```
In [5]: def pali_num(n):
            return n == int(str(n)[::-1])

        pali_num(110)
```

```
Out[5]: False
```

```
In [10]: def arm_num(n):
             p = len(str(n))
             c = 0
             for i in str(n):
                 c += int(i) ** p
             return c == n

         arm_num(1634)
```

```
Out[10]: True
```

```
In [21]: def prime(n):
             if n < 1:
                 return False
             elif n == 1:
                 return True
             else:
                 for i in range(2,n):
                     if n % i == 0:
                         return False
                     return True

         def p_range(n):
             for i in range(n+1):
                 if prime(i):
                     print(i,end = ' ')
         p_range(19)
```

```
1 3 5 7 9 11 13 15 17 19
```

```
In [22]: def i_factorial(n):
             c = 1
             for i in range(1,n+1):
                 c *= i
             return c
         i_factorial(4)
```

```
Out[22]: 24
```

```
In [6]: def r_factorial(n):
            if n <= 1:
                return n
            return n * r_factorial(n-1)

        r_factorial(4)
```

```
Out[6]:  24
```

```
In [17]: def r_fib(n):
             a,b = 0,1
             for _ in range(n):
                 a,b = b, a+b
             return a

         r_fib(10)
```

```
Out[17]:  55
```

```
In [15]: def r_fib(n):
             if n <= 1:
                 return n
             return r_fib(n-2) + r_fib(n-1)

         r_fib(10)
```

```
Out[15]:  55
```

```
In [18]: def sum_of_d(n):
             c = 0
             for i in str(n):
                 c += int(i)
             return c

         sum_of_d(123)
```

```
Out[18]:  6
```

```
In [19]: def reverse(n):
             return int(str(n)[::-1])
         reverse(321)
```

```
Out[19]:  123
```

```
In [25]: def gcd(x,y):
             for i in range( min(x,y), 1, -1):
                 if x % i == 0 and y % i ==0:
                     return i
         gcd(2,4)
```

```
Out[25]:  2
```

```
In [22]: def lcm(x,y):
             for i in range(max(x,y), x*y + 1):
                 if i%x==0 and i%y==0:
                     return i

         lcm(2,4)
```

```
Out[22]:  4
```

```
In [26]: def count_dig(n):
             c = 0
             for i in str(n):
                 c += 1
             return c
         count_dig(1234)

Out[26]: 4

In [29]: def perfect(n):
             c = 0
             for i in range(1,n):
                 if n % i == 0:
                     c += i
             return c == n

         perfect(28)

Out[29]: True

In [30]: def rev(s):
             return s[::-1]
         rev('hello')

Out[30]: 'olleh'

In [32]: def pali(s):
             return s == s[::-1]
         pali('zyx')

Out[32]: False

In [35]: def count_v(s):
             c = 0
             for i in s:
                 if i in 'aeiou':
                     c += 1
             return c
         count_v('arwse')

Out[35]: 2

In [37]: def remove_duplicate(s):
             return ''.join(dict.fromkeys(s))
         remove_duplicate('aasssdddwwwqqq')

Out[37]: 'asdwq'

In [41]: def non_repchar(s):
             c = {}
             for i in s:
                 c[i] = c.get(i,0) + 1
             for i in c:
                 if c[i] == 1:
```

```
            return i
non_repchar('aadddeerrttffglllleeoo')
```

Out[41]:
```
'g'
```

In [42]:
```
def anagram(x,y):
    return sorted(x) == sorted(y)
anagram('lisent','silent')
```

Out[42]:
```
True
```

In [45]:
```
def freq_char(x):
    s ={}
    for i in x:
        s[i] = s.get(i,0) + 1
    return s
freq_char('wwfww00eessqqdciiqqoo')
```

Out[45]:
```
{'w': 4,
 'f': 1,
 '0': 2,
 'e': 2,
 's': 2,
 'q': 4,
 'd': 1,
 'c': 1,
 'i': 2,
 'o': 2}
```

In [46]:
```
def replace_white(s):
    return s.replace(' ','%10')
replace_white('hf uhu ijw de')
```

Out[46]:
```
'hf%10uhu%10ijw%10de'
```

In [47]:
```
def long_word(s):
    return max(s.split(), key = len)
long_word('Hi this is my beautiful room')
```

