In [118…

```python
# Write a python code for Fibonacci series without recursive function
def Fibonacci_iter(num):
    """ This is the function which takes non-negative number (N) as input and prints the Fibonacci searines till N terms"""
    n1 = 0
    n2 = 1
    if num==0:
        print(n1)
    elif num ==1:
        print(n1,n2, end = ' ')
    else:
        print(n1,n2, end = " ")
        while num>2:
            n3 = n1+n2
            n1, n2 = n2, n3
            num = num-1
            print(n3, end = " ")

num = int(input("Enter the number:"))
Fibonacci_iter(num)
```

```
Enter the number:5
0 1 1 2 3
```

In [7]:

```python
# Python code to extract digits from string
def str_digits(input_str):
    """ This function takes any string as input and returns only digits prsent in the string"""
    digits = []
    for i in input_str:
        if str(i).isdigit() == True:
            digits.append(i)
    return ''.join(digits)


def str_digits1(input_str):
    """ This function takes any string as input and returns only digits prsent in the string"""

    digits = ''.join([i for i in input_str if str(i).isdigit() == True])
    return digits


input_str = "dinesh23"

for i in str_digits(input_str):
    print(i, end = "  ")

print()

for i in str_digits1(input_str):
    print(i, end = "  ")
```

```
2  3
2  3
```

In [46]:

```python
# Check if provided number is Armstrong or not
num = int(input("Enter the number:"))
def armstrong(num):
    """ This function takes any integer number as input and returns if the provided number is armstrong or not.
        Armstrong is a number where sum of cube of digits of a number equals to the number"""
    lst = []
    while num!=0:
        digit = num%10
        num = num//10
        lst.append(digit)
    return lst

cubes = [ i**3 for i in armstrong(num)]
if num == sum(cubes):
    print("Provided number is Armstrong")
else:
    print("Provided number is not Armstrong")
```

```
Enter the number:153
Provided number is Armstrong
```

In [305…

```python
# Divisible by 3, print Fizz, by 5 - print Buzz, By 15 - print FizzBuzz - WITHOUT DICTIONARY
def fizzbuzz(num):
    """ This function takes any number and return Fizz if divisible by 3, returns Buzz if divisible by 5 and return FizzBuzz
        if divisble by 15"""
    result = ""
    if num%3==0 and num%15!=0:
        result = "Fizz"
    elif num%5==0 and num%15!=0:
        result = "Buzz"
    elif num%15 == 0:
        result = "FizzBuzz"
    else:
        result = num
    return result

num = int(input("Enter the number:"))
for i in range(1,num+1):
    print(fizzbuzz(i), end = " ")
```

```
Enter the number:30
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buzz Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz
```

In [16]:

```python
# Divisible by 3, print Fizz, by 5 - print Buzz, By 15 - print FizzBuzz - WITH DICTIONARY
def FizzBuzz_withDict(num):
    """ This function takes any number and return Fizz if divisible by 3, returns Buzz if divisible by 5 and return FizzBuzz
        if divisble by 15"""
    dict = {3: 'Fizz', 5:'Buzz'}
    result = ""
    for key in dict:
        if num%key == 0:
            result = result + dict[key]
    if result:
        return result
    else:
        return num

for i in range(1,31):
    print(FizzBuzz_withDict(i), end = " ")
```

```
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16 17 Fizz 19 Buzz Fizz 22 23 Fizz Buzz 26 Fizz 28 29 FizzBuzz
```

In [60]:

```python
# Python code to flatten the nested list
def flatten_list(lst):
    flat_list = []
    for i in lst:
        if type(i) == list:
            flat_list.extend(flatten_list(i))
        else:
            flat_list.append(i)
    return flat_list

lst = [[1,2,[3]],4,[5]]
flatten_list(lst)
```

