Python Basics

1. Write a Python program to reverse a string without using any built-in string reversal functions.

In [1]:

```
s='Hello World'
s[::-1]
```

Out[1]:

```
'dlroW olleH'
```

In [2]:

```
def reverse_string(s):
    reversed_str = ""
    for i in range(len(s) - 1, -1, -1):
        reversed_str += s[i]
    return reversed_str
s='Practising '
reverse_string(s)
```

Out[2]:

```
' gnisitcarP'
```

2. Implement a function to check if a given string is a palindrome.

In [3]:

```
def palindrome(s):
    for i in range(len(s)):
        for j in range(len(s)-1,-1,-1):
            if s[i]==s[j]:
                return 'Is Palindrome'
            else:
                return 'Not a Palindrome'
```

In [4]:

```
s='madam'
palindrome(s)
```

Out[4]:

```
'Is Palindrome'
```

In [5]:

```
s='mybook'
palindrome(s)
```

Out[5]:

```
'Not a Palindrome'
```

3. Write a program to find the largest element in a given list.

In [6]:

```
def largestElement(lst):
    maxi=0
    for i in lst:
        maxi=max(maxi,i)
    return maxi
```

In [7]:

```
lst=[8,0,8,6,5]
largestElement(lst)
```

```
Out[7]: 8
```

```
In [8]:
lst=[1,2,3,4,5,6]
largestElement(lst)
```

```
Out[8]: 6
```

4. Implement a function to count the occurrence of each element in a list.

```
In [9]:
def counter(lst):
    counted={}
    for element in lst:
        if element in counted:
            counted[element]+=1
        else:
            counted[element]=1
    return counted
```

```
In [10]:
counter([1,1,1,2,2,3,3,3])
```

```
Out[10]: {1: 3, 2: 2, 3: 3}
```

```
In [11]:
counter([10,11,11,24,23,23,24,11])
```

```
Out[11]: {10: 1, 11: 3, 24: 2, 23: 2}
```

5. Write a Python program to find the second largest number in a list.

```
In [12]:
def secondLargest(lst):
    for i in lst:
        lst.sort(reverse=True)
    return lst[1]
```

```
In [13]:
secondLargest([1,2,3,60,7])
```

```
Out[13]: 7
```

```
In [14]:
def secondLargest(lst):
## Finding the largest
    maxi=0
    for i in lst:
        maxi=max(i,maxi)
## Removing the largest from list and again searching for the largest in new lost
    lst = [x for x in lst if x != maxi]
    maxi=0
    for x in lst:
        maxi=max(x,maxi)
    return maxi
```

```
In [15]:
secondLargest([1,2,3,60,7])
```

```
Out[15]: 7
```

6. Implement a function to remove duplicate elements from a list.

In [16]:

```
def removeDuplicate(lst):
    unique=list(set(lst))
    return unique
```

In [17]:

```
lst=[1,1,2,3,4,4,5,6,7,8,9]
removeDuplicate(lst)
```

```
Out[17]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [18]:

```
def removeDuplicate(lst):
    counted={}
    for i in lst:
        if i in counted:
            counted[i]+=1
        else:
            counted[i]=1
        if counted[i]>1:
            lst.remove(i)
    return lst
```

In [19]:

```
lst=[1,1,2,3,4,4,5,6,7,8,9]
removeDuplicate(lst)
```

```
Out[19]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

7. Write a program to calculate the factorial of a given number.

In [20]:

```
def factorial(n):
    if n==0 or n==1:
        return 1
    else :
        return n*factorial(n-1)
```

In [21]:

```
factorial(5)
```

```
Out[21]: 120
```

In [22]:

```
factorial(3)
```

```
Out[22]: 6
```

8. Implement a function to check if a given number is prime.

In [23]:

```
def is_prime(number):
    if number < 2:
        return False
    for i in range(2,int(number*0.5) +1):
        if number % i == 0:
            return False
    return True
```

In [24]:

```
is_prime(5)
```

Out[24]:

```
True
```

In [25]:

```
is_prime(10)
```

Out[25]:

```
False
```

9. Write a Python program to sort a list of integers in ascending order.

In [26]:

```
def quick_sort(lst):
    if len(lst)<=1:
        return lst

    pivot=lst[len(lst)//2]
    left=[x for x in lst if x < pivot]
    middle=[x for x in lst if x == pivot]
    right=[x for x in lst if x > pivot]

    return quick_sort(left)+middle+quick_sort(right)
```