Out[47]:
```
'beautiful'
```

In [50]:
```
def string_compression(s):
    c = {}
    res= ''
    for i in s:
        c[i] = c.get(i,0) + 1
    for i in c:
        res += f'{i}{c[i]}'
    return res
string_compression('rrfwfeweeewwwasssssffff')
```

Out[50]:
```
'r2f6w5e4a1s5'
```

In [51]:
```
def lar_ele(arr):
    l = float('-inf')
    for i in arr:
        if i > l:
            l = i
    return l
lar_ele([2,3,1,5,3,9])
```

```
Out[51]: 9
```

```
In [52]: def sml_ele(arr):
    s = float('inf')
    for i in arr:
        if i < s:
            s = i
    return s
sml_ele([0,2,1,3,4,1])
```

```
Out[52]: 0
```

```
In [56]: def sec_lar(arr):
    l=sl=float('-inf')
    for i in arr:
        if i > l:
            sl = l
            l = i
        elif l > i > sl:
            sl = i
    return sl
sec_lar([2,4,1,2,3,4,6])
```

```
Out[56]: 4
```

```
In [58]: def rev_arr(arr):
    l,r = 0,len(arr) - 1
    while l < r:
        arr[l], arr[r] = arr[r] , arr[l]
        l += 1
        r -= 1
    return arr
rev_arr([3,2,1,4,12,1])
```

```
Out[58]: [1, 12, 4, 1, 2, 3]
```

```
In [59]: def sort(arr):
    n = len(arr) - 1
    for i in range(n):
        for j in range(n - i - 1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

sort([2,1,4,3,1,5])
```

```
Out[59]: [1, 1, 2, 3, 4, 5]
```

In [61]:
```
def rm_dup(arr):
    res = []
    for i in arr:
        if i not in res:
            res.append(i)
    return res
rm_dup([1,1,2,2,3,3,4,4,5,5])
```

Out[61]:
```
[1, 2, 3, 4, 5]
```

In [62]:
```
def fre_ele(arr):
    c ={}
    for i in arr:
        c[i] = c.get(i,0)+1
    return c
fre_ele([3,2,1,1,2,2,1,4,3,2,5])
```

Out[62]:
```
{3: 2, 2: 4, 1: 3, 4: 1, 5: 1}
```

In [64]:
```
def rot(k,arr):
    k = k % len(arr)
    return arr[k:] + arr[:k]
rot(3,[1,2,3,4,5,6,7,8,9])
```

Out[64]:
```
[4, 5, 6, 7, 8, 9, 1, 2, 3]
```

In [65]:
```
def missing(arr):
    n = max(arr)
    sn = n * (n+1) // 2
    return sn - sum(arr)
missing([1,3,4,5,6,7])
```

Out[65]:
```
2
```

In [67]:
```
def subarray(arr,k):
    c = 0
    it = {-1:0}
    for i,v in enumerate(arr):
        c += v
        while c-k in it:
            start = it[c-k] + 1
            return arr[start:i+1]
        it[c] = i
subarray([1,2,3,4,5,6,7],9)
```

Out[67]:
```
[2, 3, 4]
```

In [87]:
```
def ratri(n):
    for i in range(1,n+1):
        print('*'*i)
ratri(4)
```

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

In [91]:
```python
def ratri_n(n):
    for i in range(1,n+1):
        print()
        for j in range(1,i+1):
            print(j,end ='')
ratri_n(4)
```

```
1
12
123
1234
```

In [95]:
```python
def h_rati(n):
    for i in range(1,n+1):
        if i == 1:
            print('*')
        elif i == n:
            print('*'*n)
        else:
            print('*'+' '*(i-2)+'*')
h_rati(6)
```

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

In [102…
```python
def ih_rati(n):
    for i in range(n,0,-1):
        if i == 1:
            print('*')
        elif i == n:
            print('*'*n)
        else:
            print('*'+' '*(i-2)+'*')
ih_rati(6)
```

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

In [106…
```python
def tri(n):
    for i in range(1,n+1):
        print(' '*(n-i)+'*'*(2*i-1))
tri(4)
```

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

In [108…

```
def h_tri(n):
    for i in range(1,n+1):
        if i == 1:
            print(' '*(n-1)+'*')
        elif i == n:
            print('*'*(2*n-1))
        else:
            print(' '*(n-i)+'*'+' '*(2*i-3)+'*')
h_tri(5)
```

