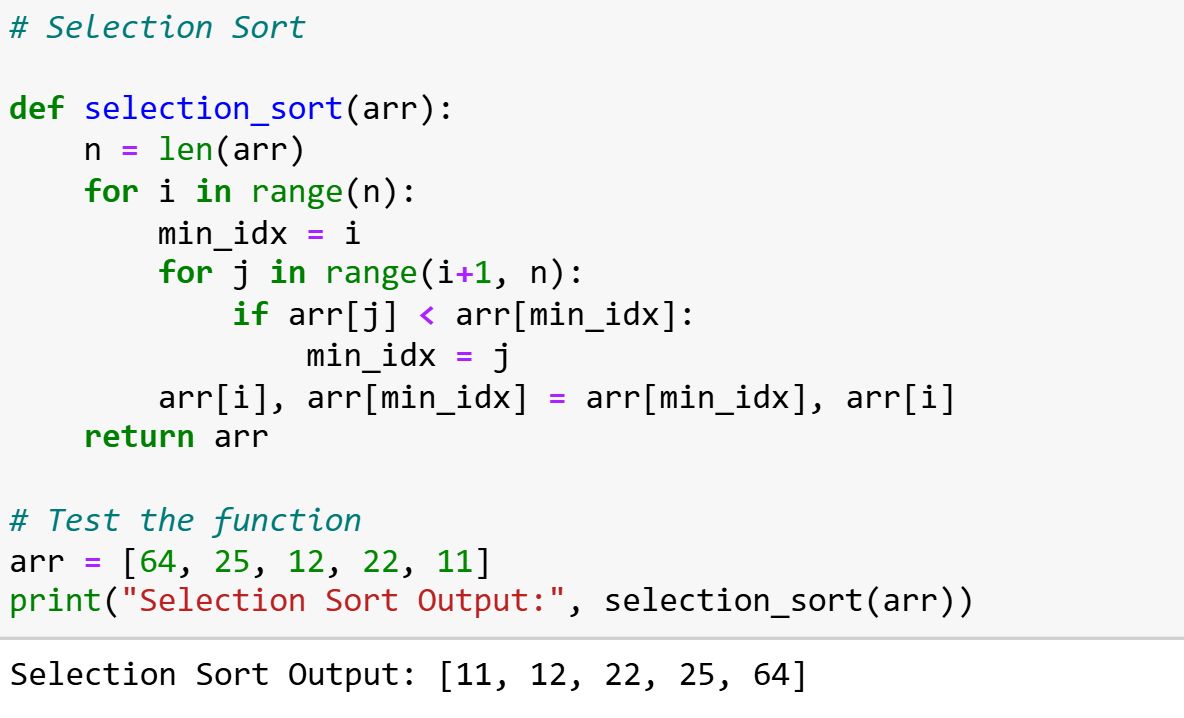
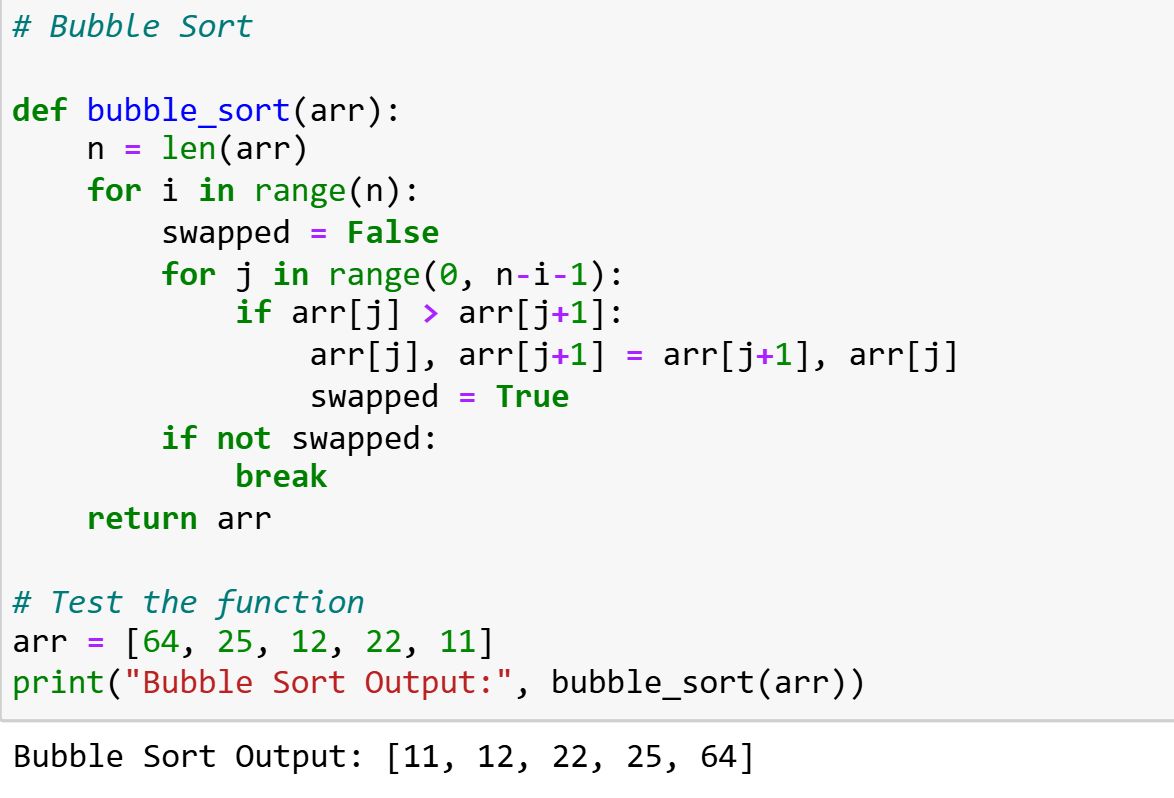