Out[60]: `[1, 2, 3, 4, 5]`

In [18]:

```python
# Find the combinations of 3 numbers having required sum
def required_sum(lst, num):
    num_lst = []
    for i in range(len(lst)):
        for j in range(i+1, len(lst)):
            if lst[i]+lst[j]<num:
                for k in range (j+1,len(lst)):
                    if lst[i]+lst[j]+lst[k] ==num:
                        num_lst.append([lst[i], lst[j], lst[k]])
            else:
                if lst[i]+lst[j]==num:
                    num_lst.append([lst[i],lst[j]])
    num_tup = [tuple(i) for i in num_lst]
    return num_tup

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

Out[18]: `[(1, 2, 6), (1, 3, 5), (2, 3, 4), (3, 6), (4, 5)]`

In [81]:

```python
# Find the combinations of 2 numbers having required sum
def sum_comb(lst, num):
    num_lst = []
    for i in range(len(lst)):
        for j in range(i+1,len(lst)):
            if lst[i]+lst[j] == num:
                num_lst.append([lst[i],lst[j]])
    tup_lst = [tuple(i) for i in num_lst]
    return tup_lst

sum_comb([1,2,3,4,5,0],5)
```

Out[81]: `[(1, 4), (2, 3), (5, 0)]`

In [21]:

```python
# Find the combinations of 2 numbers having required sum
def sum_comb_quick(lst, num):
    return [(j,m) for i,j in enumerate(lst) for l,m in enumerate(lst[i+1:]) if j+m == num]

sum_comb_quick([1,2,3,4,5,0],5)
```

Out[21]: `[(1, 4), (2, 3), (5, 0)]`

In [74]:

```python
# Find the combinations of 2 numbers having required sum and return position of numbers in list
def sum_index_quick(lst, num):
    return [(i,l+i+1) for i,j in enumerate(lst) for l,m in enumerate(lst[i+1:]) if j+m == num]

sum_index_quick([1,2,3,4,5,0],5)
```

Out[74]: `[(0, 3), (1, 2), (4, 5)]`

In [26]:

```python
# Python code to find the number of characters in string
def character_nums(input_str):
    dict = {}
    for i in input_str:
        if i in dict:
            dict[i] = dict[i] + 1
        else:
            dict[i] = 1
    return dict

character_nums("BOOK")
```

Out[26]: `{'B': 1, 'O': 2, 'K': 1}`

In [27]:

```python
# Python code to find the number of characters in string
def character_nums_quick(input_str):
    dict = {}
    for i in input_str:
        dict[i] = dict.get(i,0)+1
    return dict

character_nums_quick("BOOK")
```

Out[27]: `{'B': 1, 'O': 2, 'K': 1}`

In [306…

```python
# Reverse the provided number
def reverse_num(num):
    digits = []
    while num!=0:
        dig = num%10
        num = num//10
        digits.append(dig)
    digits = digits[::-1]
    rev_num = 0
    for i in range(len(digits)):
        rev_num = rev_num + digits[i]*10**i
    return rev_num

reverse_num(454)
```

Out[306… `454`

In [29]:

```python
# Write a python code to check if string is palindrome or not
def pallindrome(input_str):
    clean_str = ''.join(e for e in input_str if e.isalnum()).lower()
    rev_str = clean_str[::-1]
    if clean_str==rev_str:
        return "String is Pallindrome"
    else:
        return "String is not Pallindrome"

pallindrome("A man, A plan, a canal, Panama!")
```

Out[29]: `'String is Pallindrome'`

In [56]:

```python
def pallindrome_alternate(input_str):
    clean_str = ''.join(e for e in input_str if e.isalnum()).lower()
    rev_str = ""
    for i in range(len(clean_str)-1,-1,-1):
        rev_str = rev_str + clean_str[i]
    if clean_str==rev_str:
        return "String is Pallindrome"
    else:
        return "String is not Pallindrome"

pallindrome_alternate("A man, A plan, a canal, Panama!")
```

Out[56]: `'String is Pallindrome'`

In [32]:

```python
# Python code to find the sum of Factorial of digits of a given numbers
def fact_digits(num):
    digits = []
    while num!=0:
        digit = num%10
        num = num//10
        digits.append(digit)
    def fact(number):
        if number <=0:
            return 1
        else:
            return number*fact(number-1)

    return sum([fact(i) for i in digits])

fact_digits(145)
```

Out[32]: `145`

In [185…

```python
# Write python code to rotate the list by given positions in left or right

def rotate(lst,k):
    if k>= len(lst):
        k = k%len(lst)
    return lst[k:]+lst[:k]

