Question 1

Test Case 1

# Input the size of the transaction array

n = int(input("Enter the size of transaction array: "))

# Input the values of the transaction array

transactions = list(map(int, input("Enter the values of array: ").split()))

# Input the total number of targets

total\_targets = int(input("Enter the total number of targets that need to be achieved: "))

for \_ in range(total\_targets):

# Input the value of the target

target = int(input("Enter the value of target: "))

# Initialize a running total

running\_total = 0

# Iterate through transactions to find when the target is achieved

for i in range(n):

running\_total += transactions[i]

# Check if the running total is greater than or equal to the target

if running\_total >= target:

# Print the transaction number (1-based index)

print(f"Target achieved after {i+1} transactions")

break

else:

# If the loop completes without breaking, target is not achieved

print("Target is not achieved")

Test Case 2

# Input the size of the transaction array

n = int(input("Enter the size of transaction array: "))

# Input the values of the transaction array

transactions = list(map(int, input("Enter the values of array: ").split()))

# Input the total number of targets

total\_targets = int(input("Enter the total number of targets that need to be achieved: "))

for \_ in range(total\_targets):

# Input the value of the target

target = int(input("Enter the value of target: "))

# Initialize a running total

running\_total = 0

# Iterate through transactions to find when the target is achieved

for i in range(n):

running\_total += transactions[i]

# Check if the running total is greater than or equal to the target

if running\_total >= target:

# Print the transaction number (1-based index)

print(f"Target achieved after {i+1} transactions")

break

else:

# If the loop completes without breaking, target is not achieved

print("Given target is not achieved")

Question 2

Test Case 1

# Input the size of currency denominations

n = int(input("Enter the size of currency denominations: "))

# Input the currency denominations in random order

denominations = [int(input("Enter the currency denominations value: ")) for \_ in range(n)]

# Sort the denominations in descending order

denominations.sort(reverse=True)

# Input the amount you want to pay

amount = int(input("Enter the amount you want to pay: "))

# Initialize variables to keep track of the current amount and notes used

current\_amount = 0

notes\_used = {}

# Iterate through the sorted denominations

for denomination in denominations:

if current\_amount < amount:

notes = (amount - current\_amount) // denomination

if notes > 0:

current\_amount += notes \* denomination

notes\_used[denomination] = notes

# Print the minimum number of notes used for each denomination

print("Your payment approach in order to give a minimum number of notes will be:")

for denom, notes in notes\_used.items():

print(f"{denom}:{notes}")

Test Case 2

# Input the size of currency denominations

n = int(input("Enter the size of currency denominations: "))

# Input the currency denominations in random order

denominations = [int(input("Enter the currency denominations value: ")) for \_ in range(n)]

# Sort the denominations in descending order

denominations.sort(reverse=True)

# Input the amount you want to pay

amount = int(input("Enter the amount you want to pay: "))

# Initialize variables to keep track of the current amount and notes used

current\_amount = 0

notes\_used = {}

# Iterate through the sorted denominations

for denomination in denominations:

if current\_amount < amount:

notes = (amount - current\_amount) // denomination

if notes > 0:

current\_amount += notes \* denomination

notes\_used[denomination] = notes

# Print the minimum number of notes used for each denomination

print("Your payment approach in order to give a minimum number of notes will be:")

for denom, notes in notes\_used.items():

print(f"{denom}:{notes}")

Test Case 3

# Input the size of currency denominations

n = int(input("Enter the size of currency denominations: "))

# Input the currency denominations in random order

denominations = [int(input("Enter the currency denominations value: ")) for \_ in range(n)]

# Sort the denominations in descending order

denominations.sort(reverse=True)

# Input the amount you want to pay

amount = int(input("Enter the amount you want to pay: "))

# Initialize variables to keep track of the current amount and notes used

current\_amount = 0

notes\_used = {}

# Iterate through the sorted denominations

for denomination in denominations:

if current\_amount < amount:

notes = (amount - current\_amount) // denomination

if notes > 0:

current\_amount += notes \* denomination

notes\_used[denomination] = notes

# Print the minimum number of notes used for each denomination

print("Your payment approach in order to give a minimum number of notes will be:")

for denom, notes in notes\_used.items():

print(f"{denom}:{notes}")