In [27]:

```
quick_sort([7,6,5,4,3,7,8,9,11,1,12,10])
```

Out[27]:

```
[1, 3, 4, 5, 6, 7, 7, 8, 9, 10, 11, 12]
```

10. Implement a function to find the sum of all numbers in a list.

In [28]:

```
def summed(lst):
    if len(lst) <=1:
        return lst

    summed=0
    for i in lst:
        summed+=i

    return summed
```

In [29]:

```
summed([1,2,3,4,5,6,7])
```

Out[29]:

```
28
```

11. Write a program to find the common elements between two lists.

In [30]:

```
def common(lst1,lst2):
    result=[]
    for i in lst1:
        if i in lst2:
            result.append(i)
    return result
```

In [31]:

```
lst1=[1,2,3,4,5,6,7,8]
lst2=[1,3,5,7,9,11,12,23]
common(lst1,lst2)
```

```
Out[31]: [1, 3, 5, 7]
```

12. Implement a function to check if a given string is an anagram of another string.

```
In [32]:
def is_anagram(s1,s2):

    s1.lower().replace('  ','')
    s2.lower().replace('  ','')
    s1.strip()
    s2.strip()

    if len(s1)!=len(s2):
        return False

    for i in s1:
        if i in s2:
            for j in s2:
                if j in s1:
                    return True
    return False
```

```
In [33]:
s1='listen'
s2='silent'
is_anagram(s1,s2)
```

```
Out[33]: True
```

```
In [34]:
s1='good'
s2='book'
is_anagram(s1,s2)
```

```
Out[34]: True
```

13. Write a Python program to generate all permutations of a given string.

```
In [69]:
from itertools import permutations

def generate_permutations(string):
    perms = permutations(string)
    for perm in perms:
        print(''.join(perm))

# Test the function
string = input("Enter a string: ")
result=generate_permutations(string)
```

```
Sky
Syk
kSy
kyS
ySk
ykS
```

14. Implement a function to calculate the Fibonacci sequence up to a given number of terms.

In [36]:

```
def fibonacci(n):
    fib = []
    for i in range(n):
        if i == 0:
            fib.append(0)
        elif i == 1:
            fib.append(1)
        else:
            fib.append(fib[i-1] + fib[i-2])
    return fib
```

In [37]:

```
fibonacci(11)
```

Out[37]:

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

In [38]:

```
fibonacci(5)
```

Out[38]:

```
[0, 1, 1, 2, 3]
```

15. Write a program to find the median of a list of numbers.

In [39]:

```
def findMedian(lst):
    sorted_lst = sorted(lst)
    length = len(sorted_lst)
    if length % 2 != 0:
        median = sorted_lst[length // 2]
    else:
        median = (sorted_lst[length // 2 - 1] + sorted_lst[length // 2]) / 2
    return median
```

In [40]:

```
findMedian([1,2,3,4,6])
```

Out[40]:

```
3
```

In [41]:

```
findMedian([1,2,3,4])
```

Out[41]:

```
2.5
```

16. Implement a function to check if a given list is sorted in non-decreasing order.

In [42]:

```
def ascendingOrder(lst):
    for i in range(len(lst)):
        if lst[i] > lst[i+1]:
            return 'The given list is not sorted in non-decreasing order.'
        else:
            return 'The given list is sorted in non-decreasing order.'
```

In [43]:

```
ascendingOrder([2,3,4,5,6,7])
```

Out[43]:

```
'The given list is sorted in non-decreasing order.'
```

In [44]:

```
ascendingOrder([1,3,5,5,6,7])
```

Out[44]:
```
'The given list is sorted in non-decreasing order.'
```

In [45]:
```
ascendingOrder([7,3,6,9])
```

Out[45]:
```
'The given list is not sorted in non-decreasing order.'
```

17. Write a Python program to find the intersection of two lists.

In [46]:
```
def intersection(lst1,lst2):
    intersect=[]
    for i in lst1:
        if i in lst2:
            intersect.append(i)
    return intersect
```

In [47]:
```
lst1=[1,3,3,4,5,6]
lst2=[4,5,6,7,8,9]
intersection(lst1,lst2)
```

Out[47]:
```
[4, 5, 6]
```

18. Implement a function to find the maximum subarray sum in a given list.

In [48]:
```
class Solution:

    def maxSum(self, arr):

        if len(arr)==0:
            return 0

        max_sum,min_sum=arr[0],arr[0]

        result=max_sum

        for i in range(1,len(arr)):
            curr=arr[i]

            temp_max=max(curr, max_sum+curr, min_sum+curr)
            min_sum=min(curr, max_sum+curr, min_sum+curr)

            max_sum=temp_max

            result= max(max_sum,result)

        return result
```

