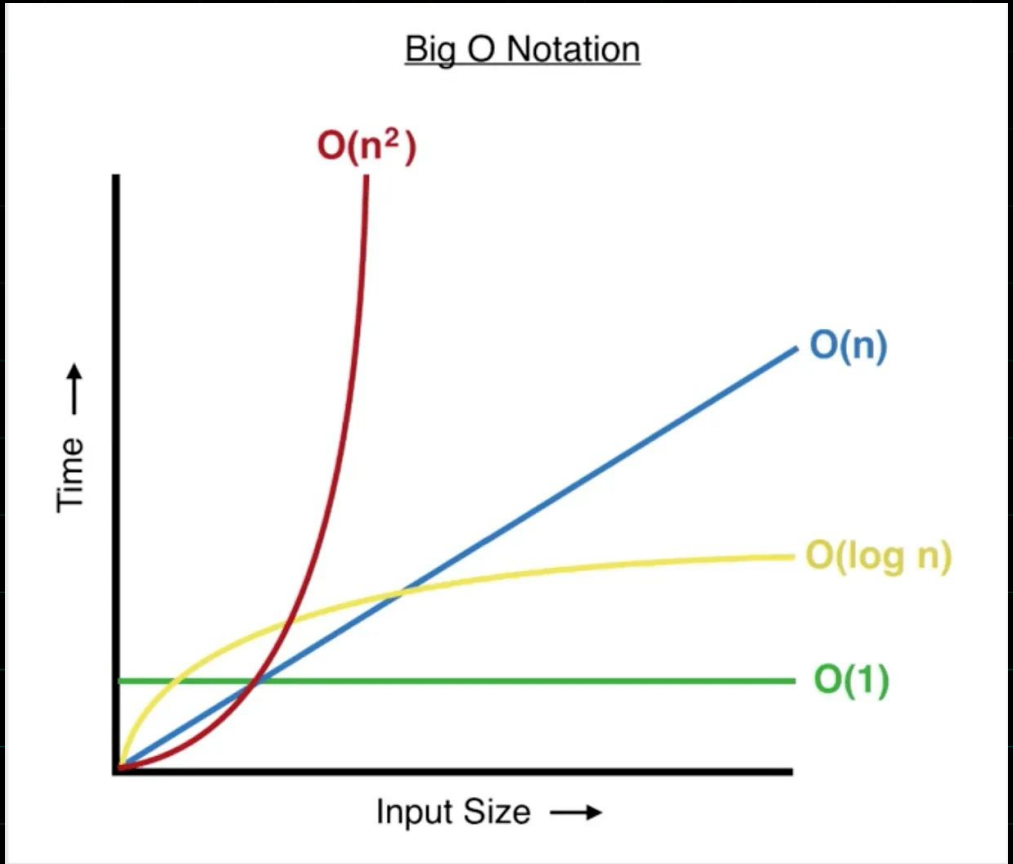
Big O; Searching, Sorting

Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity



Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity

Problem 1.py

```python
data_array = [1,2,3,4,5]
# Problem:
# How many items are there in the sorted array
no_of_items = 0
for item in data_array:
  no_of_items = no_of_items + 1

print('Number of items: ', no_of_items)
```

Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity

Problem 1.py

```python
data_array = [1,2,3,4,5,6,7,8,9,10]
# Problem:
# How many items are there in the sorted array
no_of_items = 0
for item in data_array:
  no_of_items = no_of_items + 1

print('Number of items: ', no_of_items)
```

Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity

Problem 1.py

```python
data_array = [1,2,3,4,5]
# Problem:
# How many items are there in the sorted array
no_of_items = data_array[-1]

print('Number of items: ', no_of_items)
```

Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity

Problem 1.py

```python
data_array = [1,2,3,4,5,6,7,8,9,10]
# Problem:
# How many items are there in the sorted array
no_of_items = data_array[-1]

print('Number of items: ', no_of_items)
```


Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity

O(1) ← O(2), O(100), O(1000)

O(n) ← O(2n), O(10n), O(n/2), O(2n + 100),

O(logn) ← O(3logn), O(4logn + 3n + 1)

