1_Factorial_Number.py

```
def factorial(x):
    res = 1 # (0!=1)
    for i in range(1,(x+1)):
        res *= i
    return res

def main():
    num = int(input("Enter a Number \n"))
    print(factorial(num))

if __name__ == '__main__':
    main()
```

2_Count_Number_Of_Digits.py

```
#write a program to count the number of digits
#input 2597

''' steps
1. check if the digit exists n>0
2. remove the last digit  n/=10
3. increase the value of count by 1 count+=1
4. Repeat step 1 to step 3 till digits remains
'''

'''def num_of_digits(num):
    count = 0
    while(num > 0):
        count += 1
        num = num // 10 #floor division for hole number
    return count

def main():
    num = int(input("Enter a Number \n"))
    print(num_of_digits(num))

if __name__ == '__main__':
    main()'''


def num_of_digits(num):
    count = 0
    for i in num:
        if num.isdigit:
            count += 1
    return count

def main():
    num = input("Enter a Number \n")
    print(num_of_digits(num))

if __name__ == '__main__':
    main()
```

3_Trailing_Zeros.py

```
# Given a number find the number of trailing zeros of its factorial
```

```
#input 10

def trailing_zero(num):
    res = 0
    power_of_5 = 5

    while (num >= power_of_5):
        res = res + (num//power_of_5)
        power_of_5 = power_of_5 * 5
    return res

def main():
    num = int(input("Enter a Number \n"))
    print(trailing_zero(num))

if __name__ == '__main__':
    main()
```

4_Find_GCD.py

```
# brute force apporach

def find_gcd (a,b):
    min = 0
    if (a > b):
        min = b
    else:
        min = a

    for i in range(min ,0 ,-1):
        if (a % i == 0) and (b % i == 0):
            return i

def main():
    lst= input("Enter a number \n").split()
    print(find_gcd(int(lst[0]),int(lst[1])))

if __name__ == '__main__':
    main()
```

5_Euclid_GCD.py

```
# brute force apporach

def find_gcd (a,b):
    while(a!=b):
        if (a>b):
            a = a-b
        else:
            b = b-a
    return a

def main():
    lst= input("Enter a number \n").split()
    print(find_gcd(int(lst[0]),int(lst[1])))

if __name__ == '__main__':
    main()
```

6_Gabriel_Lame_GCD.py

```
# brute force apporach

def find_gcd (a,b):
    while(a!=0 and b!=0):
        if (a>b):
            a = a%b
        else:
            b = b%a

    if (a!=0):
        return a
    else:
        return b

def main():
    lst= input("Enter a number \n").split()
    print(find_gcd(int(lst[0]),int(lst[1])))

if __name__ == '__main__':
    main()
```

7_Find_LCM.py

```
#Least common multiple

def find_lcm (a,b):
    res = max(a,b)
    while(True):
        if (res%a == 0 and res%b == 0):
            break
        res = res + 1
    return res

def main():
    lst= input("Enter a number \n").split()
    print(find_lcm(int(lst[0]),int(lst[1])))

if __name__ == '__main__':
    main()
```

8_Euclid_LCM.py

```
#Least common multiple

def find_lcm (a,b):
    return (a*b)//(find_gcd(a,b))

def find_gcd (a,b):
    while (a!=0 and b!=0):
        if a>b:
            a = a%b
        else:
            b = b%a

    if a!=0:
        return a
    else:
        return b
```

```
def main():
    lst= input("Enter a number \n").split()
    print(find_lcm(int(lst[0]),int(lst[1])))

if __name__ == '__main__':
    main()
```

9_Find_Prime.py

```
#first approach o(n)

def is_prime(num):
    for i in range (2,(num//2)+1):
        if num%i==0:
           return False
    return True

def main():
    num = int(input("Enter a Number \n"))
    print(is_prime(num))

if __name__ == '__main__':
    main()
```

10_Find_Prime_2.py

```
#second apporach o(math.square n)

import math


def is_prime(num):
    for i in range (2,int(math.sqrt(num))+1):
        if num%i==0:
           return False
    return True

def main():
    num = int(input("Enter a Number \n"))
    print(is_prime(num))

if __name__ == '__main__':
    main()
```

