**Name some commonly used library in pythons?**

os

csv

subprocess

copy

requests

re

**How to use logging in python?**

**What are features of python?**

Python is a dynamic, high level, free open source and interpreted programming language

Free and open source

Object Oriented

Interpreted language

Large standard library

Dynamically typed language

**How to check size of file using fucntion**

In python’s os.path module there is function get\_size() which returns the file size in Bytes

os.path.getsize(path) --- to get the size for given path in bytes

**How to get size of directory in python**

No builtin method, use walk function to get file size of each file and sum that value

**What is use of walk() in os module**

It walks from give path to each subdirectory/subdirectory.

It returns tuple of three variable.

path, dir, file=os.walk(path, topdown=True)

path – represents where it is

dir ---- gives all directory in current path as list form

file ---- gives all files in that path in list

**Example:**

for (path,dirs,files) in os.walk('D:\Ajanta', topdown=true):

print (path)

print (dirs)

print (files)

print ('--------------------------------')

**What is zip() function in Python**

Zip function take each argument from an iterable and create a new zip object after picking single object from each iterables.

z=zip(a,b,c..)

**Use zip function for fetching elements from three iterables and data from one iterable must come in reverse order.**

list1 = [10, 20, 30, 40]

list2 = [100, 200, 300, 400]

list3=range(4)

for x, y,z in zip(list1, list2[::-1],list3):

    print(x, y, z)

If length of each object is not equal then result will have length of minimum of them.

x=list(range(10))

y=list(range(5))

print(x)

print(y)

z=zip(x,y)

for each in z:

    print(each)

output

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

[0, 1, 2, 3, 4]

(0, 0)

(1, 1)

(2, 2)

(3, 3)

(4, 4)

**What is the difference between remove() function and del statement?**

remove() ----- delete/removes given value from list ---- takes values want to remove

del --------- it deletes/removes value from given index --- takes reference kind of thing

**WAP to get the ASCII code of each character**

We can use ord() to get the ASCII value of any character.

ord(‘e’) // 101

**Python chr()**

It returns character/string equivalent of a number. (internally it assumes inter as ascii value and convert them into it equivalent character).

The valid range of the integer is from 0 through 1,114,111.

print(chr(97)) #a

print(chr(11)) #♂

print(chr(57)) #9

**WAP a program to get values and key in a tuple or list of dictionaries.**

Let say dictionary variable name is d

Key\_tuple=tuple(d.keys()) // get keys in tuple

Key\_list=list(d.keys() // get keys in list

*Keys() or values()method of dictionary gives dictionary view that will not be iterable. We can convert them to any iterable data using required methos(list(), tuple())*

**Create a function that converts a date formatted as MM/DD/YYYY to YYYYDDMM**

**WAP to create a dictionary using below keys and values.**

keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

Method 1:

keys = ['Ten', 'Twenty', 'Thirty']

values = [10, 20, 30]

d={}

for i in range(len(keys)):

    d[keys[i]]=values[i]

[print(d)]

Method 2:

d=dict(zip(keys,values))

print(d)

**What is python dictionary comprehension?**

Dictionary comprehension is an elegant and concise way to create a new dictionary from an iterable in Python

Dictionary comprehension consists of an expression pair (key: value) followed by a for statement inside curly braces {}

Example:

# Dictionary Comprehension

squares = {x: x\*x for x in range(6)}

print(squares)

**Question**

**Create a new dictionary by extracting the following keys from a below dictionary**

given dictionary----

sampleDict = {

"name": "Kelly",

"age":25,

"salary": 8000,

"city": "New york"

}

Key to extract----

keys = ["name", "salary"]

**Method 1:** ------- Using dictionary comprehension

d={k:sampleDict[k] for k in keys}

print(d)

method 2 : ---------- Traditional way

sampleDict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

keys = ["name", "salary"]

#{'name': 'Kelly', 'salary': 8000}

sample\_keys=list(sampleDict.keys())

d={}

for each in sample\_keys:

    if each in keys:

        d[each]=sampleDict[each]

print(d)

**How can we test if a given key is present in a dictionary?**

We can do it using membership operator (IN). For dictionary membership checks for key presence.

squares = {1: 1, 3: 9, 5: 25, 7: 49, 9: 81}

print(49 in squares) /// False, key 49 is not present in dictionary

**Delete set of keys from a dictionary**

sampleDict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

keysToRemove = ["name", "salary"] # we have to remove these keys

for each in keysToRemove:

    del sampleDict[each]

print(sampleDict)

**Rename the given key in dictionary but value will be same.**

For given dictionary rename ‘city’ key to ‘location’

sampleDict = {

"name": "Kelly",

"age":25,

"salary": 8000,

"city": "New york"

}

For above given key change the key ‘city’ to location but value will be preserved.

sampleDict = {

  "name": "Kelly",

  "age":25,

  "salary": 8000,

  "city": "New york"

}

sampleDict['location']=sampleDict.pop('city')

print(sampleDict)

**Get the key of a minimum value from the following dictionary**

For given dictionary find out the key whose value is minimum.

sampleDict = {

'Physics': 82,

'Math': 65,

'history': 75

}

Method 1:

sampleDict = {

  'Physics': 82,

  'Math': 65,

  'history': 75

}

d=sorted(sampleDict.items(),key=lambda x:x[1])

print(d[0][0])

Method 2: Using zip

sampleDict = {

  'Physics': 82,

  'Math': 65,

  'history': 75,

  'chem':21

}

z=zip(sampleDict.keys(),sampleDict.values())

print(z)

d=sorted(z,key=lambda x:x[1])

print(d)

**Sort a list which is list of list in which sorting will be performed by the value at position 3 of each sublist.**

<https://www.hackerrank.com/challenges/decorators-2-name-directory/problem>

input = [['Mike', 'Thomson', '20', 'M'], ['Robert', 'Bustle', '32', 'M'], ['Andria', 'Bustle', '30', 'F']]

output=[['Mike', 'Thomson', '20', 'M'], ['Andria', 'Bustle', '30', 'F'], ['Robert', 'Bustle', '32', 'M']]

input\_val = [['Mike', 'Thomson', '20', 'M'], ['Robert', 'Bustle', '32', 'M'], ['Andria', 'Bustle', '30', 'F']]

output=[['Mike', 'Thomson', '20', 'M'], ['Andria', 'Bustle', '30', 'F'], ['Robert', 'Bustle', '32', 'M']]

l1=sorted(input\_val,key=lambda x:x[2])

print(input\_val)

print(l1)

**Concatenate below two lists index-wise**

list1 = ["M", "na", "i", "Ke"]

list2 = ["y", "me", "s", "lly"]

Method 1:

result=[list1[i]+list2[i] for i in range(len(list1))]

Method 2:

list1 = ["M", "na", "i", "Ke"]

list2 = ["y", "me", "s", "lly"]

list3 = [i + j for i, j in zip(list1, list2)]

print(list3)

**Concatenate two lists in the following order**

list1 = ["Hello ", "take "]

list2 = ["Dear", "Sir"]

Output--

['Hello Dear', 'Hello Sir', 'take Dear', 'take Sir']

list1 = ["Hello ", "take "]

list2 = ["Dear", "Sir"]

result=[i+j for i,j in zip(list1,list2)]

print(result)

**Add blank space at end of each element if given list.**

list2 = ["Dear", "Sir"]

list2 = ["Dear", "Sir"]

list2=[each+' ' for each in list2]

print(list2)

**Given a two Python list. Iterate both lists simultaneously such that list1 should display item in original order and list2 in reverse order.**

list1 = [10, 20, 30, 40]

list2 = [100, 200, 300, 400]

Method 1: Using for loop

for each in range(len(list2)):

    print(list1[each],' ',list2[-each-1])

Method 2: Using zip function

for x, y in zip(list1, list2[::-1]):

    print(x, y)

**Remove empty string from a list/iterable**

list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

output must be-

["Mike", "Emma", "Kelly", "Brad"]

Method 1 : Using filter function

list1 = ["Mike", "", "Emma", "Kelly", "", "Brad"]

result=list(filter((lambda x: len(x)!=0),list1))

print(result)

**What is sorted function?**

The sorted() function sorts the elements of a given iterable in a specific order (either ascending or descending) and returns the sorted iterable as a list.

*sorted(iterable, key=None, reverse=False)*

key--- A Function to execute to decide the order. Default is None and optional

**Sort a tuple of tuples by 2nd item**

tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))

tuple1 = tuple(sorted(list(tuple1), key=lambda x: x[1]))#tell to sort on index of 1

print(tuple1)

tuple1 = (('a', 23),('b', 37),('c', 11), ('d',29))

result=sorted(tuple1,key=lambda x:x[1])

**Sort a dictionary by value of key**

X=any dictioanry

{k: v for k, v in sorted(x.items(), key=lambda item: item[0])}

{0: 0, 2: 1, 1: 2, 4: 3, 3: 4}

**Question** --- **Sort the below list based on year.**

cars = [

{'car': 'Ford', 'year': 2005},

{'car': 'Hundai', 'year': 2000},

{'car': 'BMW', 'year': 2019},

{'car': 'Thor', 'year': 2011}

]

**Solution** :

x=sorted(cars,key=lambda x:x['year'])

**Python any()**

The any() function returns True if any element of an iterable is True. If not, any() returns False.

any(iterable) -----Syntax

it is applicable for any iterable data(list, set, tuple, dict(if dict then checks for keys))

d = {}

print(any(d))#False

d = {'0': 'False'}

print(any(d)) # True , b/c ‘0’ is string not number

**Python all()**

The all() function returns True if all elements in the given iterable are true. If not, it returns False.

all(iterable)

this is same as any() but it check for True on all elements, if on dict then check of keys.

**Python delattr()**

The delattr() deletes an attribute from the object (if the object allows it).

delattr(object\_name, var\_name) -------- syntax

e.g -- let say we have a class XYZ and then want to reomve any class varaible then we can do it using this method

* object - the object from which name attribute is to be removed
* name - a string which must be the name of the attribute to be removed from the object

class Coordinate:

  x = 10

  y = -5

  z = 0

point1 = Coordinate()

delattr(Coordinate,'z') #deleted class variable z

print(point1.z) #will get error b/c it's deleted

Python dir()

The dir() method tries to return a list of valid attributes of the object.

**What is PIP in python**

PIP is a package manager for Python packages, or modules.