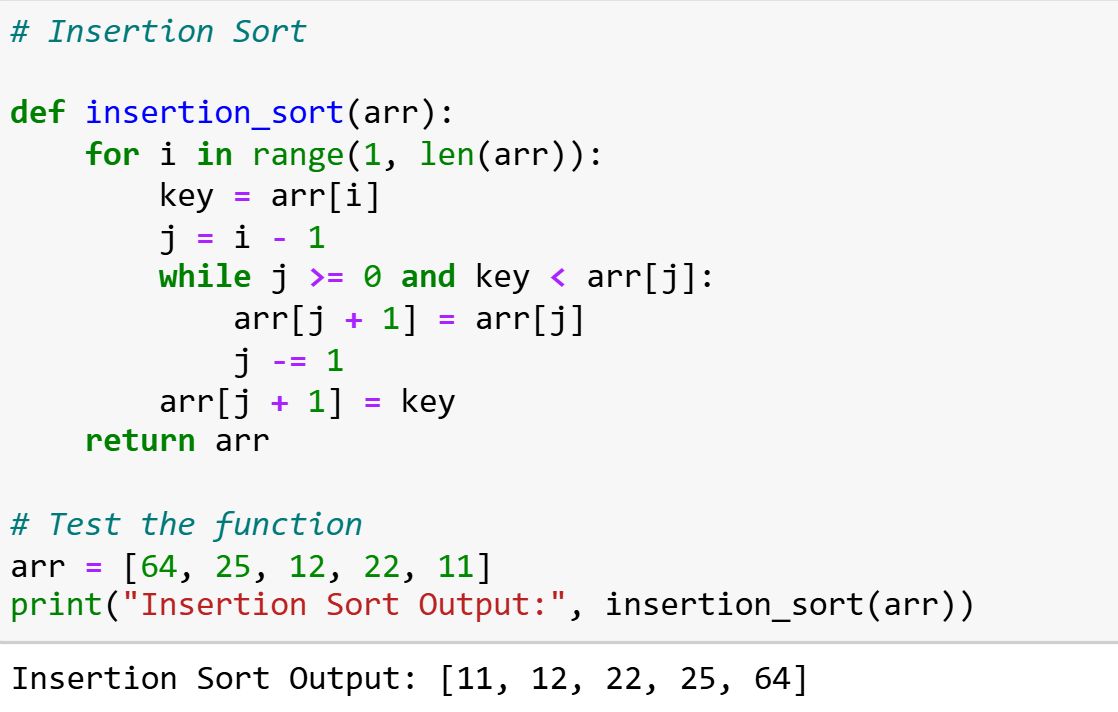
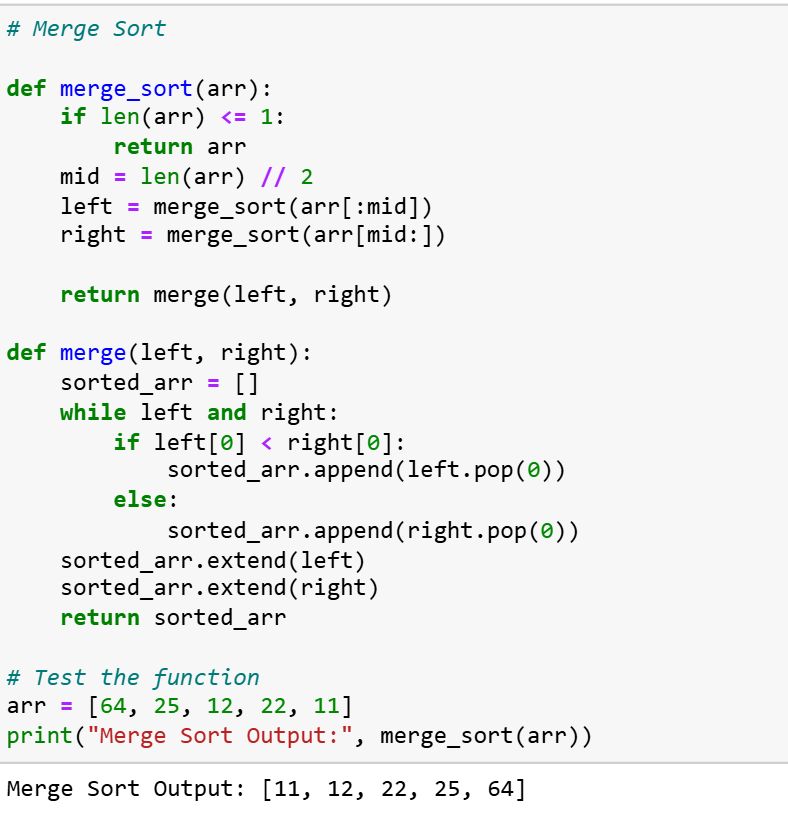
𝐒𝐨𝐫𝐭𝐢𝐧𝐠 𝐀𝐥𝐠𝐨𝐫𝐢𝐭𝐡𝐦𝐬 𝐢𝐧 𝐩𝐲𝐭𝐡𝐨𝐧

