**Answer 11.** Using pen & Paperwrite the internal mechanism for sum operation using reduce function on this given list:[47,11,42,13];

To understand how the `reduce()` function works internally for summing a list, let's break down the process step-by-step using the provided list: `[47, 11, 42, 13]`.

Internal Mechanism of `reduce()`

The `reduce()` function applies a specified function cumulatively to the items of an iterable, reducing it to a single value. Here’s how the sum operation would work for your list:

1. **Initialization:** Start with the first two elements of the list.

2. **Cumulative Application:** Use the result from the previous step and apply the function with the next element.

3. **Final Result:** Continue this process until all elements have been processed.

**Example Code**

Here's how you would implement the sum operation using `reduce()`:

from functools import reduce

# List of numbers

numbers = [47, 11, 42, 13]

# Define the sum function

def add(x, y):

return x + y

# Use reduce to compute the sum

result = reduce(add, numbers)

print(result) # Output: 113

**Breakdown of Steps**

1. **First Iteration:**

- Take the first two elements: `47` and `11`.

- Apply `add(47, 11)`, which returns `58`.

2. **Second Iteration:**

- Take the result from the first iteration (`58`) and the next element (`42`).

- Apply `add(58, 42)`, which returns `100`.

3. **Third Iteration:**

- Take the result from the second iteration (`100`) and the next element (`13`).

- Apply `add(100, 13)`, which returns `113`.

4. **Final Result:**

- The final output of `reduce(add, numbers)` is `113`.

**Summary**

The `reduce()` function essentially takes a binary function (like addition) and applies it iteratively across the elements of the iterable, resulting in a single aggregated value—in this case, the sum of the list `[47, 11, 42, 13]`, which is `113`.



