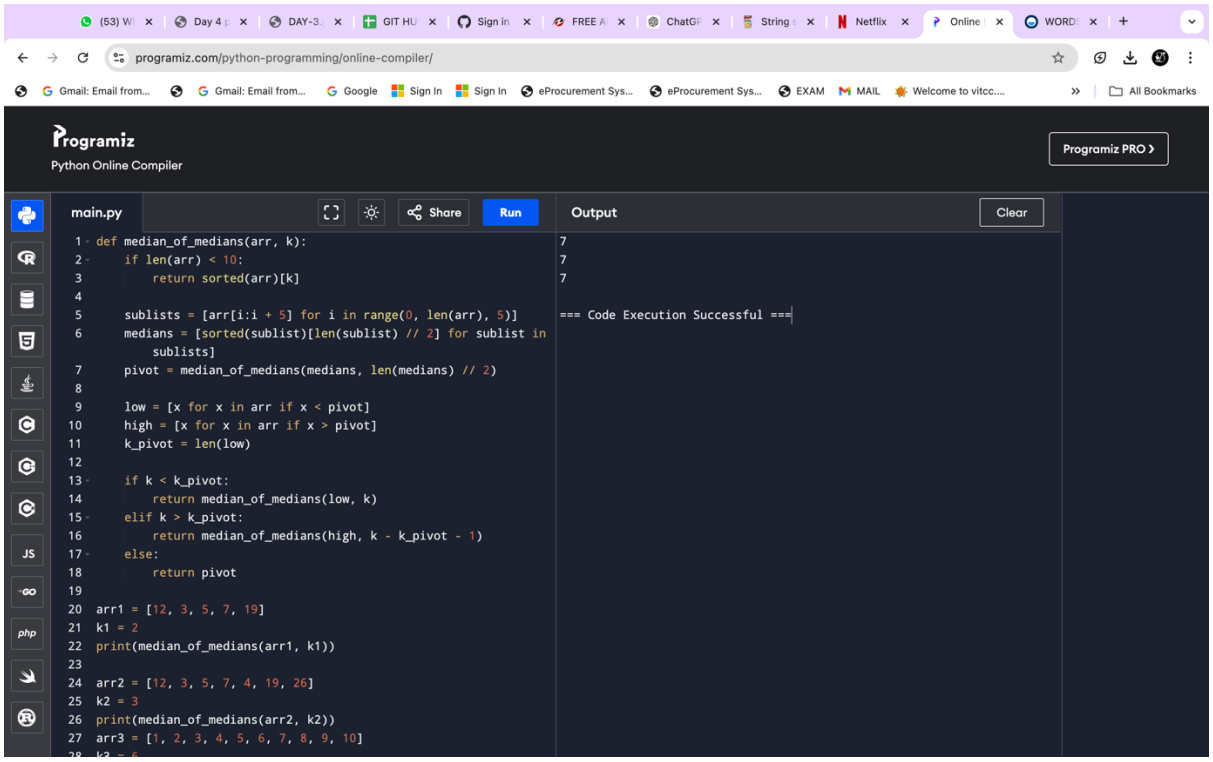
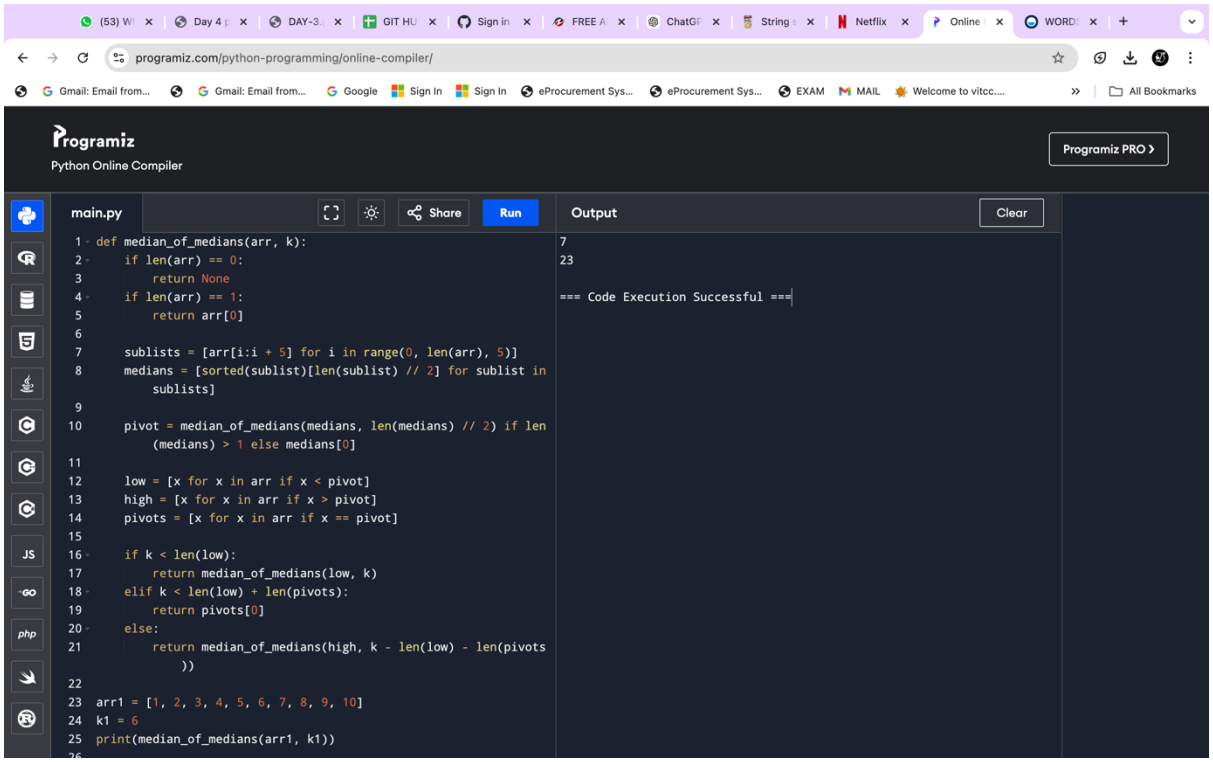
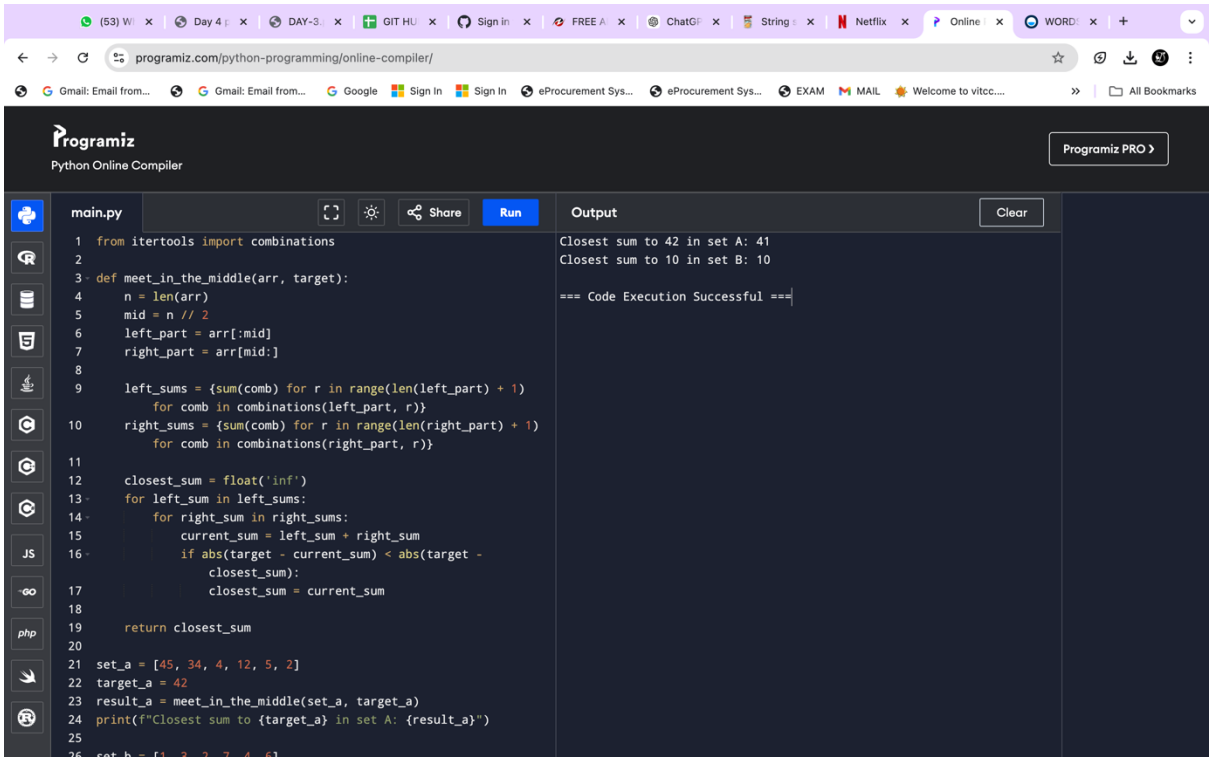
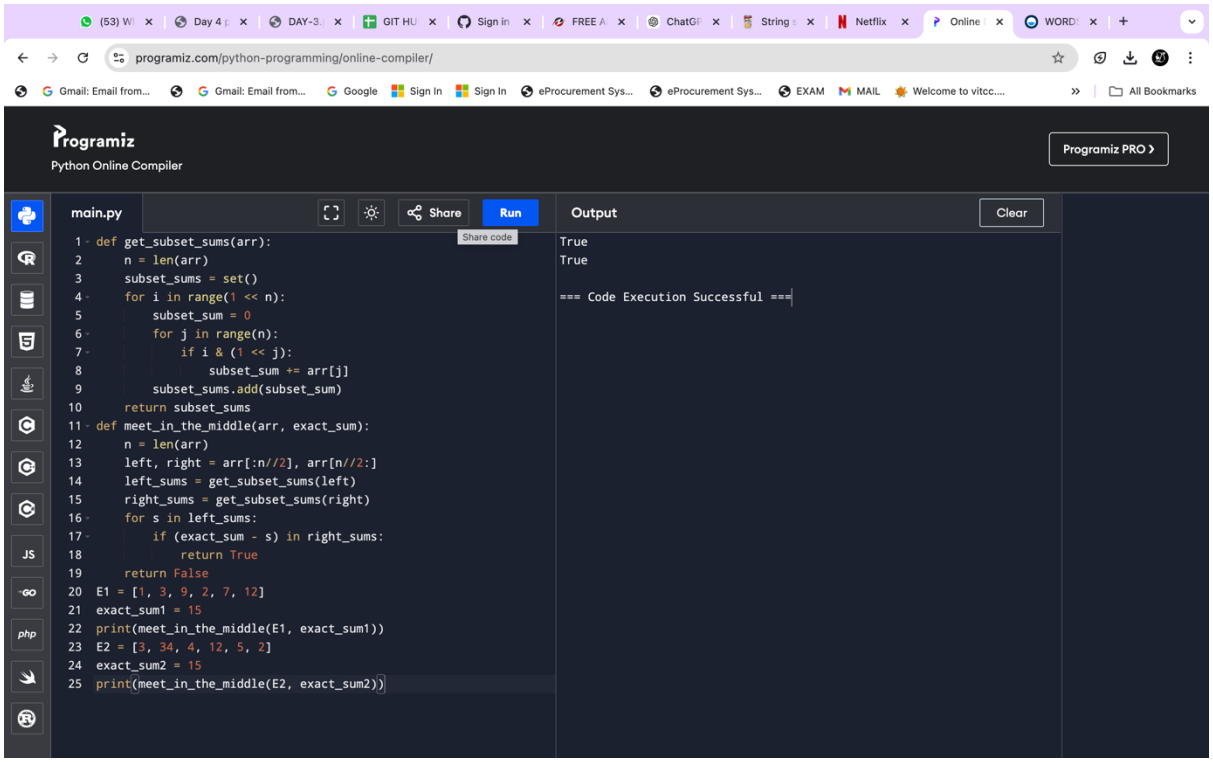
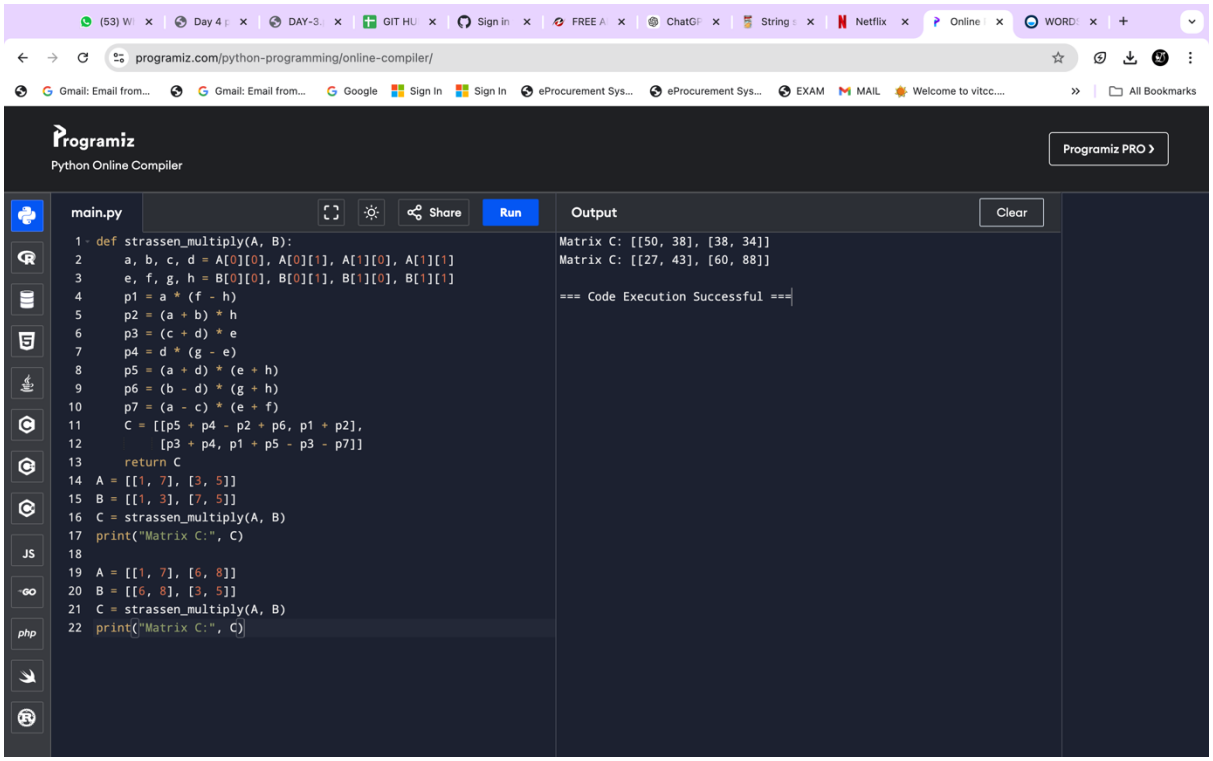
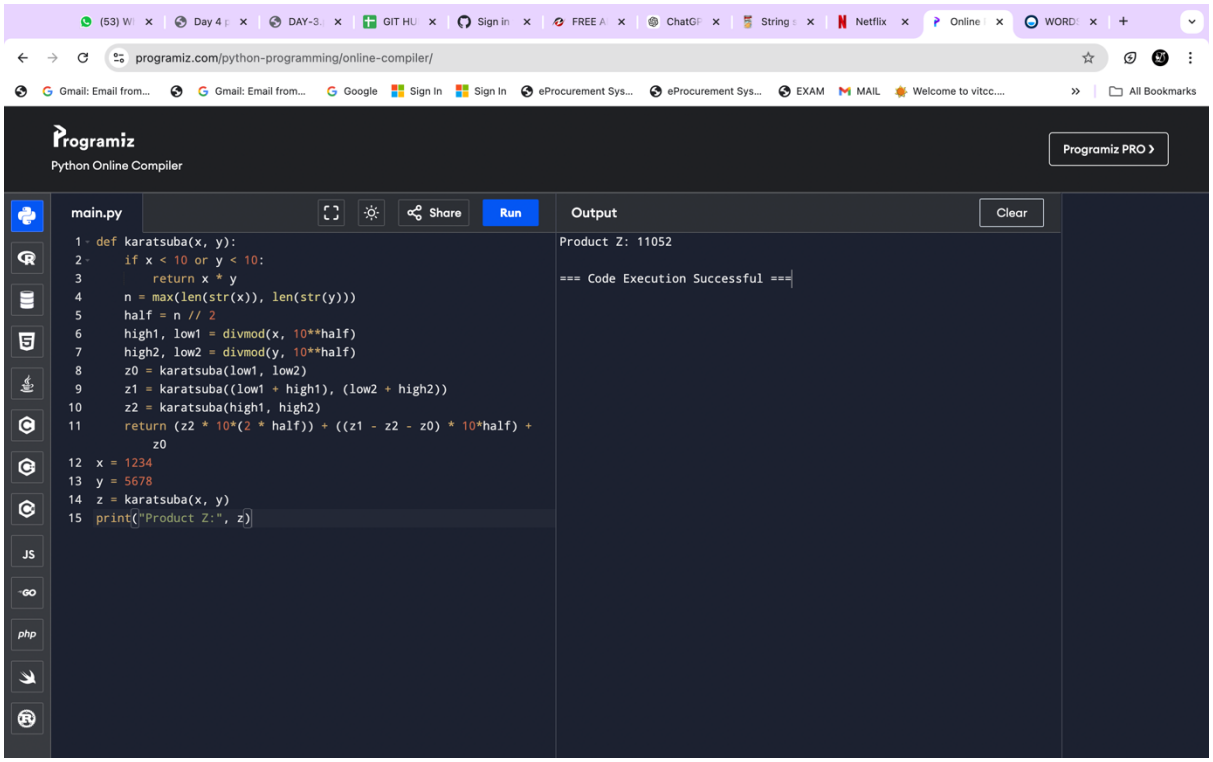

programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def median_of_medians(arr, k):
if len(arr) < 10:
return sorted(arr)[k]
sublists = [arr[i:i + 5] for i in range(0, len(arr), 5)]
medians = [sorted(sublist)[len(sublist) // 2] for sublist in sublists]
pivot = median_of_medians(medians, len(medians) // 2)
low = [x for x in arr if x < pivot]
high = [x for x in arr if x > pivot]
k_pivot = len(low)
if k < k_pivot:
return median_of_medians(low, k)
elif k > k_pivot:
return median_of_medians(high, k - k_pivot - 1)
else:
return pivot
arr1 = [12, 3, 5, 7, 19]