The most common use of pip is for installing library/packages

pip install package\_name

pip uninstall package\_name

pip –version

**What is difference between GET and POST**

GET

Get method is used for fetching/requesting/getting data.

In GET methods is passing any data then that will be visible in URL

POST

In POST methods is passing any data then that will not be visible in URL

**What is iterator and generator and difference**

The iterable object is the one you iterate over its elements.

The iterator object is the one that produces the values during the iteration and it also being returned by the iterable object.

Iterator can be created easily when iter() is called on it and next() on iterator returns the next value from the iterator.

Iterators are objects that have iter() and a next() method.

Let’s take an example of creating iterator.

l=[1,2,3]

iter\_obj=iter(l) # iter\_obj is iterator object

print(next(iter\_obj)) #using next() to get objects one by one

An iterator is an object which contains a countable number of values and it is used to iterate over iterable objects like list, tuples, sets, etc.

Generators provide an easy, built-in way to create instances of Iterators thus we can say ---

Every generator is an iterator, but not vice versa

Every Iterator is also an Iterable, but not every Iterable is an Iterator

**What is an Iterable**

In Python, an Iterable is an object that implements the **\_\_iter\_\_()** method and returns an iterator object or an object that implements \_\_getitem\_\_() method (and should raise an IndexError when indices are exhausted). Built-in iterable objects include Lists, Sets and Strings as such sequences can be iterated over -say- in a for-loop.

**What is an Iterator**

On the other hand, an Iterator in Python is an object that implements the **\_\_next\_\_()** method in a way that

**What is decorator in python --- Interview**

A decorator is a design pattern in Python that allows a user to add new functionality to an existing object without modifying its structure.

**Function based decorator in python**

def my\_decorator(func):

    #some codes

    def wrapper(\*args,\*\*kwargs):

        #some codes or actions

        return func(\*args,\*\*kwargs)

    return wrapper

class based decorator

class MyDecorator:

    def \_\_init\_\_(self,func):

        self.func=func

    def \_\_call\_\_(self, \*args,\*\*kwds):

        return self.func(\*args,\*\*kwargs)

@MyDecorator

def func1():

#statements

Example: -- Class based decorator

class MyDecorator:

def \_\_init\_\_(self,func):

print('inizialization')

self.func=func

def \_\_call\_\_(self,a,b):

print('call function')

if a>b:

return self.func(a,b)

else:

return self.func(b,a)

@MyDecorator

def div(a,b):

print('div function')

return a/b

print(div(2,5))

**how to create decorator for a function:**

Use **@function\_name** decorator for function which you have created decorator.

def my\_decorator(func):

    #some codes

    def wrapper(\*args,\*\*kwargs):

        #some codes or actions

        return func(\*args,\*\*kwargs)

    return wrapper

@my\_decorator

def m1(\*args,\*\*kwargs):

pass

**What is use of \_\_call\_\_() in python.**

The main use of \_\_call\_\_ in method is to use the class instance as a function name.

**Example:**

class Example:

    def \_\_init\_\_(self):

        print("Instance Created")

    # Defining \_\_call\_\_ method

    def \_\_call\_\_(self):

        print("Instance is called via special method")

# Instance created

e = Example() #this is call \_\_init\_\_

**# \_\_call\_\_ method will be called**

**e()**

**What is GIL and it’s action.**

GIL --- Global interpreter lock.

GIL protects the reference counter by disabling multi-threading in Python.

Python has a “reference-counter” for memory management, whenever we declare any variable then it counts number of references referencing that variable. GIL stops accidently changing the value of reference-counter.

import sys

my\_variable = 'apple'

x = my\_variable

sys.getrefcount(my\_variable) #3

First time when my\_variable is initialized, second time when value assigned to x third time when getting it’s count/value.

**Argument vs parameter from a function's perspective:**

A parameter is the variable listed inside the parentheses in the function definition.

An argument is the value that are sent to the function when it is called.

**What is pass by value and pass by reference?**

In a pass by value, the parameter value copies to another variable while, in a pass by reference, the actual parameter passes to the function.

Python’s argument-passing model is neither “Pass by Value” nor “Pass by Reference” but it is “Pass by Object Reference”. Immutable objects show “pass by value” whereas mutable objects show “pass by reference”.

|  |  |
| --- | --- |
| Pass by reference | Pass by argument/reference |
| def m1(x):      x.append(10)      return x  l=[1,2,3]  print(m1(l))  print(l)  Output:  [1, 2, 3, 10]  [1, 2, 3, 10] | def m1(x):      x.append(10)      return x  l=[1,2,3]  print(m1([1,2,3]))  print(l)  Output:  [1, 2, 3, 10]  [1, 2, 3] |

**What is list comprehension and double list comprehension? ------- Interview**

List comprehension is way to creating list in one line with iterable data.

We have below 3-4 type of list comprehension

**l1=[ x for x in iterable ]**

**l=[x for x in iterable if condition]** ----- for with if statement

----if ‘condition’ evaluates to True then it return x in list else nothing

**l= [exp if cond else on\_false\_value for x in iterable ]** ---- for with if-else

--- if ‘cond’ evaluates to True then it x goes in list else on\_false\_value goes into list.

**Double list comprehension:**

It’s using double for loop in for creating list how we use in list comprehension.

|  |  |
| --- | --- |
| res=[x+y for x in list('axeds') for y in list('hdoes')] | Output:  ['ah', 'ad', 'ao', 'ae', 'as', 'xh', 'xd', 'xo', 'xe', 'xs', 'eh', 'ed', 'eo', 'ee', 'es', 'dh', 'dd', 'do', 'de', 'ds', 'sh', 'sd', 'so', 'se', 'ss'] |

**Filtering in list comprehension?**

my\_list = [2, 3, 5, 7, 11]

squared\_list = [x\*\*2 for x in my\_list if x%2 != 0]    # list comprehension

# output => [9 , 25 , 49 , 121]

squared\_dict = {x:x\*\*2 for x in my\_list if x%2 != 0}    # dict comprehension

# output => {11: 121, 3: 9 , 5: 25 , 7: 49}

**Flattening a multi-dimensional list**

my\_list = [[10,20,30],[40,50,60],[70,80,90]]

flattened = [x for temp in my\_list for x in temp]

**What is use of extend() in python ---- Interview**

In python list extend() is used to extend data from one iterable into a list. It returns None.

l.extend(iterable ) ---------- Syntax

iterable can be list, tuple, string (all character of string will be extended), dictionary (key will be extended) , set etc.

* list ---- one list into another list
* tuple --- tuple will be extended into list.
* String ---- all character of string will be extended
* dictionary ------ key will be extended

**Question**: - Find the output of below lines-

l = [1, 2, 3]

l.extend(150)

Answer ---- Error, extend method takes iterable data but we are passing integer

**Question**: - Find the output of below lines-

l = [1, 2, 3]

l.extend("XYZ")

Answer: -- [1, 2, 3,, 'X', 'Y', 'Z']

**Question**: - Find the output of below lines-

l = [1, 2, 3]

l.append((3, 4))

Answer: ---- [1, 2, 3, (3, 4)]

**Question**: - Find the output of below lines-

l = [1, 2, 3]

l.extend({"a": 10, "b":20})

Answer:-- [1, 2, 3, 'a', 'b']

**List append vs extend**

append ---- if want to add one item/native data into another list

extend ----- Want to add iterable data into a list.

**What is use of add and update method in set.**

Set doesn’t contains duplicate values and have no ordering.

**add**

add() ---- this method is used to add one data/element into a set.

Example:

prime\_numbers = {2, 3, 5, 7}

prime\_numbers.add(11) #{{2, 3, 5, 7, 11}

**update**

--- this is used to add some iterable data into set as like extend of python

A.update(B) --- A is a set and B can be any iterable like list, set, dictionary, string, etc.

A.update(B, C, D) --- B, C, D - iterables whose items are added to set A

|  |  |
| --- | --- |
| s. add(x)   * adds the x to the existing set s if x is not present. * X must be immutable else error.   Note: Set is mutable data type | s.update(x)   * updates (add each element of x into s) the existing s by each value of x if x not present in s. * x can be immutable or mutable. |

**Example**: ---- Interview

s = {2, 3, 5, 7}

s.add(12) #{2, 3, 5, 7, 12}

s.update({'hello','hii'}) #{2, 3, 5, 'hello', 7, 12, 'hii'}

**Example: 2** ---- Interview

s = {2, 3, 5, 7}

s.update({'hello1':2,'hii':2121}) #{2, 3, 'hii', 5, 7, 'hello1'}

**Example 3**: ---- updating/adding multiple iterable into set

s = {2, 3, 5, 7}

s.update({'hello1':2,'hii':2121},[22,21]) # {'hello1', 2, 3, 5, 'hii', 7, 22, 21}

**Is python interpreted language?**

Yes, python is interpreted language.

Source code ---------------> Intermediatory code (.py) -----------> Machine level code

**Explain docstring in Python?**

The Python docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. It is all about the information purpose.

Python uses triple quotes to create docstrings even though the string fits on one line.

Docstring phrase ends with a period (.) and can be multiple lines. It may consist of spaces and other special chars.

# One-line docstrings

def hello():

"""A function to greet."""

return "hello"

**# Question #**

Letter combination of phone number from leetcode.

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order.

<https://leetcode.com/problems/letter-combinations-of-a-phone-number/description/>



**Input:** digits = "23"

**Output:** ["ad","ae","af","bd","be","bf","cd","ce","cf"]

Algo:

First create a list with empty string( say --- result\_list).

