swap two numbers without using third variable

In [1]:
```
a=9
m=8
a,m = m, a
print(a,m)
```

```
8 9
```

In [2]:
```
x=10
y=11
x=x+y #21
y=x-y #21-11=10
x=x-y #21-10=11
print(x, y)
```

```
11 10
```

Reverse a string

In [ ]:
```
#using slicing
```

In [4]:
```
def rev(n):
    return n[::-1]
n=input("enter a string: ")
print(rev(n))
```

```
enter a stringqwerty
ytrewq
```

In [ ]:
```
#using reversed keyword
```

In [5]:
```
def rev(m):
    l=reversed(m)
    return l
m=input("enter a string: ")
print(rev(m))
```

```
enter a stringqwert
<reversed object at 0x000002499D878A60>
```

In [ ]:
```
```

In [ ]:
```
#reverse a string using command line argument
```

In [ ]:
```
import sys
if len(sys.argv) < 2:
    print("Usage: python reverse_string.py <string>")
    sys.exit(1)

input_string = sys.argv[1]
reversed_string = input_string[::-1]
print("Reversed string:", reversed_string)
```

arrange 1s and 0s together in a single array scan

In [29]:
```
print([0,4]*3)
```

```
[0, 4, 0, 4, 0, 4]
```

In [32]:
```
def arrange_zeros_ones(arr):
    zeros_count = arr.count(0)
    print(zeros_count)
    ones_count = arr.count(1)
    print(ones_count)
    st_count = arr.count('e')
    return [0]*zeros_count + [1]*ones_count +['e']*st_count # concate the arra

# Example usage:
input_array = [1, 0, 0, 0, 1, 'e', 'e', 'e', 1, 0, 1]
output_array = arrange_zeros_ones(input_array)
print("Output:", output_array)
```

```
4
4
Output: [0, 0, 0, 0, 1, 1, 1, 1, 'e', 'e', 'e']
```

In [33]:
```
l=[1,0,0,1,0,1,0,0,1,1]
print(sorted(l)[::-1])
```

```
[1, 1, 1, 1, 1, 0, 0, 0, 0, 0]
```

even positions and odd positions in list

In [39]:
```
# even positions and odd positions in list with indices
arr=[0,1,3,6,7,8,9,1,0]
for index, values in enumerate(arr):
    print(index, values)
```

```
0 0
1 1
2 3
3 6
4 7
5 8
6 9
7 1
8 0
```

In [ ]:
```
# difference of even position sum and odd position sum
```

In [45]:
```
arr=[0,1,3,6,7,8,9,1,0]
even_n=arr[0::2]
odd_n=arr[1::2]
y=sum(even_n)-sum(odd_n)
print(f'even digits {even_n} | odd digits {odd_n} | difference of even sum and
```

```
even digits [0, 3, 7, 9, 0] | odd digits [1, 6, 8, 1] | difference of even s
um and odd sum 3
```

In [ ]:
```
# even positions and odd positions in list without slicing
```

In [44]:
```
arr=[0,1,3,6,7,8,9,1,0]
even_pos, odd_pos = [],[]
for i in range(len(arr)):
    if i%2==0:
        even_pos.append(arr[i])
    else:
        odd_pos.append(arr[i])
print(even_pos, odd_pos)
```

```
[0, 3, 7, 9, 0] [1, 6, 8, 1]
```

convert a matrix into lower triangular matrix

In [71]:
```
matrix = [
    [1, 2, 3], #0th row
    [4, 5, 6], #1st row
    [7, 8, 9]  #2nd row
]
```

In [49]:
```
print(matrix[0])
print(len(matrix)) # to get number of rows
print(len(matrix[0])) # to get number of columns
```

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

In [72]:
```
def l_matrix(matrix):
    rows= len(matrix)
    cols= len(matrix[0])
    for i in range(rows): #i to row-1
        for j in range(cols): #j to cols-1
            if i<j:           #checks
                matrix[i][j]=0
    return matrix
m=l_matrix(matrix)
for rows in m:
    print(*rows)
```

```
1 0 0
4 5 0
7 8 9
```

convert a matrix into upper triangular matrix

In [68]:
```
mat = [
    [1, 2], #0th row
    [4, 5], #1st row
    [7, 8],
    [8, 9]#2nd row
]
```

In [69]:
```
def u_mat(mat):
    row=len(mat)
    col=len(mat[0])
    for i in range(row):
        for j in range(col):
            if i>j:
                mat[i][j]=0
    return mat

m=u_mat(mat)
for row in m:
    print(*row)
```

```
1 2
0 5
0 0
0 0
```

factorial of number using arithmetic operations

In [74]:
```
def factorial(n):
    r=1
    for i in range(1, n+1): #range of 1 to 5
        r*=i
    return r
n = int(input())
print(factorial(n))
```

```
5
120
```

In [ ]:
```
#factorial of number using recursion'''
```

In [5]:
```
def factorial(n):
    if n<0:
        return "enter valid number"
    elif n == 0 or n==1:
        return 1
    return n*factorial(n-1)
n = int(input())
print(factorial(n))
```

```
10
3628800
```

In [ ]:
```
#factorial of number using math library
```

In [6]:
```
import math
n=5
k=math.factorial(n)
print(k)
```

```
120
```

number of pairs whose average is also present in the array,

In [15]:
```python
def count_average_pairs(arr):
    s=set(arr)
    #count=0
    pairs=[]
    for i in range(len(arr)):
        for j in range(i+1, len(arr)):
            avg = (arr[i]+arr[j])/2
            if avg in s:
                #count=count+1
                pairs.append((arr[i], arr[j]))
    return pairs
arr = [2, 4, 6, 8, 5]
v=count_average_pairs(arr)
print("Number of valid pairs:", len(v))
print("the average pairs: ", v)
```

```
Number of valid pairs: 4
the average pairs:  [(2, 6), (2, 8), (4, 6), (4, 8)]
```

binary search algorithm

In [ ]:
```python
def binary_search(arr, target):
    low=0
    high=len(arr)-1
    while low<=high:
        mid = (low+high) // 2
        if (arr[mid] == target):
            return mid
        elif (arr[mid] < target):
            low = mid + 1
        else:
            high = mid + 1
    return -1 # target not found


arr = [1, 3, 5, 7, 9, 11]
target = 7
k=binary_search(arr, target)
```

In [ ]:
```python
#prime code
#explain what is stack
#node deletion code in linkedlist via singly linked list and doubly
#middle node in linkedlist via singly linked list and doubly
#find vowel & consonants from string
#what is queue and circular queue
#n-queen problem
#merge sort and bubble sort
# explain armstrong number and
# Explain what is bitwise operator, list all the common bitwise operator and w
# explain acid properties with example
```

Bubble Sort

In [2]:
```
def bubble_sort(a):
    n = len(a)
    for i in range(n):
        swapped = False
        for j in range(0, n-i-1):
            if a[j] > a[j+1]:
                a[j], a[j+1] = a[j+1], a[j]
                swapped = True
        if not swapped:
            break
    return a
a=[6,8,4,6,0,2,9,5,46]
print(bubble_sort(a))
```

```
[0, 2, 4, 5, 6, 6, 8, 9, 46]
```

In [ ]: