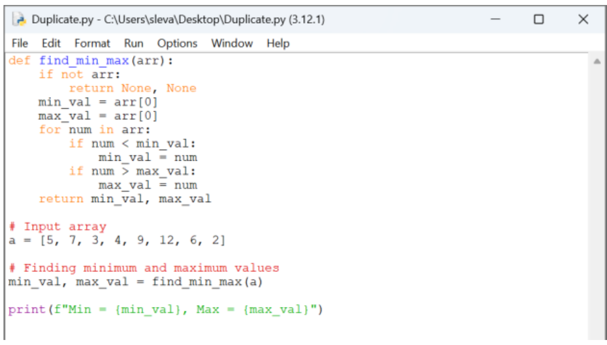
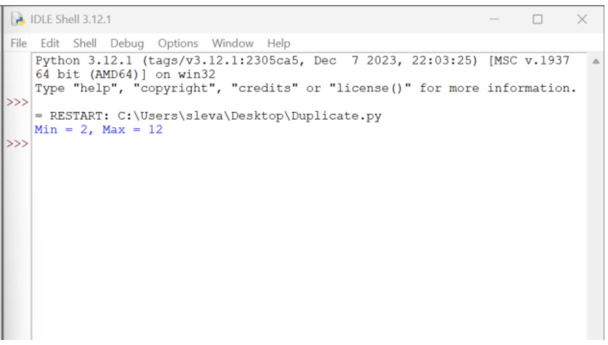
1. Max and Min

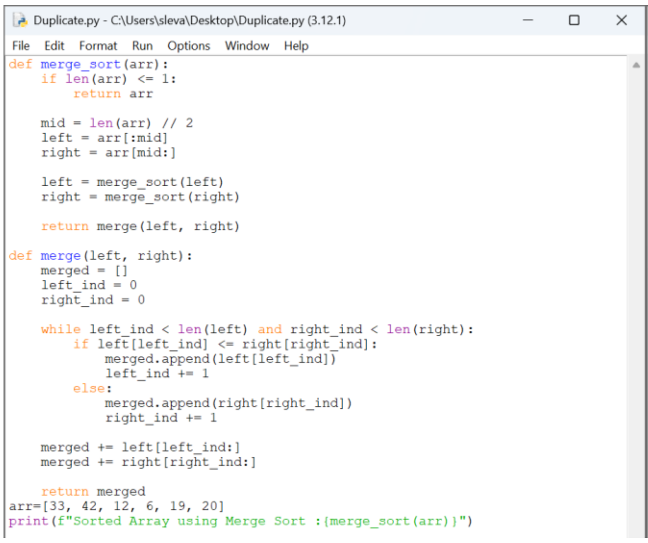

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
def find_min_max(arr):
    if not arr:
        return None, None
    min_val = arr[0]
    max_val = arr[0]
    for num in arr:
        if num < min_val:
            min_val = num
        if num > max_val:
            max_val = num
    return min_val, max_val

# Input array
a = [5, 7, 3, 4, 9, 12, 6, 2]

# Finding minimum and maximum values
min_val, max_val = find_min_max(a)

print(f"Min = {min_val}, Max = {max_val}")
```

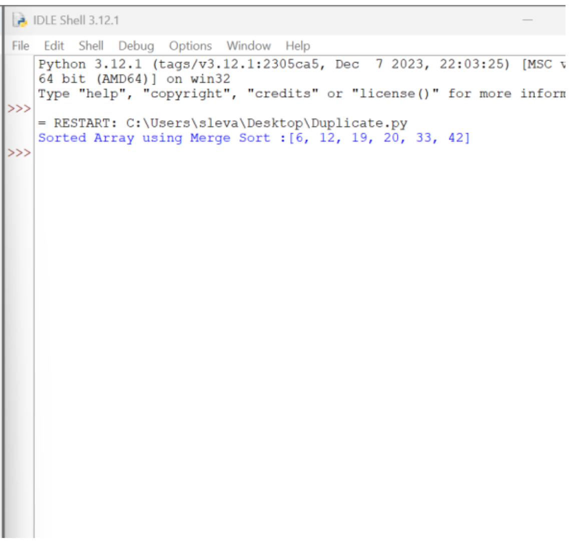


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Min = 2, Max = 12
>>>
```