k1 = 2
print(median_of_medians(arr1, k1))
arr2 = [12, 3, 5, 7, 4, 19, 26]
k2 = 3
print(median_of_medians(arr2, k2))
arr3 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
7
7
7
=== Code Execution Successful ===


programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
def median_of_medians(arr, k):
if len(arr) == 0:
return None
if len(arr) == 1:
return arr[0]
sublists = [arr[i:i + 5] for i in range(0, len(arr), 5)]
medians = [sorted(sublist)[len(sublist) // 2] for sublist in sublists]
pivot = median_of_medians(medians, len(medians) // 2) if len(medians) > 1 else medians[0]
low = [x for x in arr if x < pivot]
high = [x for x in arr if x > pivot]
pivots = [x for x in arr if x == pivot]
if k < len(low):
return median_of_medians(low, k)
elif k < len(low) + len(pivots):
return pivots[0]
else:
return median_of_medians(high, k - len(low) - len(pivots))
arr1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
k1 = 6
print(median_of_medians(arr1, k1))
7
23
=== Code Execution Successful ===


programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
from itertools import combinations
def meet_in_the_middle(arr, target):
n = len(arr)
mid = n // 2
left_part = arr[:mid]
right_part = arr[mid:]
left_sums = {sum(comb) for r in range(len(left_part) + 1)
for comb in combinations(left_part, r)}
right_sums = {sum(comb) for r in range(len(right_part) + 1)
for comb in combinations(right_part, r)}
closest_sum = float('inf')
for left_sum in left_sums:
for right_sum in right_sums:
current_sum = left_sum + right_sum
if abs(target - current_sum) < abs(target -
closest_sum):