O(nlogn) - Usually in sorting

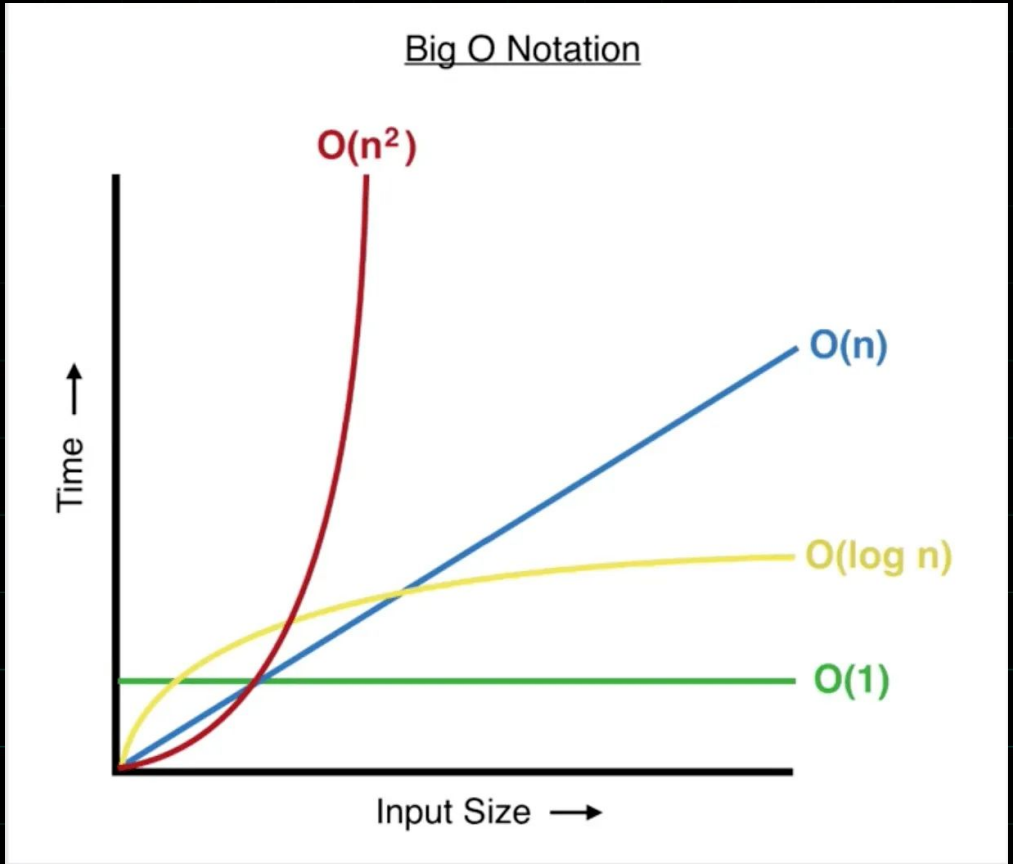
$O(n^2)$ - For every element of input perform n x n instructions

$O(n^3)$ - For every element of input perform n x n x n instructions

Strictly private and confidential

BIG O - Time Space Complexity

Time Complexity



Strictly private and confidential

BIG O - Time Space Complexity

Space Complexity

Problem 1.py

```python
numbers = [1,10,4,11] # Input
# calculate the sum of the integers
result = 0
for num in numbers:
  result = result + num
print(result)
```

Strictly private and confidential

BIG O - Time Space Complexity

Space Complexity

Problem 1.py

```python
numbers = [1,10,4,11,11,20,8,2] # Input
# calculate the sum of the integers
result = 0
for num in numbers:
  result = result + num
print(result)
```