3. Merge Sort

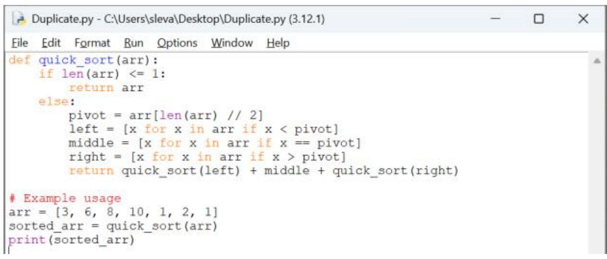

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    mid = len(arr) // 2
    left = arr[:mid]
    right = arr[mid:]

    left = merge_sort(left)
    right = merge_sort(right)

    return merge(left, right)

def merge(left, right):
    merged = []
    left_ind = 0
    right_ind = 0

    while left_ind < len(left) and right_ind < len(right):
        if left[left_ind] <= right[right_ind]:
            merged.append(left[left_ind])
            left_ind += 1
        else:
            merged.append(right[right_ind])
            right_ind += 1

    merged += left[left_ind:]
    merged += right[right_ind:]

    return merged
arr=[33, 42, 12, 6, 19, 20]
print(f"Sorted Array using Merge Sort :{merge_sort(arr)}")
```

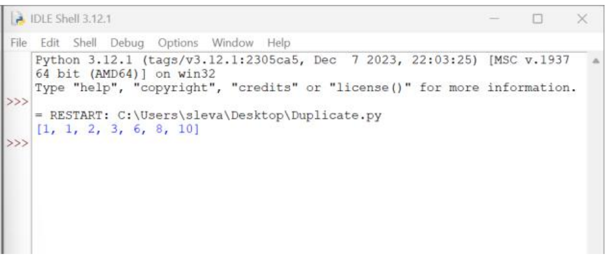


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more inform
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Sorted Array using Merge Sort :[6, 12, 19, 20, 33, 42]
>>>
```

5. Quick Sort

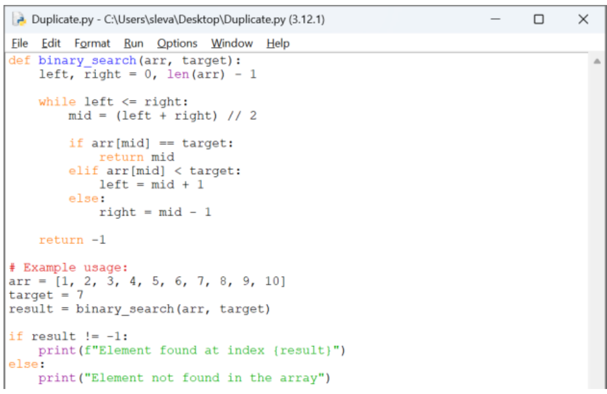

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[len(arr) // 2]
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
        return quick_sort(left) + middle + quick_sort(right)

# Example usage
arr = [3, 6, 8, 10, 1, 2, 1]
sorted_arr = quick_sort(arr)
print(sorted_arr)
```

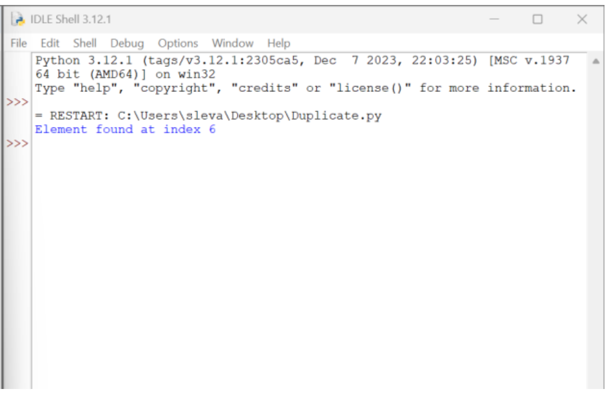


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
[1, 1, 2, 3, 6, 8, 10]
>>>
```

