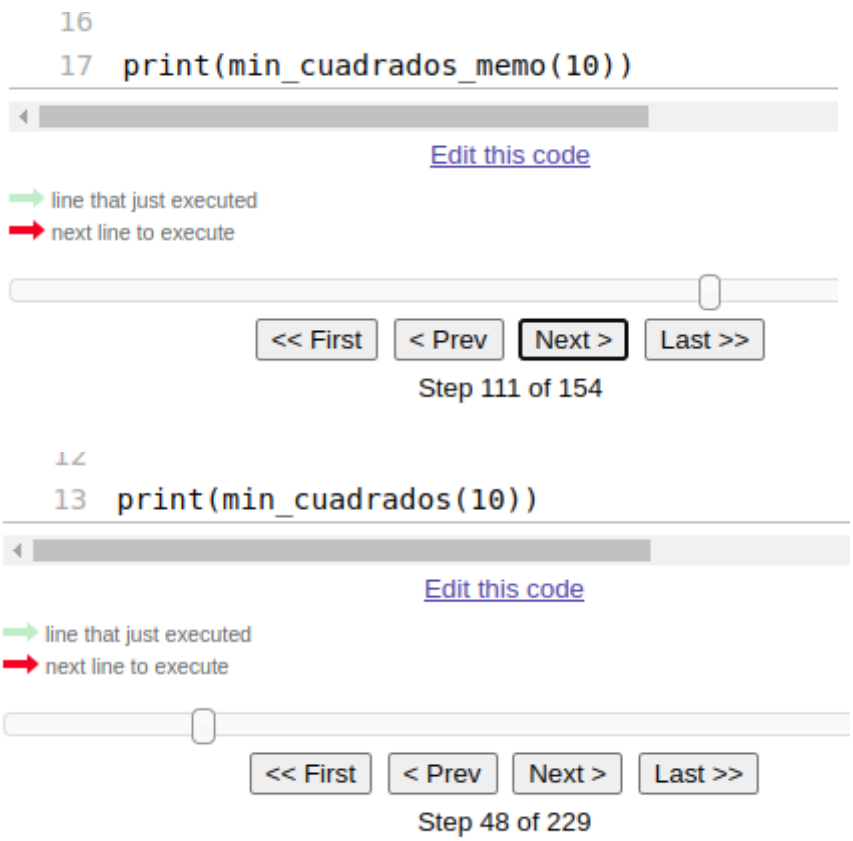
Para comenzar lo que hice fue renombrar las variables, ya que considero que el nombramiento de las variables no era el más adecuado, esto porque no se describían por si misma lo que almacenaban y podían llegar a confundir, por otro lado mejoré el algoritmo utilizando memorización, esto porque en números más grandes vamos a tener que calcular en diferentes ocasiones los mismos valores para la respuesta de un número 'n' es decir que para el n=4 siempre tendremos un valor en específico en el cual es la cantidad mínima de cuadrados que se puede representar este n, por lo tanto al almacenar esto en un diccionario cuando más adelante en otro punto de la iteración se vuelva a tener un temporal_number igual a un número que ya se calculó anteriormente lo que hace que ahorre toda la recursión nuevamente y así se reduzca el tiempo en ejecución del mismo.



En las anteriores imágenes se puede ver que con el método utilizando memorización se reducen bastante los pasos por lo que para número más grandes se hará dará solución muchísimo más rápido y permite darle más eficiencia al código.

```python
import math

def min_cuadrados_memo(number, memo_dict={}):
    if number in memo_dict:
        return memo_dict[number]
    if number <= 3:
        return number
    result = number  # En el peor de los casos, la respuesta es n (1^2 + 1^2 + ... + 1^2)

    for i in range(1, int(math.sqrt(number)) + 1):
        temporal_number = i * i
        result = min(result, 1 + min_cuadrados_memo(number - temporal_number, memo_dict))

    memo_dict[number] = result
    return result

print(min_cuadrados_memo(10))
```