Edit Format Run Options Window Help

```
median_of_medians(arr, k):
# Base case: small list → just sort and return
if len(arr) <= 5:
    return sorted(arr)[k]

# Step 1: split arr into chunks of 5 and find their medians
chunks = [arr[i:i+5] for i in range(0, len(arr), 5)]
medians = [sorted(chunk)[len(chunk)//2] for chunk in chunks]

# Step 2: recursively find pivot (median of medians)
pivot = median_of_medians(medians, len(medians)//2)

# Step 3: partition into three parts
low   = [x for x in arr if x < pivot]
high = [x for x in arr if x > pivot]
equal = [x for x in arr if x == pivot]

# Step 4: decide where k lies
if k < len(low):
    return median_of_medians(low, k)
elif k < len(low) + len(equal):
    return pivot
else:
    return median_of_medians(high, k - len(low) - len(equal))


Example usage
ns = [12, 3, 5, 7, 4, 19, 26]
= 3   # 0-based → 4th smallest element
int(f"{k+1}-th smallest element is:", median_of_medians(nums, k))
```

IDLE Shell 3.13.7 - C:/Users/DELL/fir_41.py (3.13.7)

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pui-41.py ==========
4-th smallest element is: 7
>>>
========== RESTART: C:/Users/DELL/pui-41.py ==========
4-th smallest element is: 7
>>>
```

dit Format Run Options Window Help

```
ullJustify(words, maxWidth):
res, cur, num_of_letters = [], [], 0

for w in words:
    # if adding this word exceeds the width, justify the current line
    if num_of_letters + len(w) + len(cur) > maxWidth:
        for i in range(maxWidth - num_of_letters):
            cur[i % (len(cur) - 1 or 1)] += " "
        res.append("".join(cur))
        cur, num_of_letters = [], 0
    cur.append(w)
    num_of_letters += len(w)

# last line (left-justified)
res.append(" ".join(cur).ljust(maxWidth))
return res


--------------- Example Usage ---------------
ds1 = ["This", "is", "an", "example", "of", "text", "justification."]
Width1 = 16

ds2 = ["What", "must", "be", "acknowledgment", "shall", "be"]
Width2 = 16

nt(fullJustify(words1, maxWidth1))
nt(fullJustify(words2, maxWidth2))
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [
MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information
.
>>>
========== RESTART: C:/Users/DELL/piu56.py ====
==========
['This    is    an', 'example  of text', 'justification.  ']
['What    must    be', 'acknowledgment  ', 'shall be        ']
>>>
```

```
Edit Format Run Options Window Help
F = float('inf')

f floyd_warshall(n, dist):
  """Floyd-Warshall Algorithm"""
  for k in range(n):
      for i in range(n):
          for j in range(n):
              if dist[i][k] + dist[k][j] < dist[i][j]:
                  dist[i][j] = dist[i][k] + dist[k][j]
  return dist


ef print_matrix(dist):
   for row in dist:
       print(row)


 ---------------- Main ----------------
rint("=== Floyd's Algorithm for Routers ===")

 = 6  # Routers A-F

 Initial distance matrix
NF = float('inf')
list = [[INF]*n for _ in range(n)]
or i in range(n):
    dist[i][i] = 0

# Given edges
edges = [
    (0, 1, 1),   # A -> B
    (0, 2, 5),   # A -> C
    (1, 2, 2),   # B -> C
    (1, 3, 1),   # B -> D
    (2, 4, 3),   # C -> E
    (3, 4, 1),   # D -> E
    (3, 5, 6),   # D -> F
    (4, 5, 2)    # E -> F
]

# Fill adjacency matrix
for u, v, w in edges:
```

```
IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
=== Floyd's Algorithm for Routers ===

Initial Distance Matrix:
[0, 1, 5, inf, inf, inf]
[1, 0, 2, 1, inf, inf]
[5, 2, 0, inf, 3, inf]
[inf, 1, inf, 0, 1, 6]
[inf, inf, 3, 1, 0, 2]
[inf, inf, inf, 6, 2, 0]