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

In [110…

```
def ih_tri(n):
    for i in range(n,0,-1):
        if i == 1:
            print(' '*(n-1)+'*')
        elif i == n:
            print('*'*(2*n-1))
        else:
            print(' '*(n-i)+'*'+' '*(2*i-3)+'*')
ih_tri(5)
```

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

In [112…

```
def ih_dai(n):
    for i in range(1,n+1):
        if i != n:
            if i == 1:
                print(' '*(n-1)+'*')
            else:
                print(' '*(n-i)+'*'+' '*(2*i-3)+'*')
        else:
            for i in range(n,0,-1):
                if i == 1:
                    print(' '*(n-1)+'*')
                else:
                    print(' '*(n-i)+'*'+' '*(2*i-3)+'*')

ih_dai(5)
```

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

In [113…

```
def sqr(n):
    for i in range(n):
        print('*'*n)
sqr(4)
```

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

In [117…

```
def hsqr(n):
    for i in range(1,n+1):
        if i == 1 or i == n:
            print('*'*n)
        else:
            print('*'+' '*(n-2)+'*')
hsqr(4)
```

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

In [120…

```
def linear_search(arr,k):
    for i in range(len(arr) -1):
        if arr[i] == k:
            return i
linear_search([1,4,2,5,7,9],5)
```

Out[120… 3

In [123…

```
def binary_search(arr,k):
    l = 0
    h = len(arr) - 1
    while l < h:
        mid = (l + h) // 2
        if arr[mid] == k:
            return mid
        elif arr[mid] > k:
            h = mid - 1
        else:
            l = mid + 1
    return -1
binary_search([1,3,4,5,6,8],6)
```

Out[123… 4

In [124…

```
def rec_binary_search(arr,k,l,h):
    mid = (l+h)//2
    if arr[mid] == k:
        return mid
    elif arr[mid] > k:
        return rec_binary_search(arr,k,l,mid-1)
    else:
        return rec_binary_search(arr,k,mid+1,h)
rec_binary_search([1,3,4,5,6,8],6,0,5)
```

Out[124…

```
4
```

In [126…

```
def tim_sort(arr):
    arr.sort()
    return arr
tim_sort([4,2,4,1,2,6,8])
```

Out[126…

```
[1, 2, 2, 4, 4, 6, 8]
```

In [2]:

```
def bubble_sort(arr):
    n = len(arr) - 1
    for i in range(n-1):
        swapped = False
        for j in range(n-i-1):
            if arr[j] > arr[j+1]:
                arr[j],arr[j+1] = arr[j+1],arr[j]
                swapped = True
            if swapped == False:
                break;
    return arr

bubble_sort([2,1,5,9])
```

Out[2]:

```
[1, 2, 5, 9]
```

In [3]:

```
def sequential_sort(arr):
    n = len(arr) - 1
    for i in range(n-1):
        min_ind = i
        for j in range(1,n):
            if arr[j] < arr[min_ind]:
                min_ind = j
        arr[i], arr[min_ind] = arr[min_ind], arr[i]
    return arr

sequential_sort([2,1,5,9])
```

Out[3]:

```
[1, 2, 5, 9]
```

In [4]:

```
def insertion_sort(arr):
    n = len(arr) - 1
    for i in range(1,n):
        k = arr[i]
        j = i-1
        while j>=0 and arr[j] > k:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = k
    return arr
insertion_sort([2,1,5,9])
```

```
Out[4]: [1, 2, 5, 9]
```

```
In [2]: def quick_sort(arr):
            if len(arr) <= 1:
                return arr
            p = arr[-1]
            l = []
            r = []
            for i in arr[:-1]:
                if i < p:
                    l.append(i)
                else:
                    r.append(i)
            return quick_sort(l) + [p] + quick_sort(r)

        quick_sort([2,1,5,9])
```

```
Out[2]: [1, 2, 5, 9]
```

```
In [1]: def merg_sort(arr):
            if len(arr) <=1:
                return arr
            mid = len(arr) // 2
            l = arr[:mid]
            r = arr[mid:]
            l = merg_sort(l)
            r = merg_sort(r)
            return merg(l,r)

        def merg(l,r):
            i=j=0
            res = []
            while i < len(l) and j < len(r):
                if l[i] < r[j]:
                    res.append(l[i])
                    i += 1
                else:
                    res.append(r[j])
                    j += 1
            res.extend(l[i:])
            res.extend(r[j:])
            return res

        merg_sort([2,1,5,9])
```

```
Out[1]: [1, 2, 5, 9]
```