11_Find_Prime_3.py

```
import math

#o(root n ) only but elliminating the more values

def is_prime(num):
    if num==1:
       return False
    if num==2 or num==3:
        return True
    if num%2==0 or num%3==0:
        return False
    for i in range (5,int(math.sqrt(num))+1,6):
```

```
        if num%i==0 or num%(i+1)==0:
           return False
    return True

def main():
    num = int(input("Enter a Number \n"))
    print(is_prime(num))

if __name__ == '__main__':
    main()
```

12_Find_Prime.py

```
# Find all the numbers from 1 to n

import math


def find_prime_all(n):
    for i in range (2,n+1):
        if (is_prime(i)):
            print(i)

def is_prime(num):
    if num==1:
       return False
    if num==2 or num==3:
        return True
    if num%2==0 or num%3==0:
        return False
    for i in range (5,int(math.sqrt(num))+1,6):
        if num%i==0 or num%(i+1)==0:
           return False
    return True

def main():
    num = int(input("Enter a Number \n"))
    find_prime_all(num)

if __name__ == '__main__':
    main()
```

13_Find_Prime_Sieve_Of_Eratosthenes.py

```
# Find all the numbers from 1 to n

import math


def find_prime_all(n):
    prime = [False] * (n+1)
    for i in range (2,int(math.sqrt(n))+1):
        if prime[i] == False:
            for j in range(i*i,n+1,i):
                prime[j] = True
    for i in range(2,n+1):
        if(prime[i]==False):
            print(i)
```

```
def main():
    num = int(input("Enter a Number \n"))
    find_prime_all(num)

if __name__ == '__main__':
    main()
```

14_Find_Factors.py

```
#0(n)

def find_factors(num):
    for i in range(1,num+1):
        if(num%i==0):
            print(i)

def main():
    num = int(input("Enter a Number \n"))
    find_factors(num)

main()
```

15_Find_Factors_Effective.py

```
#0(root n)
import math

def find_factors(num):
    for i in range(1,int(math.sqrt(num))+1):
        if (num%i==0):
            print(i)
            if i!=(num//i):
                print(num//i)

def main():
    num = int(input("Enter a Number \n"))
    find_factors(num)

main()
```

16_Find_Factors_Acending_Order.py

```
#0(root n)
import math

def find_factors(num):
    for i in range(1,int(math.sqrt(num))+1):
        if (num%i==0):
            print(i)
            '''if i!=(num//i):
                print(num//i)'''
    for i in range(int(math.sqrt(num)),0,-1):
        if (num%i==0 and i!=(num//i)):
            print(num//i)

def main():
    num = int(input("Enter a Number \n"))
    find_factors(num)
```

```
main()
```

17_Prime_Factors.py

```
#Find all the prime factors of a given number(o(n))

def prime_factor(n):
    i = 2
    while (n>1):
        while(n%i==0):
            print(i)
            n = n//i
        i +=1

def main():
    num = int(input("Enter a Number\n"))
    prime_factor(num)

if __name__ == '__main__':
    main()
```

18_Prime_Factors_Effective.py

```
#Find all the prime factors of a given number(o(root n * log(n)))
import math


def prime_factor(n):
    i = 2
    while(i<=math.sqrt(n)):
        while(n%i==0):
            print(i)
            n = n//i
        i +=1
    if n>1:
        print(n)

def main():
    num = int(input("Enter a Number\n"))
    prime_factor(num)

if __name__ == '__main__':
    main()
```

19_Demical_to_Binary.py

```
def decimal_to_binary(n):
    b = ' '
    while(n>=1):
        x = n%2
        n = n//2
        b = str(x) + b
    return b

def main():
    num = int(input("Enter a Number \n"))
    print(decimal_to_binary(num))

if __name__ == '__main__':
```

```
    main()
```

20_Binary_to_Decimal.py

```
def decimal_to_binary(n):
    b = ''
    while(n>=1):
        x = n%2
        n = n//2
        b = str(x) + b
    return b

def binary_to_decimal(b):
    result = 0
    pow_of_2 = 1

    for i in range(len(b)-1, -1 , -1):
        if (b[i]=='1'):
            result = result + pow_of_2
        pow_of_2 = pow_of_2 * 2

    return result

def main():
    num = int(input("Enter a Number \n"))
    b = decimal_to_binary(num)
    print(b)
    print(binary_to_decimal(b))

if __name__ == '__main__':
    main()
```