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

Out[185… `[2, 3, 4, 5, 1]`

In [33]:

```python
# Write a python program to find pythogorean triplets in given limit

def pyth_trip(limit):
    triplet = []
    for i in range(1,limit+1):
        for j in range(i+1,limit+1):
            c = (i**2+j**2)**0.5
            if c==int(c) and c<=limit:
                triplet.append((i,j,int(c)))
    return triplet

pyth_trip(15)
```

Out[33]: `[(3, 4, 5), (5, 12, 13), (6, 8, 10), (9, 12, 15)]`

In [187…

```python
# Write a python code to know if provided number is power of 2 or not
def power_of_two(num):
    Flag = False
    for i in range(1,num):
        if num == 2**i:
            print(f"{num} is 2power{i}")
            Flag = True
        if i+1==num and Flag == False:
            print(f"{num} is not power of 2")

power_of_two(16)
power_of_two(15)
```

```
16 is 2power4
15 is not power of 2
```

In [39]:

```python
# Write a python code to find number of trailing zeros in factorial of number
def trailing_zeros(num):
    product = 1
    if num==0:
        product = 1
    for i in range(1,num+1):
        product = product*i
    dict = {}
    while product!=0:
        d = product%10
        product = product//10
        if d in dict:
            dict[d] = dict[d]+1
        else:
            dict[d] = 1
    return dict[0]

trailing_zeros(5)
```

Out[39]: `1`

In [188…

```python
# Write a python code to find missing number in a list
def missing_number(lst):
    n = len(lst)+1
    esum = n*(n+1)//2
    asum = sum(lst)
    mn = esum-asum
    return mn

missing_number([1,5,2,4])
```

Out[188… `3`

In [41]:

```python
# Write a python code to find missing number in a list
def missing_number_quick(lst):
    max_num = max(lst)
    return [i for i in range(1,max_num) if i not in lst][0]

missing_number_quick([1,5,2,4])
```

Out[41]: `3`

In [16]:

```python
# Write a python code for thousand's seperator, for example, if input = 1000000, then output = 1,000,000
def thousand_sep(num):
    result = ""
    for i,j in enumerate(str(num)[::-1], start = 1):
        result = result + j
        if i%3==0 and i != len(str(num)):
            result = result + ','
    return result[::-1]

thousand_sep(500000)
```

Out[16]: `'500,000'`

In [32]:

```python
# Write a python code for thousand's seperator, for example, if input = 1000000, then output = 1,000,000
def thousand_sep_quick(num):
    num = str(num)[::-1]
    n = 3
    lst = [num[i:i+n] for i in range(0,len(num),n)]
    return ','.join(lst)[::-1]

thousand_sep_quick(500000)
```

Out[32]: `'500,000'`

In [38]:

```python
# Write a python code to print prime number in a given range
def prime_num_range(start,end):
    for num in range(start,end+1):
        if all(num%i !=0 for i in range(2,num)):
            yield num

for i in prime_num_range(100,200):
    print(i, end = " ")
```

```
101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199
```

In [39]:

```python
# Write a python code to sort the list in ascending order without using sort function
def sort_list(lst):
    for i in range(len(lst)):
        for j in range(i+1, len(lst)):
            if lst[i]>lst[j]:
                lst[i], lst[j] = lst[j], lst[i]
    return lst
sort_list([4,7,1,4,3])
```

Out[39]: `[1, 3, 4, 4, 7]`

In [56]:

```python
# Write a python code to elliminate duplicate elements (keep the unique elements only) from a given list
def unique(lst):
    return [j for i,j in enumerate(lst) if j not in lst[i+1:]]
unique([1,2,2,3,3,3,4,4,5,5])
```

Out[56]: `[1, 2, 3, 4, 5]`

In [189…

```python
# Write a python code to keep repeated elements from a given list
def unique(lst):
    return [j for i,j in enumerate(lst) if j in lst[i+1:]]
unique([1,2,2,3,3,3,4,4,5,5,6,6,6,6])
```