Strictly private and confidential

BIG O - Time Space Complexity

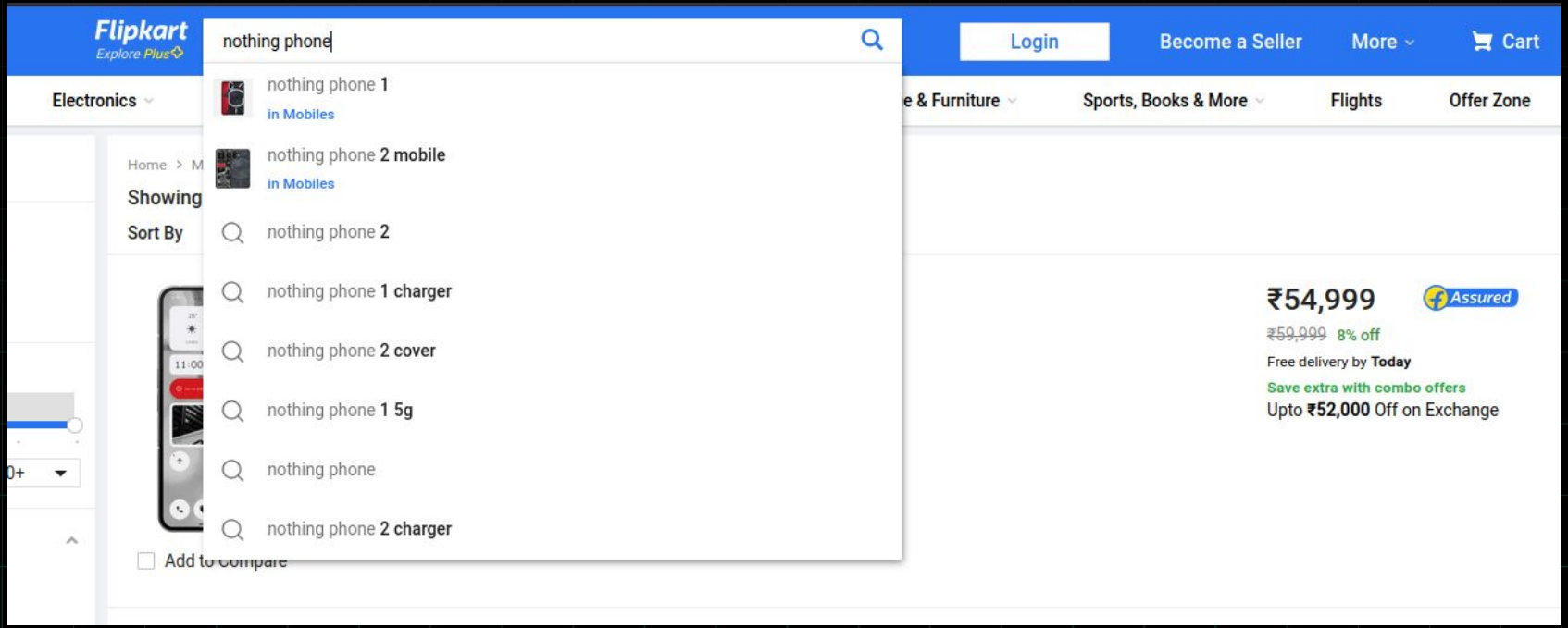
Space Complexity

Untitled-1

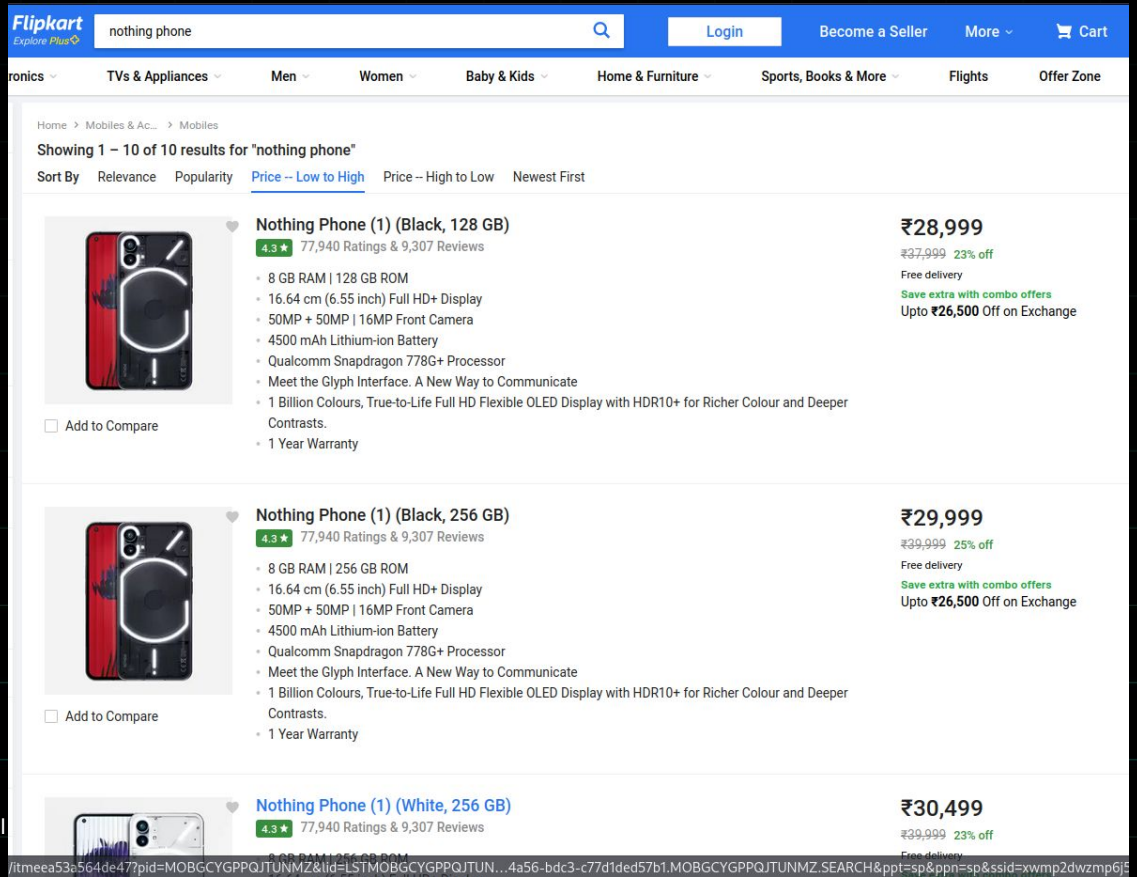
```
num = 10
# create an array contianig 10 elements
newList = []
for i in range(num):
  newList.append(i)
print(newList)
```