21_Bitwise_Operator.py

```
def bit_wise_operator():
    print(5 | 9)
    print(5 & 9)
    print(5 ^ 9)
    print(5 << 1)
    print(5 << 2)
    print(5 >> 1)
    print(5 >> 2)
    print(-5 >> 1)


bit_wise_operator()
```

22_Swith_On_Bit.py

```
def main():
    #n = 36
    n = int(input("Enter a Number \n"))
    i = 3
    on_mask = 1 << i
    print(n|on_mask)

if __name__ == '__main__':
    main()
```

23_Swith_Off_Bit.py

```
def main():
    n = int(input("Enter a Number \n"))
    i = 5
    off_mask = ~(1<<i)
    print(n & off_mask)

main()
```

24_Toggle.py

```
def main():
    n = int(input("Enter a Number \n"))
    i = 5
    t_mask = 1 << i
    print(n ^ t_mask)

main()
```

25_Check_On_Off.py

```
def main():
    n = int(input("Enter a Number \n"))
    i = 5
    c_mask = 1 << i

    if ((n&c_mask)==0):
        print("OFF")
    else:
        print("ON")

main()
```

26_Right_Most_Bit.py

```
def right_most_bit(n):
    m = 1
    pos = 0

    while((n&m)==0):
        m = m+1
        pos = pos+1
    return pos+1

def main():
    n = int(input("Enter a Number \n"))
    print(right_most_bit(n))

main()
```

27_Right_Most_Bit_Effective.py

```
import math

def right_most_bit(n):
    # a = n^(n&(n-1))
```

```
    # return  a // 8 (we want 2^3 power + 1 should return)
    return  math.log(n^(n&(n-1)),2) + 1

def main():
    n = int(input("Enter a Number \n"))
    print(right_most_bit(n))

main()
```

28_Count_Set_Bit.py

```
def count_set_bit(n):
    count = 0
    while (n >0 ):
        n = n & (n-1)
        count +=1
    return count

def main():
    num = int(input("Enter a Number \n"))
    print(count_set_bit(num))

main()
```

29_Check_Power_Of_2.py

```
def check_power_of_2(num):
    if num == 0:
        return False

    return num&(num-1) == 0

def main():
    num = int(input("Enter a Number \n"))
    print(check_power_of_2(num))

main()
```

30_Find_Lonely_Integer.py

```
# o(N log N) // brute force apporach

def lonely_integer(arr):
    arr.sort()
    for i in range(0,len(arr)-2,2):
        if (arr[i] != arr[i+1]):
            return arr[i]
    return arr[len(arr)-1]

def main():
    arr = list(map(int, input("Enter an array: ").split()))
    print(lonely_integer(arr))

if __name__ == '__main__':
    main()
```

31_Find_Lonely_Integer2.py

```
# o(N) // but space is increased

def lonely_integer(arr):
    s = set()
    for i in arr:
        if i not in s:
            s.add(i)
        else:
            s.remove(i)
    return s.pop()

def main():
    arr = list(map(int, input("Enter an array: ").split()))
    print(lonely_integer(arr))

if __name__ == '__main__':
    main()
```

32_Find_Lonely_Integer3.py

```
# o(N) // but space reduced

def lonely_integer(arr):
    result = 0

    for i in arr:
        result = result ^ i

    return result

def main():
    arr = list(map(int, input("Enter an array: ").split()))
    print(lonely_integer(arr))

if __name__ == '__main__':
    main()
```

33_Is_Consecutive_Set.py

```
def is_consecutive(n):
    return (n & (n<<1) != 0)

def main():
    num = int(input("Enter a Number \n"))
    print(is_consecutive(num))

main()
```

34_Longest_Consecutive.py

```
#time o (log N)

def max_consecutive(n):
    count = 0
```

```
    while(n>0):
        n = n & (n<<1)
        count += 1

    return count

def main():
    num = int(input("Enter a Number \n"))
    print(max_consecutive(num))

main()
```

35_Swap_Odd_Even.py

```
# o(1)

def swap_ood_even_places(n):
    return ( (n & 0Xaaaaaaaa ) >> 1| ( n & 0X55555555 ) << 1)

def main():
    n = int(input("Enter a Number \n"))
    print(swap_ood_even_places(n))

main()
```

