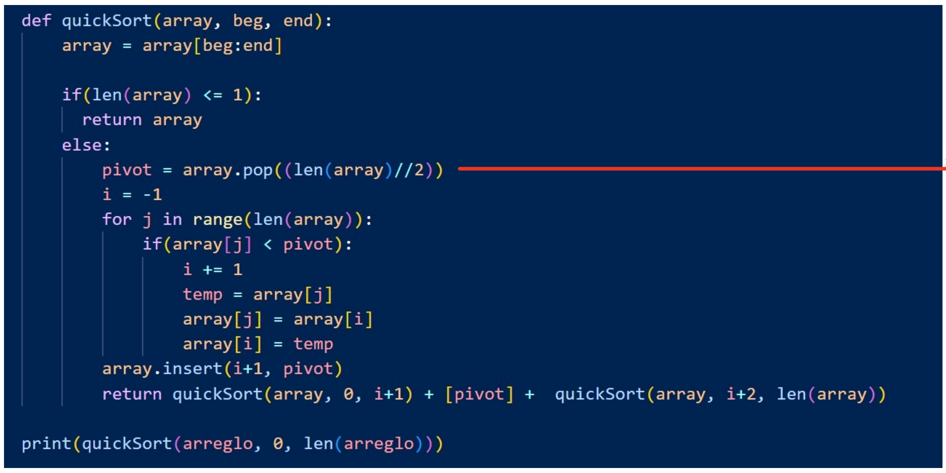
Quick Sort

Recursion

It is a sorting algorithm based on divide and conquer strategy that picks an element as a pivot and sorts the minor elements to the left of the pivot, and the greater elements to the right.

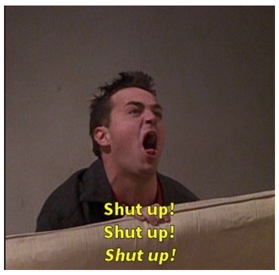


```python
def quickSort(array, beg, end):
    array = array[beg:end]

    if(len(array) <= 1):
      return array
    else:
        pivot = array.pop((len(array)//2))
        i = -1
        for j in range(len(array)):
            if(array[j] < pivot):
                i += 1
                temp = array[j]
                array[j] = array[i]
                array[i] = temp
        array.insert(i+1, pivot)
        return quickSort(array, 0, i+1) + [pivot] +  quickSort(array, i+2, len(array))

print(quickSort(arreglo, 0, len(arreglo)))
```

It does matter which pivot do we choose, depending on if it's in the middle, the beginning or the end.

In the beginning we re-define the array limits, including where it starts and where and where it ends.
The base case for our recursive algorithm will be: returning the array when there is only one or less elements left.
Right after, the algorithm chooses a (middle) pivot (and also pops it out) and starts comparing the elements in the array with it. When it finds a smaller value swaps it with the last checked element.
After checking all the array, we add the pivot in its corresponding place and call the quickSort function for the right sub-array and the left one.

Sometimes the selection it's done randomly, but the best scenario is when it's a middle element.



Time complexity:

Best case: when the pivot is in the middle.

O(nLogn)

Worst case: when the pivot is greater or less than the other elements.

O(n^2)

Average case:

O(nLogn)