Strictly private and confidential

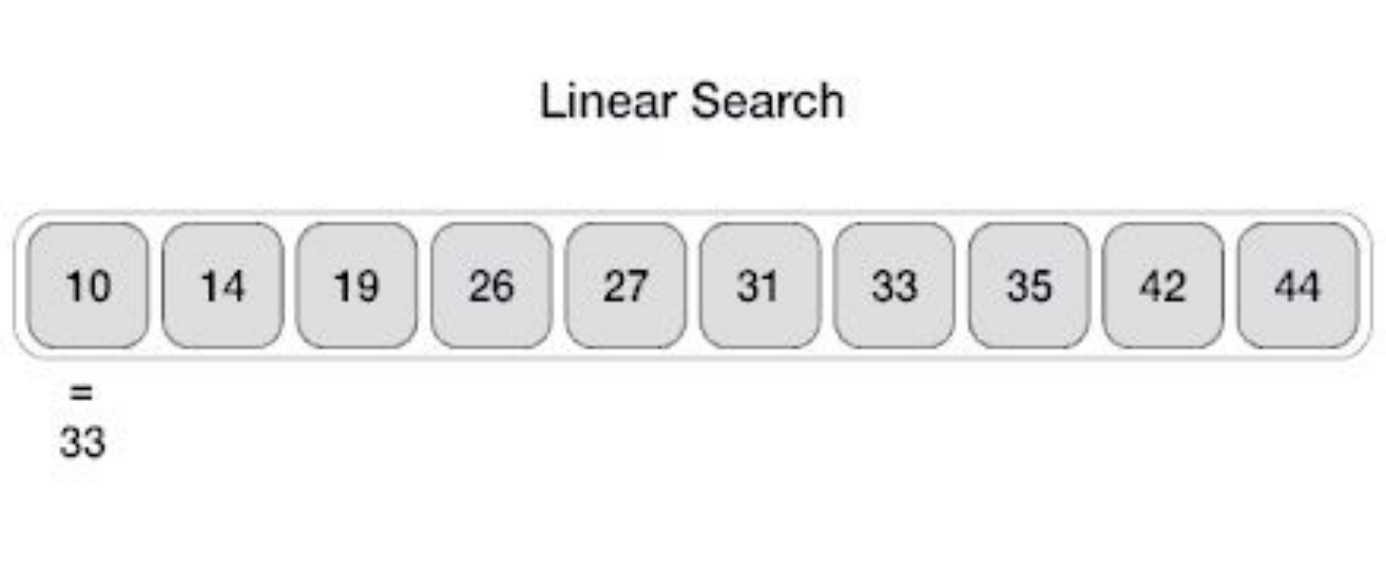
Searching & Sorting



Searching & Sorting



Linear Search



Strictly private and confidential

Linear Search

Untitled-1

```python
def linear_search(arr, target):
  for i in range(len(arr)):
    if arr[i] == target:
      return i
  return -1

# Example usage
arr = [10, 15, 20, 5, 30]
target = 30

result = linear_search(arr, target)
if result != -1:
  print("Element found at index", result)
else:
  print("Element not found")
```

Strictly private and confidential

Searching - Binary Search

Binary Search

Searching for...

