**Comparative Programming Language Coding Project**

**Srikanth Reddy Challa - 2882709**

**Quick Sort Algorithm in Python Programming Language**

**# Function to swap two elements in a list**

def swap(a,i,j):

**# Swapping elements at index i and j**

a[i],a[j]=a[j],a[i]

**# Function to perform QuickSort on a list**

def Quicksort(a,s,e):

if s<e:

**# Partition the list and get the pivot index**

p=partition(a,s,e)

**# Recursively sort elements before pivot**

Quicksort(a,s,p-1)

**# Recursively sort elements after pivot**

Quicksort(a,p+1,e)

**# Function to partition the list**

def partition(a,s,e):

**# Choose the rightmost element as pivot**

pivot=a[e]

**# Pointer for greater element**

pindex=s

for i in range(s,e):

if a[i]<=pivot: **# If element smaller than pivot is found**

swap(a,i,pindex) **# Swap it with the greater element pointed by pIndex**

pindex +=1

swap(a,e,pindex) **# Swap the pivot element with the greater element specified by pIndex**

return pindex

**# Input list on which sorting should be performed**

a=[12,5,6,23,1,16,9]

**# Print list before sorting**

print("Before sorting {}".format(a))

**# Perform QuickSort on the list**

Quicksort(a,0,len(a)-1)

**# Print list after sorting**

print("After sorting {}".format(a))

Expected output:

A screenshot of a computer

Description automatically generated

# The scope of a variable is determined at the time of its definition based on the lexical structure of the code, hence it is static scope

# parameter passing methods – it follows "call by object reference" or "call by sharing," where changes to the mutable object are visible outside the function.

# garbage collector automatically identifies and reclaims memory occupied by objects that are no longer referenced or in use, ensuring efficient memory management in the code.

# code is executed line by line by the Python interpreter, and there is no separate compilation step as in compiled languages.

# It follows a procedural programming paradigm.

**Quick Sort Algorithm for Java Programming Language**

import java.util.\*;

public class QuickSort {

**// Main method**

public static void main(String[] args) {

**// Input Array on which sorting need to be performed**

int[] a = new int[]{ 10,3,5,12,4,23};

**//Displaying array before sorting**

System.out.println("Before sorting " + Arrays.toString(a));

**// Call quicksort method**

quicksort(a, 0, a.length-1);

**// Displaying array after quick sort**

System.out.println("After sorting " + Arrays.toString(a));

}

**// Method to swap two elements in an array**

static void swap(int[] a, int i, int j) {

int temp = a[i];

a[i] = a[j];

a[j] = temp;

}

// Quicksort method

static void quicksort(int[] a, int start, int end) {

if (start < end) {

**// get the pivot index by partitioning the array**

int pIndex = partition(a, start, end);

**// Recursively sort elements before pivot**

quicksort(a, start, pIndex - 1);

**// Recursively sort elements after pivot**

quicksort(a, pIndex + 1, end);

}

}

// Partition method

static int partition(int[] a, int start, int end) {

**// Choosing the rightmost element as pivot**

int pivot = a[end];

int pIndex = start;

for (int i = start; i < end; i++) {

**// If element smaller than pivot is found**

**// swap it with the greater element pointed by pIndex**

if (a[i] <= pivot) {

**// Swap elements at i and pIndex**

swap(a, i, pIndex);

pIndex++;

}

}

**// Swap the pivot element with the greater element specified by pIndex**

swap(a, end, pIndex);

return pIndex;

}

}

Expected output:

A screenshot of a computer

Description automatically generated

// Follows static scoping. In Java, the variable scope is determined by the block structure of the code and is defined statically at compile-time.

//The parameter passing mechanism is "call by value". When you pass a primitive type to a method, you are passing the value of the variable. When you pass an array or an object to a method, you are passing the value of the reference to that array or object.

//The garbage collector automatically identifies and reclaims memory occupied by objects that are no longer reachable or referenced.

//Both compiler and interpreter are used. The Java source code is first compiled into an intermediate form called bytecode by the Java compiler. This bytecode is then interpreted and executed by the Java Virtual Machine (JVM) at runtime.

//It is an object-oriented programming paradigm. It defines a class (**QuickSort**) with methods (**quicksort**, **partition**, and **swap**), showcasing encapsulation, abstraction, and the use of objects.