36_Trailing_Zero_Count.py

```
import math

def tailing_zero(n):
    # a = n^(n&(n-1))
    # return  a // 8 (we want 2^3 power + 1 should return)
    return  math.log(n^(n&(n-1)),2)

def main():
    n = int(input("Enter a Number \n"))
    print(tailing_zero(n))

main()
```

37_Reversing_32_Bit.py

```
#time o(log N)

def reverse_binary(n):
    f = 31
    l = 0
    rev = 0

    while (f>l):
        if ((n & (1<<f)) != 0):
            rev = rev | (1<<l)
        if ((n & (1<<l)) != 0):
            rev = rev | (1<<f)
        f = f - 1
        l = l + 1

    return rev
```

```
def main():
    n = int(input("Enter a Number \n"))
    print(reverse_binary(n))

main()
```

38_nth_number_palindrome.py

```
# write a program to find the nth number , whose binary representation is a palindrome

import math

def reverse_binary(n,length):
    f = length - 1
    l = 0
    rev = 0

    while (f>l):
        if ((n & (1<<f)) != 0):
            rev = rev | (1<<l)
        if ((n & (1<<l)) != 0):
            rev = rev | (1<<f)
        f = f - 1
        l = l + 1

    return rev

def nth_palin_binary(n):
    length = 0
    count = 0

    while (count<n):
        length += 1
        count += int(math.pow(2, (length-1)//2))

    count -= int(math.pow(2, (length-1)//2))
    elem = n - count - 1
    ans = ( (1<< (length-1)) | (elem<<(length//2)))
    ans = ans | reverse_binary(ans,length)
    return ans

n = int(input("Enter a Number \n"))
print(bin(nth_palin_binary(n)))
```

39_Factorial_Number.py

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

def main():
    n = int(input("Enter a Number \n"))
    print(fact(n))

main()
```

40_Fibonacci_Series.py

```
def fibonacci_series(n):
    if n == 1 or n == 2:
        return 1

    return fibonacci_series(n-1) + fibonacci_series(n-2)

def main():
    n = int(input("Enter a Number \n"))
    print(fibonacci_series(n))

main()
```

41_First_N_Natural_Number.py

```
def n_natural_number(n):
    if n == 0:
        return
    print(n)
    n_natural_number(n-1)
    #print(n)

n = int(input("Enter a Number \n"))
n_natural_number(n)
```

42_Count_Digits.py

```
def count_digits(n):
    if n == 0:
        return n

    return count_digits(n//10) + 1

n = int(input("Enter a Number \n"))
print(count_digits(n))
```

43_Sum_Of_Digits.py

```
def count_digits(n):
    if n == 0:
        return n

    return count_digits(n//10) + n%10

n = int(input("Enter a Number \n"))
print(count_digits(n))
```

44_Reverse_String_Recurstion.py

```
def reverse_string(s,r,i):
    if i < 0:
        return r

    return reverse_string(s, r + s[i] , i-1)

s = input("Enter a String \n")
```

```
print(reverse_string(s,"",len(s)-1))
```

45_Palindrome_Recursion.py

```
def is_palindrome(s,i,j):
    if (s[i] != s[j]):
        return False
    if (j<=i):
        return True

    return is_palindrome(s, i+1, j-1)

s = input("Enter a String \n")
print(is_palindrome(s,0,len(s)-1))
```

46_Sum_Of_Array.py

```
def sum_of_array(arr,i):
    if (len(arr) == i):
        return 0

    return sum_of_array(arr,i+1) + arr[i]

s = list(map(int , input("Enter an array \n").split()))
print(sum_of_array(s,0))
```

48_Balanced_Parenthesis.py

```
def balanced_parenthesis(arr,n,i,o,c):
    if i == len(arr):
       print("".join(arr))

    if (o<n):
        arr[i] = '('
        balanced_parenthesis(arr,n,i+1,o+1,c)

    if (c<o):
        arr[i] = ')'
        balanced_parenthesis(arr,n,i+1,o,c+1)


n = int(input("Enter A Number \n"))

list = [""] * (n*2)

balanced_parenthesis(list,n,0,0,0)
```

49_Letter_Compination_Phone_Number.py

```
keypad = ["","","abc","def","ghi","jkl","mno","pqrs","tuv","wxyz"]

def possible_words(s, ans):

    if len(s) == 0:
        print(ans)
        return

    key = keypad[ int(s[0]) ]
```

```
    for i in key:
        possible_words(s[1:] , ans+i)

def main():
    s = input("Enter a value \n")
    possible_words(s,'')

main()
```

50_Possible_Combinations.py

```
lst = []

def possible_combinations(s, ans):
    if len(s) == 0:
        #print(ans)
        lst.append(ans)
        return

    possible_combinations(s[1:] , ans+s[0])
    possible_combinations(s[1:], ans)

def main():
    s = input()
    possible_combinations(s,'')
    print(lst)
main()
```

51_Permutations.py