Out[189… `[2, 3, 3, 4, 5, 6, 6, 6]`

In [57]:

```python
# Write a python code to sum the digit of numbers and return the count of numbers that matches with sum
def sum_of_digits(num):
    digit_sum = []
    for i in str(num):
        digits = [int(j) for j in i.split()]
        digit_sum.append(sum(digits))
    return sum(digit_sum)
lst = []
def count_digit_sum(start,end):
    for i in range(start, end+1):
        lst.append(sum_of_digits(i))
    return len([j for i,j in enumerate(lst) if j in lst[i+1:]])
count_digit_sum(4,17)
```

Out[57]: `5`

In [90]:

```python
# Write a python code to show the index of repeated characters in string
def index_rep_char(input_str):
    return [(i,l+i+1) for i,j in enumerate(input_str) for l,m in enumerate(input_str[i+1:]) if j==m]

index_rep_char("BOOK")
```

Out[90]: `[(1, 2)]`

In [117…

```python
# Write a python code to Remove the repeated characters from the string and print the remaining string
def remove_repeated(input_str):
    dict = {}
    for i in input_str:
        if i in dict:
            dict[i] = dict[i]+1
        else:
            dict[i] = 1
    return {k:v for (k,v) in zip(dict.keys(),dict.values()) if v<=1}

remove_repeated("BOOK")
```

Out[117… `{'B': 1, 'K': 1}`

In [127…

```python
# Write a python code for Fibonacci series using recursive function
def fibbonacci_recursive(num):
    if num<2:
        return num
    else:
        return fibbonacci_recursive(num-1)+fibbonacci_recursive(num-2)

for i in range(5):
    print(fibbonacci_recursive(i), end = " ")
```

```
0 1 1 2 3
```

In [133…

```python
# Write a python code for factorial of number using recursive function
def factorial(num):
    if num<2:
        return num
    else:
        return num * factorial(num-1)

for i in range(5):
    print(factorial(i), end = " ")
```

```
0 1 2 6 24
```

In [136…

```python
# Python code to locate index position of user provided number on list
def index_pos(lst, num):
    return [i for i,j in enumerate(lst) if j==num][0]

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

Out[136… `2`

In [163…

```python
# Check if given number is pallindrome or not
def number_pallindrome(num):
    n = num
    lst = []
    while num!=0:
        digit = num%10
        num = num//10
        lst.append(digit)
    lst = lst[::-1]
    rev_num = 0
    for i,j in enumerate(lst):
        rev_num = rev_num + j*10**i
    if rev_num == n:
        return f'{n} is pallindrome'
    else:

        return f'{n} is not pallindrome'

number_pallindrome(121)
```

Out[163… `'121 is pallindrome'`

In [59]:

```python
# Write a python code to find the second largest number from the given list
def second_largest(lst):
    largest = float('-inf')
    sec_largest = float('-inf')
    for num in lst:
        if num> largest:
            sec_largest = largest
            largest = num

        elif num>= sec_largest and num != largest:
            sec_largest = num
    if sec_largest == float('-inf'):
        return "No second largest number"
    else:
        return sec_largest

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

Out[59]: `7`

In [251…

```python
# Write a python code to find the sum of max two numbers from the given list
def second_large(lst):
    largest = float('-inf')
    second_largest = float('-inf')
    for num in lst:
        if num>largest:
            second_largest = largest
            largest = num
        elif num>=second_largest and num!=largest:
            second_largest = num
    return second_largest+largest
second_large([8,4,6,3,7,9])
```

Out[251… `17`

In [252…

```python
# Write a python code to find the sum of the digits of the numbers
def sum_of_digits(num):
    lst = []
    while num!=0:
        digit = num%10
        num = num//10
        lst.append(digit)
    return sum(lst)
sum_of_digits(452)
```

Out[252… `11`

In [264…

```python
# Write a python code to find the armstrong numbers from given range
def armstrong_number(num):
    lst = []
    while num!=0:
        digits = num%10
        num = num//10
        lst.append(digits)
    return sum([i**3 for i in lst])

for i in range(1,1000):
    if i == armstrong_number(i):
        print(i, end = " ")
```

```
1 153 370 371 407
```

