**Dice throw problem**

import random

def dice\_throw():

return random.randint(1, 6)

# Simulate dice throws

num\_throws = 10

throws = [dice\_throw() for \_ in range(num\_throws)]

print(throws)

**Subset sum problem**

def is\_subset\_sum(arr, n, target):

if target == 0:

return True

if n == 0 and target != 0:

return False

if arr[n-1] > target:

return is\_subset\_sum(arr, n-1, target)

return is\_subset\_sum(arr, n-1, target) or is\_subset\_sum(arr, n-1, target-arr[n-1])

arr = [3, 34, 4, 12, 5, 2]

target\_sum = 9

n = len(arr)

if is\_subset\_sum(arr, n, target\_sum):

print("Subset with the given sum exists")

else:

print("No subset with the given sum")

**Assembly line scheduling**

def assembly\_line\_scheduling(processing\_time, switching\_time, entry\_time, exit\_time):

f1 = [0] \* len(processing\_time)

f2 = [0] \* len(processing\_time)

f1[0] = entry\_time[0] + processing\_time[0][0]

f2[0] = entry\_time[1] + processing\_time[1][0]

for i in range(1, len(processing\_time)):

f1[i] = min(f1[i - 1] + processing\_time[0][i], f2[i - 1] + switching\_time[1] + processing\_time[0][i])

f2[i] = min(f2[i - 1] + processing\_time[1][i], f1[i - 1] + switching\_time[0] + processing\_time[1][i])

f1[-1] += exit\_time[0]

f2[-1] += exit\_time[1]

return min(f1[-1], f2[-1])

# Example input

processing\_time = [[7, 9, 3, 4, 8], [8, 5, 6, 4, 5]]

switching\_time = [2, 3]

entry\_time = [2, 3]

exit\_time = [3, 2]

print(assembly\_line\_scheduling(processing\_time, switching\_time, entry\_time, exit\_time))

**Longest Palindromic subsequence**

def longest\_palindromic\_subsequence(s):

n = len(s)

dp = [[0] \* n for \_ in range(n)]

for i in range(n-1, -1, -1):

dp[i][i] = 1

for j in range(i+1, n):

if s[i] == s[j]:

dp[i][j] = 2 + dp[i+1][j-1]

else:

dp[i][j] = max(dp[i+1][j], dp[i][j-1])

return dp[0][n-1]

# Example

s = "character"

print(longest\_palindromic\_subsequence(s))