```
def permutations(ar , fi):

    if (fi == len(ar)-1):
        print("".join(ar))
        return

    for i in range(fi , len(ar)):
        ar[fi] , ar[i]= ar[i] , ar[fi]
        permutations(ar , fi+1)
        ar[fi] , ar[i]= ar[i] , ar[fi]

def main():
    s = input()
    permutations (list(s) , 0)

main()
```

52_Rope_Cutting_Problem.py

```
def max_pieces(n, a, b, c):

    if (n == 0):
        return 0
    elif (n < 0) :
        return -1

    # temp1 = max_pieces(n-a , a, b, c)
    # temp2 = max_pieces(n-b , a, b, c)
```

```
    # temp3 = max_pieces(n-c , a, b, c)

    # max(temp1 , temp2 , temp3)

    pieces = max(max_pieces(n-a , a, b, c) , max_pieces(n-b , a, b, c) , max_pieces(n-c , a

    if pieces == -1:
        return -1

    return pieces + 1


def main():
    print(max_pieces(15,1,2,2))

main()


# def max_pieces(n, a, b, c):
#     if n == 0:
#         return 0, []  # base case: 0 pieces, empty path
#     elif n < 0:
#         return -1, None  # invalid path

#     # Recursive calls
#     res_a, path_a = max_pieces(n - a, a, b, c)
#     res_b, path_b = max_pieces(n - b, a, b, c)
#     res_c, path_c = max_pieces(n - c, a, b, c)

#     # Find the max among valid results
#     max_val = max(res_a, res_b, res_c)

#     if max_val == -1:
#         return -1, None  # no valid cut

#     # Pick the corresponding path
#     if max_val == res_a:
#         return res_a + 1, path_a + [a]
#     elif max_val == res_b:
#         return res_b + 1, path_b + [b]
#     else:
#         return res_c + 1, path_c + [c]

# def main():
#     count, cuts = max_pieces(15, 5, 8, 7)
#     print("Max pieces:", count)
#     print("Cuts used:", cuts)

# main()
```

53_Sub_Set.py

```
def count_subsets(arr, sum, i):

    if sum == 0:
        return 1
    if sum < 0 :
        return 0
    if i == len(arr):
        return 0
```

```
    return count_subsets(arr , sum - arr[i] , i+1) + count_subsets(arr, sum , i+1)

def main():
    ar = [10,15,20,5]
    print(count_subsets(ar,25,0))

main()
```

54_Lucky_Number.py

```
def is_lucky_number(n,counter):

    if n < counter:
       return True

    if n % counter == 0:
        return False

    return is_lucky_number( n-(n//counter) , counter+1)

def main():
    print(is_lucky_number(9,2))

main()
```

55_Tower_Of_Honai.py

```
def tower_of_honai(n , src , aux , dest):
    if n == 1:
       print(src,'-->',dest)
       return
    tower_of_honai(n-1,src,dest,aux)
    tower_of_honai(1,src,aux,dest)
    tower_of_honai(n-1,aux,src,dest)

def main():
   tower_of_honai(4,'A','B','C')

main()
```

56_Power_Of.py

```
def power_of(x,y):

    if y == 0:
       return 1

    if y % 2 == 0:
       res = power_of(x,y//2)
       return res * res
    else:
       return power_of(x,y-1) * x

def main():
   print(power_of(5,2))

main()
```

57_Linear_Search.py

```
def liner_search(a,key):
    for i in range(0,len(a)):
        if key == a[i] :
           return i

    return -1

def main():
    a = [10,20,50,77,90]
    key = 99
    print(liner_search(a,key))

main()
```

58_Binary_Search.py

```
def binary_search(arr,key):
    low , high , mid = 0 , len(arr)-1 , 0

    while(low<=high):
        #mid = (low+high)//2
        mid = low + (high - low) // 2
        if arr[mid] == key:
           return mid
        elif arr[mid] > key:
            high = mid - 1
            #low = low
        else:
            low = mid + 1
            #high = high
    return -1

def main():
    arr = [14,5,67,89,2,3,0]
    arr.sort()
    print("Sorted array:", arr)
    print(binary_search(arr,89))
main()
```

59_span_of_List.py