7. Binary Search

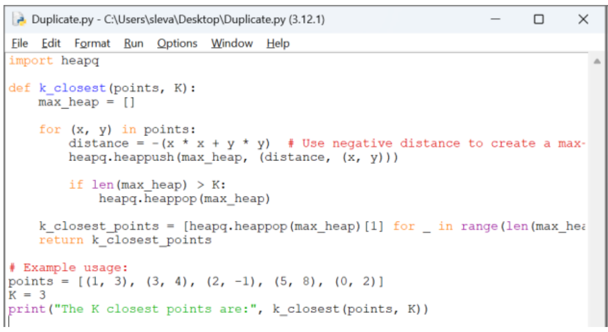

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
        mid = (left + right) // 2

        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    return -1

# Example usage:
arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
target = 7
result = binary_search(arr, target)

if result != -1:
    print(f"Element found at index {result}")
else:
    print("Element not found in the array")
```

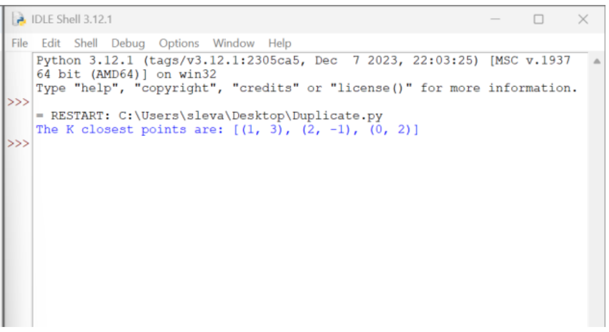


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Element found at index 6
>>>
```

9. Kth Smallest element

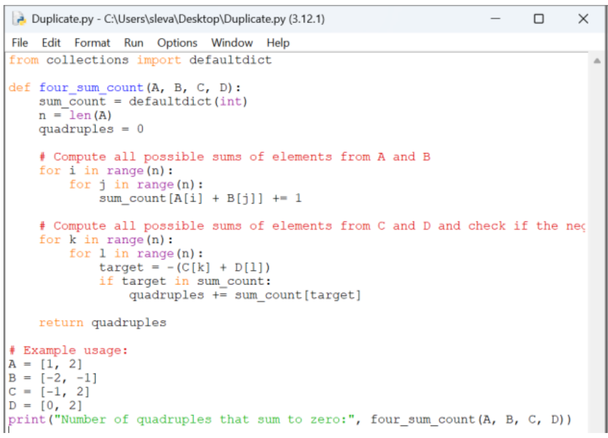

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```
import heapq

def k_closest(points, K):
    max_heap = []

    for (x, y) in points:
        distance = -(x * x + y * y)  # Use negative distance to create a max-
        heapq.heappush(max_heap, (distance, (x, y)))

        if len(max_heap) > K:
            heapq.heappop(max_heap)

    k_closest_points = [heapq.heappop(max_heap)[1] for _ in range(len(max_hea
    return k_closest_points

# Example usage:
points = [(1, 3), (3, 4), (2, -1), (5, 8), (0, 2)]
K = 3
print("The K closest points are:", k_closest(points, K))
```

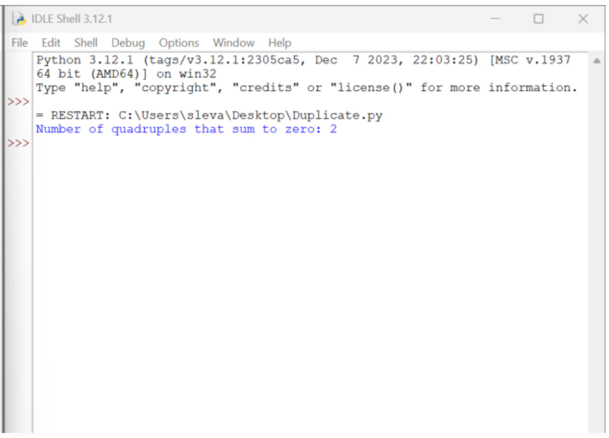


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
The K closest points are: [(1, 3), (2, -1), (0, 2)]
>>>
```