For each character value of number, append the character to each value of result list and update the result\_list.

class Solution:

    def letterCombinations(self, digits):

        d={'2':'abc',

        '3':'def',

        '4':'ghi',

        '5':'jkl',

        '6':'mno',

        '7':'pqrs',

        '8':'tuv',

        '9':'wxyz'}

        if len(digits)==0:

            return []

        results=[""]

        for each in digits:

            print(each)

            temp=[]

            for ch in d[each]:

                print(ch)

                for res in results:

                    temp.append(res+ch)

            results=temp

        return results

Method 2:

class Solution:

    def letterCombinations(self, digits):

        d={'2':'abc',

        '3':'def',

        '4':'ghi',

        '5':'jkl',

        '6':'mno',

        '7':'pqrs',

        '8':'tuv',

        '9':'wxyz'}

        if len(digits)==0:

            return []

        characters=list(d[digits[0]])

        print(characters)

        for num in digits[1:]:

            #print(num)

            temp=[]

            for ch in characters:

                for new in list(d[num]):

                    temp.append(ch+new)

            characters=temp

        print(sorted(characters))

**# Leetcode question# -- Search Insert Position**

Given a sorted array(nums) of distinct integers and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

Algo:

It’s given that numbers are in sorted order in question

If target < nums[0] ---- insert at index 0

If target > num [-1] ---- insert at last or at index=len(nums)

If we do (target-nums[i]) --- then this will be always in descending order b/c nums in ascending

Now check each value of (target-nums[i]) where ever it’s<0 then target should be inserted at that position/index

class Solution:

    def searchInsert(self, nums, target):

        d={}

        if target<nums[0]:

            return 0

        if target>nums[-1]:

            return len(nums)

        if target in nums:

            for i in range(len(nums)):

                if nums[i]==target:

                    return i

        l=[target-each for each in nums]

        min\_value=0

        #iterate in each value of diff

        for i in range(len(l)):

            if l[i]<0:#mean we have to insert at this index

                return i

**Method 2 ---- Using binary search or Olog(n) complexity**

class Solution:

    def searchInsert(self, nums: List[int], target: int) -> int:

        #check for bounry condition

        if nums[-1]<target:

            return len(nums)

        if nums[0]>target:

            return 0

        #do a binary search to get the position where target-val is 0

        N=len(nums)

        l=0

        r=len(nums)-1

        while l<r:

            middle=(l+r)//2

            if target<=nums[middle]:

                r=middle

            else:

                l=middle+1

        return l

Method 3: O(log(n))

class Solution:

    def searchInsert(self, nums, target):

        if target in nums:

            return nums.index(target)

        if target<nums[0]:

            print('\*')

            return 0

        if target>nums[-1]:

            return len(nums)

        else:

            l=0

            r=len(nums)-1

            while l<r:

                middle=(l+r)//2

                mid\_value=nums[middle]

                #if target is suppose to be in left

                if mid\_value>target:

                    if middle-1>=0 and nums[middle-1]<target:

                        return middle

                    else:

                        r=middle-1

                if mid\_value<target:

                    if middle+1<len(nums) and nums[middle+1]>target:

                        return middle+1

                    else:

                        l=middle+1

        return l

s=Solution()

print(s.searchInsert(nums = [1,3], target = 2))

**#Question --- Remove nth element from end of linkedlist#**

<https://leetcode.com/problems/remove-nth-node-from-end-of-list/description/>

**Method 1:- Using two pointer solution**

Algo:-

Nth from last= length\_from\_linkedlist-n-1

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def removeNthFromEnd(self, head: Optional[ListNode], n: int) -> Optional[ListNode]:

        print(n)

        l=0

        current=head

        while current:

            l=l+1

            current=current.next

        if l==n:

            return head.next

        elif l==1:

            return []

        else:

            start=0

            n\_th\_from\_start=l-n-1

            #reach to one prevous of node which need to be remoed

            current=head

            while start<n\_th\_from\_start:

                current=current.next

                start=start+1

            current.next=current.next.next

            return head

Method 2: --- Solve it using one pass:

In one pass, we will maintain two pointer, slow and fast. For given value of n we will not move the slow pointer but fast pointer will keep on moving.

When fast pointer reaches to next slow will be one node behind which need to be removed.



Now at this point – slow.next=slow.next.next

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def removeNthFromEnd(self, head: Optional[ListNode], n: int) -> Optional[ListNode]:

        dummy=slow=fast=ListNode(0,next=head)

        #skip moving slow pointer till n times

        for i in range(n):

            fast=fast.next

        #Now move the slow and fast to next node till fast reached to end of linked list

        while fast:

            fast=fast.next

            slow=slow.next

        #now move next of slow-->next to next of sloq

        slow.next=slow.next.next

        return dummy

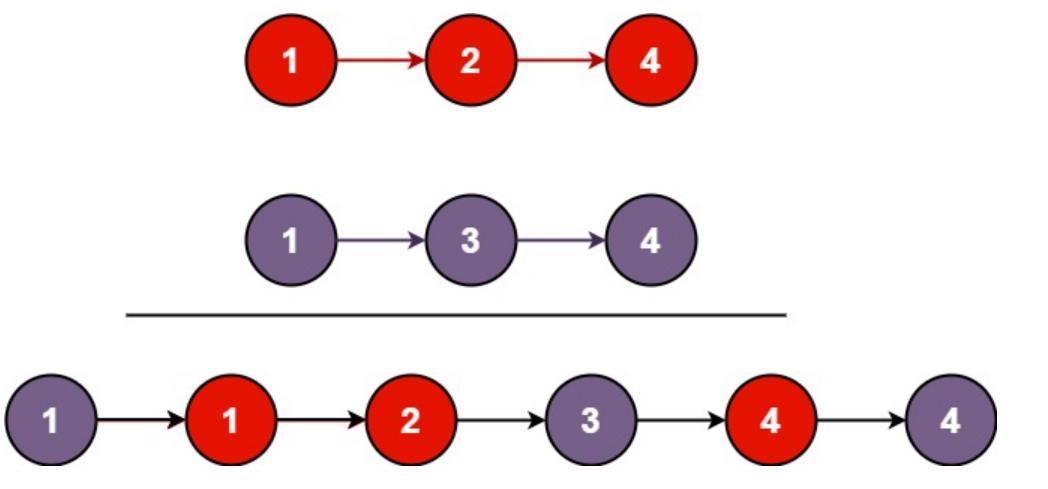
# Question --- Merge two sorted linked list#

<https://leetcode.com/problems/merge-two-sorted-lists/description/>

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists.

Return the head of the merged linked list.



**Method 1: --** **appending data into list (list\_data) and then creating linked list from list (list\_data)**

Algo:

Get the data from both linked list in one list( say – list\_data) and sort it.

Now create a head and start pointer initialize with value – 0.

Now for each value of list\_data create a new node and assign next of start to this value and update the start pointer.

At end return the head.next.

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:

        l=[]

        while list1:

            l.append(list1.val)

            list1=list1.next

        while list2:

            l.append(list2.val)

            list2=list2.next

        #got all data in l

        l=sorted(l)

        print(l)

        start=my\_head=ListNode(val=0,next=None)

        #now for each value of l update the next value of start and start

        for each in l:

            #next of start node => new node

            start.next=ListNode(val=each,next=None)

            #new start => start.next

            start=start.next

        return my\_head.next

Method 2:

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def mergeTwoLists(self, list1: Optional[ListNode], list2: Optional[ListNode]) -> Optional[ListNode]:

        # A dummy node to store the result

        dummyNode = ListNode(0)

        # Tail stores the last node

        tail = dummyNode

        while True:

            # If any of the list gets completely empty

            # directly join all the elements of the other list

            if list1 is None:

                tail.next = list2

                break

            if list2 is None:

                tail.next = list1

                break

            # Compare the data of the lists and whichever is smaller is

            # appended to the last of the merged list and the head is changed

            if list1.val <= list2.val:

                tail.next = list1

                list1 = list1.next

            else:

                tail.next = list2

                list2 = list2.next

            # Advance the tail

            tail = tail.next

        # Returns the head of the merged list

        return dummyNode.next

Question --- LeetCode

Check valid parenthesis

class Solution:

    def isValid(self, s: str) -> bool:

        #if lenght of s is null then it's defenitely unmatched paranthesis

        if len(s)%2!=0:

            return False

        else:

            stack=[]

            open\_bracket\_dict={'(':')','{':'}','[':']'}

            for each in s:

                #if it is opening backet tyhen simply push to stack

                if each in open\_bracket\_dict.keys():

                    stack.append(each)

                #if not opening backet

                else:

                    #get he last brack from stack which will be open type

                    if len(stack)>0:

                        last\_from\_stack=stack[-1]

                        #close bracket for last value of stack

                        close\_par=open\_bracket\_dict[last\_from\_stack]

                        #close par of string and from dict must be same

                        if close\_par==each:

                            stack.pop()

                            continue

                        else:

                            return False

                    else:

                        return False

            return stack==[]

Hint:

<https://geekflare.com/valid-parentheses-checking-python/>

**Question --- Generate parenthesis---- Leetcode**

You are given a number n from that you have to generate valid parenthesis.

Example-

**Input:** n = 1

**Output:** ["()"]

**Input:** n = 3

**Output:** ["((()))","(()())","(())()","()(())","()()()"]

**Method 1:** -------- Brut force. Generate all possible combination of parenthesis and filter valid.

Method 2: ---- Using recursion

Algo/Hint:

Us recursive approach.

def generate\_parenthesis(self,n):

        out=[]

        def rec(left,right,left\_par\_cnt\_in\_str,my\_str):

            if left==0 and right==0:

                out.append(my\_str)

                return

            if left>0:

                left=left-1

                my\_str=my\_str+'('

                left\_par\_cnt\_in\_str=left\_par\_cnt\_in\_str+1

                rec(left,right,left\_par\_cnt\_in\_str,my\_str)

            #if right>0 and there are some open parenthesis in string

            if right>0 and left\_par\_cnt\_in\_str>0:

                right=right-1

                left\_par\_cnt\_in\_str=left\_par\_cnt\_in\_str-1

                my\_str=my\_str+')'

                rec(left,right,left\_par\_cnt\_in\_str,my\_str)

        rec(n,n,0,'')

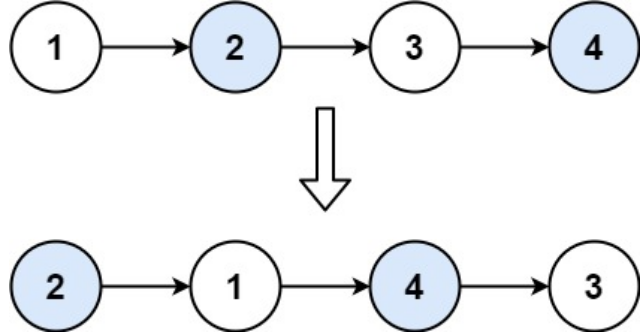
        return out

**Question --- Swap nodes in pairs---- Leetcode**

<https://leetcode.com/problems/swap-nodes-in-pairs/description/>

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example-



**Input:** head = [1,2,3,4]

**Output:** [2,1,4,3]

**Input:** head = []

**Output:** []

**Input:** head = [1]

**Output:** [1]

Hint:

If list have only one element or list have not element then return it as it is.