41

1	2	7	8	22	28	41	58	67	71	94
0	1	2	3	4	5	6	7	8	9	10

Strictly private and confidential

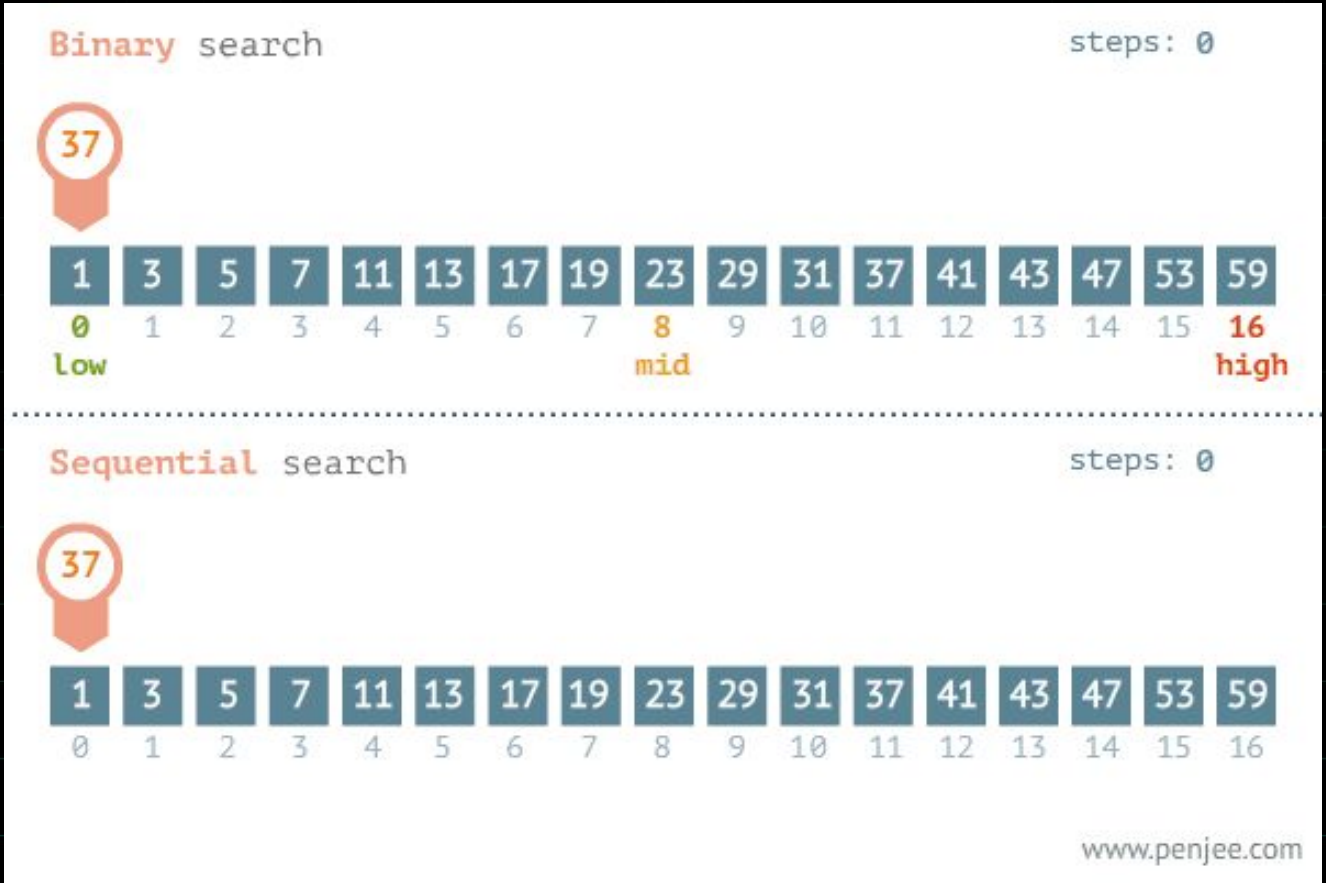
Searching - Binary Search

Untitled-1

```python
def binary_search(arr, x):
  left = 0
  right = len(arr) - 1

  while left <= right:
    mid = (left + right) // 2

    if arr[mid] == x:
      return mid

    if arr[mid] < x:
      left = mid + 1
    else:
      right = mid - 1

  return -1

# Example usage
arr = [2, 4, 6, 8, 10, 12]
x = 10

result = binary_search(arr, x)

if result != -1:
  print("Element found at index", result)
else:
  print("Element not found")
```

Strictly private and confidential

Searching - Linear vs Binary Search



