Python Concepts Summary

### Merge Sort:  
- \*\*Definition\*\*: Merge Sort is a stable, divide-and-conquer sorting algorithm.  
- \*\*Process\*\*: It splits the array into halves, recursively sorts them, and then merges the sorted halves.  
- \*\*Time Complexity\*\*: The average-case time complexity is O(n log n).  
- \*\*Python Code\*\*:  
 ```python  
 def merge\_sort(arr):  
 if len(arr) > 1:  
 mid = len(arr) // 2  
 left\_half = arr[:mid]  
 right\_half = arr[mid:]  
  
 merge\_sort(left\_half)  
 merge\_sort(right\_half)  
 merge(arr, left\_half, right\_half)  
  
 def merge(arr, left, right):  
 i = j = k = 0  
 while i < len(left) and j < len(right):  
 if left[i] < right[j]:  
 arr[k] = left[i]  
 i += 1  
 else:  
 arr[k] = right[j]  
 j += 1  
 k += 1  
 while i < len(left):  
 arr[k] = left[i]  
 i += 1  
 k += 1  
 while j < len(right):  
 arr[k] = right[j]  
 j += 1  
 k += 1  
  
 # Example usage:  
 my\_list = [64, 34, 25, 12, 22, 11, 90]  
 merge\_sort(my\_list)  
 print("Sorted array:", my\_list)  
 ```  
  
### Quick Sort:  
- \*\*Definition\*\*: Quick Sort is a divide-and-conquer sorting algorithm.  
- \*\*Process\*\*: It selects a pivot element, partitions the array into elements less than and greater than the pivot, and recursively sorts the partitions.  
- \*\*Time Complexity\*\*: The average-case time complexity is O(n log n); worst-case is O(n²).  
- \*\*Python Code\*\*:  
 ```python  
 def quick\_sort(arr):  
 if len(arr) <= 1:  
 return arr  
  
 pivot = arr[len(arr) // 2]  
 left = [x for x in arr if x < pivot]  
 middle = [x for x in arr if x == pivot]  
 right = [x for x in arr if x > pivot]  
  
 return quick\_sort(left) + middle + quick\_sort(right)  
  
 # Example usage:  
 my\_list = [64, 34, 25, 12, 22, 11, 90]  
 sorted\_list = quick\_sort(my\_list)  
 print("Sorted array:", sorted\_list)  
 ```  
  
### Decorators in Python:  
- \*\*Definition\*\*: Decorators are functions that modify the behavior of other functions or methods.  
- \*\*Usage\*\*: Applied using the `@decorator` syntax.  
- \*\*Example\*\*:  
 ```python  
 def my\_decorator(func):  
 def wrapper():  
 print("Before the function.")  
 func()  
 print("After the function.")  
 return wrapper  
  
 @my\_decorator  
 def say\_hello():  
 print("Hello!")  
  
 say\_hello()  
 ```  
  
### Pandas Library in Python:  
- \*\*Definition\*\*: Pandas is an open-source data manipulation and analysis library.  
- \*\*Key Structures\*\*:  
 - \*\*DataFrame\*\*: A 2-dimensional labeled data structure with columns of potentially different types.  
 - \*\*Series\*\*: A 1-dimensional labeled array.  
- \*\*Operations\*\*: Supports filtering, grouping, merging, etc.  
- \*\*Examples\*\*:  
 ```python  
 import pandas as pd  
  
 # Creating a DataFrame  
 data = {'Name': ['Alice', 'Bob', 'Charlie'],  
 'Age': [25, 30, 35],  
 'City': ['New York', 'San Francisco', 'Los Angeles']}  
 df = pd.DataFrame(data)  
  
 # Creating a Series  
 ages = pd.Series([25, 30, 35], name='Age')  
  
 # Filtering data  
 df\_young = df[df['Age'] < 30]  
  
 # Grouping data  
 grouped\_data = df.groupby('City')['Age'].mean()  
 ```