If more than two more element then swap data of current and next node and move the current and next node pointer as below-

Current --- > next of next\_pointer

Next ---- > next to next of next\_pointer

Solution:

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def swapPairs(self, head: Optional[ListNode]) -> Optional[ListNode]:

        if head==None or head.next==None:

            return head

        slow=head

        fast=head.next

        while slow and fast:

            slow.val,fast.val=fast.val,slow.val

            if fast.next and fast.next.next:

                fast=fast.next.next

                slow=slow.next.next

            else:

                break

        return head

# Questions # ---- Maximum Subarray ---- Leetcode

<https://leetcode.com/problems/maximum-subarray/>

Given an integer array nums, find the subarray of consecutive element which has the largest sum and return its sum.

class Solution:

    def maxSubArray(self, nums: List[int]) -> int:

        max\_sum,curr\_sum=float("-inf"),0

        for each in nums:

            curr\_sum=max(curr\_sum+each,each)

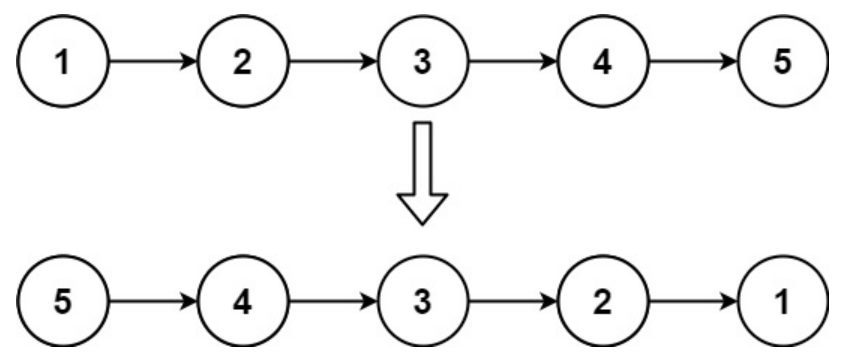
            max\_sum=max(max\_sum,curr\_sum)

        return max\_sum

**#Question# --- Reverse the linked list --- Leetcode**

Given the head of a singly linked list, reverse the list, and return the reversed list.

<https://leetcode.com/problems/reverse-linked-list/description/>



Algo:

First get the value of linked list one list.

Start from head and keep replacing value of each node by the last value of list.

At end return the head of linked list

Solution:

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:

        current=head

        l=[]

        while current:

            l.append(current.val)

            current=current.next

        #now reverse it

        current=head

        while current:

            current.val=l[-1]

            l.pop()

            current=current.next

        return head

**Method 2: ------ Using dumy node**

**Algo:**

Create a dumy node and store the next node of dumy node in a variable (temp\_node) and follow below operation for each node of given linked list

dumy.next=ListNode(current.val) # current points to each node of given linked list

dumy.next.next=temp\_node

class Solution:

    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:

        dumy1=head1=ListNode(0)

        if head==None:

            return head

        current=head

        while current:

            temp=dumy1.next

            dumy1.next=ListNode(current.val)

            dumy1.next.next=temp

            current=current.next

        return head1.next

**#Question # ----- Check the first bad version of product ----- Leetcode**

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the versions after a bad version are also bad.

Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following ones to be bad.

You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to find the first bad version. **You should minimize the number of calls to the API.**

Hint:

Same as previous set three pointer left and right. Left=1 and right=n and bad\_vr=None

Find middle as (left+right)//2, if version as middle is good then bad version will be in right, update the left pointer as left=middle+1 , if it is bad then update right =middle-1 and bad\_vr=middle

Keep on looping this algo till left<right to find the root bad version.

Note:

When loop breaks then l==r but loop will not check if it is bad to at end , check if it is bad. In case bad then return l (that version) else return the version which was found as bad (bad\_vr)

# The isBadVersion API is already defined for you.

# def isBadVersion(version: int) -> bool:

class Solution:

    def firstBadVersion(self, n: int) -> int:

        l=1

        r=n

        bad\_vr=''

        while l<r:

            middle=(l+r)//2

            print(l,r,middle)

            if isBadVersion(middle)==True:

                r=middle-1

                bad\_vr=middle

            else:

                l=middle+1

        #here both l and r will be same, check if this is bad or not

        #loop does not checks this value of l and r

        print(l,r)

        if isBadVersion(l)==True:

            return l

        else:

            return bad\_vr

**Question ---- Longest Substring Without Repeating Characters**

<https://leetcode.com/problems/longest-substring-without-repeating-characters/description/>

Solution:

class Solution:

    def lengthOfLongestSubstring(self, s: str) -> int:

        d={}

        max\_len=0

        start=0

        for num,ch in enumerate(s):

            if ch in d:

                start = max(start,d[ch])

            max\_len=max(max\_len,num+1-start)

            d[ch]=num+1

        return max\_len

Question ---- Longest palindrome substring

class Solution:

    def longestPalindrome(self, s: str) -> str:

        l=''

        #create this method for getting palindrome string

        def helper(l,r):

            while l>=0 and r<=len(s)-1:

                if s[l]==s[r]:

                    l=l-1

                    r=r+1

                else:

                    break

            return s[l+1:r]

        res=''

        #now call helper function for each index of s

        for i in range(len(s)):

            substr=helper(i,i)

            if len(substr)>len(res):

                res=substr

            substr=helper(i,i+1)

            if len(substr)>len(res):

                res=substr

        return res

# Question # ------- Intersection of Two Arrays II --------- LeetCode

Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must appear as many times as it shows in both arrays and you may return the result in any order.

**Easy Solution**

class Solution:

    def intersection(self, nums1: List[int], nums2: List[int]) -> List[int]:

        s1=set(nums1)

        s2=set(nums2)

        s=s1.intersection(s2)

        print(s)

        return list(s)

**Method 1: ----- Brut Force way ---- This is based on hashmap, which stores the value if it was already visited for any other matching value**

Create a dictionary which will contains the index of values which matched in both table

If in iteration both value are same and value is not in dictionary then update dictionary.

If in iteration both value are same and it’s in dictionary then check value for this key. If value is less than the index of value in num2 then add in dictionary and break the inner loop.

class Solution:

    def intersect(self, nums1: List[int], nums2: List[int]) -> List[int]:

        res=[]

        s=set(list(nums1))

        d={}

        for i in range(len(nums1)):

            for j in range(len(nums2)):

                if nums1[i]==nums2[j] and nums1[i] in d:

                    x=d[nums1[i]]

                    if j>x:

                        res.append(nums1[i])

                        d[nums1[i]]=j

                        break

                elif nums1[i]==nums2[j] and nums1[i] not in d:

                    d[nums1[i]]=j

                    res.append(nums1[i])

                    break

        return res

Method 2: ---- Classical solution

First sort both the array and set two pointers l=r=0

If for any value of l,r values in nums1 and nums2 are same then add in result variable the increase both l,r by 1.

Since array are sorted then if any value is matching in both array then l,r will not again come to same value for duplicate value of l and r.

class Solution:

    def intersect(self, nums1, nums2):

        l,r=0,0

        nums1,nums2=sorted(nums1),sorted(nums2)

        output=[]

        while l<len(nums1) and r<len(nums1):

            if nums1[l]<nums2[r]:

                l+=1

            elif nums1[l]>nums2[r]:

                r+=1

            else:

                output.append(l)

                l+=1

                r+=1

Question –---- Best Time to Buy and Sell Stock

You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.

**Method 1: ----- Brut Force**

class Solution:

    def maxProfit(self, prices: List[int]) -> int:

        max\_profit=0

        sell\_price=0

        for i in range(len(prices)-1):

            buy=prices[i]

            sell\_value=max(prices[i+1:])

            if sell\_value-buy>max\_profit:

                max\_profit=sell\_value-buy

        return max\_profit

**Method 2: ---- Classical solution**

class Solution:

    def maxProfit(self, prices: List[int]) -> int:

        max\_profit = 0

        min\_buy = float('inf')

        for price in prices:

            min\_buy=min(min\_buy, price)

            max\_profit=max(max\_profit, price-min\_buy)

        return max\_profit

Question ----- First Unique Character in a String ---- Leetcode

Method 1: ---- Historical way------ use nested for loop

**Method 2: ---- Classical solution ---- using only one for loop**

class Solution:

    def firstUniqChar(self, s: str) -> int:

        non\_uniq=[]

        for i in range(len(s)):

            if s[i] not in s[i+1:] and s[i] not in non\_uniq:

                return i

            else:

                non\_uniq.append(s[i])

        return -1

**Question ----- Reverse the given linked list ------- Leetcode**

Given the head of a singly linked list, reverse the list, and return the reversed list.

**Method 1:** -- using two loops

In first looping get the element in list

Reverse the list data

Now change the data of each node from the reversed data of linked list(list)

class Solution:

    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:

        current=head

        l=[]

        while current:

            l.append(current.val)

            current=current.next

        #now reverse it

        current=head

        while current:

            current.val=l[-1]

            l.pop()

            current=current.next

        return head

**Method 2:** - Using single loop

Create a dumy node

Iterate thought the linked list and create a temp\_node=ListNode(head.val) and do below operation-

temp\_node.next= dumy.next

dumy.next=temp\_node

class Solution:

    def reverseList(self, head: Optional[ListNode]) -> Optional[ListNode]:

        current=head

        dumy=ListNode(0)

        while head:

            ln=ListNode(head.val)

            ln.next=dumy.next

            dumy.next=ln

            head=head.next

        return dumy.next

Question ---- Remove duplicate from sorted linked list --- LeetCode

**Method 1:** --- Get the value of linked list in a list/set, then remove duplicate and create new linked list and return it.

**Method 2:** ---- Using single loop ------ Classical solution

class Solution:

    def deleteDuplicates(self, head: Optional[ListNode]) -> Optional[ListNode]:

        dumy=root=ListNode(0)

        dumy.next=head

        while head:

            while head.next and  head.val==head.next.val:

                head=head.next

                #now reach to point where data is not same

            dumy.next=head

            dumy=dumy.next

            head=head.next

        return root.next

**Question ----- Permutation in String ------ Leetcode**

Given two strings s1 and s2, return true if s2 contains a permutation of s1, or false otherwise.