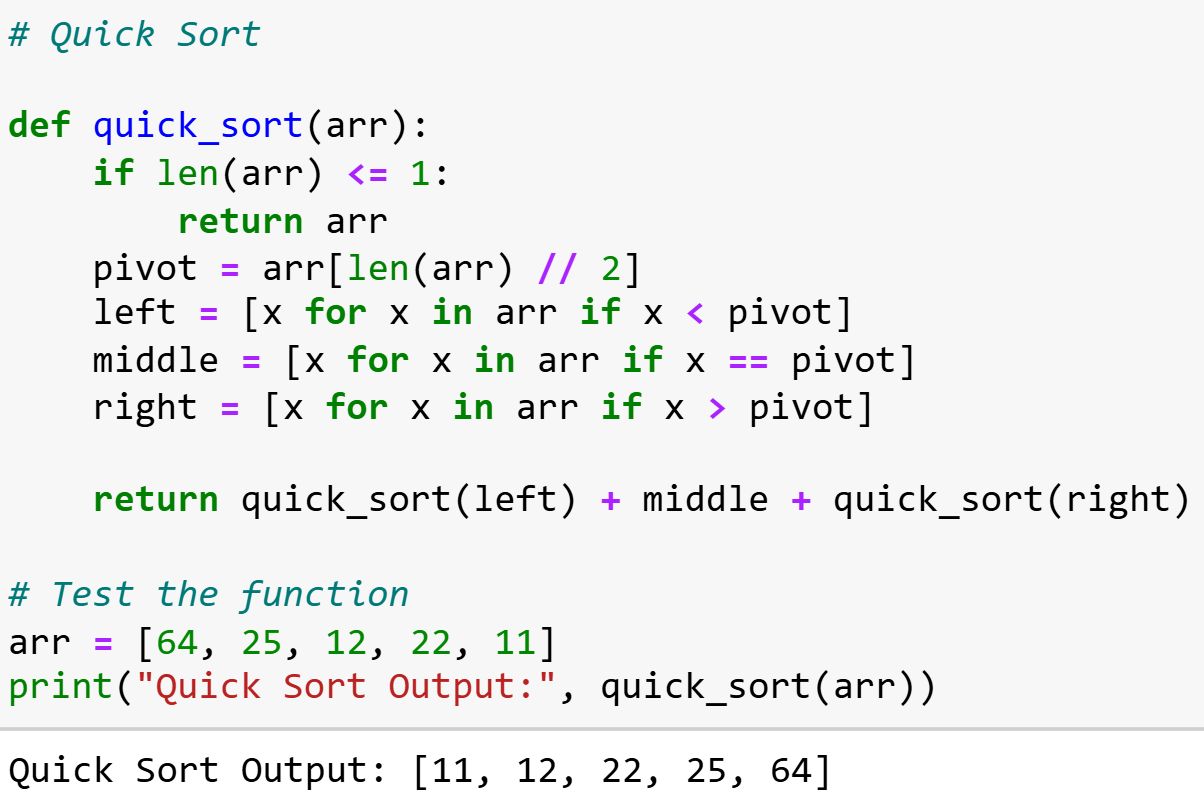
1. 𝐒𝐞𝐥𝐞𝐜𝐭𝐢𝐨𝐧 𝐒𝐨𝐫𝐭  
     
   Selection sort works by repeatedly finding the minimum element from the unsorted portion of the list and swapping it with the first unsorted element.   
     
   𝐓𝐢𝐦𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲:  
   Best, Worst, and Average: O(n²)  
   Explanation: It uses two nested loops: one to iterate over each element and one to find the minimum element.  
     
   𝐒𝐩𝐚𝐜𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲: O(1)  
   Explanation: The algorithm sorts the list in place, requiring constant extra space.  
     
   2. 𝐁𝐮𝐛𝐛𝐥𝐞 𝐒𝐨𝐫𝐭  
     
   Bubble sort works by repeatedly stepping through the list, comparing adjacent elements, and swapping them if they are in the wrong order. This process is repeated until the list is sorted.  
     
   𝐓𝐢𝐦𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲:  
   Best: O(n) (if the list is already sorted)  
   Worst and Average: O(n²)  
   Explanation: The worst-case occurs when the list is in reverse order, requiring n-1 comparisons for each of the n elements.  
     
   𝐒𝐩𝐚𝐜𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲: O(1)  
   Explanation: The sorting is done in place, so no additional space is required.  
     
   3. 𝐈𝐧𝐬𝐞𝐫𝐭𝐢𝐨𝐧 𝐒𝐨𝐫𝐭  
     
   Insertion sort builds the final sorted array one element at a time. It takes each element from the unsorted portion and inserts it into its correct position in the sorted portion.  
     
   𝐓𝐢𝐦𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲:  
   Best: O(n) (if the list is already sorted)  
   Worst and Average: O(n²)  
   Explanation: The worst case occurs when the list is sorted in reverse order, resulting in the maximum number of shifts.  
     
   𝐒𝐩𝐚𝐜𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲: O(1)  
   Explanation: The sorting is done in place without requiring additional space.  
     
   4. 𝐌𝐞𝐫𝐠𝐞 𝐒𝐨𝐫𝐭  
     
   This is divide-and-conquer algorithm that divides the array into two halves, recursively sorts them, and then merges the two sorted halves. This is done by dividing the array until each subarray has a single element and then merging them in sorted order.  
     
