In [ ]:
```
from random import randint

def create_array(size=5, max=10):
    array = [randint(0, max) for _ in range(size)]
    return array
```

In [ ]:
```
create_array() # first test run
```

In [ ]:
```
create_array() # second test run - check that next array is a different one
```

In [ ]:
```
# this function sorts a list using Insert Sort algorithm
def insert_sort(x):
    '''
    Pseudocode:
    1. First element is sorted
    2. Take the next element and insert it into a sorted list
    '''
    c = 0
    print(f'{c:3} : {x}')
    for top in range(1, len(x)):
        k = top
        while k > 0 and a[k-1] > a[k]:
            tmp = x[k]
            x[k] = x[k-1]
            x[k-1] = tmp
            k = k - 1
            c += 1
            print(f'{c:3} : {x}') # we print the list after every swap of elements
    return x
```

In [ ]:
```
a = create_array()
print (a)
b = insert_sort(a)
print (b)
```

In [ ]:
```
# this function sort a list using Choice Sort algorithm
def choice_sort(x):
    '''
    Pseudocode:
    1. Go through the list and compare every element with the first element
    2. Swap the element with the first element, if the first element is greater
    3. Repeat steps for the rest of the list
    '''
    c = 0
    print(f'{c:3} : {x}')
    for pos in range(0,len(x)-1):
        for k in range(pos+1, len(x)):
            if x[k] < x[pos]:
                tmp = x[k]
                x[k] = x[pos]
                x[pos] = tmp
                c += 1
                print(f'{c:3} : {x}') # we print the list after every operation to see the cha
    return x
```

In [ ]:
```
a = create_array()
print (a)
b = choice_sort(a)
print (b)
```

In [ ]:
```
# this function sort a list using bubble Sort algorithm
def bubble_sort(x):
    c = 0
    for bypass in range(1,len(x)):
        for k in range(0, len(x) - bypass):
            if x[k] > x[k+1]:
                tmp = x[k]
                x[k] = x[k+1]
                x[k+1] = tmp
                c += 1
                print(f'{c:3} : {x}') # we print the list after every operation to see the cha
    return x
```

In [ ]:
```
a = create_array()
print (a)
b = bubble_sort(a)
print (b)
```

In [ ]:
```
def merge(a, b):
    c = [] # final output array
    ai, bi = 0, 0
    while ai < len(a) and bi < len(b):
        if a[ai] < b[bi]:
            c.append(a[ai])
            ai += 1
        else:
            c.append(b[bi])
            bi += 1
    if ai == len(a):c.extend(b[bi:])
    else:c.extend(a[ai:])
    return c

def merge_sort(a, debug = False):
    n, half = len(a), len(a)//2 # determine number of elements and it's half value
    if n <= 1: return a # a list of zero or one element is sorted, by definition
    # left, right = merge_sort(a[:half], debug), merge_sort(a[half:], debug)
    left, right = a[:half], a[half:]
    if debug:
        print (left, right, '<=', a)
    left, right = merge_sort(a[:half], debug), merge_sort(a[half:], debug)
    return merge(left, right)
```

In [ ]:
```
a = create_array()
print(a)
s = merge_sort(a, True)
print(s)
```

In [ ]:
```
def quicksort(a, debug=False):
    n = len(a) # determine number of elements
    if n <= 1:
        return a # a list of zero or one element is sorted, by definition
    smaller, equal, larger = [], [], []
    #pivot = a[randint(0, n -1)]
    pivot = a[0]
    for x in a:
        if x < pivot :
            smaller.append(x)
        elif x == pivot:
            equal.append(x)
        else:
            larger.append(x)
    if debug:
        print (smaller, equal, larger)
    return quicksort(smaller, debug) + equal + quicksort(larger, debug)
```

In [ ]:
```
a = create_array()
print(f'unsorted: {a}')
s = quicksort(a, True)
print(f'sorted:   {s}')
```