In other words, return true if one of s1's permutations is the substring of s2.

Method 1: ------- Historical way

Slice the substring of length s1 from s2 and sort it, after sorting compare if it matched with s1.

If it matched then permutation of s1 exists in s2 and return True.

class Solution:

    def checkInclusion(self, s1: str, s2: str) -> bool:

        #start from index 0 and go till index len(s2)-len(s1)+1

        for i in range(0,len(s2)-len(s1)+1):

            #take a substring from s2

            sub2=s2[i:i+len(s1)]

            #sort the s1

            sorted\_s1=sorted(s1)

            #sort the substring

            sorted\_sub=sorted(sub2)

            #Check if both sorted sub string and s1 same

            if sorted\_s1==sorted\_sub:

                return True

        return False

Question ---- Permutation of a string/list/iterable

Method 1: Using permutation from itertools library

Method 2: Using recursion

<https://www.youtube.com/watch?v=s7AvT7cGdSo>

class Solution:

    def permute(self, nums: List[int]) -> List[List[int]]:

        result=[]

        if len(nums)==1:

            return [nums[:]]

        else:

            for each in nums:

                #first to bahar nikal do

                n=nums.pop(0)

                perms=self.permute(nums)

                #jo bahar aaya the add kro each perm me

                for perm in perms:

                    perm.append(n)

                result.extend(perms)

                nums.append(n)

            return result

**Question -------- Group Anagrams -------- Leetcode**

Given an array of strings strs, group the anagrams together. You can return the answer in any order.

Example: -

**Input:** strs = ["eat","tea","tan","ate","nat","bat"]

**Output:** [["bat"],["nat","tan"],["ate","eat","tea"]]

Algo:

Create a empty dictionary.

Get each word of string list and sort it ,

if this sorted word is not present in dictionary then add this word and key and value will be list type of this unsorted word.

If present then append the current unsorted word into the values of this sorted key.

class Solution:

    def groupAnagrams(self, strs: List[str]) -> List[List[str]]:

        d={}

        for s in strs:

#if sorted of s not in dict then add the current value as list

            if "".join(sorted(s)) not in d.keys():

                d["".join(sorted(s))]=[s]

            else:

#If present then append in existing value

                x=d["".join(sorted(s))]

                x.append(s)

                d["".join(sorted(s))]=x

        return list(d.values())

Questions --- Merged Interval ---- LeetCode

class Solution:

    def merge(self, intervals: List[List[int]]) -> List[List[int]]:

        merged\_interval=[]

        intervals=sorted(intervals,key=lambda x : x[0])

        i=0

        for interval in intervals:

            #if merged\_interval is empty or current interval don't overlaps with

            # last interval of merged\_interval

            if len(merged\_interval)==0 or interval[0]>merged\_interval[-1][1]:

                merged\_interval.append(interval)

            #if overlaps

            else:

                #if last value of current inetval <last value of last interval

                #then just update the last value of last interval of merged interval

                if merged\_interval[-1][1]<=interval[1]:

                    merged\_interval[-1][1]=interval[1]

                else:

                    pass

        return merged\_interval

**Question ----- Rotate linkedlist List -----LeetCode**

Algo:

First get the value of linked list in a list

Slice the last values from this list of same length as many number of rotation required and delete those value from list

Now extend the value of list into sliced values and say this list is new\_list

Now change the value of linked list with new\_list

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def rotateRight(self, head,k):

        #first make the node as circular linked list and get length

        if head==None or k==0:

            return head

        value\_list=[]

        current=head

        while current:

            value\_list.append(current.val)

            current=current.next

        #check actually how many rotation required

        k=k%len(value\_list)

        #get the last k values

        l1=value\_list[len(value\_list)-k:]

        #delete those last k value from list

        del value\_list[len(value\_list)-k:]

        #now extend value\_list into l1

        l1.extend(value\_list)

        #assign value of l1 to value\_list

        value\_list=l1

        current=head

        #interchange the value of linked list

        #with newly created list for interchanging

        while current:

            x=value\_list.pop(0)

            current.val=x

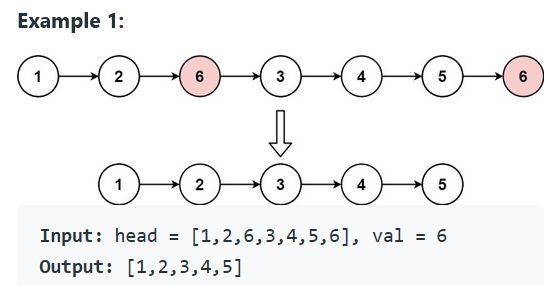
            current=current.next

        return head

**Questions ------ LeetCode Interview**

<https://leetcode.com/assessment/2/>

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, and return the new head.



**Method 1:** ---- historical method

Get all data in list and then delete the value from list and recreate the linked list using that value.

(either by linked list value replacement or create a new linked list)

**Method 2**: --- Classiacal

# Definition for singly-linked list.

# class ListNode:

#     def \_\_init\_\_(self, val=0, next=None):

#         self.val = val

#         self.next = next

class Solution:

    def removeElements(self, head: Optional[ListNode], val: int) -> Optional[ListNode]:

        #if head is none or empty then retiurn

        if head==None:

            return head

        #delete all occurance of val if it is in starting

        while head!=None and head.val==val:

            head=head.next

        #if after removing no data exits in linked list

        if head==None:

            return head

        #if exist then start to delete that node

        prev=head

        current=head.next

        while current:

            if current.val==val:

                prev.next=current.next

                current=current.next

            else:

                prev=current

                current=current.next

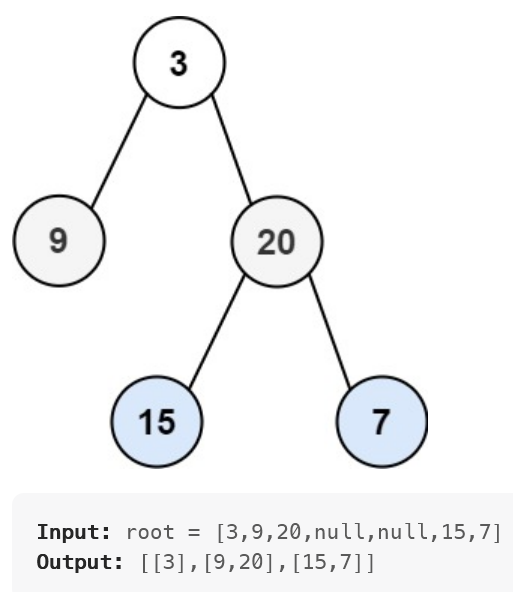
        return head

Question --- Leetcode

<https://leetcode.com/problems/binary-tree-level-order-traversal/description/>

Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by level).

Example:



**Answer:**

class Solution:

    def levelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:

        result=[]

        def helper(nodes):

            data=[]

            child\_nodes=[]

            #if node have no data then return

            if nodes==None or nodes==[]:

                return

            for each in nodes:

                #append the value of each node in data

                data.append(each.val)

                #append the left node of current node in child\_node

                if each.left:

                    child\_nodes.append(each.left)

                #append the right node of current node in child\_node

                if each.right:

                    child\_nodes.append(each.right)

            #append the data in result

            result.append(data)

            #Call the helper function with child\_nodes

            helper(child\_nodes)

        if root==None:

            return []

        #callign helpder funtion bu supplying current node in list form

        helper([root])

        return result

**Algorithm:**

Create a helpder function for below work-

which will store the value of each level of node in a list.

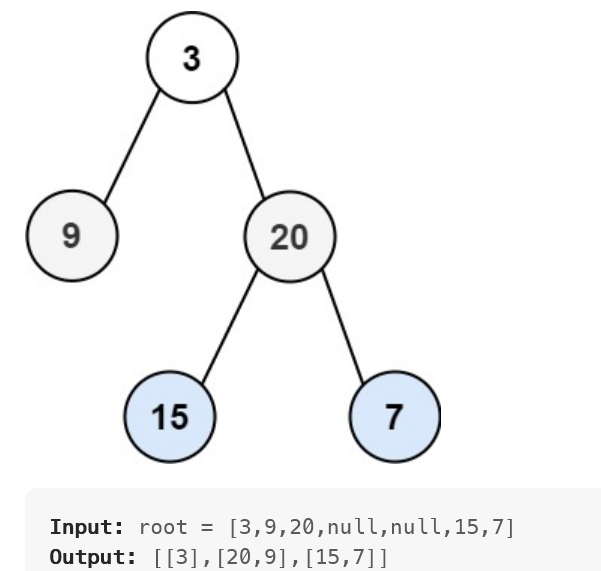
At each vel it will store the childs (left and right node) and will call the helper function for these child.

**Question ------ Binary tree zig zag traversal --- Good**

<https://leetcode.com/problems/binary-tree-zigzag-level-order-traversal/description/>

Given the root of a binary tree, return the zigzag level order traversal of its nodes' values. (i.e., from left to right, then right to left for the next level and alternate between).

Example:



**Answer:**

class Solution:

    def zigzagLevelOrder(self, root: Optional[TreeNode]) -> List[List[int]]:

        if root==None:

            return

        result=[]

        s1=[root]

        s2=[]

        while s1 or s2:

            level\_res=[]

            while s1:

                lvl\_node=s1.pop()

                level\_res.append(lvl\_node.val)

                if lvl\_node.left:

                    s2.append(lvl\_node.left)

                if lvl\_node.right:

                    s2.append(lvl\_node.right)

            result.append(level\_res)

            level\_res=[]

            while s2:

                lvl\_node=s2.pop()

                level\_res.append(lvl\_node.val)

                if lvl\_node.right:

                    s1.append(lvl\_node.right)

                if lvl\_node.left:

                    s1.append(lvl\_node.left)

            if level\_res!=[]:

                result.append(level\_res)

            level\_res=[]

        return result

**Algo:**

Create two stack one for odd level another of even level. (let say s1 for even, s2 for odd)

Pop the item from one stack then store the value of that level in one list.

If child nodes of current node exists then append in another node.

Follow above two steps for another stack, and keep on doing this till both stacks are empty.

