1. Two Sum

```
"""
1. Two Sum
Question:
Given an array of integers, find two numbers such that they add up to a specific target number.
The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2. Please note that your returned answers (both index1 and index2) are not zero-based.
You may assume that each input would have exactly one solution.
"""

class Solution:
    def twoSum(self, nums, target):
        """
        :type nums: List[int]
        :type target: int
        :rtype: List[int]
        """
        m = {}
        for i in range(len(nums)):
            if target - nums[i] in m:
                return [i, m[target - nums[i]]]
            else:
                m[nums[i]] = i
```

2. Two Sum II – Input array is sorted

```
"""
2. Two Sum II – Input array is sorted
Question:
Similar to Question [1. Two Sum], except that the input array is already sorted in ascending order.
"""

class Solution:
    def twoSum(self, numbers, target):
        """
        :type numbers: List[int]
        :type target: int
        :rtype: List[int]
        """
        i = 0
```

```
    j = len(numbers) - 1
    for x in range(len(numbers)):
        if numbers[i] + numbers[j] > target:
            j -= 1
        elif numbers[i] + numbers[j] < target:
            i += 1
        elif numbers[i] + numbers[j] == target:
            return [i+1, j+1]
```

3. Two Sum III – Data structure design

```
"""
3. Two Sum III – Data structure design

Question:
Design and implement a TwoSum class. It should support the following operations: add
and find.
add(input) – Add the number input to an internal data structure.
find(value) – Find if there exists any pair of numbers which sum is equal to the value.
For example,
add(1); add(3); add(5); find(4)  true; find(7)  false
"""


class TwoSum:
        """docstring for ClassName"""
        def __init__(self):
                self.table = dict()

        def add(self, number):
                if number in self.table:
                        self.table[number] += 1
                else:
                        self.table[number] = 1

        def find(self, value):
                for key in self.table:
                        if value == key * 2:
                                if self.table[key] > = 2:
                                        return true

                        else:
                                if value - key in self.table:
                                        return True
```

```
                return False
```

4. Valid Palindrome

```
"""
4. Valid Palindrome

Question:
Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.
For example,
"A man, a plan, a canal: Panama" is a palindrome.
"race a car" is not a palindrome.
Example Questions Candidate Might Ask:
Q: What about an empty string? Is it a valid palindrome?
A: For the purpose of this problem, we define empty string as valid palindrome.
"""

import re

class Solution:
    def isPalindrome(self, s):
        """
        :type s: str
        :rtype: bool
        """
        i = 0
        x = s.lower()
        x = re.sub('[^a-z0-9]+' , "", x)
        j = len(x) - 1
        print (x)
        while i <= j:
            for k in range(len(x)):
                if x[i] == x[j]:
                    i += 1
                    j -= 1
                else:
                    return False

        return True
```

5. Implement strstr()

```
"""
5. Implement strstr()
```

Question:
Implement strstr(). Returns the index of the first occurrence of needle in haystack, or –1 if needle is not part of haystack.

Example 1:
Input: haystack = "hello", needle = "ll"
Output: 2

Example 2:
Input: haystack = "aaaaa", needle = "bba"
Output: -1
"""

```
class Solution:
    def strStr(self, haystack, needle):
        """
        :type haystack: str
        :type needle: str
        :rtype: int
        """
        # Edge Case
        if len(needle) > len(haystack):
            return -1

        for i in range (0, len(haystack)-len(needle)+1):
            if haystack[i: i+len(needle)] == needle:
                    return i

        return -1
```

6. Reverse Words in a String

"""
6. Reverse Words in a String

Question:
Given an input string s, reverse the string word by word.
For example, given s = "the sky is blue", return "blue is sky the".
Example Questions Candidate Might Ask:
Q: What constitutes a word?
A: A sequence of non-space characters constitutes a word.

