Question 1

Create a function that takes a list of non-negative **integers** and **strings** and return a new list without the strings.

**Examples**

filter\_list([1, 2, "a", "b"]) ➞ [1, 2]

filter\_list([1, "a", "b", 0, 15]) ➞ [1, 0, 15]

filter\_list([1, 2, "aasf", "1", "123", 123]) ➞ [1, 2, 123]

Ans: You can create a Python function to filter out the strings from a list of non-negative integers and strings. Here's the function:

def filter\_list(lst):

return [item for item in lst if isinstance(item, int)]

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(filter\_list([1, 2, "a", "b"])) # Output: [1, 2]

print(filter\_list([1, "a", "b", 0, 15])) # Output: [1, 0, 15]

print(filter\_list([1, 2, "aasf", "1", "123", 123])) # Output: [1, 2, 123]

In this function, we use a list comprehension to iterate through each item in the input list `lst`. We include only those items in the new list for which the condition `isinstance(item, int)` is true, meaning they are non-negative integers. This filters out the strings from the list.

Example usage:

filter\_list([1, 2, "a", "b"]) ➞ [1, 2]

filter\_list([1, "a", "b", 0, 15]) ➞ [1, 0, 15]

filter\_list([1, 2, "aasf", "1", "123", 123]) ➞ [1, 2, 123]

Question 2

The "Reverser" takes a string as input and returns that string in reverse order, with the opposite case.

### Examples

reverse("Hello World") ➞ "DLROw OLLEh"

reverse("ReVeRsE") ➞ "eSrEvEr"

reverse("Radar") ➞ "RADAr"

ANS: def reverse(input\_str):

reversed\_str = input\_str[::-1]

return ''.join(char.swapcase() for char in reversed\_str)

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(reverse("Hello World")) # Output: "DLROw OLLEh"

print(reverse("ReVeRsE")) # Output: "eSrEvEr"

print(reverse("Radar")) # Output: "RADAr"

Example :

reverse("Hello World") ➞ "DLROw OLLEh"

reverse("ReVeRsE") ➞ "eSrEvEr"

reverse("Radar") ➞ "RADAr"

Question 3

You can assign variables from lists like this:

lst = [1, 2, 3, 4, 5, 6]

first = lst[0]

middle = lst[1:-1]

last = lst[-1]

print(first) ➞ outputs 1

print(middle) ➞ outputs [2, 3, 4, 5]

print(last) ➞ outputs 6

With Python 3, you can assign variables from lists in a much more succinct way. Create variables first, middle and last from the given list using **destructuring assignment** (check the **Resources** tab for some examples), where:

first ➞ 1

middle ➞ [2, 3, 4, 5]

last ➞ 6

Your task is to unpack the list writeyourcodehere into three variables, being first, middle, and last, with middle being everything in between the first and last element. Then print all three variables.

Ans: You can use destructuring assignment in Python to unpack the elements of a list into separate variables. Here's how you can do it for the given list:

writeyourcodehere = [1, 2, 3, 4, 5, 6]

first, \*middle, last = writeyourcodehere

print(first) # Output: 1

print(middle) # Output: [2, 3, 4, 5]

print(last) # Output: 6

In this code, we use `first, \*middle, last` to unpack the elements of the `writeyourcodehere` list. The variable `first` will be assigned the first element (1), `last` will be assigned the last element (6), and the variable `middle` will be assigned the elements in between the first and last element ([2, 3, 4, 5]). The `\*` before `middle` is used to collect all the elements between the first and last element into the list `middle`.

When you run this code, it will produce the desired output:

1

[2, 3, 4, 5]

6

Question 4

Write a function that calculates the **factorial** of a number **recursively**.

### Examples

factorial(5) ➞ 120

factorial(3) ➞ 6

factorial(1) ➞ 1

factorial(0) ➞ 1

ANS: def factorial(n):

if n == 0 or n == 1:

return 1

else:

return n \* factorial(n - 1)

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(factorial(5)) # Output: 120

print(factorial(3)) # Output: 6

print(factorial(1)) # Output: 1

print(factorial(0)) # Output: 1

Example usage:

factorial(5) ➞ 120

factorial(3) ➞ 6

factorial(1) ➞ 1

factorial(0) ➞ 1

Question 5

Write a function that moves all elements of one type to the **end** of the list.

### Examples

move\_to\_end([1, 3, 2, 4, 4, 1], 1) ➞ [3, 2, 4, 4, 1, 1]

# Move all the 1s to the end of the array.

move\_to\_end([7, 8, 9, 1, 2, 3, 4], 9) ➞ [7, 8, 1, 2, 3, 4, 9]

move\_to\_end(["a", "a", "a", "b"], "a") ➞ ["b", "a", "a", "a"]

ANS: def move\_to\_end(lst, element):

count = lst.count(element)

lst = [item for item in lst if item != element]

lst.extend([element] \* count)

return lst

# Test the function

if \_\_name\_\_ == "\_\_main\_\_":

print(move\_to\_end([1, 3, 2, 4, 4, 1], 1)) # Output: [3, 2, 4, 4, 1, 1]

print(move\_to\_end([7, 8, 9, 1, 2, 3, 4], 9)) # Output: [7, 8, 1, 2, 3, 4, 9]

print(move\_to\_end(["a", "a", "a", "b"], "a")) # Output: ["b", "a", "a", "a"]

Example usage:

move\_to\_end([1, 3, 2, 4, 4, 1], 1) ➞ [3, 2, 4, 4, 1, 1]

move\_to\_end([7, 8, 9, 1, 2, 3, 4], 9) ➞ [7, 8, 1, 2, 3, 4, 9]

move\_to\_end(["a", "a", "a", "b"], "a") ➞ ["b", "a", "a", "a"]