<https://www.youtube.com/watch?v=Fy_udhB395A>

Question --- LeetCode ----- Check if Tree is mirror image of itself

<https://leetcode.com/problems/symmetric-tree/description/>

class Solution:

def isSymmetric(self, root: Optional[TreeNode]) -> bool:

return self.isSameTreeReverse(root.left, root.right)

def isSameTreeReverse(self, p, q):

if p is None and q is None:

return True

if p and q and p.val == q.val:

return self.isSameTreeReverse(p.left, q.right) and self.isSameTreeReverse(p.right, q.left)

return False

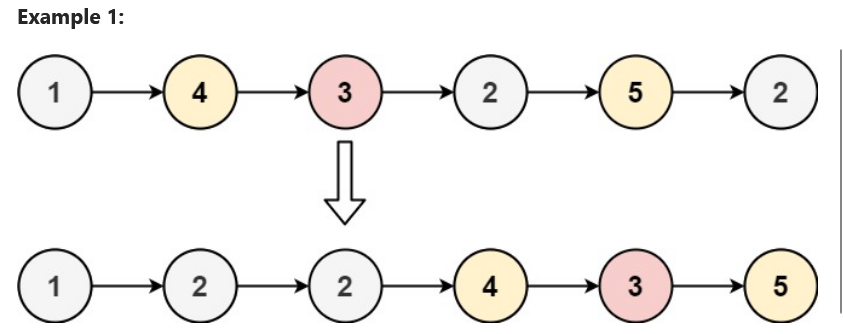
Approach:

Moving from the isSameTree solution.  
Let's imagine we have 2 subtrees: left and right ones.  
To solve the problem we will have to compare:

* left node of the left subtree to the right node of the right subtree
* right node of the left subtree to the left node of the right subtree
* repeat recursively for the child nodes

**Questions --- Partition List ----Leetcode**

Given the head of a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x.



Method 1:

Using two dummy node

class Solution:

    def partition(self, head: Optional[ListNode], x: int) -> Optional[ListNode]:

        if head==None:

            return

        dumy1=head1=ListNode(0)

        dumy2=head2=ListNode(0)

        current=head

        prev=None

        while current:

            #if val<x then add in dumy1

            if current.val<x:

                dumy1.next=current

                dumy1=current

                current=current.next

            else:

                dumy2.next=current

                dumy2=current

                current=current.next

        dumy2.next=None

        dumy1.next=head2.next

        return head1.next

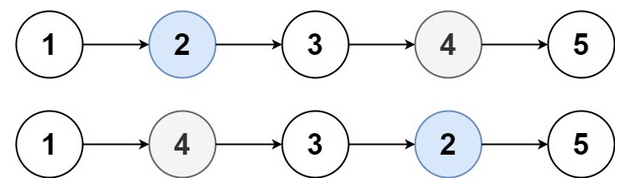
**Question ---- Swap nth nodes from last and start --- Leetcode**

<https://leetcode.com/problems/swapping-nodes-in-a-linked-list/>

You are given the head of a linked list, and an integer k.

Return the head of the linked list after swapping the values of the kth node from the beginning and the kth node from the end (the list is 1-indexed).

Example:



**Method 2**: Get the data in list and swap the value (kth from start and end) then re-assign the value of list node with list.

**Method 2:** Using two pointers

**Algo:**

Have two pointer

First move the fast pointer (k-1) time and store this node in a temporary variable. (this is kth from starting)

Then keep moving fast and slow pointer then fast will be at end slow will be on kth from end node.

Now swap the values of temporary node and slow node.

class Solution:

    def swapNodes(self, head: Optional[ListNode], k: int) -> Optional[ListNode]:

        if head==None:

            return root

        slow=fast=head

        for i in range(1,k):

            fast=fast.next

        kth\_from\_start=fast

        while fast:

            if fast.next:

                fast=fast.next

                slow=slow.next

            else:

                break

        kth\_from\_start.val,slow.val=slow.val,kth\_from\_start.val

        return head

Question ---- Leetcode ----

<https://leetcode.com/assessment/3/>

You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not contain any leading 0's.

Example:

**Input:** digits = [1,2,3]

**Output:** [1,2,4]

**Input:** digits = [4,3,2,1]

**Output:** [4,3,2,2]

Answer:

class Solution:

def plusOne(self, digits: List[int]) -> List[int]:

l=digits

carry=0

need\_to\_add=1

for i in range(-1,-len(digits)-1,-1):

s=digits[i]+need\_to\_add+carry

need\_to\_add=0

print(digits[i],s)

if s>=10:

modulo=s%10

carry=s//10

l[i]=modulo

else:

l[i]=s

carry=0

break

#after doing all sum check still carry over is there

if carry>0:

x=[carry]

x.extend(l)

return x

return l

Question --- Leetcode

<https://leetcode.com/assessment/3/>

Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input

**Example:1**

**Input:** intervals = [[1,3],[2,6],[8,10],[15,18]]

**Output:** [[1,6],[8,10],[15,18]]

**Explanation:** Since intervals [1,3] and [2,6] overlap, merge them into [1,6].

**Example: 2**

**Input:** intervals = [[1,4],[2,3]]

**Output:** [1,4]

**Example: 3**

**Input:** intervals = [[0,4],[2,3]]

**Output:** [0,4]

**Solution:**

class Solution:

def merge(self, intervals: List[List[int]]) -> List[List[int]]:

result=[]

intervals=sorted(intervals,key=lambda x: x[0])

for each in intervals:

if result==[]:

result.append(each)

#check if end value of last interval of result overlap with current

elif result[-1][1]>=each[0]:

if result[-1][0]>each[0]:

result[-1][0]=each[0]

if result[-1][1]>each[1]:

pass

else:

result[-1][1]=each[1]

else:

result.append(each)

return result

**Question ----- Loop detection in LinkedList**

We can detect loop using two way-

Method 1: ---

Start from head node and store each node in one list and keep moving to next next.

Keep on checking while moving to next node that it’s already in list.

If current node is in list then there is loop.

Leetcode baklole accept nhi krta hai ye

    def hasCycle(self, head: Optional[ListNode]) -> bool:

        if head.next!=None:

            return False

        current=head

        visited=[]

        while current:

            if current in visited:

                return True

            else:

                visited.append(current)

                current=current.next

                if type(current.next) is int and

        return False

Method 2: ----- Using two pointer.

Take two pointers slow and fast, and keep moving these two pointers to next node.

fast=fast.next.next

slow=slow.next

If there is loop then they will start pointing to same node.

if not head or not head.next:

            return False

        slow=head

        fast=head.next

        while(fast and fast.next):

            if slow==fast:

                return True

            slow=slow.next

            fast=fast.next.next

        return False

Method 3: ---- Using two pointer

Questions ---- Subarray with given sum ---- Leetcode

<https://practice.geeksforgeeks.org/problems/subarray-with-given-sum-1587115621/1?page=1&sortBy=submissions>

Find the start and end position of array element if those are added then result to given sum value.

(assuming first element is at position 1, )

Example: N = 5, S = 12

A[] = {1,2,3,7,5}

**Output:** 2 4

Method 1: ---- Using brutforcce ---- Using two loops ---- We can do it

**Method 2: ---- Using sliding window**

The idea is simple as we know that all the elements in subarray are positive so, If a subarray has sum greater than the given sum then there is no possibility that adding elements to the current subarray will be equal to the given sum. So the Idea is to use a similar approach to a sliding window.

Start with an empty subarray

add elements to the subarray until the sum is less than x( given sum ).

If the sum is greater than x, remove elements from the start of the current subarray.

Algo:

* Create two variables, start=0, currentSum = arr[0]
* Traverse the array from index 1 to end.
* Update the variable currentSum by adding current element-

currentSum = currentSum + arr[i]

* If the currentSum is greater than the given sum, update the variable currentSum as-

currentSum = currentSum - arr[start], and update start as, start++.

* If the currentSum is equal to given sum, print the subarray and break the loop.

class Solution:

    def subArraySum(self,arr, n, s):

        current\_sum=arr[0]

        start=0

        i=1

        while i<=n-1:

            current\_sum=current\_sum+arr[i]

            while current\_sum>s and start<=i-1:

                current\_sum=current\_sum-arr[start]

                start=start+1

            if current\_sum==s:

                #print(start+1,i+1)

                return [start+1,i+1]

            if current\_sum<s:

                i=i+1

Question ---- Find leaders in array ---- GeeksForGeeks

[https://practice.geeksforgeeks.org/problems/leaders-in-an-array-1587115620/1?page=1&difficulty[]=0&sortBy=submissions](https://practice.geeksforgeeks.org/problems/leaders-in-an-array-1587115620/1?page=1&difficulty%5b%5d=0&sortBy=submissions)

Given an array A of positive integers. Your task is to find the leaders in the array. An element of array is leader if it is greater than or equal to all the elements to its right side. The rightmost element is always a leader.

**Example:**

Input:

n = 6

A[] = {16,17,4,3,5,2}

Output: 17 5 2

Method 1: Brut Force

**Method 2:** ---- Using monotonic stack/list

Iterate through the list and compare the each value of array from the last element of stack.

If current value of stack is grater then last value of stack then append current value in stack.

    def leaders(self, A, N):

        #Code here

        st = []

        for i in A:

            while st and i > st[-1]:

                st.pop()

            st.append(i)

        return st

**Method3: ----** Brut force

Start from right most value of array/list and keep moving towards right

For looping check the current element is greater than all value in it’s right, if yes then this is leader.

Question ------ Checking valid BST ---- Leetcode and GKFGKs

Method1: --- By checking data with left anf right range

    def isBST(self):

        def valid(node,left, right):

            if not node:

                return True

            if not (node.data<right and node.data>left):

                return False

            return (valid(node.left, left,node.data) and valid(node.right, node.data,right))

        return valid(self,float("-inf"),float("inf"))

Method 2: ---- Using BST property (in order traversal is in ascending order and no duplicate)

def helper(root):

            element=[]

            if root.left:

                element+=helper(root.left)

            element.append(root.val)

            if root.right:

                element+=helper(root.right)

            return element

        data=helper(root)

        sorted\_data=sorted(data)

        if data==sorted\_data and len(sorted\_data)==len(set(data)):

            return True

        return False

**Question ---- Find the equilibrium point of array ----- GKSFRGKS**

[https://practice.geeksforgeeks.org/problems/equilibrium-point-1587115620/1?page=1&difficulty[]=0&sortBy=submissions](https://practice.geeksforgeeks.org/problems/equilibrium-point-1587115620/1?page=1&difficulty%5b%5d=0&sortBy=submissions)

Given an array A of n positive numbers. The task is to find the first Equilibrium Point in an array.

