1.Finding the maximum and minimum

```
File  Edit  Format  Run  Options  Window  Help
def find(arr, low, high):
    if low == high:
        return (arr[low], arr[low])
    if high == low + 1:
        if arr[low] < arr[high]:
            return (arr[low], arr[high])
        else:
            return (arr[high], arr[low])
    mid = (low + high) // 2

    left_min, left_max = find(arr, low, mid)
    right_min, right_max = find(arr, mid + 1, high)

    final_min = min(left_min, right_min)
    final_max = max(left_max, right_max)

    return (final_min, final_max)
arr = [2,4,1,5,6]
min_val, max_val = find(arr, 0, len(arr) - 1)
print("MIN:",min_val)
print("MAX:",max_val)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:\Users\balas\OneDrive\Documents\Day 8.1.py ============
MIN: 1
MAX: 6
>>>
```

2.Merge sort

```
File  Edit  Format  Run  Options  Window  Help
def mergeSort(array):
    if len(array) > 1:
        r = len(array)//2
        L = array[:r]
        M = array[r:]
        mergeSort(L)
        mergeSort(M)
        i = j = k = 0
        while i < len(L) and j < len(M):
            if L[i] < M[j]:
                array[k] = L[i]
                i += 1
            else:
                array[k] = M[j]
                j += 1
            k += 1
        while i < len(L):
            array[k] = L[i]
            i += 1
            k += 1

        while j < len(M):
            array[k] = M[j]
            j += 1
            k += 1
array = [4,1,5,3,2]
mergeSort(array)
print("Sorted array:",array)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============ RESTART: C:\Users\balas\OneDrive\Documents\Day 8.2.py ============
Sorted array: [1, 2, 3, 4, 5]
>>>
```

3.Quick sort

```
File  Edit  Format  Run  Options  Window  Help
def partition(array, low, high):
    pivot = array[high]
    i = low - 1
    for j in range(low, high):
        if array[j] <= pivot:
            i = i + 1
            (array[i], array[j]) = (array[j], array[i])

    (array[i + 1], array[high]) = (array[high], array[i + 1])
    return i + 1

def quicksort(array, low, high):
    if low < high:
        pi = partition(array, low, high)
        quicksort(array, low, pi - 1)
        quicksort(array, pi + 1, high)


array = [13,5,8,2,1]
N = len(array)
quicksort(array, 0, N - 1)
print('Sorted array:')
for x in array:
    print(x, end=" ")
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
>>>
============ RESTART: C:\Users\balas\OneDrive\Documents\Day 8.3.
Sorted array:
1 2 5 8 13
>>>
```

4.Binary search

```
File  Edit  Format  Run  Options  Window  Help
list1=[1,5,2,8,9]
list2=sorted(list1)
n=8
low = 0
high = len(list2) - 1
mid = 0
flag=0
while low <= high:
        mid = (high + low) // 2
        if list1[mid] < n:
            low = mid + 1
        elif list1[mid] > n:
            high = mid - 1
        else:
            flag=1
            print("found at:",mid)
            break
if flag==0:
    print("not found")
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.193
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informatic
>>>
============= RESTART: C:\Users\balas\OneDrive\Documents\Day 8.4.py ==
found at: 3
>>>
```