10. Four Sum count

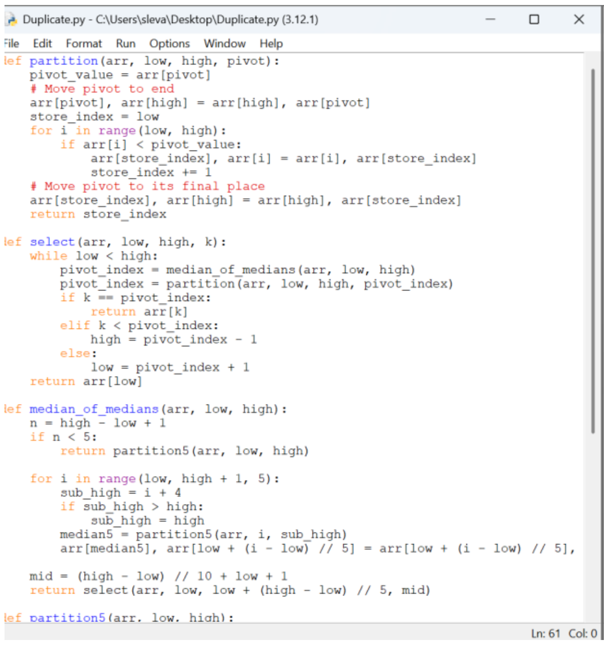

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```
from collections import defaultdict

def four_sum_count(A, B, C, D):
    sum_count = defaultdict(int)
    n = len(A)
    quadruples = 0

    # Compute all possible sums of elements from A and B
    for i in range(n):
        for j in range(n):
            sum_count[A[i] + B[j]] += 1

    # Compute all possible sums of elements from C and D and check if the neg
    for k in range(n):
        for l in range(n):
            target = -(C[k] + D[l])
            if target in sum_count:
                quadruples += sum_count[target]

    return quadruples

# Example usage:
A = [1, 2]
B = [-2, -1]
C = [-1, 2]
D = [0, 2]
print("Number of quadruples that sum to zero:", four_sum_count(A, B, C, D))
```

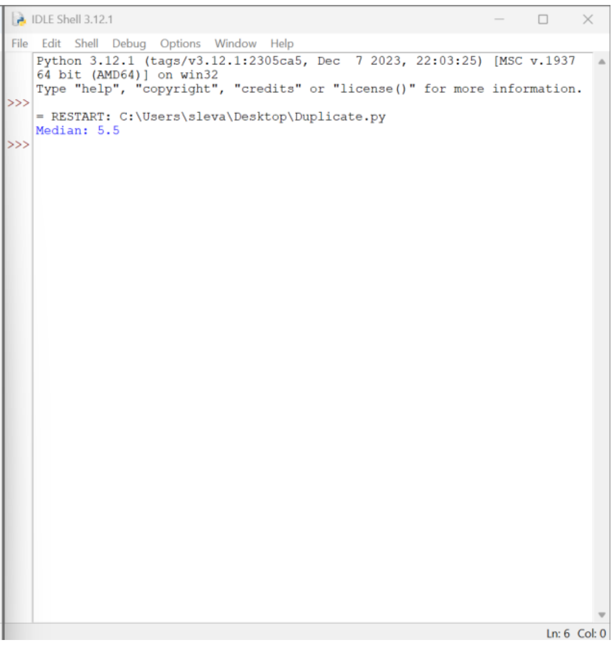


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Number of quadruples that sum to zero: 2
>>>
```