```
def span_of_list(a):
    max = a[0]
    min = a[0]

    for i in range(0,len(a)):
        if a[i] > max:
           max = a[i]
        if a[i] < min:
            min = a[i]
    return max - min

def main():
    a = [10,20,40,99,6]
    print(span_of_list(a))

main()
```

60_Second_Largest_Element.py

```
def second_largest(arr):
    max1 , max2 = 0 , 0

    if arr[0] > arr[1]:
        max1 , max2 = arr[0], arr[1]
    else:
        max1 , max2 = arr[1], arr[0]

    for i in range(2,len(arr)):
        if max1 < arr[i]:
           max2 , max1 = max1 , arr[i]
        elif max2 < arr[i]:
            max2 = arr[i]

    return max2

def main():
    arr = [20,42,6,25,30,88]
    print(second_largest(arr))

main()
```

62_Second_Smallest_Element.py

```
def second_smallest(arr):
    max1 , max2 = 0 , 0

    if arr[0] < arr[1]:
        max1 , max2 = arr[0], arr[1]
    else:
        max1 , max2 = arr[1], arr[0]

    for i in range(2,len(arr)):
        if max1 > arr[i]:
           max2 , max1 = max1 , arr[i]
        elif max2 > arr[i]:
            max2 = arr[i]

    return max2

def main():
    arr = [20,42,6,25,30,88]
    print(second_smallest(arr))

main()
```

63_Ceil_And_Floor.py

```
def ceil(arr,key):
    low , high , mid = 0 , len(arr)-1 , 0

    while(low<=high):
        #mid = (low+high)//2
        mid = low + (high - low) // 2
        if arr[mid] == key:
           return arr[mid]
        elif arr[mid] > key:
```

```
            high = mid - 1
            #low = low
        else:
            low = mid + 1
            #high = high
    if low < len(arr):
        return arr[low]
    else:
        return -1

def floor(arr,key):
    low , high , mid = 0 , len(arr)-1 , 0

    while(low<=high):
        #mid = (low+high)//2
        mid = low + (high - low) // 2
        if arr[mid] == key:
           return arr[mid]
        elif arr[mid] > key:
            high = mid - 1
            #low = low
        else:
            low = mid + 1
            #high = high
    if high >= 0:
        return arr[high]
    else:
        return -1

def main():
    arr = [19,23,56,61,72,88,92]
    print(ceil(arr,68))
    print(floor(arr,70))

main()
```

64_Bitonic_Array.py

```
def ascending_binary_search(arr,key):
    low , high , mid = 0 , len(arr)-1 , 0

    while(low<=high):
        #mid = (low+high)//2
        mid = low + (high - low) // 2
        if arr[mid] == key:
           return mid
        elif arr[mid] > key:
            high = mid - 1
            #low = low
        else:
            low = mid + 1
            #high = high
    return -1

def decending_binary_search(arr,key):
    low , high , mid = 0 , len(arr)-1 , 0

    while(low<=high):
        #mid = (low+high)//2
        mid = low + (high - low) // 2
        if arr[mid] == key:
```

```
            return mid
        elif arr[mid] > key:
            low = mid + 1
            #low = low
        else:
            high = mid - 1
            #high = high
    return -1

def bitonic_element(a):
    l,r,m = 0, len(a)-1 , 0

    while(l<=r):
        m = l + (r-l)//2
        if a[m]>a[m+1] and a[m]>a[m-1]:
            return m
        elif a[m]>a[m-1] and a[m]<a[m+1]:
            l = m
        else:
            r=m
    return -1

def main():
    a = [5,6,7,8,9,10,3,2,1]
    key = 1
    bitonic_index = bitonic_element(a)

    if a[bitonic_index] == key:
        print(bitonic_index)
    else:
        index = ascending_binary_search(a[:bitonic_index+1], key)
        if index != -1:
            print(index)
        else:
            index = decending_binary_search(a[bitonic_index+1:], key)
            if index != -1:
                print(bitonic_index + 1 + index)
            else:
                print(-1)

main()
```

65_Count_Smaller_Or_Equal_Element.py

```
def count_smaller_equal(arr, key):
    low, high = 0, len(arr) - 1
    result = -1

    while low <= high:
        mid = low + (high - low) // 2
        if arr[mid] <= key:
            result = mid
            low = mid + 1
        else:
            high = mid - 1
    return result + 1

def count_greater_equal(arr, key):
    low, high = 0, len(arr) - 1
    result = -1
```