Equilibrium Point in an array is a position such that the sum of elements before it is equal to the sum of elements after it.

**Input:**

n = 5

A[] = {1,3,5,2,2}

**Output: 3**

Method 1: ---- Using Brut Force

**Method 2: ---- Using binary search**

**Algo:**

Have two pointers l and r and middle=(l+r)//2 and calculate left and right sum

left\_sum=A[0:middle] , right\_sum=A[middle+1:]

if left\_sum>right\_sum --- then r=middle-1

if left\_sum<right\_sum --- then l=middle+1

if left\_sum==right\_sum ---- then return middle+1

**keep above operation in loop while l<=r**

class Solution:

    def equilibriumPoint(self,A):

        if len(A)==1:

            return True

        l=0

        r=len(A)

        while l<=r:

            middle=(l+r)//2

            left\_sum=sum(A[:middle])

            right\_sum=sum(A[middle+1:])

            if sum(A[:middle])==sum(A[middle+1:]):

                return middle+1

            if left\_sum>right\_sum:

                r=middle-1

            else:

                l=middle+1

        return -1

Question ---- Find the peak element of an array

[https://practice.geeksforgeeks.org/problems/peak-element/1?page=1&difficulty[]=-1&difficulty[]=0&sortBy=submissions](https://practice.geeksforgeeks.org/problems/peak-element/1?page=1&difficulty%5b%5d=-1&difficulty%5b%5d=0&sortBy=submissions)

An element is called a peak element if its value is not smaller than the value of its adjacent elements(if they exists).

Method 1:

class Solution:

def peakElement(self,arr,n):

# first or last element is peak element

if (n == 1) :

return 0

if (arr[0] >= arr[1]) :

return 0

if (arr[n - 1] >= arr[n - 2]) :

return n - 1

# check for every other element

for i in range(1, n - 1) :

# check if the neighbors are smaller

if (arr[i] >= arr[i - 1] and arr[i] >= arr[i + 1]) :

return i

Method 2:

Using binary search

class Solution(object):

   def findPeakElement(self, nums):

      low = 0

      high = len(nums)-1

      while low<high:

         mid = low + (high - low+1)//2

         if (mid-1>=0 and nums[mid-1]<=nums[mid]):

            low = mid

         else:

            high = mid-1

      return nums[low+1]

**Question ---- Right view of binary Tree ----- Leetcode**

<https://leetcode.com/problems/binary-tree-right-side-view/description/>

Given the root of a binary tree, imagine yourself standing on the **right side** of it, return the values of the nodes you can see ordered from top to bottom.

|  |  |
| --- | --- |
|  | **Input:** root = [1,2,3,null,5,null,4]  **Output:** [1,3,4] |

Hint:

Think about the level order traversal.

Solution:

class Solution:

    def rightSideView(self, root: Optional[TreeNode]) -> List[int]:

        if root==None:

            return []

        result=[]

        temp\_stack=[]

        stack=[root]

        while stack:

            x=[]

            for each in stack:

                if x==[]:

                    x.append(each.val)

                if each.right:

                    temp\_stack.append(each.right)

                if each.left:

                    temp\_stack.append(each.left)

            stack=temp\_stack

            temp\_stack=[]

            result.extend(x)

        return result

**Question ---- Reverse the linked list in pars**

<https://practice.geeksforgeeks.org/problems/bae68b4d6a2a77fb6bd459cf7447240919ebfbf5/1>

Given a linked list and a number k. You have to reverse first part of linked list with k nodes and the second part with n-k nodes.

Example:

**Input:** 1 -> 2 -> 3 -> 4 -> 5

k = 2

**Output:** 2 -> 1 -> 5 -> 4 -> 3

**Explanation:** As k = 2 , so the first part 2

nodes: 1 -> 2 and the second part with 3 nodes:

3 -> 4 -> 5. Now after reversing the first part:

2 -> 1 and the second part: 5 -> 4 -> 3.

So the output is: 2 -> 1 -> 5 -> 4 -> 3

Solution:

class Solution:

    def reverse(self, head : Optional['Node'], k : int) -> Optional['Node']:

        # code here

        if head==[] or head==None:

            return head

        l=[]

        current=head

        #first get all data from linked list

        while current:

            l.append(current.data)

            current=current.next

        #get first k values from list and rever them

        l1=l[:k][::-1]

        #get remaining n-k value and reverse it

        l2=l[k:][::-1]

        #extend l2 in l1

        l1.extend(l2)

        dumy\_node=dumy\_head=Node(0)

        #now create new linked list with values

        for each in l1:

            dumy\_node.next=Node(each)

            dumy\_node=dumy\_node.next

        return dumy\_head.next

**Maximum product of array --- GeeksForGeeks -------- Kadens algo**

[https://practice.geeksforgeeks.org/problems/maximum-product-subarray3604/1?page=2&difficulty[]=0&difficulty[]=1&sortBy=submissions](https://practice.geeksforgeeks.org/problems/maximum-product-subarray3604/1?page=2&difficulty%5b%5d=0&difficulty%5b%5d=1&sortBy=submissions)

Given an array **Arr[]** that contains **N** integers (may be **positive**, **negative** or **zero**). Find the product of the maximum product subarray.

Example:

Arr[] = {6, -3, -10, 0, 2}

**Output:** 180

def maxProduct(arr, n):

    # Variables to store maximum and

    # minimum product till ith index.

    minVal = arr[0]

    maxVal = arr[0]

    maxProduct = arr[0]

    for i in range(1, n, 1):

        # When multiplied by -ve number,

        # maxVal becomes minVal

        # and minVal becomes maxVal.

        if (arr[i] < 0):

            temp = maxVal

            maxVal = minVal

            minVal = temp

        # maxVal and minVal stores the

        # product of subarray ending at arr[i].

        maxVal = max(arr[i], maxVal \* arr[i])

        minVal = min(arr[i], minVal \* arr[i])

        # Max Product of array.

        maxProduct = max(maxProduct, maxVal)

    # Return maximum product found in array.

    return maxProduct

Maximum sum of subarray ----- GeeksForGeeks ------- Kaden’s algo

[https://practice.geeksforgeeks.org/problems/kadanes-algorithm-1587115620/1?page=1&difficulty[]=0&difficulty[]=1&sortBy=submissions](https://practice.geeksforgeeks.org/problems/kadanes-algorithm-1587115620/1?page=1&difficulty%5b%5d=0&difficulty%5b%5d=1&sortBy=submissions)

Given an array **Arr[]**of **N** integers. Find the contiguous sub-array(containing at least one number) which has the maximum sum and return its sum.

class Solution:

    ##Complete this function

    #Function to find the sum of contiguous subarray with maximum sum.

    def maxSubArraySum(self,arr,N):

        ##Your code here

        curr\_sum=0

        max\_sum=float('-inf')

        for each in arr:

            curr\_sum=max(each,curr\_sum+each)

            max\_sum=max(max\_sum,curr\_sum)

        #print('for input ',arr,' max sum is: ',max\_sum)

        return max\_sum

Question --- 3Sum Closet ---- Finding the sum closest to given target

<https://leetcode.com/problems/3sum-closest/description/>

Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target

Example:

**Input:** nums = [-1,2,1,-4], target = 1

**Output:** 2

**Explanation:** The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

Method 1: ---- Brut force

**Method 2:** ---- Using two pointer

Sort the array first

First initialize result=nums[0] + nums[1] + nums[ len(num) – 1 ]

Have two pointer l and r and do following for i=0 to i-len(nums)-2.

l=i+1 and r=len(nums)-1

now while l<r and l>I do following:

now get the sum for nums[i]+nums[l]+nums[r]

if abs( sum\_value -target ) < abs (result – target )

result = sum\_value

if nums>target then r=r-1

if nums<target then l=l+1

if nums=target then return target

class Solution(object):

    def threeSumClosest(self, nums, target):

        if nums==[]:

            return 0

        if len(nums)==2:

            return sum(nums)

        nums=sorted(nums)

        result=nums[0]+nums[1]+nums[len(nums)-1]

        for i in range(len(nums)-2):

            l=i+1

            r=len(nums)-1

            while l<r and l>i:

                sum\_value=nums[i]+nums[l]+nums[r]

                #check are we more close to target value

                if abs(sum\_value-target)<abs(result-target):

                    result=sum\_value

                if sum\_value>target:

                    r=r-1

                elif sum\_value<target:

                    l=l+1

                else:

                    return target

        return result

**Questions --- Balanced Binary Tree ------ Leetcode**

<https://leetcode.com/problems/balanced-binary-tree/description/>

Given a binary tree, determine if it is **height-balanced**

**Solution:**

Check Neetcode Solution

**Question --- 4sum ---- Leetcode**

<https://leetcode.com/problems/4sum/>

Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that:

* 0 <= a, b, c, d < n
* a, b, c, and d are distinct.
* nums[a] + nums[b] + nums[c] + nums[d] == target

You may return the answer in any order.

Solution: Recursively reducing it to 2 pointer solution

class Solution:

    def fourSum(self, nums, target):

        def findNsum(l, r, target, N, result, results):

            if r-l+1 < N or N < 2 or target < nums[l]\*N or target > nums[r]\*N:  # early termination

                return

            if N == 2: # two pointers solve sorted 2-sum problem

                while l < r:

                    s = nums[l] + nums[r]

                    if s == target:

                        results.append(result + [nums[l], nums[r]])

                        l += 1

                        while l < r and nums[l] == nums[l-1]:

                            l += 1

                    elif s < target:

                        l += 1

                    else:

                        r -= 1

            else: # recursively reduce N

                for i in range(l, r+1):

                    if i == l or (i > l and nums[i-1] != nums[i]):

                        findNsum(i+1, r, target-nums[i], N-1, result+[nums[i]], results)

        nums.sort()

        results = []

        findNsum(0, len(nums)-1, target, 4, [], results)

        return results

#Question ---- GKFGS#

<https://practice.geeksforgeeks.org/problems/1f05c7c12b1084f270c57566b2110967c046730d/1>

You are given an integer **N**. Consider an array **arr** having **N** elements where **arr[i]** = **2\*i+1**. (The array is 0-indexed)

You are allowed to perform the given operation on the array any number of times:

1) Select two indices **i** and **j** and increase **arr[i]** by 1 and decrease **arr[j]** by 1.