In [276…

```python
# Write a Python code to print AABBCCCDDDD as A2B2C3D4

def string_pattern(input_str):
    dict = {}
    for i in input_str:
        if i in dict:
            dict[i] = dict[i] +1
        else:
            dict[i] = 1
    result = ""
    for k,v in dict.items():
        result = result + k + str(v)
    return result
string_pattern("AABBCCCDDDD")
```

Out[276… `'A2B2C3D4'`

In [287…

```python
# Python code to find sublist of continous elements which has max sum
# lst = [1,-2,3,4,-1,2,1,-5,4]
lst = [1,2,-3,-4,5,6]
def sublist(lst):
    sum_lst = []
    for i in range(len(lst)):
        for j in range(i+1, len(lst)+1):
            new_lst = lst[i:j]
            sum_temp = sum(new_lst)
            sum_lst.append(sum_temp)
            if sum_temp == max(sum_lst):
                start = i
                end = j
                r_sum = sum_temp
    return (lst[start:end],r_sum)
sublist(lst)
```

Out[287… `([5, 6], 11)`

In [309…

```python
# Write a python code to divide the given list in given number of equal chunks
def equal_chunks(lst,k):
    n = len(lst)//k
    return [lst[i:i+n] for i in range(0,len(lst),n)]
equal_chunks(lst,3)
```

Out[309… `[[1, 2], [-3, -4], [5, 6]]`

In [296…

```python
# Explain the decorator to add third number to function of addition of two numbers
def decorator_to_add_3rd_num(func):
    def wrapper(a,b,c):
        result = func(a,b)+c
        return result
    return wrapper
@decorator_to_add_3rd_num
def sum_two(a,b):
    return a+b
sum_two(5,10,3)
```

Out[296… `18`

In [301…

```python
# Python code to demonstrate the concept of nested decorator
def string_upper(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result.upper()
    return wrapper
def string_excl(func):
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        return result+'!'
    return wrapper
@string_upper
@string_excl
def greet(name):
    return f"Hello, {name}"

greet("Dinesh")
```

Out[301… `'HELLO, DINESH!'`

In [3]:

```python
def roman_to_int(roman):
    roman_numerals = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    result = 0
    prev_value = 0
    for numeral in reversed(roman):
        value = roman_numerals[numeral]
        if value < prev_value:
            result -= value
        else:
            result += value
        prev_value = value
    return result
# Example usage
roman_numeral = "MIV"
numeric_value = roman_to_int(roman_numeral)
print(f'The numeric value of {roman_numeral} is: {numeric_value}')
```

```
The numeric value of MIV is: 1004
```

In [32]:

```python
# Python code to print simple traingle pattern
def sip_tri(num):
    for i in range(1,num+1):
        print("* "*i)
sip_tri(5)
```

```
*
* *
* * *
* * * *
* * * * *
```

In [36]:

```python
# Python code to print simple traingle pattern with numbers
def sip_tri_num(num):
    for i in range(1,num+1):
        for j in range(1, i+1):
            print(str(j)+" ", end = "")
        print()
sip_tri_num(5)
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

In [30]:

```python
# Python code to print tringle pattern with numbers
def triangle(num):
    for i in range(1,num+1):
        print(" "*(num-i), end = "")
        for j in range(1, i+1):
            print(str(j)+" ", end = "")
        print()
triangle(5)
```

```
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

In [37]:

```python
# Python code to print tringle pattern with letters
def triangle_chr(num):
    alpha = 65
    for i in range(1,num+1):
        print(" "*(num-i), end = "")
        for j in range(1, i+1):
            print(chr(alpha), end = " ")
            alpha +=1
        alpha = 65
        print()
triangle_chr(5)
```

```
    A
   A B
  A B C
 A B C D
A B C D E
```

In [46]:

```python
# Python code to print tringle pattern with symbols
def triangle1(num):
    for i in range(1,num+1):
        print(' '*(num-i)+ '* '*i)
triangle1(5)
```

```
    *
   * *
  * * *
 * * * *
* * * * *
```

In [52]:

```python
# Python code to print reverse triagnle pattern
def rev_tri(num):
    for i in range(1, num+1):
        print(" "*(num-i)+"*"*i)
rev_tri(5)
```

```
    *
   **
```