In [49]:
```
Solution().maxSum([1,2,3,4,-1,-2,3])
```

Out[49]:
```
10
```

In [50]:
```
Solution().maxSum([1, -2, 3, 4, -5, 6])
```

Out[50]:
```
8
```

19. Write a program to remove all vowels from a given string.

In [51]:

```
def RemoveVowels(s):
    result=''
    vowels=['a','e','i','o','u','A','E','I','O','U']
    for i in s:
        if i in vowels:
            continue
        else:
            result+=i
    return result
```

In [52]:

```
RemoveVowels('Ball')
```

Out[52]:

```
'Bll'
```

In [53]:

```
RemoveVowels('All Vowels should be removed')
```

Out[53]:

```
'll Vwls shld b rmvd'
```

20. Implement a function to reverse the order of words in a given sentence.

In [54]:

```
def reversal(s):
    result=''
    for i in range(len(s)-1,-1,-1):
        result+=s[i]
    return result
```

In [55]:

```
reversal('Hello Wolrd')
```

Out[55]:

```
'drloW olleH'
```

21. Write a Python program to check if two strings are anagrams of each other.

In [56]:

```
def isAnagram(s1,s2):
    if len(s1)!=len(s2):
        return 'The given strings are not anagrams.'
    for i in s1:
        if i in s2:
            return 'The given strings are anagrams of each other'
```

In [57]:

```
isAnagram('silent','listen')
```

Out[57]:

```
'The given strings are anagrams of each other'
```

In [58]:

```
isAnagram('bad credit','debit card')
```

Out[58]:

```
'The given strings are anagrams of each other'
```

22. Implement a function to find the first non-repeating character in a string.

In [59]:

```
def FirstNonRepeating(s):
    count={}
    for i in s:
        if i not in count:
            count[i]=1
        else:
            count[i]+=1
    for i in s:
        if count[i]==1:
            return f'The first non repeating charcter is {i}.'
```

In [60]:

```
FirstNonRepeating('repeatedly repeating balance')
```

Out[60]:

```
'The first non repeating charcter is d.'
```

23. Write a program to find the prime factors of a given number.

In [61]:

```
def prime_factors(n):
    factors = []
    i = 2
    while i <= n:
        if n % i == 0:
            factors.append(i)
            n = n // i
        else:
            i += 1
    return factors
```

In [62]:

```
prime_factors(24)
```

Out[62]:

```
[2, 2, 2, 3]
```

24. Implement a function to check if a given number is a power of two.

In [63]:

```
def is_power_of_two(n):
    if n <= 0:
        return False
    while n % 2 == 0:
        n //= 2
    return n == 1
```

In [64]:

```
is_power_of_two(10)
```

Out[64]:

```
False
```

In [65]:

```
is_power_of_two(1024)
```

Out[65]:

```
True
```

25. Write a Python program to merge two sorted lists into a single sorted list

In [88]:

```
def mergeSorted(arr1, arr2):
    if len(arr1) == 0:
        return arr2
    if len(arr2) == 0:
        return arr1

    merged = []
    i = j = 0

    while i < len(arr1) and j < len(arr2):
        if arr1[i] <= arr2[j]:
            merged.append(arr1[i])
            i += 1
        else:
            merged.append(arr2[j])
            j += 1

    while i < len(arr1):
        merged.append(arr1[i])
        i += 1

    while j < len(arr2):
        merged.append(arr2[j])
        j += 1

    return merged
```

In [89]:

```
arr1=[1,2,3,4,5,6]
arr2=[0,1,4,5,7,8]
mergeSorted(arr1,arr2)
```

Out[89]:

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

26. Implement a function to find the mode of a list of numbers

In [144…

```
def Mode(nums):

    count={}

    for num in nums:
        if num not in count:
            count[num]=1
        else:
            count[num]+=1

    max_freq = max(count.values())

    mode = [num for num, freq in count.items() if freq == max_freq]

    return f'The Mode for this list is {mode}'
```

In [145…

```
Mode([1,2,2,3,3,4,4,4])
```

Out[145…

```
'The Mode for this list is [4]'
```

In [146…

```
Mode([33,44,55,33,44,77,66,55,11,33])
```

Out[146…

```
'The Mode for this list is [33]'
```

27. Write a program to find the greatest common divisor (GCD) of two numbers

In [163…

```
def GCD(num1,num2):

    D1,D2=[],[]
    for i in range(1,num1):
        if (num1 % i ==0):
            D1.append(i)

    for j in range(1,num2):
        if num2 % j ==0:
            D2.append(j)

    CommonDivisors=[]
    for n in D1:
        if n in D2:
            CommonDivisors.append(n)
    return max(CommonDivisors)
```