   𝐓𝐢𝐦𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲:  
   Best, Worst, and Average: O(n log n)  
   Explanation: The array is split into two halves recursively, and then merging them requires linear time.  
     
   𝐒𝐩𝐚𝐜𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲: O(n)  
   Explanation: An auxiliary array is required for merging.  
     
   5. 𝐐𝐮𝐢𝐜𝐤 𝐒𝐨𝐫𝐭  
     
   This is divide-and-conquer algorithm that works by selecting a pivot element, partitioning the array into two parts: elements less than the pivot and elements greater than the pivot. It recursively sorts the two parts.  
     
   𝐓𝐢𝐦𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲:  
   Best and Average: O(n log n)  
   Worst: O(n²) (when the pivot element is always the smallest or largest element)  
   Explanation: The worst case occurs when the pivot divides the array into unbalanced partitions, requiring n² comparisons.  
     
   𝐒𝐩𝐚𝐜𝐞 𝐂𝐨𝐦𝐩𝐥𝐞𝐱𝐢𝐭𝐲: O(log n)  
   Explanation: The space complexity comes from the recursive stack space, which is O(log n) in the best case (balanced partitions) and O(n) in the worst case.

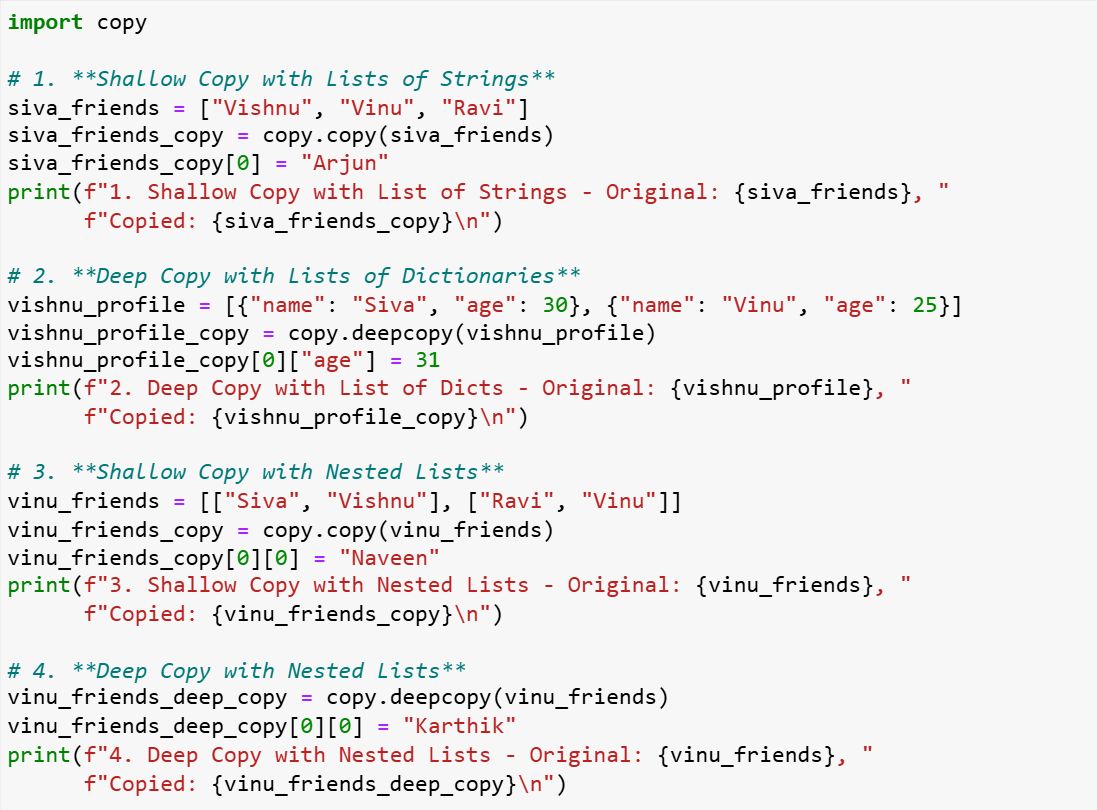


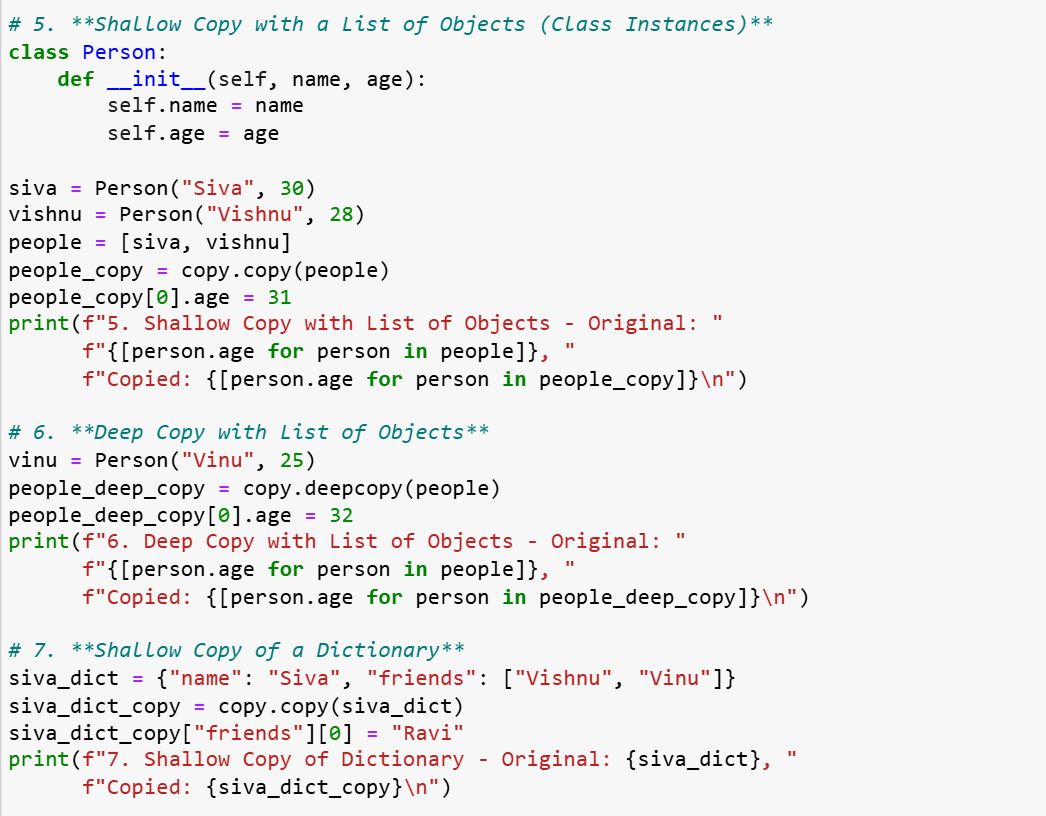


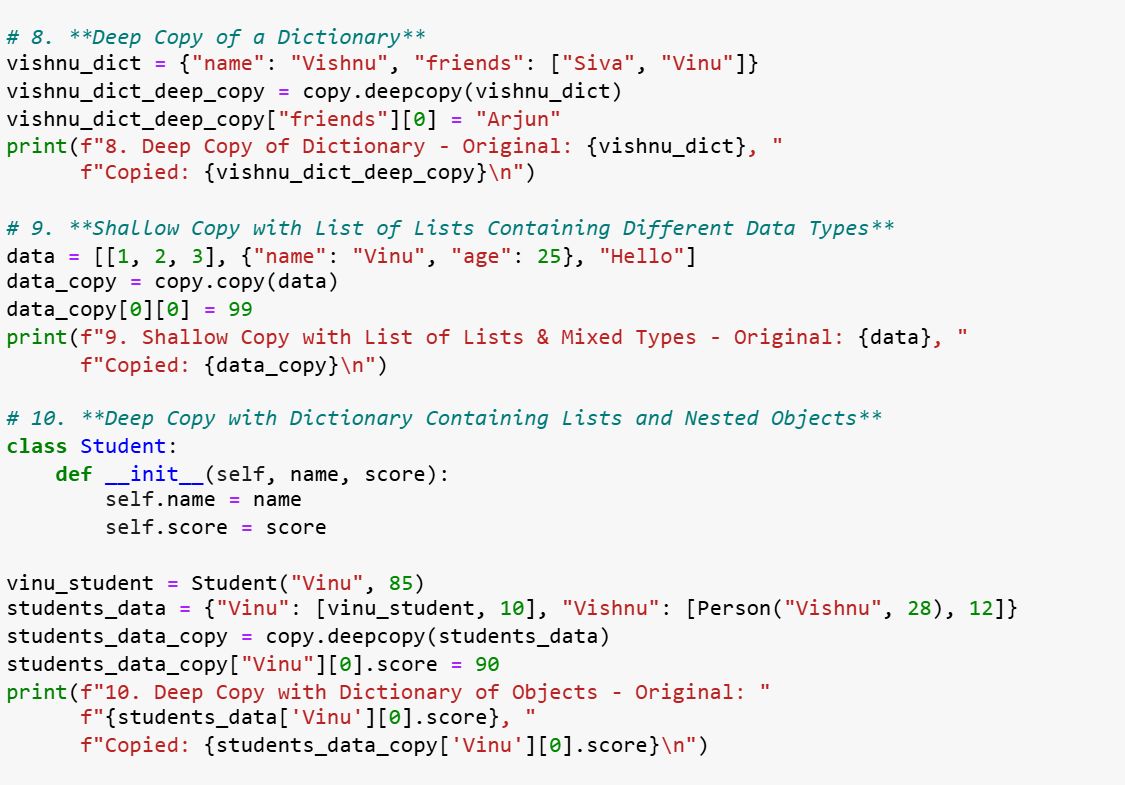


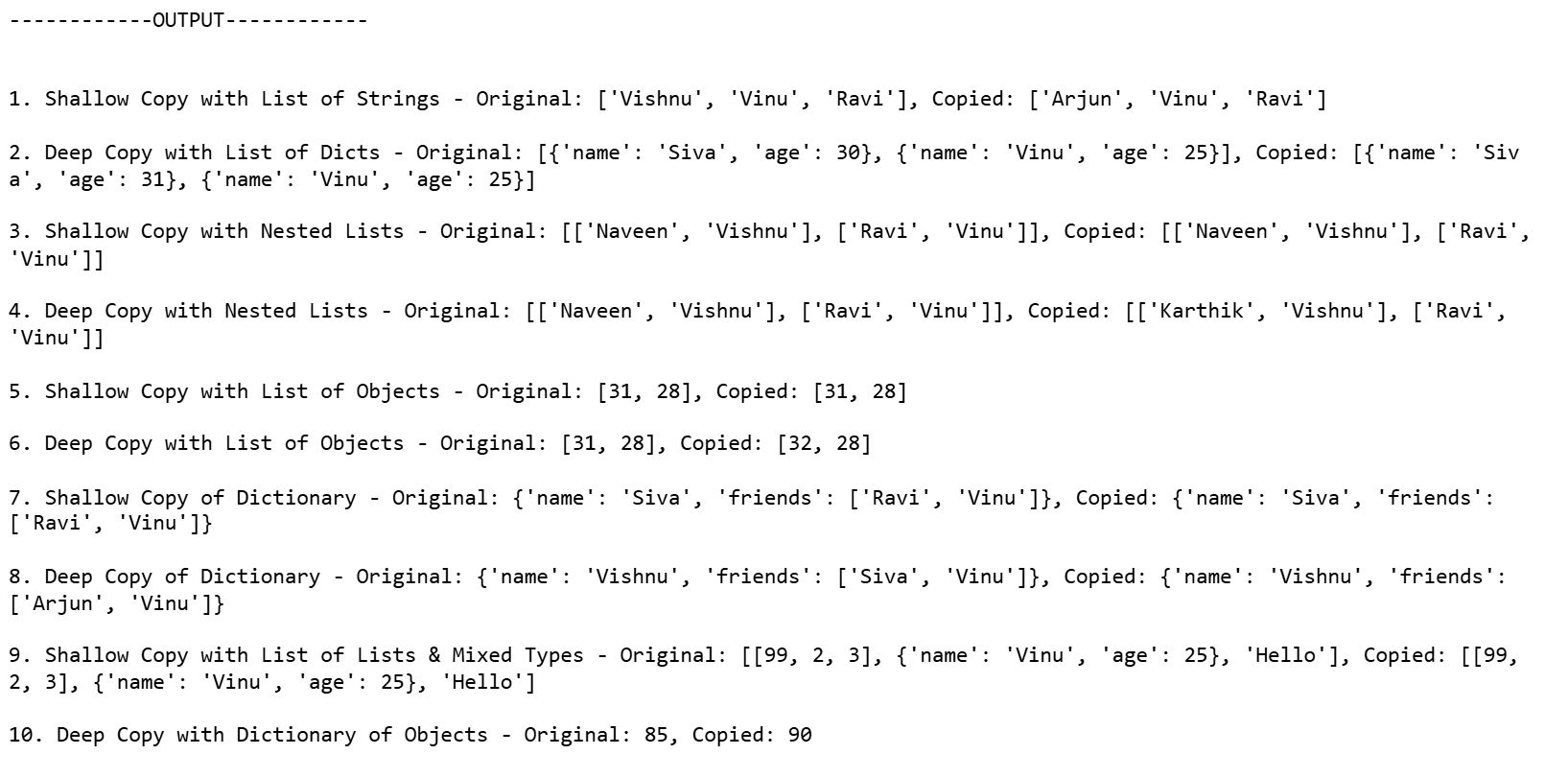




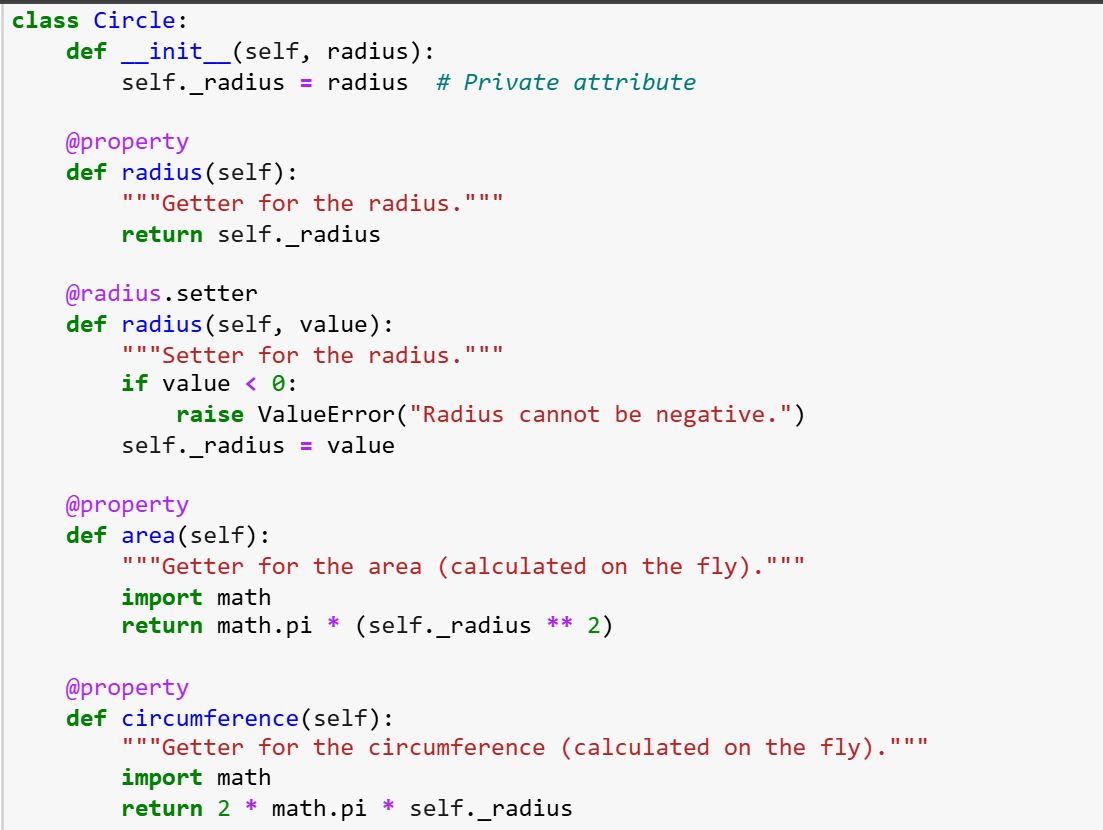


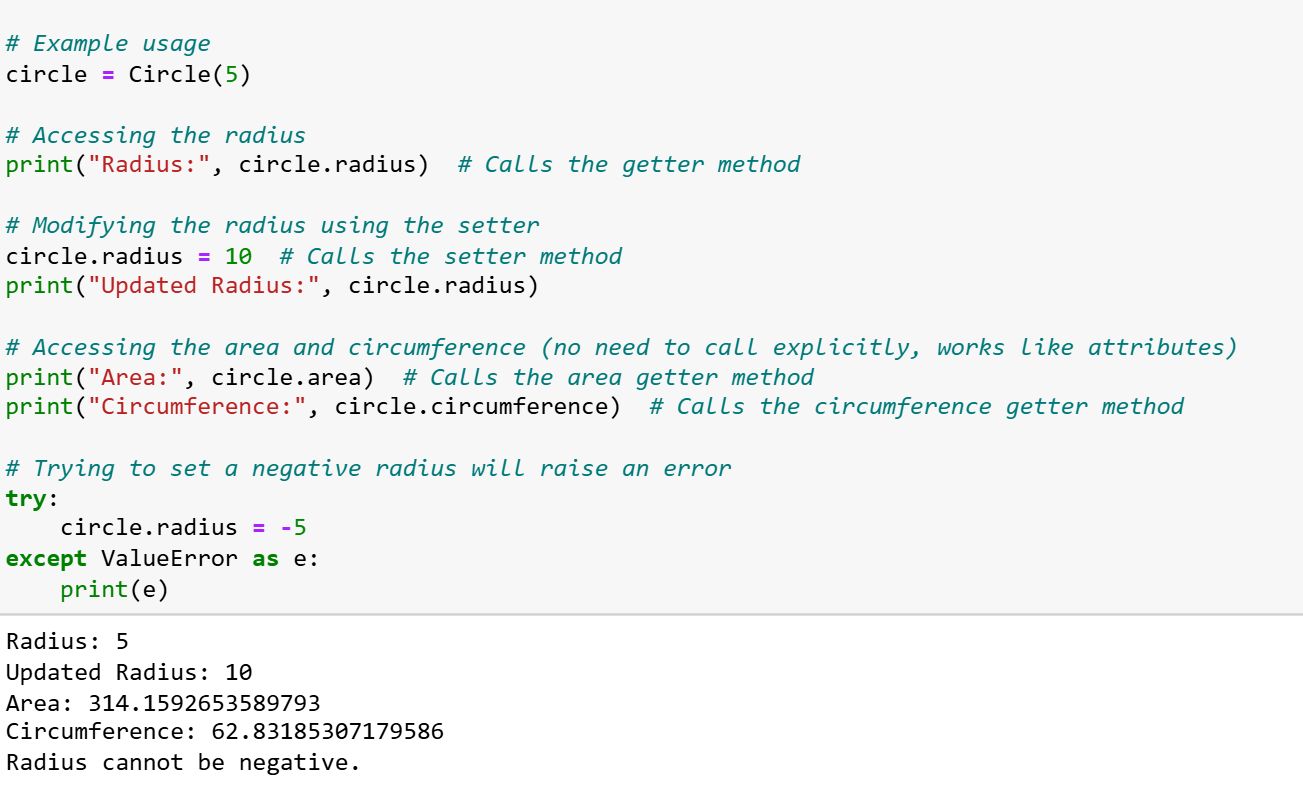






𝐖𝐡𝐚𝐭 𝐢𝐬 𝐭𝐡𝐞 𝐩𝐮𝐫𝐩𝐨𝐬𝐞 𝐨𝐟 𝐭𝐡𝐞 @𝐩𝐫𝐨𝐩𝐞𝐫𝐭𝐲 𝐝𝐞𝐜𝐨𝐫𝐚𝐭𝐨𝐫 𝐢𝐧 𝐏𝐲𝐭𝐡𝐨𝐧? 𝐆𝐢𝐯𝐞 𝐚𝐧 𝐞𝐱𝐚𝐦𝐩𝐥𝐞?  
  
The @property decorator in Python is used to define methods that can be accessed like attributes, allowing you to encapsulate getter, setter, and deleter functionality for class attributes. This provides a way to add logic to attribute access and modification without changing how the attribute is accessed in code.  
  
𝐏𝐮𝐫𝐩𝐨𝐬𝐞:  
𝐄𝐧𝐜𝐚𝐩𝐬𝐮𝐥𝐚𝐭𝐢𝐨𝐧: You can use @property to control the access to an attribute. For example, you can compute an attribute on the fly or validate it before setting it.  
  