Q: Does tab or newline character count as space characters?
A: Assume the input does not contain any tabs or newline characters.
Q: Could the input string contain leading or trailing spaces?
A: Yes. However, your reversed string should not contain leading or trailing spaces.
Q: How about multiple spaces between two words?
A: Reduce them to a single space in the reversed string.
"""

```
class Solution(object):
    def reverseWords(self, s):
        """
        :type s: str
        :rtype: str
        """
        return " ".join(s.strip().split()[::-1])



# rough work
"""
class Solution(object):
    def reverseWords(self, s):
        """
        :type s: str
        :rtype: str
        """
        print(" ".join(s.strip().split()[::-1]))
        # print(words)
        # for word in words:
        #     word = word[::-1]
        #     print(word+" ")

s = Solution()
s.reverseWords("the    sky is blue   ")
```

7. Reverse Words in a String II

"""
7. Reverse Words in a String II

Question:
Similar to Question [6. Reverse Words in a String], but with the following constraints:
“The input string does not contain leading or trailing spaces and the words are always separated by a single space.”

Could you do it in-place without allocating extra space?
"""
This is a java soution, since python strings are immutable it's not possible to change them in place.

```
public void reverseWords(char[] s) {
 reverse(s, 0, s.length);
 for (int i = 0, j = 0; j <= s.length; j++) {
 if (j == s.length || s[j] == ' ') {
 reverse(s, i, j);
 i = j + 1;
 }
 }
}
private void reverse(char[] s, int begin, int end) {
 for (int i = 0; i < (end - begin) / 2; i++) {
 char temp = s[begin + i];
 s[begin + i] = s[end - i - 1];
 s[end - i - 1] = temp;
 }
}
```

8. Pass

9. Valid Number

"""
9. Valid Number

Question:
Validate if a given string is numeric.
Some examples:
"0" true
"0.1" true
"abc" false
Example Questions Candidate Might Ask:
Q: How to account for whitespaces in the string?
A: When deciding if a string is numeric, ignore both leading and trailing whitespaces.
Q: Should I ignore spaces in between numbers – such as “1 1”?
A: No, only ignore leading and trailing whitespaces. “1 1” is not numeric.

Q: If the string contains additional characters after a number, is it considered valid?
A: No. If the string contains any non-numeric characters (excluding whitespaces and decimal point), it is not numeric.
Q: Is it valid if a plus or minus sign appear before the number?
A: Yes. “+1” and “-1” are both numeric.
Q: Should I consider only numbers in decimal? How about numbers in other bases such as hexadecimal (0xFF)?
A: Only consider decimal numbers. “0xFF” is not numeric.
Q: Should I consider exponent such as “1e10” as numeric?
A: No. But feel free to work on the challenge that takes exponent into consideration. (The Online Judge problem does take exponent into account.)

Solution:
This problem is very similar to Question [8. String to Integer (atoi)]. Due to many corner cases, it is helpful to break the problem down to several components that can be solved individually.
A string could be divided into these four substrings in the order from left to right:
s1. Leading whitespaces (optional).
s2. Plus (+) or minus (–) sign (optional).
s3. Number.
s4. Optional trailing whitespaces (optional).
We ignore s1, s2, s4 and evaluate whether s3 is a valid number. We realize that a number could either be a whole number or a decimal number. For a whole number, it is easy: We evaluate whether s3 contains only digits and we are done.
17
On the other hand, a decimal number could be further divided into three parts:
a. Integer part
b. Decimal point
c. Fractional part
The integer and fractional parts contain only digits. For example, the number “3.64” has integer part (3) and fractional part (64). Both of them are optional, but at least one of them must present. For example, a single dot ‘.’ is not a valid number, but “1.”, “.1”, and “1.0” are all valid. Please note that “1.” is valid because it implies “1.0”.
By now, it is pretty straightforward to translate the requirements into code, where the main logic to determine if s3 is numeric from line 6 to line 17.
"""

```
class Solution:
    def isNumber(self, s):
        """
        :type s: str
        :rtype: bool
        """
```

```
#Ignore all leading white spaces
i = 0
n = len(s)
print(n)
# print(s[i])

while i < n and s[i] == " ":
    i += 1
    # print (i)

#if you encounter + or - its okay go past it
if i < n:
    if (s[i] == "+" or s[i] == "-"):
        i += 1

#set isnumber to false by default
isnumber = False

while i < n and s[i].isdigit():
    i += 1
    isnumber =True

if i < n and s[i] == ".":
    i += 1
    while i < n and s[i].isdigit():
        i += 1
        isnumber = True

if isnumber and i < n and s[i] == "e":
    i += 1
    isnumber = False
    if i < n and (s[i] == "+" or s[i] == "-"):
        i += 1
    while i < n and s[i].isdigit():
        i += 1
        isnumber = True

while i < n and s[i] == " ":
    i += 1
return isnumber and i == n
```