```
    while low <= high:
        mid = low + (high - low) // 2
        if arr[mid] >= key:
            result = mid
            high = mid - 1
        else:
            low = mid + 1

    if result == -1:
        return 0
    return len(arr) - result


def main():
    arr = [1, 2, 2, 2, 3, 5, 6]
    print(count_smaller_equal(arr,6))
    print(count_greater_equal(arr,6))

main()
```

66_Wood_Cutting_Problem.py

```
def find_wood_count(ht,m):
    wc = 0
    for i in ht:
        if i > m:
            wc = wc + (i-m)

    return wc

def find_maxHeight(ht,b):
    max = 0
    for i in ht:
        if max < i:
            max = i

    l , h, m = 0 , max  , 0

    while(l<=h):
        m = l + (h-l)//2
        wc = find_wood_count(ht,m)

        if wc == b or l== m:
            return m
        elif wc > b:
            l=m
        else:
            h=m

    return -1

def main():
    ht = [20,15,10,17]
    b = 7
    print(find_maxHeight(ht,b))

main()
```

67_Find_Median.py

```
def find_median(ar1,ar2):
    i,j,k = 0,0,0
    m=[]

    while i<len(ar1) and j<len(ar2):
        if ar1[i] < ar2[j]:
            m.append(ar1[i])
            i = i+1
            k = k+1
        else:
            m.append(ar2[j])
            j = j+1
            k = k+1


    while i<len(ar1):
        m.append(ar1[i])
        i = i+1
        k = k+1

    while j<len(ar2):
        m.append(ar2[j])
        j = j+1
        k = k+1

    mid = len(m) // 2
    if len(m) % 2 == 0:
        return (m[mid] + m[mid-1])/2
    else:
        return m[mid]

def main():
    ar1 = [1,3,8,17]
    ar2 = [5,6,7,19,21,25]
    print(find_median(ar1,ar2))

main()
```

68_Find_Median_LOG.py

```
import sys


def find_median(ar1,ar2):
    if len(ar1) > len(ar2):
        return find_median(ar2,ar1)

    l = 0
    h = len(ar1)

    while l<=h:
        m1 = l + (h-l)//2
        m2 = (len(ar1) + len(ar2) + 1)//2 - m1

        l1 = (sys.maxsize * -1) if (m1==0) else (ar1[m1-1])
        r1 = (sys.maxsize) if (m1==len(ar1)) else (ar1[m1])

        l2 = (sys.maxsize * -1) if (m2==0) else (ar2[m2-1])
        r2 = (sys.maxsize) if (m2==len(ar1)) else (ar2[m2])
```

```
        if l1 <= r2 and l2 <= r1:
            if (len(ar1) + len(ar2) ) % 2 == 0:
                return ((max(l1,l2)) + min(r1,r2))/2
            else:
                return (max(l1,l2))

        elif l2 > r1:
            l = m1 + 1
        else:
            h = m1 - 1

def main():
    ar1 = [1,3,8,17]
    ar2 = [5,6,7,19,21,25]
    print(find_median(ar1,ar2))

main()
```

69_Allocate_Books.py

```
def max_page(ar,b):
    if b > len(ar):
        return -1

    l = ar[0]
    h = 0

    # for i in ar:
    #     if l > i:
    #         l = i
    #     h = h + i

    l = max(ar)
    h = sum(ar)

    res = -1
    while l <= h:
        m = (l+h)//2

        if is_possible_sol(ar,b,m) == True:
            res = m
            h = m-1
        else:
            l = m+1

    return res

def is_possible_sol(ar,b,m):
    students = 1
    spc = 0

    for i in ar:
        if i > m:
            return False
        if spc + i <= m:
            spc = spc + i
        else:
            students = students + 1
            if students > b:
                return False
            spc = i
```

