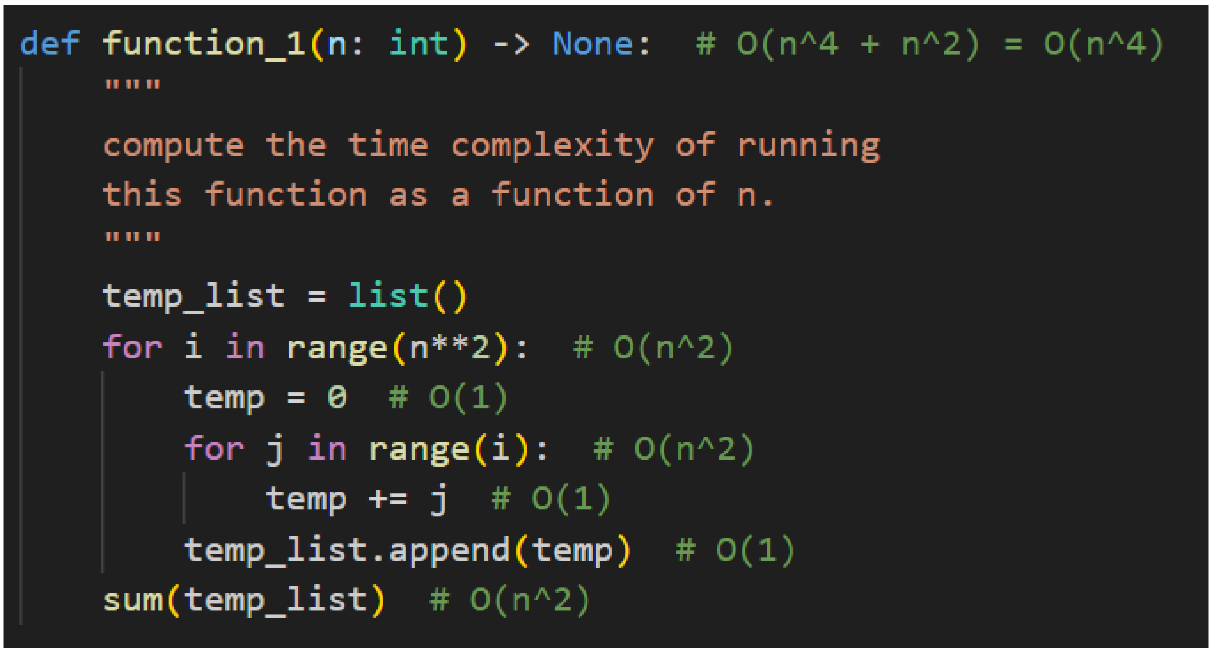
Complexities

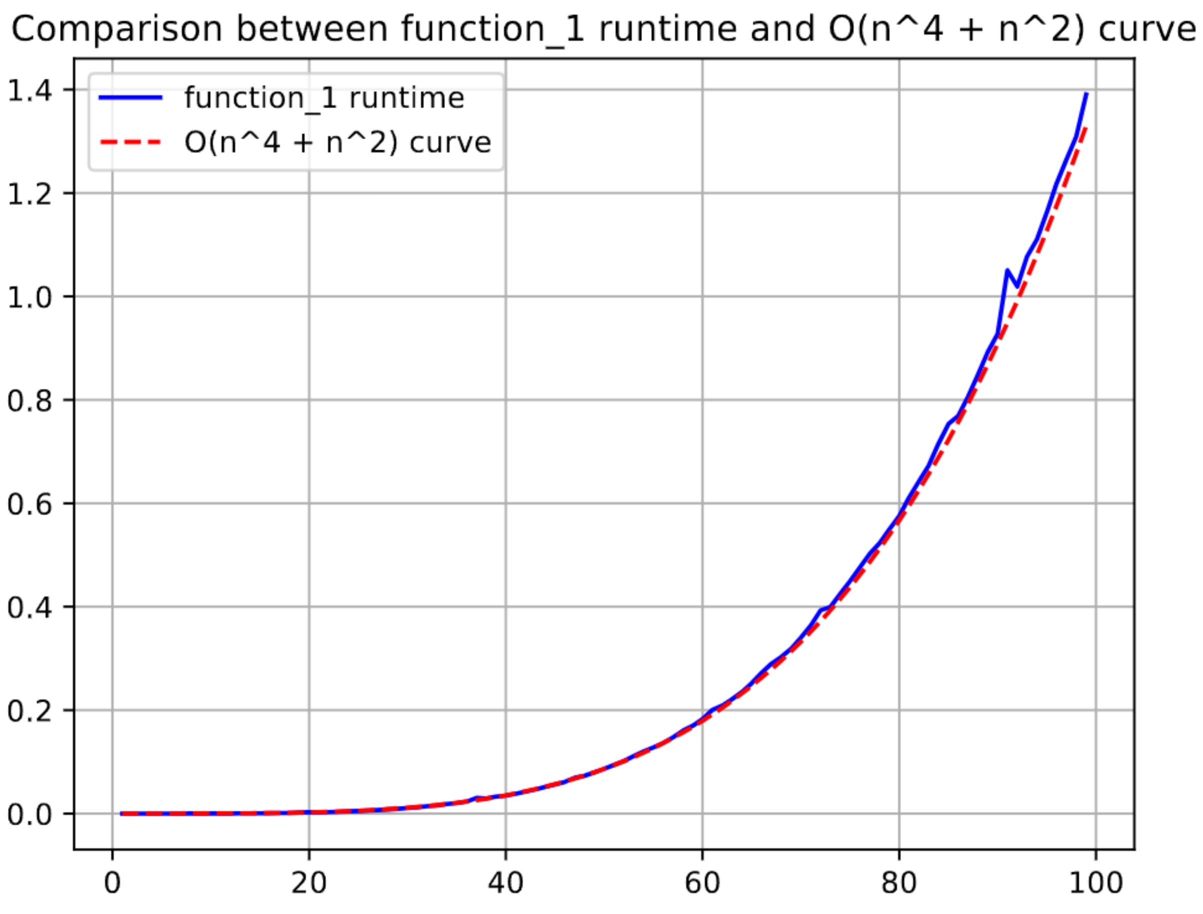
Solution function_1 :

La function_1 est en O(n^4)



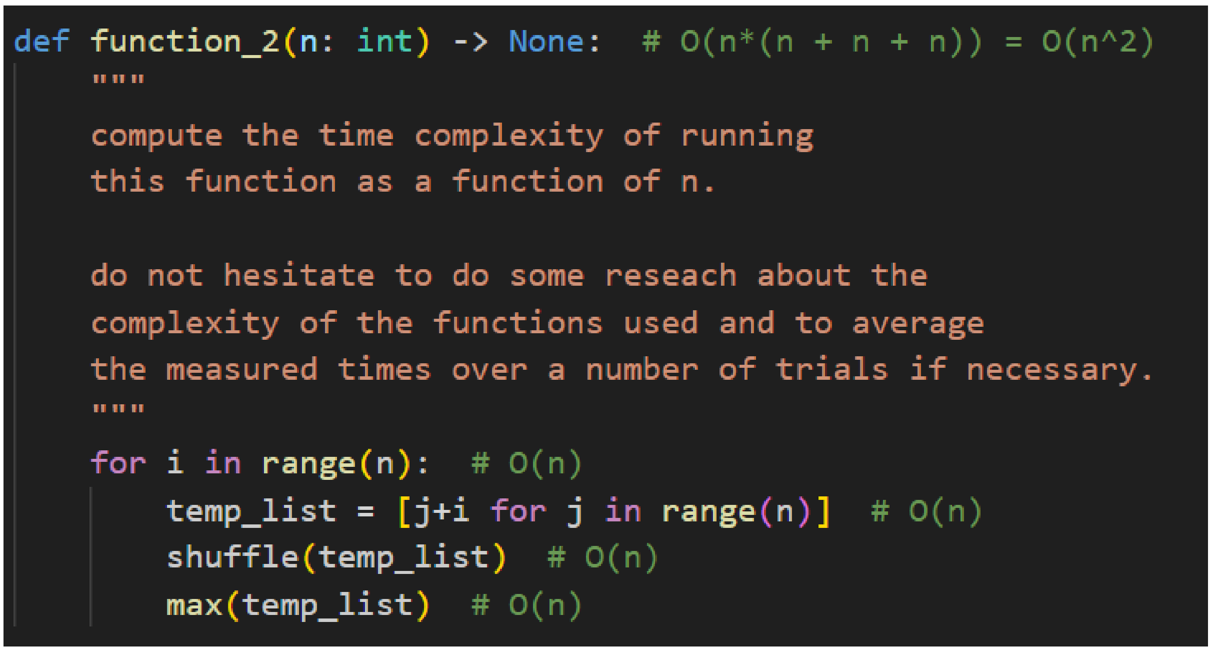
```python
def function_1(n: int) -> None:  # O(n^4 + n^2) = O(n^4)
    """
    compute the time complexity of running
    this function as a function of n.
    """
    temp_list = list()
    for i in range(n**2):  # O(n^2)
        temp = 0  # O(1)
        for j in range(i):  # O(n^2)
            temp += j  # O(1)
        temp_list.append(temp)  # O(1)
    sum(temp_list)  # O(n^2)
```