10. **Pass**
11. **Pass**
12. **Pass**
13. **Longest Palindromic Substring**

"""
13. Longest Palindromic Substring

Question:
Given a string S, find the longest palindromic substring in S. You may assume that the maximum length of S is 1000, and there exists one unique longest palindromic substring.

Hint:
First, make sure you understand what a palindrome means. A palindrome is a string which reads the same in both directions. For example, “aba” is a palindome, “abc” is not.

A common mistake:
Some people will be tempted to come up with a quick solution, which is unfortunately flawed (however can be corrected easily):
Reverse S and become S’. Find the longest common substring between S and S’, which must also be the longest palindromic substring.
This seemed to work, let’s see some examples below.
For example, S = “caba”, S’ = “abac”.
The longest common substring between S and S’ is “aba”, which is the answer.
Let’s try another example: S = “abacdfgdcaba”, S’ = “abacdgfdcaba”.
The longest common substring between S and S’ is “abacd”. Clearly, this is not a valid palindrome.
We could see that the longest common substring method fails when there exists a reversed copy of a non-palindromic substring in some other part of S. To rectify this, each time we find a longest common substring candidate, we check if the substring’s indices are the same as the reversed substring’s original indices. If it is, then we attempt to update the longest palindrome found so far; if not, we skip this and find the next candidate.

"""

"""
ALGO :

for each character start from the middle and go out until the string is not a palindrome :
isplaindrome() does this

```
record and compare from previous largest palindrome substring : res stores this
return the largest at the end. : res
"""
class Solution:
    def longestPalindrome(self, s):
        """
        :type s: str
        :rtype: str
        """
        res = ""
        for i in range(len(s)):
            # for odd cases such as a, aba, ababa
            l = self.ispalindrome(s, i, i)
            # print("odd : "+l)
            if len(l) > len(res):
                res = l

            #for even cases such as aa, abba, abbaabba
            l = self.ispalindrome(s, i, i+1)
            # print("even : "+l)
            if len(l) > len(res):
                res = l
        return res

    def ispalindrome(self, s, l, r):
        while l >= 0 and r < len(s) and s[l] == s[r]:
            # print(str(l)+" : "+str(r))
            # print(s[l], s[r])
            l -= 1
            r += 1
        return s[l+1:r]
```

14. Pass
15. Pass
16. Pass
17. Reverse Integer

```
"""
```

17. Reverse Integer

Question:
Reverse digits of an integer. For example: x = 123, return 321.
Example Questions Candidate Might Ask:
Q: What about negative integers?
A: For input x = –123, you should return –321.
Q: What if the integer's last digit is 0? For example, x = 10, 100, …
A: Ignore the leading 0 digits of the reversed integer. 10 and 100 are both reversed as 1.
Q: What if the reversed integer overflows? For example, input x = 1000000003.
A: In this case, your function should return 0.
"""

```
class Solution:
    def reverse(self, x):
        """
        :type x: int
        :rtype: int
        """
        reverse = 0
        y = abs(x)
        while y > 0:
            reverse = reverse * 10 + y % 10
            y = y // 10
        ans = -reverse if x < 0 else reverse
        if reverse >= (2**31 -1) or reverse <= (-2**31):
            return 0
        return ans
```

18. Plus One

"""
18. Plus One

Question:
Given a number represented as an array of digits, plus one to the number.
Example Questions Candidate Might Ask:
Q: Could the number be negative?
A: No. Assume it is a non-negative number.
Q: How are the digits ordered in the list? For example, is the number 12 represented by [1,2] or [2,1]?