In [164…

```
GCD(8,12)
```

Out[164…

```
4
```

In [165…

```
GCD(36,72)
```

Out[165…

```
18
```

28. Implement a function to calculate the square root of a given number.

In [167…

```
def sqrt(number):
    if number < 0:
        raise ValueError("Square root is not defined for negative numbers.")

    guess = number
    previous_guess = 0

    while abs(guess - previous_guess) > 0.0001:
        previous_guess = guess
        guess = (guess + number / guess) / 2

    return guess
```

In [168…

```
sqrt(2)
```

Out[168…

```
1.4142135623746899
```

In [169…

```
sqrt(3)
```

Out[169…

```
1.7320508100147274
```

29. Write a Python program to check if a given string is a valid palindrome ignoring non-alphanumeric characters.

In [180…

```
def isPalindrome(s):

    for i in range(len(s)):
        if s[i].isalnum()==False:
            continue
        else:
            for j in range (len(s)-1,-1,-1):
                if s[i]==s[j]:
                    return 'The given string is a Palindrome.'
                else:
                    return 'The given string is not a Palindrome.'
```

In [181…

```
s='madam'
isPalindrome(s)
```

Out[181…

```
'The given string is a Palindrome.'
```

In [182…

```
s='level'
isPalindrome(s)
```

Out[182…

```
'The given string is a Palindrome.'
```

In [183…

```
s='hello'
isPalindrome(s)
```

Out[183…

```
'The given string is not a Palindrome.'
```

30. Implement a function to find the minimum element in a rotated sorted list.

In [184…

```
def find_minimum(nums):
    left = 0
    right = len(nums) - 1

    while left < right:
        mid = left + (right - left) // 2

        if nums[mid] > nums[right]:
            left = mid + 1
        else:
            right = mid

    return nums[left]
```

In [185…

```
find_minimum([3,4,5,6,2,1])
```

Out[185…

```
1
```

In [186…

```
find_minimum([34,44,45,46,24,31])
```

Out[186…

```
24
```

31. Write a program to find the sum of all even numbers in a list.

In [187…

```
def SumOfEven(nums):
    result=0
    for i in nums:
        if i % 2 !=0:
            continue
        else:
            result+=i
    return result
```

In [188…

```
SumOfEven([1,2,3,4,5,6,8,7])
```

Out[188…

```
20
```

32. Implement a function to calculate the power of a number using recursion.

In [190…

```
def power(base, exponent):
    if exponent == 0:
        return 1
    elif exponent > 0:
        return base * power(base, exponent - 1)
    else:
        return 1 / power(base, -exponent)
```

In [191…

```
power(2,10)
```

Out[191…

```
1024
```

In [192…

```
power(3,12)
```

Out[192…

```
531441
```

33. Write a Python program to remove duplicates from a list while preserving the order.

In [198…

```
def removeDuplicates(nums):
    result = []
    for num in nums:
        if num not in result:
            result.append(num)
    return result
```

In [199…

```
removeDuplicates([1,2,3,6,4,4,5,6])
```

Out[199…

```
[1, 2, 3, 6, 4, 5]
```

34. Implement a function to find the longest common prefix among a list of strings.

In [200…

```
def longestCommonPrefix(strs):
    if not strs:
        return ""

    prefix = strs[0]
    for string in strs[1:]:
        while not string.startswith(prefix):
            prefix = prefix[:-1]
            if not prefix:
                return ""

    return prefix
```

In [203…

```
longestCommonPrefix(['repeater','repeating','repeated','repetetive'])
```

Out[203…

```
'repe'
```

35. Write a program to check if a given number is a perfect square.

In [204…

```
def isPerfectSquare(n):

    import math

    sqrt = int(math.sqrt(n))
    return sqrt * sqrt == n
```

In [205…

```
isPerfectSquare(24)
```

Out[205…

```
False
```

In [206…

```
isPerfectSquare(1024)
```

Out[206…

```
True
```

36. Implement a function to calculate the product of all elements in a list.

In [215…

```
def Product(nums):
    if len(nums)==0:
        return 'Please enter a valid list.'
    elif len(nums)==1:
        return nums[0]
    else:
        return nums[0] * Product(nums[1:])
```

In [216…

```
nums=[1,2,3,5,10]
Product(nums)
```

Out[216…

```
300
```

In [217…

```
nums=[7,6,5,2,3,4]
Product(nums)
```

Out[217…

```
5040
```

37. Write a Python program to reverse the order of words in a sentence while preserving the word order.

In [3]:

```
def reverse_sentence(sentence):
    words = sentence.split()
    reversed_words = words[::-1]
    reversed_sentence = ' '.join(reversed_words)
    return reversed_sentence
```

