Shallow Copy and Deep Copy in Python

Shallow Copy

- **Definition**: A shallow copy creates a new object but does not create copies of the objects that the original object references.
- **Key Points**:
  - It creates a new object and inserts references into it to the objects found in the original.
  - Modifications to the elements within the copied object will reflect in the original if those elements are mutable (like lists, dictionaries).
  - Shallow copy can be created using the `copy()` method or the `copy.copy()` function.

Example of Shallow Copy

```
import copy

original_list = [1, 2, [3, 4]]
shallow_copied_list = copy.copy(original_list)

# Modify the inner list
shallow_copied_list[2][0] = 100

print(original_list)  # Output: [1, 2, [100, 4]]
print(shallow_copied_list)  # Output: [1, 2, [100, 4]]
```

- In the above example, the inner list `[3, 4]` is shared between `original_list` and `shallow_copied_list`. Therefore, changes in one reflect in the other.

Deep Copy

- **Definition**: A deep copy creates a new object and recursively copies all objects found within the original object.
- **Key Points**:
  - It creates a new object and copies all nested objects found within the original object.
  - Modifications to the elements within the copied object will **not** reflect in the original as all elements are independently copied.
  - Deep copy can be created using the `copy.deepcopy()` function.

Example of Deep Copy

```
import copy

original_list = [1, 2, [3, 4]]
deep_copied_list = copy.deepcopy(original_list)

# Modify the inner list
deep_copied_list[2][0] = 100

print(original_list)  # Output: [1, 2, [3, 4]]
print(deep_copied_list)  # Output: [1, 2, [100, 4]]
```

- In this example, the inner list `[3, 4]` is independently copied for `deep_copied_list`, so changes do not affect the original.

Exercise

1. **Exercise 1**: Create a shallow copy of a list that contains another list as an element. Modify the inner list and observe the behavior in both the original and the copied list.
2. **Exercise 2**: Create a deep copy of a nested dictionary. Modify one of the nested dictionaries and observe the behavior in both the original and the copied dictionary.
3. **Exercise 3**: Write a Python program to differentiate between shallow copy and deep copy using a custom object containing a list.

Interview Questions

1. **Question 1**: What is the difference between shallow copy and deep copy in Python?
2. **Question 2**: In which scenarios would you prefer using a deep copy over a shallow copy?
3. **Question 3**: How does Python's `copy` module help in managing memory with objects containing nested mutable structures?
4. **Question 4**: Can you explain how Python handles object references in the context of shallow and deep copying?
5. **Question 5**: How would you implement a deep copy manually without using the `copy` module?