5.Strassens matrix multiplication

```
def add_matrices(A, B):
    n = len(A)
    C = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            C[i][j] = A[i][j] + B[i][j]
    return C
def subtract_matrices(A, B):
    n = len(A)
    C = [[0] * n for _ in range(n)]
    for i in range(n):
        for j in range(n):
            C[i][j] = A[i][j] - B[i][j]
    return C
def split_matrix(M):
    n = len(M)
    mid = n // 2
    A11 = [[M[i][j] for j in range(mid)] for i in range(mid)]
    A12 = [[M[i][j] for j in range(mid, n)] for i in range(mid)]
    A21 = [[M[i][j] for j in range(mid)] for i in range(mid, n)]
    A22 = [[M[i][j] for j in range(mid, n)] for i in range(mid, n)]
    return A11, A12, A21, A22
def combine_matrices(C11, C12, C21, C22):
    n = len(C11) * 2
    C = [[0] * n for _ in range(n)]
    mid = n // 2
    for i in range(mid):
        for j in range(mid):
            C[i][j] = C11[i][j]
            C[i][j + mid] = C12[i][j]
            C[mid + i][j] = C21[i][j]
            C[mid + i][j + mid] = C22[i][j]
    return C
def strassen(A, B):
    n = len(A)
    if n == 1:
        return [[A[0][0] * B[0][0]]]
    else:
        A11, A12, A21, A22 = split_matrix(A)
        B11, B12, B21, B22 = split_matrix(B)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 6
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============= RESTART: C:\Users\balas\OneDrive\Documents\Day 8.5.py =====
[19, 22]
[43, 50]
>>>
```

```
            C[i][j + mid] = C12[i][j]
            C[mid + i][j] = C21[i][j]
            C[mid + i][j + mid] = C22[i][j]
    return C
def strassen(A, B):
    n = len(A)
    if n == 1:
        return [[A[0][0] * B[0][0]]]
    else:
        A11, A12, A21, A22 = split_matrix(A)
        B11, B12, B21, B22 = split_matrix(B)
        M1 = strassen(add_matrices(A11, A22), add_matrices(B11, B22))
        M2 = strassen(add_matrices(A21, A22), B11)
        M3 = strassen(A11, subtract_matrices(B12, B22))
        M4 = strassen(A22, subtract_matrices(B21, B11))
        M5 = strassen(add_matrices(A11, A12), B22)
        M6 = strassen(subtract_matrices(A21, A11), add_matrices(B11, B12))
        M7 = strassen(subtract_matrices(A12, A22), add_matrices(B21, B22))
        C11 = add_matrices(subtract_matrices(add_matrices(M1, M4), M5), M7)
        C12 = add_matrices(M3, M5)
        C21 = add_matrices(M2, M4)
        C22 = add_matrices(subtract_matrices(add_matrices(M1, M3), M2), M6)
        return combine_matrices(C11, C12, C21, C22)

A = [
    [1, 2],
    [3, 4]
]
B = [
    [5, 6],
    [7, 8]
]
C = strassen(A, B)
for row in C:
    print(row)
```

6.Karatsuba algorithm for multiplication

```python
def karat(x, y):
    if x < 10 or y < 10:
        return x * y
    n = max(len(str(x)), len(str(y)))
    m = (n + 1) // 2
    high1, low1 = split_at(x, m)
    high2, low2 = split_at(y, m)
    z0 = karat(low1, low2)
    z1 = karat((low1 + high1), (low2 + high2))
    z2 = karat(high1, high2)

    return (z2 * 10**(2*m)) + ((z1 - z2 - z0) * 10**m) + z0

def split_at(number, m):
    high = number // 10**m
    low = number % 10**m
    return high, low

def testcase():
    x = 1234
    y = 5678
    result = karat(x, y)
    print(f"Karatsuba({x}, {y}) = {result}")
testcase()
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
>>>
========== RESTART: C:\Users\balas\OneDrive\Documents\Day 8.6.p
Karatsuba(1234, 5678) = 7006652
>>>
```