Distance Matrix After Floyd-Warshall:
[0, 1, 3, 2, 3, 5]
[1, 0, 2, 1, 2, 4]
[3, 2, 0, 3, 3, 5]
[2, 1, 3, 0, 1, 3]
[3, 2, 3, 1, 0, 2]
[5, 4, 5, 3, 2, 0]

Shortest Path Router A to Router F (before failure): 5

=== Simulating Link Failure: Router B-D removed ===

Distance Matrix After Link Failure (before Floyd-Warshall):
[0, 1, 5, inf, inf, inf]
[1, 0, 2, inf, inf, inf]
[5, 2, 0, inf, 3, inf]
[inf, inf, inf, 0, 1, 6]
[inf, inf, 3, 1, 0, 2]
[inf, inf, inf, 6, 2, 0]

Distance Matrix After Floyd-Warshall (with B-D failure):
[0, 1, 3, 7, 6, 8]
[1, 0, 2, 6, 5, 7]
[3, 2, 0, 4, 3, 5]
[7, 6, 4, 0, 1, 3]
[6, 5, 3, 1, 0, 2]
[8, 7, 5, 3, 2, 0]

Shortest Path Router A to Router F (after failure): 8
>>>
```

```
Edit Format Run Options Window Help
= float('inf')

floyd_warshall_with_path(n, edges, src, dest):
# Step 1: Initialize distance and next matrices
dist = [[INF] * n for _ in range(n)]
next_hop = [[-1] * n for _ in range(n)]

for i in range(n):
    dist[i][i] = 0
    next_hop[i][i] = i

for u, v, w in edges:
    dist[u][v] = w
    next_hop[u][v] = v

print("Initial Distance Matrix:")
for row in dist:
    print(row)

# Step 2: Floyd-Warshall
for k in range(n):
    for i in range(n):
        for j in range(n):
            if dist[i][k] + dist[k][j] < dist[i][j]:
                dist[i][j] = dist[i][k] + dist[k][j]
                next_hop[i][j] = next_hop[i][k]

print("\nDistance Matrix After Floyd-Warshall:")
for row in dist:
    print(row)

# Step 3: Reconstruct path from src to dest
if next_hop[src][dest] == -1:
    return None, dist[src][dest]

path = [src]
while src != dest:
    src = next_hop[src][dest]
    path.append(src)

return path, dist[path[0]][path[-1]]
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64
 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/piu59.py ==========
======
=== Test Case (a): 4 Cities ===
Initial Distance Matrix:
[0, 3, 8, -4]
[inf, 0, 4, 1]
[2, inf, 0, inf]
[inf, 6, -5, 0]

Distance Matrix After Floyd-Warshall:
[-7, -4, -9, -11]
[-2, 0, -4, -6]
[-5, -2, -7, -9]
[-10, -7, -12, -14]

Shortest Path City 1 to City 3: [0, 3, 2], Distance = -9

=== Test Case (b): 6 Routers ===
Initial Distance Matrix:
[0, 1, 5, inf, inf, inf]
[inf, 0, 2, 1, inf, inf]
[inf, inf, 0, inf, 3, inf]
[inf, inf, inf, 0, 1, 6]
[inf, inf, inf, inf, 0, 2]
[inf, inf, inf, inf, inf, 0]

Distance Matrix After Floyd-Warshall:
[0, 1, 3, 2, 3, 5]
[inf, 0, 2, 1, 2, 4]
[inf, inf, 0, inf, 3, 5]
[inf, inf, inf, 0, 1, 3]
[inf, inf, inf, inf, 0, 2]
[inf, inf, inf, inf, inf, 0]

Shortest Path Router A to Router F: [0, 1, 3, 4, 5], Distance = 5
>>>
```

Edit Format Run Options Window Help