11. Median of Medians

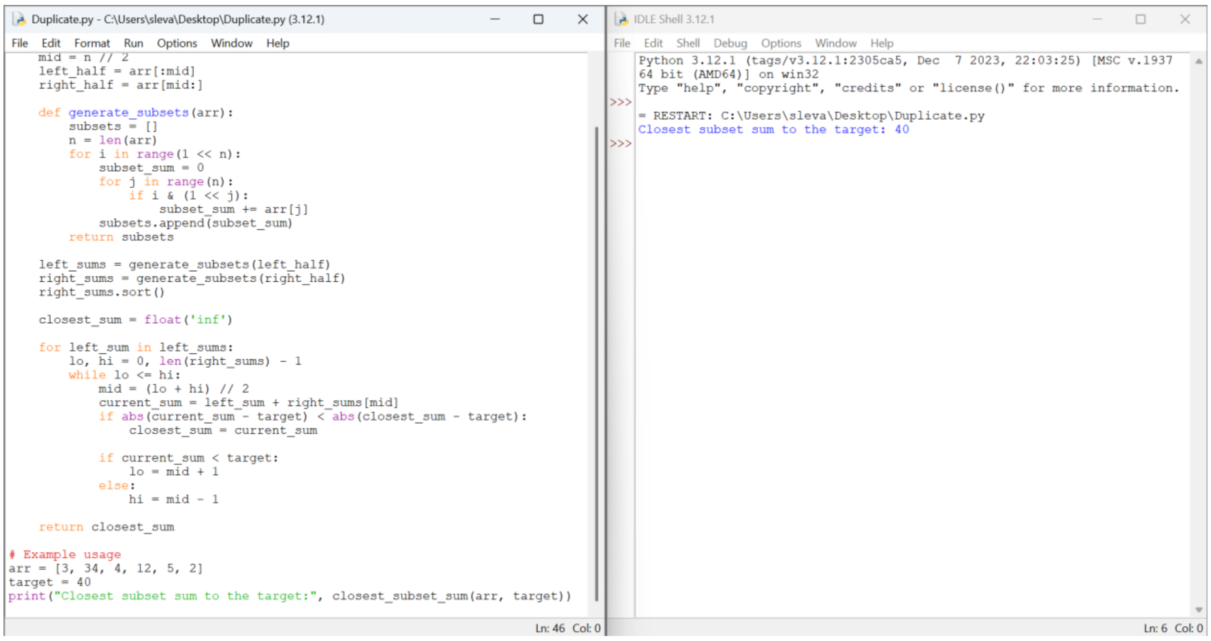

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
Ln: 61 Col: 0

```
lef partition(arr, low, high, pivot):
    pivot_value = arr[pivot]
    # Move pivot to end
    arr[pivot], arr[high] = arr[high], arr[pivot]
    store_index = low
    for i in range(low, high):
        if arr[i] < pivot_value:
            arr[store_index], arr[i] = arr[i], arr[store_index]
            store_index += 1
    # Move pivot to its final place
    arr[store_index], arr[high] = arr[high], arr[store_index]
    return store_index

lef select(arr, low, high, k):
    while low < high:
        pivot_index = median_of_medians(arr, low, high)
        pivot_index = partition(arr, low, high, pivot_index)
        if k == pivot_index:
            return arr[k]
        elif k < pivot_index:
            high = pivot_index - 1
        else:
            low = pivot_index + 1
    return arr[low]

lef median_of_medians(arr, low, high):
    n = high - low + 1
    if n < 5:
        return partition5(arr, low, high)

    for i in range(low, high + 1, 5):
        sub_high = i + 4
        if sub_high > high:
            sub_high = high
        median5 = partition5(arr, i, sub_high)
        arr[median5], arr[low + (i - low) // 5] = arr[low + (i - low) // 5],

    mid = (high - low) // 10 + low + 1
    return select(arr, low, low + (high - low) // 5, mid)

lef partition5(arr, low, high):
```



```
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\sleva\Desktop\Duplicate.py
Median: 5.5
>>>
```

