)

['Red', 'Blue', 'Green', 'Black', 'Yellow']
```

Insert an item at a given position

See the following statements:

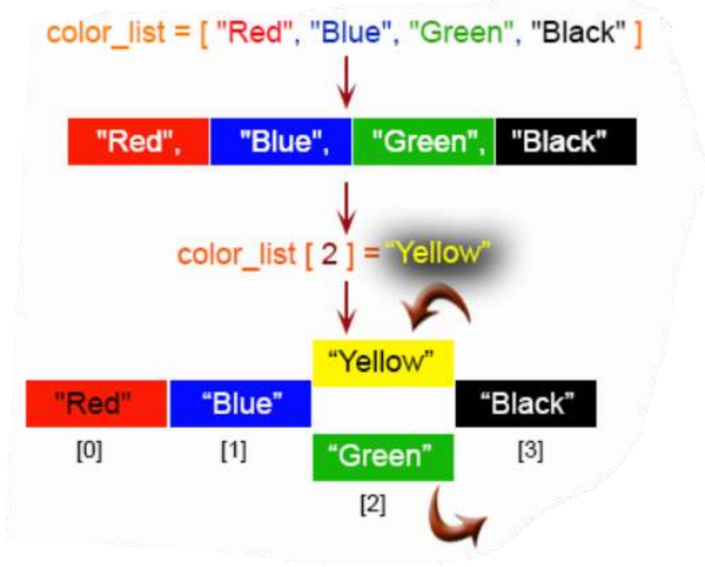


```
1. color_list=["Red", "Blue", "Green", "Black"]
2. print(color_list)

   ['Red', 'Blue', 'Green', 'Black']

1. color_list.insert(2, "White") #Insert an item at third position
2. print(color_list)
   ['Red', 'Blue', 'White', 'Green', 'Black']
```

Modify an element by using the index of the element

See the following statements:

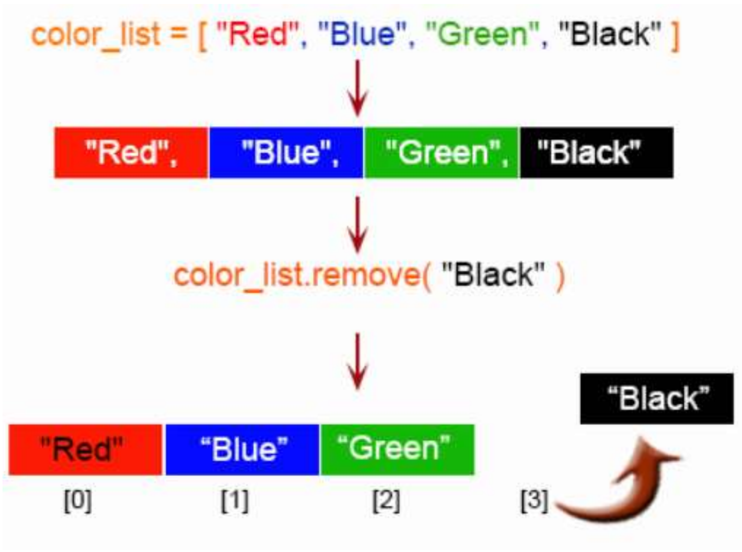
```
1. color_list=["Red", "Blue", "Green",
   "Black"]
2. print(color_list)
   ['Red', 'Blue', 'Green', 'Black']
1. color_list[2]="Yellow"  #Change the
   third color
2. print(color_list)
   ['Red', 'Blue', 'Yellow', 'Black']
```


color_list = ["Red", "Blue", "Green", "Black"]
"Red",
"Blue",
"Green",
"Black"
color_list [2] = "Yellow"
"Yellow"
"Red"
"Blue"
"Black"
[0]
[1]
"Green"
[3]
[2]

Remove an item from the list

See the following statements:

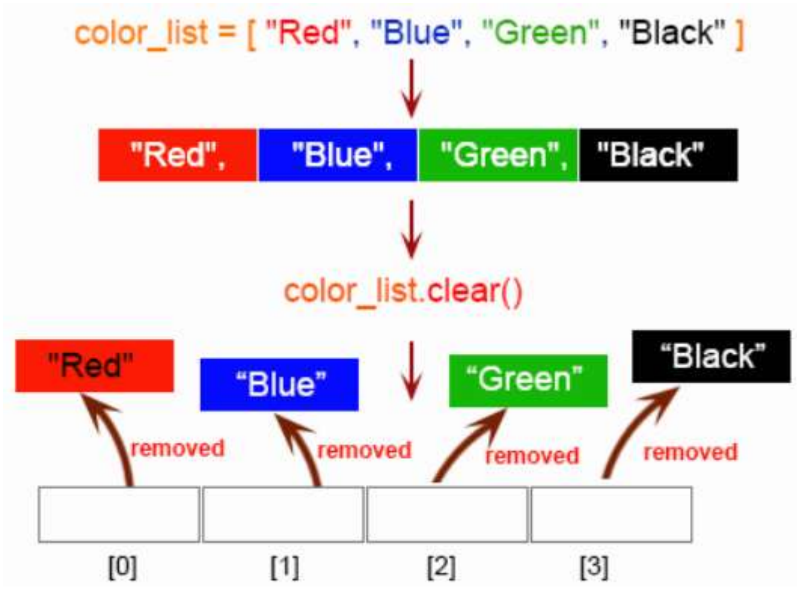
```
1. color_list=["Red", "Blue", "Green",
   "Black"]
2. print(color_list)
   ['Red', 'Blue', 'Green', 'Black']
1. color_list.remove("Black")
2. print(color_list)
   ['Red', 'Blue', 'Green']
```


color_list = ["Red", "Blue", "Green", "Black"]
"Red",
"Blue",
"Green",
"Black"
color_list.remove("Black")
"Black"
"Red"
"Blue"
"Green"
[0]
[1]
[2]
[3]

Remove all items from the list

See the following statements:

```
1. color_list=["Red", "Blue", "Green",
   "Black"]
2. print(color_list)
   ['Red', 'Blue', 'Green', 'Black']
1. color_list.clear()
2. print(color_list)
   []
```



List Slices

Lists can be sliced like strings and other sequences.

Syntax:

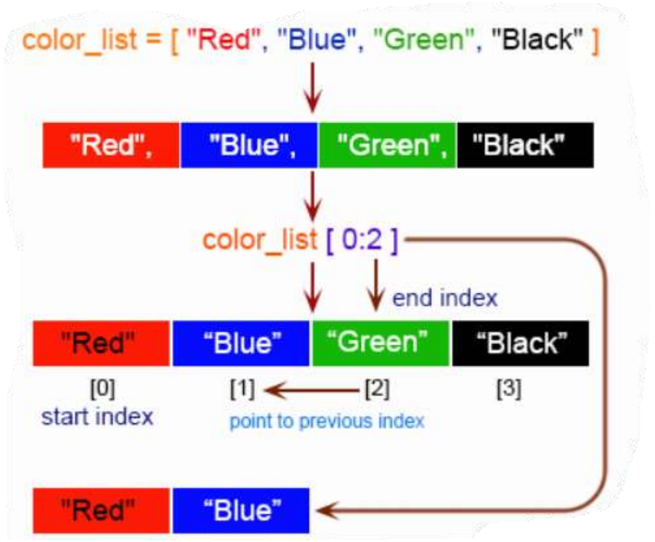
```
sliced_list = List_Name[startIndex:endIndex]
```

This refers to the items of a list starting at index startIndex and stopping just before index endIndex. The default values for list are 0 (startIndex) and the end (endIndex) of the list. If you omit both indices, the slice makes a copy of the original list.

List Slices

Cut first two items from a list:

See the following statements:

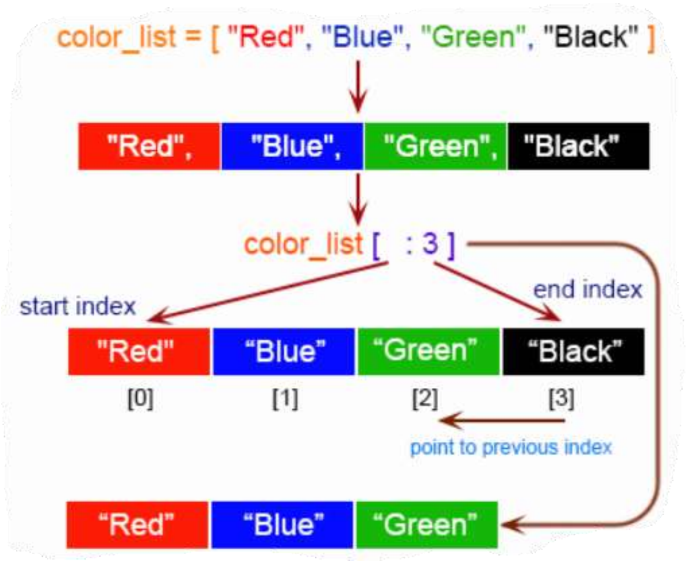