```
= float("inf")

floyd_warshall(n, edges):
# Step 1: Initialize distance matrix
dist = [[INF] * n for _ in range(n)]
next_node = [[-1] * n for _ in range(n)]  # for path reconstruction

for i in range(n):
    dist[i][i] = 0

# fill initial edges
for u, v, w in edges:
    dist[u][v] = w
    next_node[u][v] = v

print("Initial Distance Matrix:")
for row in dist:
    print(row)

# Step 2: Floyd-Warshall update
for k in range(n):
    for i in range(n):
        for j in range(n):
            if dist[i][k] + dist[k][j] < dist[i][j]:
                dist[i][j] = dist[i][k] + dist[k][j]
                next_node[i][j] = next_node[i][k]

print("\nFinal Distance Matrix (after Floyd-Warshall):")
for row in dist:
    print(row)

return dist, next_node

Function to reconstruct path
ef get_path(u, v, next_node):
if next_node[u][v] == -1:
    return []
path = [u]
while u != v:
    u = next_node[u][v]
    path.append(u)
return path
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11
) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more informat
ion.
>>>
========== RESTART: C:/Users/DELL/piu58.py =
==========
Initial Distance Matrix:
[0, 3, inf, inf]
[inf, 0, 1, 4]
[inf, inf, 0, 1]
[inf, inf, inf, 0]

Final Distance Matrix (after Floyd-Warshall):
[0, 3, 4, 5]
[inf, 0, 1, 2]
[inf, inf, 0, 1]
[inf, inf, inf, 0]

Shortest path from City 0 to City 3:
[0, 1, 2, 3]
Distance: 5

=== Test Case 2 ===
Initial Distance Matrix:
[0, 3, 8, -4]
[inf, 0, 4, 1]
[2, inf, 0, inf]
[inf, 6, -5, 0]

Final Distance Matrix (after Floyd-Warshall):
[-7, -4, -9, -11]
[-2, 0, -4, -6]
[-5, -2, -7, -9]
[-10, -7, -12, -14]

Shortest path from City 0 to City 2:
[0, 3, 2]
Distance: -9
>>>
```

Ln: 37 Col:

```
Edit Format Run Options Window Help
ss WordFilter:
 def __init__(self, words):
     self.lookup = {}
     for i, word in enumerate(words):
         # generate all prefix + suffix combinations
         for p in range(len(word) + 1):
             for s in range(len(word) + 1):
                 key = (word[:p], word[len(word) - s:])
                 self.lookup[key] = i   # store largest index (overwrite old one)

 def f(self, pref, suff):
     return self.lookup.get((pref, suff), -1)


--------------- Example Usage ---------------
dFilter = WordFilter(["apple"])
nt(wordFilter.f("a", "e"))  # Output: 0
nt(wordFilter.f("ap", "le"))  # Output: 0
nt(wordFilter.f("b", "e"))   # Output: -1
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15
:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more infor
mation.
>>>
========== RESTART: C:/Users/DELL/piu57.p
y ==========
0
0
-1
>>>
```

Edit Format Run Options Window Help

```
word_break_with_segmentation(s, wordDict):
word_set = set(wordDict)
n = len(s)

# dp[i] stores list of words that end at position i (for backtracking)
dp = [[] for _ in range(n + 1)]
dp[0] = [""]  # base case: empty string

for i in range(1, n + 1):
    for j in range(i):
        word = s[j:i]
        if dp[j] and word in word_set:
            for prev in dp[j]:
                if prev:
                    dp[i].append(prev + " " + word)
                else:
                    dp[i].append(word)

# If we have any segmentation ending at n, return Yes and examples
if dp[n]:
    print("Yes")
    print("Possible segmentations:")
    for seg in dp[n]:
        print(seg)
else:
    print("No")

--------------- Example Usage ---------------
rdDict = {"i", "like", "sam", "sung", "samsung", "mobile",
          "ice", "cream", "icecream", "man", "go", "mango"}