A: The digits are stored such that the most significant digit is at the head of the list.
Q: Could the number contain leading zeros, such as [0,0,1]?
A: No
"""

```
class Solution(object):
    def plusOne(self, digits):
        """
        :type digits: List[int]
        :rtype: List[int]
        """
        for i in range(len(digits)-1, -1, -1):
            if digits[i] < 9:
                digits[i] += 1
                return digits
            else:
                digits[i] = 0
        digits.insert(0, 1)
        return digits
```

19. Palindrome Number

"""
19. Palindrome Number

Question:
Determine whether an integer is a palindrome. Do this without extra space.
Example Questions Candidate Might Ask:
Q: Does negative integer such as –1 qualify as a palindrome?
A: For the purpose of discussion here, we define negative integers as non-palindrome.
"""

```
class Solution:
    def isPalindrome(self, x):
        """
        :type x: int
        :rtype: bool
        """
        if x < 0:
            return False
        y = str(x)
        i = 0
```

```
        j = len(y) - 1
        while i < j:
            if y[i] == y[j]:
                i += 1
                j -= 1
            else:
                return False
        return True
```

20. Pass
21. Pass
22. Swap Nodes in Pairs

```
"""
22. Swap Nodes in Pairs

Question:
Given a linked list, swap every two adjacent nodes and return its head.
For example,
Given 1  2  3  4, you should return the list as 2  1  4  3.
Your algorithm should use only constant space. You may not modify the values in the
list, only nodes itself can be changed.
Example Questions Candidate Might Ask:
Q: What if the number of nodes in the linked list has only odd number of nodes?
A: The last node should not be swapped.

"""


# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None
# https://www.youtube.com/watch?v=HuBxD8pWpJ0
class Solution:
    def swapPairs(self, head):
        """
        :type head: ListNode
        :rtype: ListNode
        """
        # print(head.val)
        dummy = curr = ListNode(0)
        dummy.next = head
```

```
while head and head.next:
    p = head.next.next
    curr.next = head.next
    curr.next.next = head
    head.next = p
    curr = head
    head = p

return dummy.next
```

23. **Pass**
24. **Pass**
25. **Pass**
26. **Pass**
27. **Pass**
28. **Pass**
29. **Pass**
30. **Pass**
31.

Top 100 Liked Questions

- **Reverse String II**

"""
541. Reverse String II

Given a string and an integer k, you need to reverse the first k characters for every 2k characters counting from the start of the string. If there are less than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k characters, then reverse the first k characters and left the other as original.
Example:
Input: s = "abcdefg", k = 2
Output: "bacdfeg"
Restrictions:
The string consists of lower English letters only.
Length of the given string and k will in the range [1, 10000]

"""

```
class Solution(object):
    def reverseStr(self, s, k):
        """
        :type s: str
        :type k: int
        :rtype: str
        """
        a = list(s)
        for i in range(0, len(a), 2*k):
            a[i:i+k] = reversed(a[i:i+k])
        return "".join(a)
```

- **Remove Nth Node From End of List**

"""
19. Remove Nth Node From End of List

Given a linked list, remove the n-th node from the end of list and return its head.

Example:

Given linked list: 1->2->3->4->5, and n = 2.

After removing the second node from the end, the linked list becomes 1->2->3->5.
Note:

Given n will always be valid.

Follow up:

Could you do this in one pass?
"""

"""
ALGO - use 'n' to separate your fast and slow pointers by appropriate distance and then simple point the slow pointer to its next.next at the end

Edge case - [1], 1
Solution - if not fast:
                    return head.next
                    # i.e if fast becomes none(and fast.net doesn't exist) it imples we've gone ahead so return head.next as thats the only possibility here.
"""

```
# Definition for singly-linked list.
# class ListNode:
#     def __init__(self, x):
#         self.val = x
#         self.next = None

class Solution:
    def removeNthFromEnd(self, head, n):
        """
        :type head: ListNode
        :type n: int
        :rtype: ListNode
        """
        fast = slow = head
        i = 0

        while i < n:
            fast = fast.next
```

```
            i += 1

        if not fast:
            return head.next

        while fast.next != None:
            fast = fast.next
            slow = slow.next

        slow.next = slow.next.next
        return head
```