Sorting - Bubble Sort

Strictly private and confidential

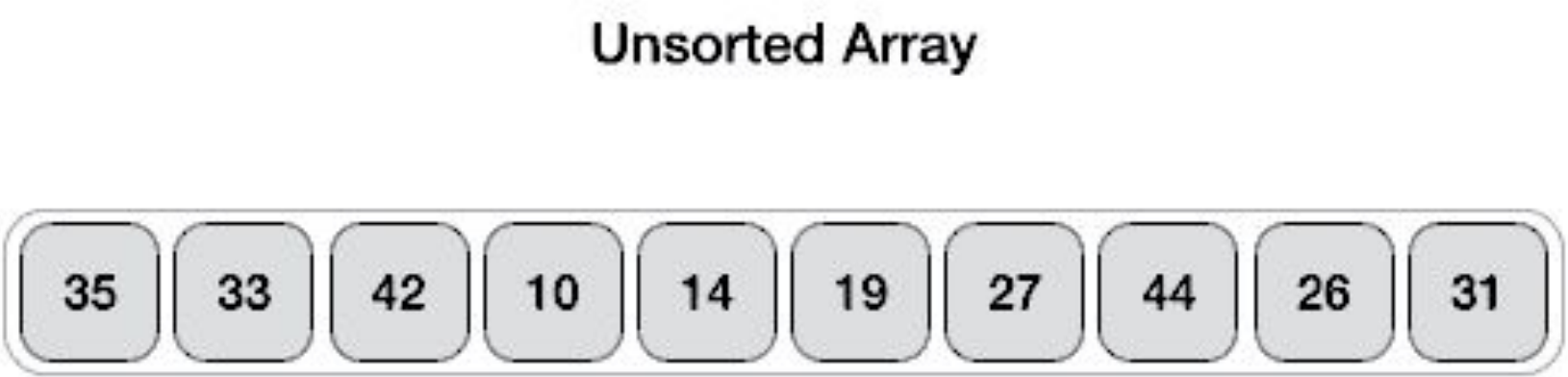
Sorting - Bubble Sort

Untitled-1

```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:")
for i in range(len(arr)):
  print(arr[i])
```

Strictly private and confidential

Sorting - Quick Sort



Strictly private and confidential

Sorting - Quick Sort

Untitled-1

```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]
n = len(arr)
quicksort(arr, 0, n-1)
print(arr)
```

Strictly private and confidential

Continuous Practical Assignment I

- **Task Automation using Python**

Strictly private and confidential

FAQs

Thank you!

douglas.cst@rub.edu.bt

Strictly private and confidential