 = "ilike"
 = "ilikesamsung"

int(f"Input: {s1}")
rd_break_with_segmentation(s1, wordDict)
int("\nInput:", s2)
rd_break_with_segmentation(s2, wordDict)
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 6
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/Pictures/screenshots/pyi-55.py =====
Input: ilike
Yes
Possible segmentations:
i like

Input: ilikesamsung
Yes
Possible segmentations:
i like samsung
i like sam sung
>>>
```

i-54.py - C:/Users/DELL/pyi-54.py (3.13.7)

Edit Format Run Options Window Help

```
word_break(s: str, wordDict: list) -> bool:
word_set = set(wordDict)  # for O(1) lookups
n = len(s)

# dp[i] = True if s[0:i] can be segmented into words in wordDict
dp = [False] * (n + 1)
dp[0] = True  # empty string can be segmented

for i in range(1, n + 1):
    for j in range(i):
        if dp[j] and s[j:i] in word_set:
            dp[i] = True
            break  # no need to check further

return dp[n]

---------- Example Usage ----------
= "leetcode"
rdDict1 = ["leet", "code"]

= "applepenapple"
rdDict2 = ["apple", "pen"]

= "catsandog"
rdDict3 = ["cats","dog","sand","and","cat"]

rint(word_break(s1, wordDict1))  # True
rint(word_break(s2, wordDict2))  # True
rint(word_break(s3, wordDict3))  # False
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-54.py ==========
True
True
False
>>>
```

Ln: 8

Edit Format Run Options Window Help

```
length_of_longest_substring(s: str) -> int:
char_index = {}  # stores last index of each character
max_len = 0
left = 0  # left pointer of sliding window

for right, char in enumerate(s):
    if char in char_index and char_index[char] >= left:
        # Move left pointer past the last occurrence
        left = char_index[char] + 1
    char_index[char] = right
    max_len = max(max_len, right - left + 1)

return max_len

--------------- Example Usage ---------------
= "abcabcbb"
= "bbbbb"
= "pwwkew"

nt(f"Input: {s1} -> Output: {length_of_longest_substring(s1)}")
nt(f"Input: {s2} -> Output: {length_of_longest_substring(s2)}")
nt(f"Input: {s3} -> Output: {length_of_longest_substring(s3)}")
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-53.py ==========
Input: abcabcbb -> Output: 3
Input: bbbbb -> Output: 1
Input: pwwkew -> Output: 3
>>>
```

lit Format Run Options Window Help

```
ongest_palindrome(s: str) -> str:
f not s or len(s) == 0:
    return ""

start, end = 0, 0

def expand(left, right):
    # Expand while characters match
    while left >= 0 and right < len(s) and s[left] == s[right]:
        left -= 1
        right += 1
    return left + 1, right - 1

for i in range(len(s)):
    # Odd length palindrome
    l1, r1 = expand(i, i)
    # Even length palindrome
    l2, r2 = expand(i, i + 1)

    if r1 - l1 > end - start:
        start, end = l1, r1
    if r2 - l2 > end - start:
        start, end = l2, r2

return s[start:end+1]

---------- Example Usage ----------
= "babad"
= "cbbd"

int("Input:", s1, "-> Longest palindrome:", longest_palindrome(s1))
int("Input:", s2, "-> Longest palindrome:", longest_palindrome(s2))
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025
, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for mor
e information.
>>>
========== RESTART: C:/Users/DELL/p
yi-52.py ==========
Input: babad -> Longest palindrome: bab
Input: cbbd -> Longest palindrome: bb
>>>
```

Ln: 7 Col: 0