- **Container With Most Water**

```
"""
11. Container With Most Water

Given n non-negative integers a1, a2, ..., an , where each represents a point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

Example:

Input: [1,8,6,2,5,4,8,3,7]
Output: 49
"""

"""
ALGo -First check for the farthest distance apart and then start cinverging to the middle and keep updating the maxarea accordingly.
"""

class Solution:
    def maxArea(self, height):
        """
        :type height: List[int]
        :rtype: int
        """
        # maxvol = 0
```

```
        # for i in range(len(height)):
        #     for j in range(1, len(height)):
        #         vol = (j - i)*min(height[i], height[j])
        #         if vol > maxvol:
        #             maxvol = vol
        # return maxvol

        i = 0
        j = len(height) - 1
        maxarea = 0
        while i < j:
            if height[i] < height[j]:
                area = height[i]*(j - i)
                if area > maxarea:
                    maxarea = area
                i += 1

            else:
                area = height[j]*(j - i)
                if area > maxarea:
                    maxarea = area
                j -= 1
        return maxarea
```

- **Generate Parentheses**

"""
22. Generate Parentheses

Given n pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

For example, given n = 3, a solution set is:

```
[
  "((()))",
  "(()())",
  "(())()",
  "()(())",
  "()()()"
]
```

"""

"""
https://www.youtube.com/watch?v=LxwiwlUDOk4
"""

```
class Solution:
    def generateParenthesis(self, n):
        """
        :type n: int
        :rtype: List[str]
        """
        if n == 0:
            return []

        return self.helper("", n, 0)

    def helper(self, cur, open_available, unclosed):
            # base case when all available open parents have been used
        if open_available == 0:
            return [cur + ")" * unclosed]
        # when there are no unclosed parent present
        elif (unclosed == 0):
            return self.helper(cur + "(", open_available - 1, unclosed + 1)
        # when there are both i.e available parents > 0 and some parents are still unclosed
        return self.helper(cur + "(", open_available - 1, unclosed + 1) + self.helper(cur + ")",
open_available, unclosed - 1)
```

- **Letter Combinations of a Phone Number**

"""
17. Letter Combinations of a Phone Number

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

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

Example:

Input: "23"

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

Although the above answer is in lexicographical order, your answer could be in any order you want.
"""

"""
ALGO - lets take for eg - 23
empty list = result
for each digit in digits i.e 2 and 3 in 23
        current list = cur
        for each character mapped to 2 in dictionary(a list or tuple can also be used to save space) i.e. a b c in abc
                for i in result(starts with 0)
                        cur.append(i + character)

"""

```
class Solution:
    def letterCombinations(self, digits):
        """
        :type digits: str
        :rtype: List[str]
        """
        mapped_digit = {
            '1': '',
            '2': 'abc',
            '3': 'def',
            '4': 'ghi',
            '5': 'jkl',
            '6': 'mno',
            '7': 'pqrs',
            '8': 'tuv',
            '9': 'wxyz',
            '0': ' '}

        res = ['']

        for d in digits:
            cur = []
            print("current list : ")
            print(cur)
```

```
            for chars in mapped_digit[d]:
                print("chars : "+chars)
                for c in res:
                    cur.append(c + chars)
                    print("current list appended : ")
                    print (cur)
            res = cur

        return res if digits else []

"""
OUTPUT for 23

current list :
[]
chars : a
current list appended :
['a']
chars : b
current list appended :
['a', 'b']
chars : c
current list appended :
['a', 'b', 'c']
current list :
[]
chars : d
current list appended :
['ad']
current list appended :
['ad', 'bd']
current list appended :
['ad', 'bd', 'cd']
chars : e
current list appended :
['ad', 'bd', 'cd', 'ae']
current list appended :
['ad', 'bd', 'cd', 'ae', 'be']
current list appended :
['ad', 'bd', 'cd', 'ae', 'be', 'ce']
chars : f
current list appended :
['ad', 'bd', 'cd', 'ae', 'be', 'ce', 'af']
```

current list appended :
['ad', 'bd', 'cd', 'ae', 'be', 'ce', 'af', 'bf']
current list appended :
['ad', 'bd', 'cd', 'ae', 'be', 'ce', 'af', 'bf', 'cf']
"""