7.Closest pair of points using divide and conquer

```python
def eucl(p, q):
    return ((p[0] - q[0])**2 + (p[1] - q[1])**2)**0.5

def brute(points):
    min_dist = float('inf')
    best_pair = (None, None)
    for i in range(len(points)):
        for j in range(i + 1, len(points)):
            dist = eucl(points[i], points[j])
            if dist < min_dist:
                min_dist = dist
                best_pair = (points[i], points[j])
    return best_pair[0], best_pair[1], min_dist

def closesplit(points, delta):
    n = len(points)
    mid_x = points[n // 2][0]
    strip = [p for p in points if abs(p[0] - mid_x) < delta]
    strip.sort(key=lambda p: p[1])

    best_dist = delta
    best_pair = (None, None)

    for i in range(len(strip)):
        for j in range(i + 1, min(i + 7, len(strip))):
            dist = eucl(strip[i], strip[j])
            if dist < best_dist:
                best_dist = dist
                best_pair = (strip[i], strip[j])

    return best_pair[0], best_pair[1], best_dist

def recursive(points):
    n = len(points)
    if n <= 3:
        return brute(points)

    mid = n // 2
    Q = points[:mid]
    R = points[mid:]
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\balas\OneDrive\Documents\Day 8.7.py
The closest pair of points are (2, 3) and (3, 4)
>>>
```

```python
    for i in range(len(strip)):
        for j in range(i + 1, min(i + 7, len(strip))):
            dist = eucl(strip[i], strip[j])
            if dist < best_dist:
                best_dist = dist
                best_pair = (strip[i], strip[j])

    return best_pair[0], best_pair[1], best_dist

def recursive(points):
    n = len(points)
    if n <= 3:
        return brute(points)

    mid = n // 2
    Q = points[:mid]
    R = points[mid:]

    (p1, q1, dist1) = recursive(Q)
    (p2, q2, dist2) = recursive(R)

    delta = min(dist1, dist2)
    (p3, q3, dist3) = closesplit(points, delta)

    if dist3 < delta:
        return p3, q3, dist3
    else:
        if dist1 < dist2:
            return p1, q1, dist1
        else:
            return p2, q2, dist2

def closest_pair(points):
    points.sort(key=lambda p: p[0])
    return recursive(points)
points = [(2, 3), (12, 30), (40, 50), (5, 1), (12, 10), (3, 4)]
p1, p2, dist = closest_pair(points)
print(f"The closest pair of points are {p1} and {p2}")
```

```
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informati
>>>
= RESTART: C:\Users\balas\OneDrive\Documents\Day 8.7.py
The closest pair of points are (2, 3) and (3, 4)
>>>
```

8.Median of medians

```python
def partition(arr, pivot):
    less = []
    equal = []
    greater = []
    for element in arr:
        if element < pivot:
            less.append(element)
        elif element == pivot:
            equal.append(element)
        else:
            greater.append(element)
    return less, equal, greater

def med(arr, k):
    if len(arr) <= 5:
        arr.sort()
        return arr[k-1]
    sublists = [arr[i:i + 5] for i in range(0, len(arr), 5)]
    medians = [sorted(sublist)[len(sublist) // 2] for sublist in sublists]
    medpivot = med(medians, len(medians) // 2)
    less, equal, greater = partition(arr, medpivot)
    if k <= len(less):
        return med(less, k)
    elif k <= len(less) + len(equal):
        return medpivot
    else:
        return med(greater, k - len(less) - len(equal))
arr = [12, 3, 5, 7, 4, 19, 26]
k = 3
result = med(arr, k)
print(result)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" f
>>>
============= RESTART: C:\Users\balas\OneDrive\Docum
5
>>>
```

9.Meet in middle technique

```python
def subsetsum(subset):
    sums = set()
    n = len(subset)
    for i in range(1 << n):
        current_sum = 0
        for j in range(n):
            if i & (1 << j):
                current_sum += subset[j]
        sums.add(current_sum)
    return sums

def mim(arr, target):
    n = len(arr)
    left_part = arr[:n//2]
    right_part = arr[n//2:]

    left_sums = subsetsum(left_part)
    right_sums = subsetsum(right_part)

    for sum in left_sums:
        if (target - sum) in right_sums:
            return True
    return False
arr = [3, 34, 4, 12, 5, 2]
target = 9
result = mim(arr, target)
print(result)
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec
AMD64)] on win32
Type "help", "copyright", "credits" or "
>>>
============= RESTART: C:\Users\balas\Or
True
>>>
```