1. Python Program for Recursive Insertion Sort
2. Python Program for QuickSort
3. Python Program for Iterative Quick Sort
4. Python Program for Selection Sort
5. Python Program for Bubble Sort
6. Python Program for Merge Sort
7. Python Program for Iterative Merge Sort
8. Python Program for Heap Sort
9. Python Program for Counting Sort
10. Python Program for ShellSort

Solve

Here's the Python code for each of the sorting algorithms you requested:

### 1. Python Program for Recursive Insertion Sort

```python

def insertion\_sort\_recursive(arr, n):

if n <= 1:

return

# Sort the first n-1 elements

insertion\_sort\_recursive(arr, n - 1)

# Insert the nth element in its correct position

last = arr[n - 1]

j = n - 2

while j >= 0 and arr[j] > last:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = last

# Example usage

arr = [12, 11, 13, 5, 6]

insertion\_sort\_recursive(arr, len(arr))

print("Sorted array:", arr)

```

### 2. Python Program for QuickSort

```python

def partition(arr, low, high):

pivot = arr[high]

i = low - 1

for j in range(low, high):

if arr[j] < pivot:

i += 1

arr[i], arr[j] = arr[j], arr[i]

arr[i + 1], arr[high] = arr[high], arr[i + 1]

return i + 1

def quicksort(arr, low, high):

if low < high:

pi = partition(arr, low, high)

quicksort(arr, low, pi - 1)

quicksort(arr, pi + 1, high)

# Example usage

arr = [10, 7, 8, 9, 1, 5]

quicksort(arr, 0, len(arr) - 1)

print("Sorted array:", arr)

```

### 3. Python Program for Iterative Quick Sort

```python

def partition(arr, low, high):

pivot = arr[high]

i = low - 1

for j in range(low, high):

if arr[j] < pivot:

i += 1

arr[i], arr[j] = arr[j], arr[i]

arr[i + 1], arr[high] = arr[high], arr[i + 1]

return i + 1

def quick\_sort\_iterative(arr):

size = len(arr)

stack = [0] \* size

top = -1

top += 1

stack[top] = 0

top += 1

stack[top] = size - 1

while top >= 0:

high = stack[top]

top -= 1

low = stack[top]

top -= 1

pi = partition(arr, low, high)

if pi - 1 > low:

top += 1

stack[top] = low

top += 1

stack[top] = pi - 1

if pi + 1 < high:

top += 1

stack[top] = pi + 1

top += 1

stack[top] = high

# Example usage

arr = [4, 3, 2, 10, 12, 1, 5, 6]

quick\_sort\_iterative(arr)

print("Sorted array:", arr)

```

### 4. Python Program for Selection Sort

```python

def selection\_sort(arr):

for i in range(len(arr)):

min\_idx = i

for j in range(i + 1, len(arr)):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

# Example usage

arr = [64, 25, 12, 22, 11]

selection\_sort(arr)

print("Sorted array:", arr)

```

### 5. Python Program for Bubble Sort

```python

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

for j in range(0, n - i - 1):

if arr[j] > arr[j + 1]:

arr[j], arr[j + 1] = arr[j + 1], arr[j]

# Example usage

arr = [64, 34, 25, 12, 22, 11, 90]

bubble\_sort(arr)

print("Sorted array:", arr)

```

### 6. Python Program for Merge Sort

```python

def merge\_sort(arr):

if len(arr) > 1:

mid = len(arr) // 2

L = arr[:mid]

R = arr[mid:]

merge\_sort(L)

merge\_sort(R)

i = j = k = 0

while i < len(L) and j < len(R):

if L[i] < R[j]:

arr[k] = L[i]

i += 1

else:

arr[k] = R[j]

j += 1

k += 1

while i < len(L):

arr[k] = L[i]

i += 1

k += 1

while j < len(R):

arr[k] = R[j]

j += 1

k += 1

# Example usage

arr = [12, 11, 13, 5, 6, 7]

merge\_sort(arr)

print("Sorted array:", arr)

```

### 7. Python Program for Iterative Merge Sort

```python

def merge(arr, temp\_arr, left, mid, right):

i = left

j = mid + 1

k = left

while i <= mid and j <= right:

if arr[i] <= arr[j]:

temp\_arr[k] = arr[i]

i += 1

else:

temp\_arr[k] = arr[j]

j += 1

k += 1

while i <= mid:

temp\_arr[k] = arr[i]

i += 1

k += 1

while j <= right:

temp\_arr[k] = arr[j]

j += 1

k += 1

for i in range(left, right + 1):

arr[i] = temp\_arr[i]

def merge\_sort\_iterative(arr):

current\_size = 1

n = len(arr)

temp\_arr = [0] \* n

while current\_size < n:

left = 0

while left < n - 1:

mid = min(left + current\_size - 1, n - 1)

right = min((2 \* current\_size) + left - 1, n - 1)

merge(arr, temp\_arr, left, mid, right)

left += 2 \* current\_size

current\_size \*= 2

# Example usage

arr = [12, 11, 13, 5, 6, 7]

merge\_sort\_iterative(arr)

print("Sorted array:", arr)

```

### 8. Python Program for Heap Sort

```python

def heapify(arr, n, i):

largest = i

l = 2 \* i + 1

r = 2 \* i + 2

if l < n and arr[i] < arr[l]:

largest = l

if r < n and arr[largest] < arr[r]:

largest = r

if largest != i:

arr[i], arr[largest] = arr[largest], arr[i]

heapify(arr, n, largest)

def heap\_sort(arr):

n = len(arr)

for i in range(n // 2 - 1, -1, -1):

heapify(arr, n, i)

for i in range(n - 1, 0, -1):

arr[i], arr[0] = arr[0], arr[i]

heapify(arr, i, 0)

# Example usage

arr = [12, 11, 13, 5, 6, 7]

heap\_sort(arr)

print("Sorted array:", arr)

```

### 9. Python Program for Counting Sort

```python

def counting\_sort(arr):

max\_val = max(arr)

m = max\_val + 1

count = [0] \* m

for a in arr:

count[a] += 1

i = 0

for a in range(m):

for c in range(count[a]):

arr[i] = a

i += 1

# Example usage

arr = [4, 2, 2, 8, 3, 3, 1]

counting\_sort(arr)

print("Sorted array:", arr)

```

### 10. Python Program for ShellSort

```python

def shell\_sort(arr):

n = len(arr)

gap = n // 2

while gap > 0:

for i in range(gap, n):

temp = arr[i]

j = i

while j >= gap and arr[j - gap] > temp:

arr[j] = arr[j - gap]

j -= gap

arr[j] = temp

gap //= 2

# Example usage

arr = [12, 34, 54, 2, 3]

shell\_sort(arr)

print("Sorted array:", arr)

```

Each code snippet provides an example of the specific sorting algorithm requested. You can run these programs in a Python environment to see how they sort an array of numbers.