```
    return True

def main():
    ar = [2,3,4,1]
    b = 2
    print(max_page(ar,b))

main()
```

70_Painters_Partition.py

```
def is_possible_soln(ar,a,m):
    painters = 1
    pbc = 0

    for i in ar:
        if m < i:
            return False
        if pbc + i <= m:
            pbc = pbc + i
        else:
            painters = painters + 1

            if painters > a:
                return False
            pbc = i

    return True

def max_time(ar,a,b):
    l = 0
    h = 0

    res = -1

    # for i in ar:
    #     h = h + i

    h = sum(ar)

    while l <= h:
        m = l + (h-l)//2

        if is_possible_soln(ar,a,m) == True:
            res = m
            h = m - 1
        else:
            l = m + 1

    return res * b

def main():
    ar = [10,20,30,40]
    a = 2
    b = 2
    print(max_time(ar,a,b))

main()
```

71_Minimum_Days_To_Make_Bouquets.py

```
def is_possible_soln(ar,boq,flowers,m):
    adj , bc = 0 , 0

    for i in ar:
        if i <= m:
           adj = adj + 1
           if adj == flowers:
               bc = bc + 1
               if bc == boq:
                   return True
               adj = 0
        else:
            adj = 0
    return False

def min_day_to_make_bouquets(ar,boq,flowers):
    if boq * flowers > len(ar):
        return -1

    l = ar[0]
    h = ar[0]
    # for i in ar:
    #     if i > h:
    #         h = i
    #     if i < l:
    #         l = i
    l = min(ar)
    h = max(ar)
    res = -1

    while l <= h:
        m = l + (h-l)//2

        if is_possible_soln(ar,boq,flowers,m) == True:
            res = m
            h = m - 1
        else:
            l = m + 1

    return res

def main():
    ar = [2,5,2,9,3,10,4,6,5,6]
    boq = 4
    flowers = 2
    print(min_day_to_make_bouquets(ar,boq,flowers))

main()
```

72_Is_Array_Sorted.py

```
def is_sorted(ar):

    for i in range(1, len(ar)):
        if ar[i] < ar[i-1]:
           return False

    return True
```

```
def main():
    ar = [2,4,6,8,10,12,14]
    print(is_sorted(ar))

main()
```

73_Sqaure_Root_Number_Floor.py

```
def sqrt(n):
    if n==0 or n==1:
        return n
    l = 2
    h = n//2

    res = 0

    while l <= h:
        m = l + (h-l)//2

        if m * m == n:
            return m
        elif m * m < n:
            res = m
            l = m + 1
        else:
            h = m - 1
            #res = m // ceil
    return res

def main():
    n = 24
    print(sqrt(n))

main()
```

sample.py

```
import os
from reportlab.lib.pagesizes import A4
from reportlab.pdfgen import canvas

# Folder path
folder_path = r"E:\Python_Code_Practice"
output_path = os.path.join(folder_path, "All_Python_Files.pdf")

# Canvas setup
c = canvas.Canvas(output_path, pagesize=A4)
width, height = A4
margin = 50
line_height = 12

# Start from top of the first page
y = height - margin

# Helper: sorting key based on numeric prefix
def extract_sort_key(filename):
    parts = filename.split('_')
    try:
        return (int(parts[0]), filename)
    except ValueError:
```

```
        return (float('inf'), filename)

# Get and sort Python files
py_files = sorted(
    [f for f in os.listdir(folder_path) if f.endswith(".py")],
    key=extract_sort_key
)

# Write lines with paging
def write_lines(lines, font_name="Courier", font_size=10):
    global y
    c.setFont(font_name, font_size)
    for line in lines:
        if y < margin:
            c.showPage()
            y = height - margin
            c.setFont(font_name, font_size)
        c.drawString(margin, y, line)
        y -= line_height

# Write each file
for filename in py_files:
    file_path = os.path.join(folder_path, filename)
    with open(file_path, "r", encoding="utf-8") as f:
        code_lines = f.readlines()

    # Heading (file name)
    write_lines([filename], font_name="Courier-Bold", font_size=12)
    y -= line_height  # Add spacing below heading

    # Code
    clean_lines = [line.rstrip() for line in code_lines]
    write_lines(clean_lines)

    y -= line_height * 2  # Space before next file

# Save PDF
c.save()
print(f"PDF created at: {output_path}")
```