Your task is to find the minimum number of such operations required to make all the elements of the array equal.

Hint:

If N==2 ---- then only iteration

If N<2 ---- then no iteration

If N is odd then we will need sum(2,4,6 … 2\*mid\_index\_minus\_1 ) --- that will be n(n+1)

If N is even then we will need sum(1,3,5… 2\*mid\_indx+1 ) ----- that will be n\*n

**Solution**

class Solution:

    def minOperations(self, N):

        # Code here

        #if n is even

        if N<2:

            return 0

        if N==2:

            return 1

        else:

            #if N is even

            if N%2==0:

                #work for even case

                mid\_indx=N//2

                return (mid\_indx)\*\*2

            if N%2!=0:

                mid\_indx=N//2

                return mid\_indx\*(mid\_indx+1)

#Question # ---- Leetcode ---- Sorted Array to Height Balanced BST

<https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/>

Given an integer array nums where the elements are sorted in ascending order, convert it to a

height-balanced

binary search tree.

Hint:

Root will be middle of array

eft node of root will be middle of left part of sub array

right node will be middle of right part of sub array

keep this iteration

class Solution:

    def sortedArrayToBST(self, nums: List[int]) -> Optional[TreeNode]:

#if array is empty then return None

        if len(nums)==0 or nums==None:

            return None

        middle=len(nums)//2

        root=TreeNode(nums[middle])

        #now create the left node

        root.left=self.sortedArrayToBST(nums[:middle])

        #now create right node

        root.right=self.sortedArrayToBST(nums[middle+1:])

        return root

#Question# ---- Leetcode

Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.

<https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/description/>

Hint:

<https://www.youtube.com/watch?v=ihj4IQGZ2zc>

class Solution:

    def buildTree(self, preorder: List[int], inorder: List[int]) -> Optional[TreeNode]:

        if not preorder and not inorder:

            return None

        root=TreeNode(preorder[0])

        indx=inorder.index(preorder[0])

        #new preorder will be

        root.left=self.buildTree(preorder[1:indx+1],inorder[0:indx])

        root.right=self.buildTree(preorder[indx+1:],inorder[indx+1:])

        return root

#Qurstion#

Find the next greater element for an array.

Method 1: --- Here element is it’s next greater element will be displayed. Input and output order will change

class Solution:

    def nextLargerElement(self,a,n):

        s = []

        for i in range(len(a)):

            # If the stack is empty, push the current element.

            if len(s) == 0:

                s.append(a[i])

                continue

            while s and s[-1] < a[i]:

                print(str(s[-1]), "-->", str(a[i]))

                s.pop()

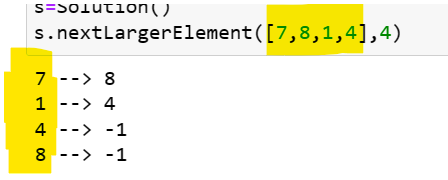
            s.append(a[i])

        while len(s) > 0:

            print(str(s[-1]), "--> -1")

            s.pop()

Example:



Method 2: --- Want result in same order of input.

class Solution:

    def nextLargerElement(self,arr,n):

        #code here

        st=[]

        st.append(arr[-1])

        ans=[-1]\*n

        for i in range(n-2,-1,-1):

            while len(st) and st[-1]<=arr[i]:

                st.pop()

            if not len(st):

                ans[i]=-1

            else:

                ans[i]=st[-1]

            st.append(arr[i])

        return ans

#

|  |  |
| --- | --- |
|  | next greater of 7 --- 8  next greater of 8 --- -1  next greater of 1 --- 4  next greater of 4 --- -1 |

**Finding duplicates in a array**

<https://leetcode.com/problems/find-all-duplicates-in-an-array/description/>

Method 1: ---- Using brut force approach

Method 2: --- Using counter class.

class Solution:

    def findDuplicates(self, nums: List[int]) -> List[int]:

        from collections import Counter

        x = Counter(nums)

        res=[each for each in x.keys() if x[each]>1]

        return res

Method 3: --- Using algo

class Solution:

    def findDuplicates(self, nums: List[int]) -> List[int]:

        ans = []

        for num in nums:

            if nums[abs(num)-1] < 0:

                ans.append(abs(num))

            else:

                nums[abs(num)-1] = -1\*nums[abs(num)-1]

**Question ------ Remove the balls ----- GKG**

<https://practice.geeksforgeeks.org/problems/546ea68f97be7283a04ddcc8057e09b46a686471/1>

Imagine an imaginary array of length N containing balls. Given 2 arrays color and radius of length N each, where color[i] represents the color of the ith ball while radius[i] represents the radius of ith ball. If two consecutive balls have the same color and size, both are removed from the array. Geek wants to know the length of the final imaginary array

**Input**:

N = 3

color[] = {2, 2, 5}

radius[] = {3, 3, 4}

**Output:**

1

Hint:

* Create two stack, radius\_stack and color\_stack.
* Iterate through range(N) and pick each element from color and radius and compare with last element of radius\_stack and color\_stack.
* If last element of stack and each elemt of radius and color same then pop from stack.
* If last element of stack and each elemt of radius and color are not same then push into stack.
* At end return length of stack.

Solution:

class Solution:

    def finLength(self, N : int, color : List[int], radius : List[int]) -> int:

        # code here

        color\_stack=[]

        radius\_stack=[]

        #loop through each value of radius and color

        for i in range(N):

            #if color or radius stack empty then push element

            if color\_stack==[]:

                color\_stack.append(color[i])

                radius\_stack.append(radius[i])

                continue

            #if last element of radius and color stack

            #are same as current value of color and rasuis

            #then pop element from stack and check for next element

            if color\_stack[-1]==color[i] and radius\_stack[-1]==radius[i]:

                color\_stack.pop()

                radius\_stack.pop()

                continue

            #else append values in color and radius stack

            else:

                color\_stack.append(color[i])

                radius\_stack.append(radius[i])

        return len(color\_stack)

# Question # --- Missing and repeating ---- GFGS

**Find the missing and repeating number from an array.**

Solution:

class Solution:

    def findTwoElement( self,arr,n):

        duplicate=''

        l1=list(range(1,n+1))

        missing\_value=list(set(l1).difference(set(arr)))[0]

        temp = [0] \* n

        for i in range(n):

            temp[arr[i] - 1] += 1

            if temp[arr[i] - 1] > 1:

                duplicate = arr[i]

                break

        return [duplicate,missing\_value]

Algo used:

For missing number set to find the missing number.

For duplicate/repeating number use below algo-

* Create a temp array temp[] of size n with all initial values as 0.
* Traverse the input array arr[], and do the following for each arr[i]
  + if(temp[arr[i]-1] == 0), set temp[arr[i]-1] = 1;
  + if(temp[arr[i]-1] == 1) output “arr[i]” //repeating number
* Traverse temp[] and output ‘i+1’ corresponding to the element of array temp[] having value as 0. (This is the missing number)

For finding the repeating number we have many algo, we can see all on below link

<https://www.geeksforgeeks.org/find-a-repeating-and-a-missing-number/>

1. using of count
2. using enumerate
3. using loop

**# Question # ---- Make Palindrome ---- GFG**

You are given an array of strings **arr** of size **n**. You have to find out if it is possible to make a palindromic string by concatenating the strings in any order. Provided that all the strings given in the array are of **equal length**.

Example:

**Input:**

n = 4

arr = {"djfh", "gadt", "hfjd", "tdag"}

**Output:**

YES

**Explanation:**

You can make the string "djfhgadttdaghfjd", by concatenating the given strings which is a palindrome.

Solution:

class Solution:

    def makePalindrome(self, arr):

        # code here

        res=[]

        #do below operation when there are more than one element in arr

        while len(arr)>1:

            x=arr[0]

            #if reverse of x is in arr

            #then remove the x and it's reverse

            if arr[0][::-1] in arr:

                arr.remove(x)

                arr.remove(x[::-1])

                continue

            else:

                return False

        #in case arr len is even

        if len(arr)==0:

            return True

        #if arr is odd

        elif arr[0][::-1]==arr[0]:

            return True

        #finally return False b/c we can not create palindrome string

        return False

**# Question # -- Finding middle element in a linked list --- GFG**

<https://practice.geeksforgeeks.org/problems/finding-middle-element-in-a-linked-list/1?page=2&sortBy=submissions>

Given a singly linked list of N nodes. The task is to find the middle of the linked list.

Assume that in linked list number of node will be >1

Solution:

class Solution:

    #  Should return data of middle node. If linked list is empty, then  -1

    def findMid(self, head):

        stack=[]

        i=0

        current=head

        #store all nodes in a stack/list

        #i is used to coutn number of node

        #if i is not used then we can get as -- len(stack)

        while current:

            stack.append(current)

            i=i+1

            current=current.next

        #get the middle index

        mid=i//2

        return stack[mid].data

# Question # --- Reverse array in groups --- GFG

<https://practice.geeksforgeeks.org/problems/reverse-array-in-groups0255/1?page=2&sortBy=submissions>

Given an array arr[] of positive integers of size N. Reverse every sub-array group of size K.

class Solution:

    #Function to reverse every sub-array group of size k.

    def reverseInGroups(self, arr, N, K):

        # code here

        #start will be used to take portion from arr for reversion

        start=0

        check=True

        #loop till we have revired all elements

        while check:

            portion=arr[start:start+K]

            portion=portion[::-1]

            #now replace the original value with replaced value

            arr[start:start+K]=portion

            start=start+K

            #if we revered all element

            if start>=len(arr):

                break

        return arr

#Question#

<https://www.codewars.com/kata/5a793fdbfd8c06d07f0000d5/train/python>

(on code war)

solve("3(ab)") = "ababab" -- because "ab" repeats 3 times

solve("2(a3(b))") = "abbbabbb" -- because "a3(b)" == "abbb", which repeats twice.

Hint:

Reverse the string and try using method isalpha() and isdigit.

Solution

def solve(s):

s1 = s[::-1]

s2 = ''

for i in s1:

if i.isalpha():

s2 += i

elif i.isdigit():

s2 = s2\*int(i)

return s2[::-1]