𝐆𝐞𝐭𝐭𝐞𝐫, 𝐒𝐞𝐭𝐭𝐞𝐫, 𝐚𝐧𝐝 𝐃𝐞𝐥𝐞𝐭𝐞𝐫: The @property decorator can be used to define a getter method, while additional methods @<property\_name>.setter and @<property\_name>.deleter can be defined to control setting and deleting the property.  
  
𝐑𝐞𝐚𝐝𝐚𝐛𝐥𝐞 𝐒𝐲𝐧𝐭𝐚𝐱: It allows you to write more readable code, as you can use methods in a way that feels like accessing attributes.





Top 34 𝐩𝐲𝐭𝐡𝐨𝐧 Interview Questions for 𝐃𝐚𝐭𝐚 𝐄𝐧𝐠𝐢𝐧𝐞𝐞𝐫𝐢𝐧𝐠 Interview  
  
🥰follow [SIVA VISHNU](https://www.linkedin.com/in/sivavishnu1/) for more🥰  
  
1, What are Python decorators, and how do they work?  
  
2, Explain the difference between deep copy and shallow copy in Python?  
  
3, What is the Global Interpreter Lock (GIL) in Python, and how does it affect multithreading?  
  
4, How does Python handle memory management and garbage collection?  
  
5, What is a generator in Python? How is it different from a normal function?  
  
6, Explain the concept of \*args and \*\*kwargs in Python functions?  
  
7, What are Python's built-in data structures, and when would you use them (e.g., list, tuple, set, dict)?  
  
8, What is the purpose of the with statement in Python, and how does it work?  
  
9, What is the difference between \_\_str\_\_ and \_\_repr\_\_ methods in Python?  
  
10, Explain how to handle exceptions in Python. What is the difference between try-except, try-except-finally, and try-else?  
  
11, What is the purpose of the @property decorator in Python? Give an example?  
  
12, What are Python's comprehensions (list, dict, set) and how do they work?  
  
13, What are the four pillars of Object-Oriented Programming (OOP)?  
  
14, What is method overriding, and how does it work in Python?  
  
15, What is method overloading in Python?  
  
16, Explain the concept of encapsulation in Python with an example?  
  
17, What is polymorphism in Python, and how can it be implemented?  
  
18, How can you define an abstract class in Python? Provide an example?  
  
19, What are Python's special methods (dunder methods)? Give some examples?  
  
20, How do you prevent a class from being instantiated in Python (e.g., making it an abstract class)?  
  
21, Explain how you would use composition instead of inheritance in Python?  
  
22, Write a Python function to check whether a given string is a palindrome?  
  
23, Write a Python function that flattens a nested list (list of lists) into a single list?  
  
24, Write a Python program that removes duplicates from a list while preserving the original order of elements?  
  
25, Write a Python function to find the longest substring without repeating characters in a string.?  
  
26, Write a Python function that calculates the factorial of a number recursively?  
  
27, Write a Python function to reverse a string without using the reverse() method?  
  
28, Write a Python program to implement binary search on a sorted list?  
  
29, Write a Python function that takes a list of numbers and returns the pair of numbers that sum up to a given target?  
  
30, How would you read large datasets (e.g., CSV, Parquet) efficiently in Python?  
  
31, What is the difference between NumPy arrays and Python lists in terms of performance and memory usage?  
  
32, How do you handle missing or corrupted data in a dataset using Python (e.g., with pandas)?  
  
33, How would you optimize the performance of a Pandas operation on a large dataset?  
  
34, How would you store and retrieve large datasets in a distributed system using Python (e.g., with HDFS or Amazon S3)?

𝐐𝐮𝐞𝐬𝐭𝐢𝐨𝐧 𝟏  
𝐒𝐭𝐨𝐫𝐞 𝐑𝐞𝐯𝐞𝐧𝐮𝐞 𝐃𝐢𝐜𝐭𝐢𝐨𝐧𝐚𝐫𝐲  
  
Given two lists—one with store IDs and the other with corresponding revenues—write a function to return a dictionary with store IDs as keys and revenues as values.?  
  
𝐐𝐮𝐞𝐬𝐭𝐢𝐨𝐧 𝟐  
𝐆𝐫𝐨𝐮𝐩 𝐃𝐚𝐭𝐚 𝐛𝐲 𝐊𝐞𝐲 𝐚𝐧𝐝 𝐒𝐮𝐦𝐦𝐚𝐫𝐢𝐳𝐞  
  
Write a Python program to group a list of dictionaries by a specific key and calculate summary statistics (e.g., total, average) for each group.?  
  
𝐐𝐮𝐞𝐬𝐭𝐢𝐨𝐧 𝟑  
𝐋𝐨𝐧𝐠𝐞𝐬𝐭 𝐂𝐨𝐧𝐬𝐞𝐜𝐮𝐭𝐢𝐯𝐞 𝐒𝐞𝐪𝐮𝐞𝐧𝐜𝐞 𝐨𝐟 𝐔𝐧𝐢𝐪𝐮𝐞 𝐍𝐮𝐦𝐛𝐞𝐫𝐬  
  
Write a function to find the longest consecutive sequence of unique numbers in a list.?  
  
𝐐𝐮𝐞𝐬𝐭𝐢𝐨𝐧 𝟒  
𝐃𝐢𝐜𝐭𝐢𝐨𝐧𝐚𝐫𝐲 𝐌𝐚𝐧𝐢𝐩𝐮𝐥𝐚𝐭𝐢𝐨𝐧  
  
Build a dictionary, add items, modify an entry, and display it sorted by key.?