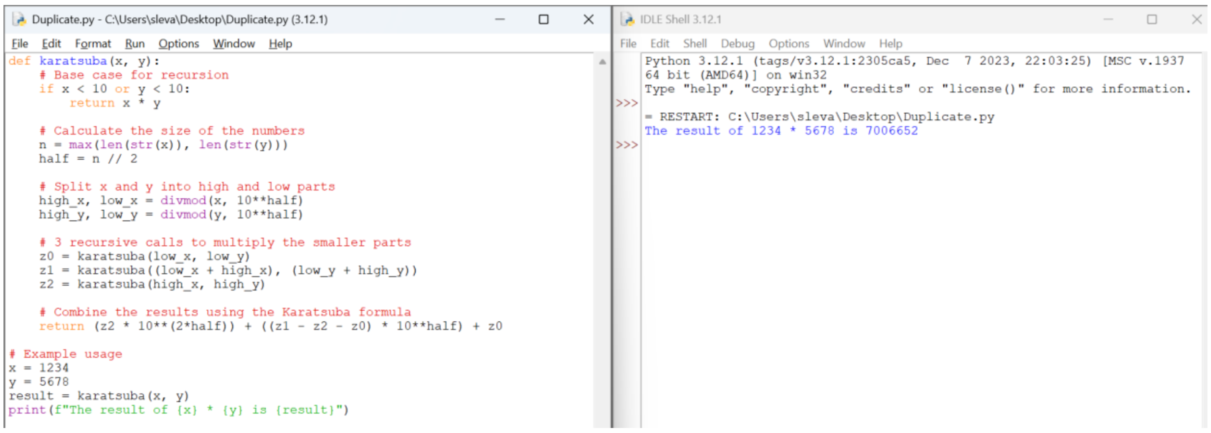
13. Meet in the middle


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
Ln: 46 Col: 0
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Ln: 6 Col: 0

```python
    mid = n // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    def generate_subsets(arr):
        subsets = []
        n = len(arr)
        for i in range(1 << n):
            subset_sum = 0
            for j in range(n):
                if i & (1 << j):
                    subset_sum += arr[j]
            subsets.append(subset_sum)
        return subsets

    left_sums = generate_subsets(left_half)
    right_sums = generate_subsets(right_half)
    right_sums.sort()

    closest_sum = float('inf')

    for left_sum in left_sums:
        lo, hi = 0, len(right_sums) - 1
        while lo <= hi:
            mid = (lo + hi) // 2
            current_sum = left_sum + right_sums[mid]
            if abs(current_sum - target) < abs(closest_sum - target):
                closest_sum = current_sum

            if current_sum < target:
                lo = mid + 1
            else:
                hi = mid - 1

    return closest_sum

# Example usage
arr = [3, 34, 4, 12, 5, 2]
target = 40
print("Closest subset sum to the target:", closest_subset_sum(arr, target))
```

```
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\sleva\Desktop\Duplicate.py
Closest subset sum to the target: 40
>>>
```

16. Karatsuba algorithm


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
def karatsuba(x, y):
    # Base case for recursion
    if x < 10 or y < 10:
        return x * y

    # Calculate the size of the numbers
    n = max(len(str(x)), len(str(y)))
    half = n // 2

    # Split x and y into high and low parts
    high_x, low_x = divmod(x, 10**half)
    high_y, low_y = divmod(y, 10**half)

    # 3 recursive calls to multiply the smaller parts
    z0 = karatsuba(low_x, low_y)
    z1 = karatsuba((low_x + high_x), (low_y + high_y))
    z2 = karatsuba(high_x, high_y)

    # Combine the results using the Karatsuba formula
    return (z2 * 10**(2*half)) + ((z1 - z2 - z0) * 10**half) + z0

# Example usage
x = 1234
y = 5678
result = karatsuba(x, y)
print(f"The result of {x} * {y} is {result}")
```

```
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\sleva\Desktop\Duplicate.py
The result of 1234 * 5678 is 7006652
>>>
```