```
dit Format Run Options Window Help
rt itertools

ties
es = ['A', 'B', 'C', 'D', 'E']

mmetric distance matrix
der: A, B, C, D, E
 = [
 [0, 10, 15, 20, 25],  # A
 [10, 0, 35, 25, 30],  # B
 [15, 35, 0, 30, 20],  # C
 [20, 25, 30, 0, 15],  # D
 [25, 30, 20, 15, 0]   # E


 len(cities)
 distance = float('inf')
t_route = []

enerate all permutations of cities excluding starting city A (index 0
 perm in itertools.permutations(range(1, n)):
  route = [0] + list(perm) + [0]  # Start and end at A
  total_dist = sum(dist[route[i]][route[i+1]] for i in range(n))

  if total_dist < min_distance:
      min_distance = total_dist
      best_route = route

Print result
st_route_names = [cities[i] for i in best_route]
int("Shortest route:", " -> ".join(best_route_names))
int("Total distance:", min_distance)
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-51.py ==========
Shortest route: A -> B -> D -> E -> C -> A
Total distance: 85
>>>
```

Ln: 7 Co

```
lit Format Run Options Window Help
t sys

ction to find minimum of two numbers
in_val(a, b):
eturn a if a < b else b

cursive TSP function
tsp(dist, visited, pos, n):
# Base case: all cities visited, return to starting city
if visited == (1 << n) - 1:
    return dist[pos][0]

ans = sys.maxsize

for city in range(n):
    if (visited & (1 << city)) == 0:  # if city not visited
        new_ans = dist[pos][city] + tsp(dist, visited | (1 << city),
        ans = min_val(ans, new_ans)

return ans

------------------ Test Cases ------------------

4  # Number of cities

est Case 1
t1 = [
  [0, 10, 15, 20],
  [10, 0, 35, 25],
  [15, 35, 0, 30],
  [20, 25, 30, 0]


Test Case 2
st2 = [
  [0, 10, 10, 10],
  [10, 0, 10, 10],
  [10, 10, 0, 10],
  [10, 10, 10, 0]


Test Case 3
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-50.py ==========
Test Case 1: Minimum path distance = 80
Test Case 2: Minimum path distance = 40
Test Case 3: Minimum path distance = 14
>>>
```

Ln: 8

Edit Format Run Options Window Help

```
def min_time_three_lines(station_times, transfer, entry=None, exit=No
    """
    station_times: list of lists, station_times[line][station]
    transfer: matrix transfer[i][j] cost to move from line i to line
    entry: list of entry times for each line (optional, default 0)
    exit: list of exit times for each line (optional, default 0)
    Returns: (min_time, path_lines) where path_lines is list of chose
    """
    m = len(station_times)     # number of lines
    n = len(station_times[0]) # number of stations (assumes all lines
    if entry is None:
        entry = [0]*m
    if exit is None:
        exit = [0]*m

    # dp[l][s] = minimum time to complete station s on line l
    dp = [[float('inf')]*n for _ in range(m)]
    parent = [[None]*n for _ in range(m)]  # to reconstruct path

    # Base case: first station
    for l in range(m):
        dp[l][0] = entry[l] + station_times[l][0]
        parent[l][0] = -1  # start

    # Fill DP table
    for s in range(1, n):
        for l in range(m):
            best_prev = None
            best_val = float('inf')
            for p in range(m):
                cand = dp[p][s-1] + transfer[p][l] + station_times[l]
                if cand < best_val:
                    best_val = cand
                    best_prev = p
            dp[l][s] = best_val
            parent[l][s] = best_prev

    # Add exit times and find best final line
    best_final = None
    best_total = float('inf')
    for l in range(m):
        total = dp[l][n-1] + exit[l]
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-49.py ==========
Minimum total production time: 17
Chosen line for each station (0-based indices): [0, 0, 0]
Chosen line names (1-based): [1, 1, 1]
>>>
```

i-48.py - C:/Users/DELL/pyi-48.py (3.13.7)

Edit Format Run Options Window Help

```
assembly_line(a1, a2, t1, t2, e1, e2, x1, x2):
n = len(a1)