closest_sum = current_sum
return closest_sum
set_a = [45, 34, 4, 12, 5, 2]
target_a = 42
result_a = meet_in_the_middle(set_a, target_a)
print(f"Closest sum to {target_a} in set A: {result_a}")
Closest sum to 42 in set A: 41
Closest sum to 10 in set B: 10
=== Code Execution Successful ===


programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Programiz PRO
main.py
Share
Run
Share code
Output
Clear
def get_subset_sums(arr):
n = len(arr)
subset_sums = set()
for i in range(1 << n):
subset_sum = 0
for j in range(n):
if i & (1 << j):
subset_sum += arr[j]
subset_sums.add(subset_sum)
return subset_sums
def meet_in_the_middle(arr, exact_sum):
n = len(arr)
left, right = arr[:n//2], arr[n//2:]
left_sums = get_subset_sums(left)
right_sums = get_subset_sums(right)
for s in left_sums:
if (exact_sum - s) in right_sums:
return True
return False
E1 = [1, 3, 9, 2, 7, 12]
exact_sum1 = 15
print(meet_in_the_middle(E1, exact_sum1))
E2 = [3, 34, 4, 12, 5, 2]
exact_sum2 = 15
print(meet_in_the_middle(E2, exact_sum2))
True
True
=== Code Execution Successful ===


programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
1 def strassen_multiply(A, B):
2 a, b, c, d = A[0][0], A[0][1], A[1][0], A[1][1]
3 e, f, g, h = B[0][0], B[0][1], B[1][0], B[1][1]
4 p1 = a * (f - h)
5 p2 = (a + b) * h
6 p3 = (c + d) * e
7 p4 = d * (g - e)
8 p5 = (a + d) * (e + h)
9 p6 = (b - d) * (g + h)
10 p7 = (a - c) * (e + f)
11 C = [[p5 + p4 - p2 + p6, p1 + p2],
12 [p3 + p4, p1 + p5 - p3 - p7]]
13 return C
14 A = [[1, 7], [3, 5]]
15 B = [[1, 3], [7, 5]]
16 C = strassen_multiply(A, B)
17 print("Matrix C:", C)
18
19 A = [[1, 7], [6, 8]]
20 B = [[6, 8], [3, 5]]
21 C = strassen_multiply(A, B)
22 print("Matrix C:", C)
Matrix C: [[50, 38], [38, 34]]
Matrix C: [[27, 43], [60, 88]]
=== Code Execution Successful ===


programiz.com/python-programming/online-compiler/
Programiz
Python Online Compiler
Programiz PRO
main.py
Share
Run
Output
Clear
1 def karatsuba(x, y):
2 if x < 10 or y < 10:
3 return x * y
4 n = max(len(str(x)), len(str(y)))
5 half = n // 2
6 high1, low1 = divmod(x, 10**half)
7 high2, low2 = divmod(y, 10**half)
8 z0 = karatsuba(low1, low2)
9 z1 = karatsuba((low1 + high1), (low2 + high2))
10 z2 = karatsuba(high1, high2)
11 return (z2 * 10*(2 * half)) + ((z1 - z2 - z0) * 10*half) +
z0
12 x = 1234
13 y = 5678
14 z = karatsuba(x, y)
15 print("Product Z:", z)
Product Z: 11052
=== Code Execution Successful ===