Preuve :



Solution function_2 :

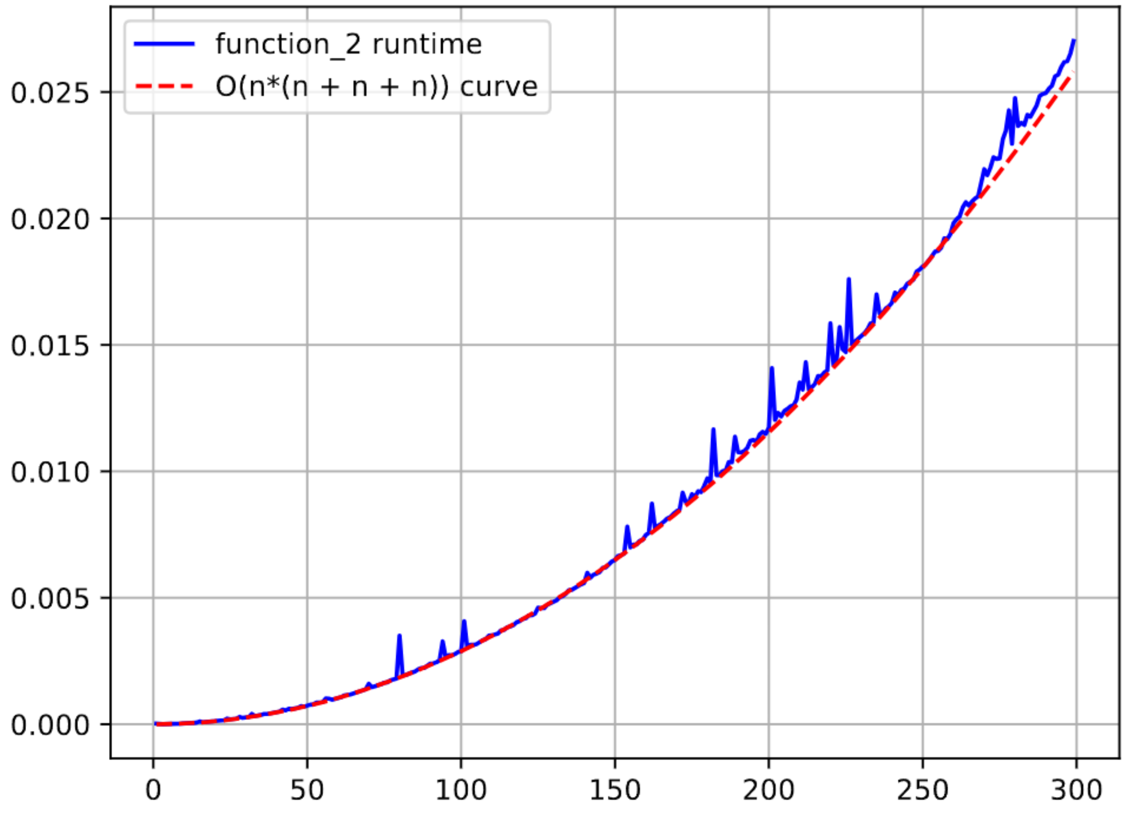
La function_2 est en O(n^2)



```python
def function_2(n: int) -> None:  # O(n*(n + n + n)) = O(n^2)
    """
    compute the time complexity of running
    this function as a function of n.

    do not hesitate to do some reseach about the
    complexity of the functions used and to average
    the measured times over a number of trials if necessary.
    """
    for i in range(n):  # O(n)
        temp_list = [j+i for j in range(n)]  # O(n)
        shuffle(temp_list)  # O(n)
        max(temp_list)  # O(n)
```

Preuve :


Comparison between function_2 runtime and O(n*(n + n + n)) curve