# DP arrays
f1 = [0] * n
f2 = [0] * n

# Base case
f1[0] = e1 + a1[0]
f2[0] = e2 + a2[0]

# Fill DP
for i in range(1, n):
    f1[i] = min(f1[i-1] + a1[i], f2[i-1] + t2[i-1] + a1[i])
    f2[i] = min(f2[i-1] + a2[i], f1[i-1] + t1[i-1] + a2[i])

# Final answer
return min(f1[n-1] + x1, f2[n-1] + x2)


Example Test Case
= 4
 = [4, 5, 3, 2]
 = [2, 10, 1, 4]
 = [7, 4, 5]   # transfer from line1 -> line2
 = [9, 2, 8]   # transfer from line2 -> line1
, e2 = 10, 12  # entry times
, x2 = 18, 7   # exit times

int("Minimum time required:", assembly_line(a1, a2, t1, t2, e1, e2, x1, x2))
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.194
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-48.py ==========
Minimum time required: 35
>>>
```

Edit Format Run Options Window Help

```
dice_throw(num_sides, num_dice, target):
# DP table: (num_dice+1) × (target+1)
dp = [[0] * (target + 1) for _ in range(num_dice + 1)]

# Base case
dp[0][0] = 1

# Fill DP table
for d in range(1, num_dice + 1):
    for t in range(1, target + 1):
        for face in range(1, num_sides + 1):
            if t - face >= 0:
                dp[d][t] += dp[d - 1][t - face]

return dp[num_dice][target]


Test Case 1
int("Test Case 1:")
ys1 = dice_throw(6, 2, 7)
int("Number of ways to reach sum 7:", ways1)

Test Case 2
int("\nTest Case 2:")
ys2 = dice_throw(4, 3, 10)
int("Number of ways to reach sum 10:", ways2)
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-47.py ==========
Test Case 1:
Number of ways to reach sum 7: 6

Test Case 2:
Number of ways to reach sum 10: 6
>>>
```

```
dit Format Run Options Window Help
caratsuba(x, y):
# Base case for small numbers
if x < 10 or y < 10:
    return x * y

# Calculate size of numbers
n = max(len(str(x)), len(str(y)))
m = n // 2  # Split position

# Split x and y
high1, low1 = divmod(x, 10**m)
high2, low2 = divmod(y, 10**m)

# 3 recursive multiplications
z0 = karatsuba(low1, low2)
z2 = karatsuba(high1, high2)
z1 = karatsuba(low1 + high1, low2 + high2) - z2 - z0

# Combine results
return (z2 * 10**(2*m)) + (z1 * 10**m) + z0


est Case 1
y = 1234, 5678
= karatsuba(x, y)
int(f"{x} × {y} = {z}")
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-46.py ==========
1234 × 5678 = 7006652
>>>
```

Ln: 6 Col: 0

Edit Format Run Options Window Help

```
strassen_2x2(A, B):
 # Extract elements
 a, b, c, d = A[0][0], A[0][1], A[1][0], A[1][1]
 e, f, g, h = B[0][0], B[0][1], B[1][0], B[1][1]

 # Strassen's formulas
 M1 = (a + d) * (e + h)
 M2 = (c + d) * e
 M3 = a * (f - h)
 M4 = d * (g - e)
 M5 = (a + b) * h
 M6 = (c - a) * (e + f)
 M7 = (b - d) * (g + h)

 # Compute result matrix
 C11 = M1 + M4 - M5 + M7
 C12 = M3 + M5
 C21 = M2 + M4
 C22 = M1 - M2 + M3 + M6

 return [[C11, C12],
         [C21, C22]]


 Test Case 1
 = [[1, 7],
    [3, 5]]
 = [[6, 8],
    [4, 2]]

 = strassen_2x2(A, B)
print("Result Matrix C =")
for row in C:
    print(row)
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-45.py ==========
Result Matrix C =
[34, 22]
[38, 34]
>>>
```