```
1. color_list=["Red", "Blue", "Green", "Black"] # The list have
   four elements
2. indices start at 0 and end at 3
3. print(color_list[0:2]) # cut first two items
   ['Red', 'Blue']
```

List Slices

Cut first three items from a list:

See the following statements:

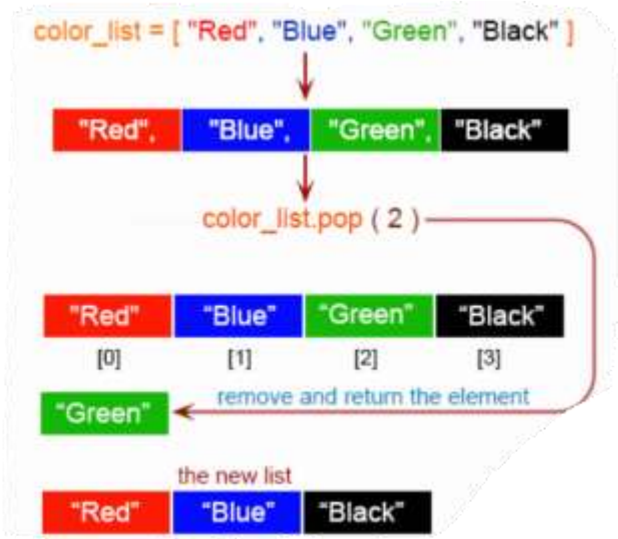

color_list = ["Red", "Blue", "Green", "Black"]
"Red", "Blue", "Green", "Black"
color_list [: 3]
start index
end index
"Red" "Blue" "Green" "Black"
[0] [1] [2] [3]
point to previous index
"Red" "Blue" "Green"

```
1. color_list=["Red", "Blue", "Green", "Black"] # The list have
   four elements
2. indices start at 0 and end at 3
3. print(color_list[:3]) # cut first three items
   ['Red', 'Blue', 'Green']
```

Remove the item at the given position in the list, and return it

See the following statements:

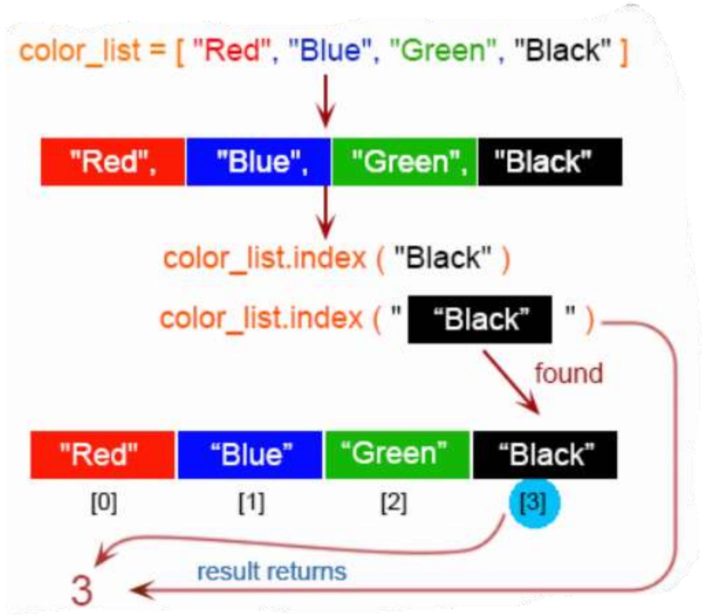
```
1. color_list=["Red", "Blue", "Green",
   "Black"]
2. print(color_list)
   ['Red', 'Blue', 'Green', 'Black']
1. color_list.pop(2) # Remove second item
   and return it
2. 'Green'
3. print(color_list)
   ['Red', 'Blue', 'Black']
```



Return the index in the list of the first item whose value is x

See the following statements:

```
1. color_list=["Red", "Blue", "Green",
   "Black"]
2. print(color_list)
   ['Red', 'Blue', 'Green', 'Black']
1. color_list.index("Red")
   0
1. color_list.index("Black")
   3
```



color_list = ["Red", "Blue", "Green", "Black"]
"Red",
"Blue",
"Green",
"Black"
color_list.index ("Black")
color_list.index (" "Black" ")
found
"Red"
"Blue"
"Green"
"Black"
[0]
[1]
[2]
[3]
3
result returns

Contest

link

Descriptors:

- solve the task by using lists
- perform arithmetic operation with list elements
- select list items by condition

Task 1. Running sum of an array - 0.25 marks

Given an array nums. We define a running sum of an array as runningSum[i] = sum(nums[0]…nums[i]).

Return the running sum of nums.

***PS: first line include len of a list**

Example 1:

Input:

```
len =4
1
2
3
4
```

Output:

```
[1,3,6,10]
```

Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

Task 2. Concatenation of Array - 0.25 marks

Given an integer array nums of length n, you want to create an array ans of length 2n where ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (0-indexed).

Specifically, ans is the concatenation of two nums arrays.

Return the array ans.

***PS: first line include len of a list**

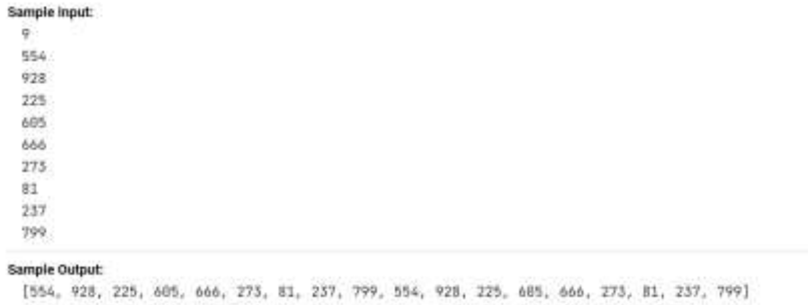
Example 1:

Input:

```
len=3
1
2
1
```

Output:

```
[1,2,1,1,2,1]
```



Sample Input:

```
9
554
928
225
605
666
273
81
237
799
```

Sample Output:

```
[554, 928, 225, 605, 666, 273, 81, 237, 799, 554, 928, 225, 685, 666, 273, 81, 237, 799]
```

Task 3. Shuffle the Array - 0.25 marks

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].

Return the array in the form [x1,y1,x2,y2,...,xn,yn].

Example 1:

Input: nums = [2,5,1,3,4,7], n = 3
Output: [2,3,5,4,1,7]
Explanation: Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the answer is [2,3,5,4,1,7].

Task 4. Count Pairs Whose Sum is Less than Target - 0.25 marks

Given a **0-indexed** integer array `nums` of length `n` and an integer `target`, return *the number of pairs* `(i, j)` *where* `0 <= i < j < n` *and* `nums[i] + nums[j] < target`.

*PS: First line include len of the list

Example 1:

```
Input: nums = [-1,1,2,3,1], target = 2
Output: 3
Explanation: There are 3 pairs of indices that satisfy the conditions in the statement:
- (0, 1) since 0 < 1 and nums[0] + nums[1] = 0 < target
- (0, 2) since 0 < 2 and nums[0] + nums[2] = 1 < target
- (0, 4) since 0 < 4 and nums[0] + nums[4] = 0 < target
Note that (0, 3) is not counted since nums[0] + nums[3] is not strictly less than the target.
```

Example 2:

```
Input: nums = [-6,2,5,-2,-7,-1,3], target = -2
Output: 10
Explanation: There are 10 pairs of indices that satisfy the conditions in the statement:
- (0, 1) since 0 < 1 and nums[0] + nums[1] = -4 < target
- (0, 3) since 0 < 3 and nums[0] + nums[3] = -8 < target
- (0, 4) since 0 < 4 and nums[0] + nums[4] = -13 < target
- (0, 5) since 0 < 5 and nums[0] + nums[5] = -7 < target
- (0, 6) since 0 < 6 and nums[0] + nums[6] = -3 < target
- (1, 4) since 1 < 4 and nums[1] + nums[4] = -5 < target
- (3, 4) since 3 < 4 and nums[3] + nums[4] = -9 < target
- (3, 5) since 3 < 5 and nums[3] + nums[5] = -3 < target
- (4, 5) since 4 < 5 and nums[4] + nums[5] = -8 < target
- (4, 6) since 4 < 6 and nums[4] + nums[6] = -4 < target
```