<u>Task Write a Python function</u>:

fast_copy(source, count) that:

● Allocates a destination structure large enough to hold count elements.

● Determines how many complete transfer groups of eight elements are required.

● Performs any necessary initial element transfers using structured control flow rather than iteration.
● Completes the remaining transfers in a repetitive block that assigns exactly eight elements per iteration.

● Returns the destination structure containing the copied elements

<u>Example for how it will work</u>
<u>Source Buffer</u>:
[A B ----C D E F G H I J---- K L M N O P Q R]
<u>Destination Buffer</u>:
Step 1: Copy Leftover Elements
[A B --]

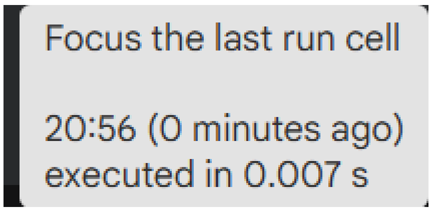
Step 2: Destination after full copy:
[A B C D E F G H I J K L M N O P Q R]

```
def fast_copy(S, count):

     D = [None] * count

   i = 0

    R = count % 8

   groups = count // 8

   print(f"Total: {count} | Partial (R): {R} | Full Groups: {groups}")

   if R >= 7: D[i] = S[i]; i += 1

   if R >= 6: D[i] = S[i]; i += 1

   if R >= 5: D[i] = S[i]; i += 1

   if R >= 4: D[i] = S[i]; i += 1

   if R >= 3: D[i] = S[i]; i += 1

   if R >= 2: D[i] = S[i]; i += 1

   if R >= 1: D[i] = S[i]; i += 1



while groups > 0:
        D[i]     = S[i]
        D[i + 1] = S[i + 1]
        D[i + 2] = S[i + 2]
        D[i + 3] = S[i + 3]
        D[i + 4] = S[i + 4]
        D[i + 5] = S[i + 5]
        D[i + 6] = S[i + 6]
        D[i + 7] = S[i + 7]
            i += 8
        groups -= 1
    return D
source_data = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
result = fast_copy(source_data, 10)
print("Result in D:", result)
```

Output

```
Total: 10 | Partial (R): 2 | Full Groups: 1
Result in D: [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

Running Time for above program


Focus the last run cell

20:56 (0 minutes ago)
executed in 0.007 s

1. First, I create a new empty list called D (Destination). I make it the same size as the source data so there is enough space to copy everything. I also set a counter i = 0 to keep track of where I am while copying.

2. Since copying data in larger blocks is faster, I decide to move the elements in groups of eight. To handle this properly, I calculate the remainder (R) to find out how many elements are left over that don’t fit neatly into groups of eight.

3. Before starting the main fast loop, I take care of these leftover elements. Instead of using a loop that keeps checking a condition again and again, I use a few simple if statements. These statements copy the remaining 1 to 7 elements quickly. This helps align the data so the remaining part can be processed cleanly in groups of eight.

4. After that, I move on to the main copying section. Here, I use a while loop, but inside the loop I manually write eight copy operations one after another. This way, the loop condition is checked only once for every eight elements instead of after each individual copy.

5. Because of this approach, the program spends less time checking conditions and more time actually copying data. Once the loop completes, all the elements from S (Source) have been successfully copied into D (Destination). Finally, I return the destination list