```
Format Run Options Window Help
eet_in_middle_exact(nums, target):
 = len(nums)
eft, right = nums[:n//2], nums[n//2:]

 Function to generate all subset sums
ef subset_sums(arr):
    sums = []
    def backtrack(i, current):
        if i == len(arr):
            sums.append(current)
            return
        backtrack(i+1, current)           # exclude
        backtrack(i+1, current + arr[i]) # include
    backtrack(0, 0)
    return sums

 left_sums = subset_sums(left)
 right_sums = subset_sums(right)
 right_sums_set = set(right_sums)  # for fast lookup

 # Check if any pair forms the exact target
 for s in left_sums:
     if target - s in right_sums_set:
         return True
 return False


Example a)
r1 = [1, 3, 9, 2, 7, 12]
rget1 = 15
int("Subset with exact sum exists (Example a):", meet_in_middle_exact(arr1, tar

 Example b)
rr2 = [3, 34, 4, 12, 5, 2]
arget2 = 15
rint("Subset with exact sum exists (Example b):", meet_in_middle_exact(arr2, tar
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1
944 64 bit (AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-44.py ==========
==========
Subset with exact sum exists (Example a): True
Subset with exact sum exists (Example b): True
>>>
```

Edit Format Run Options Window Help

```
median_of_medians(arr, k):
# Base case: if array small, just sort and return k-th
if len(arr) <= 5:
    return sorted(arr)[k-1]   # k-1 because of 1-based indexing

# Step 1: split arr into chunks of 5
chunks = [arr[i:i+5] for i in range(0, len(arr), 5)]

# Step 2: find medians of each chunk
medians = [sorted(chunk)[len(chunk)//2] for chunk in chunks]

# Step 3: recursively find pivot (median of medians)
pivot = median_of_medians(medians, (len(medians)+1)//2)

# Step 4: partition array around pivot
low  = [x for x in arr if x < pivot]
equal = [x for x in arr if x == pivot]
high = [x for x in arr if x > pivot]

# Step 5: choose where k falls
if k <= len(low):
    return median_of_medians(low, k)
elif k <= len(low) + len(equal):
    return pivot
else:
    return median_of_medians(high, k - len(low) - len(equal))


Example runs
r1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
 = 6
int(f"{k1}-th smallest element in arr1:", median_of_medians(arr1, k1))

r2 = [23, 17, 31, 44, 55, 21, 20, 18, 19, 27]
 = 5
int(f"{k2}-th smallest element in arr2:", median_of_medians(arr2, k2))
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pui-42.py ==========
6-th smallest element in arr1: 6
5-th smallest element in arr2: 21
>>>
```

```
Edit Format Run Options Window Help
ort bisect

E meet_in_middle(nums, target):
  # Step 1: split into two halves
  n = len(nums)
  left, right = nums[:n//2], nums[n//2:]

  # Step 2: generate all subset sums
  def subset_sums(arr):
      sums = []
      def backtrack(i, current):
          if i == len(arr):
              sums.append(current)
              return
          backtrack(i+1, current)           # exclude
          backtrack(i+1, current + arr[i]) # include
      backtrack(0, 0)
      return sums

  left_sums = subset_sums(left)
  right_sums = subset_sums(right)

  # Step 3: sort right sums
  right_sums.sort()

  # Step 4: find best sum close to target
  best_sum = None
  min_diff = float("inf")

  for s in left_sums:
      remaining = target - s
      # find closest in right_sums to 'remaining'
      pos = bisect.bisect_left(right_sums, remaining)

      # check candidate at pos
      if pos < len(right_sums):
          total = s + right_sums[pos]
          if abs(target - total) < min_diff:
              min_diff = abs(target - total)
              best_sum = total

      # check candidate before pos
```

IDLE Shell 3.13.7

File Edit Shell Debug Options Window Help

```
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (
AMD64)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/DELL/pyi-43.py ==========
Closest sum to 42 : 41
Closest sum to 10 : 10
>>>
```