In [4]:

```
reverse_sentence('Hello I am a Student')
```

Out[4]:

```
'Student a am I Hello'
```

38. Implement a function to find the missing number in a given list of consecutive numbers.

In [23]:

```
def find_missing_number(nums):
    n = len(nums) + 1
    expected_sum = (n * (n + 1)) // 2
    actual_sum = sum(nums)
    missing_number = expected_sum - actual_sum
    return missing_number
```

In [26]:

```
nums=[1,2,3,5,6]

find_missing_number([1,2,3,5,6])
```

Out[26]:

```
4
```

39. Write a program to find the sum of digits of a given number.

In [53]:

```
def SummedDigits(n):
    total_sum = 0
    while n > 0:
        digit = n % 10
        total_sum += digit
        n //= 10
    return total_sum
```

In [55]:

```
SummedDigits(1024)
```

Out[55]:

```
7
```

In [56]:

```
SummedDigits(2233)
```

Out[56]:

```
10
```

40. Implement a function to check if a given string is a valid palindrome considering case sensitivity.

In [63]:

```
def isPalindrome(s):

    for i in range(len(s)):
        for j in range (len(s)-1,-1,-1):

            if s[i]==s[j]:
                return 'The given string is a valid Palindrome.'
            else:
                return 'The given string is not a valid Palindrome.'
```

In [64]:

```
isPalindrome('MadAm')
```

Out[64]:

```
'The given string is not a valid Palindrome.'
```

In [65]:

```
isPalindrome('Level')
```

Out[65]:

```
'The given string is not a valid Palindrome.'
```

In [66]:

```
isPalindrome('LeveL')
```

Out[66]:

```
'The given string is a valid Palindrome.'
```

41. Write a Python program to find the smallest missing positive integer in a list.

In [78]:

```
def find_smallest_missing_positive(nums):
    num_set = set(nums)
    smallest_missing = 1

    while smallest_missing in num_set:
        smallest_missing += 1

    return smallest_missing
```

In [80]:

```
find_smallest_missing_positive([1,2,3,4,5,6,9])
```

Out[80]:

```
7
```

42. Implement a function to find the longest palindrome substring in a given string.

In [3]:

```
def longest_palindrome_substring(s):
    n = len(s)
    longest = ""

    for i in range(n):
        for j in range(i, n):
            substring = s[i:j+1]
            if substring == substring[::-1] and len(substring) > len(longest):
                longest = substring

    return longest
```

In [4]:

```
longest_palindrome_substring('babad')
```

Out[4]:

```
'bab'
```

43. Write a program to find the number of occurrences of a given element in a list.

In [5]:

```
def counter(lst):
    counter={}
    for i in lst:
        if i not in counter:
            counter[i]=1
        else:
            counter[i]+=1
    return counter
```

In [6]:

```
counter([1,2,3,4,4,1,2,2,1,3])
```

Out[6]:

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

44. Implement a function to check if a given number is a perfect number.

In [10]:

```
def perfect_number(num):

    Divisors_list=[]
    for i in range(1,num):
        if (num % i ==0):
            Divisors_list.append(i)
    return sum(Divisors_list)==num
```

In [11]:

```
perfect_number(6)
```

Out[11]:

```
True
```

In [12]:

```
perfect_number(12)
```

Out[12]:

```
False
```

45. Write a Python program to remove all duplicates from a string.

In [34]:

```
def duplicated(s):
    s1 = s.split(' ')
    count={}
    for i in s1:
        if i not in count:
            count[i]=1
        else:
            s1.remove(i)
    return ' '.join(s1)
```

In [35]:

```
s='Hello Hello'
duplicated(s)
```

Out[35]:

```
'Hello'
```

In [36]:

```
s='My name is name is XYZ'
duplicated(s)
```

Out[36]:

```
'My is name is XYZ'
```

46. Implement a function to find the first missing positive

In [37]:

```
def find_missing_number(nums):
    n = len(nums) + 1
    expected_sum = (n * (n + 1)) // 2
    actual_sum = sum(nums)
    missing_number = expected_sum - actual_sum
    return missing_number
```

In [38]:

```
find_missing_number([1,2,3,5,6,7,8])
```

Out[38]:

```
4
```

In [39]:

```
find_missing_number([1,2,3,4,5,6,7,8,9,10,11])
```

Out[39]:

```
12
```

The End