Research Work: Sets and Hashing

Q1:

How does a set handle hash collisions?

Ans:

A set uses a hash table. When collisions occur, Python resolves them internally (like probing or reassigning) so data stays unique and accessible.

Q2:

Why must set elements be hashable?

Ans:

Set elements must be hashable because their hash values need to remain fixed. Mutable objects can change, breaking the hashing system, so only immutable objects can be used.

Q3:

How does garbage collection affect memory for large sets?

Ans:

When a large set is deleted, the garbage collector frees the memory. Calling `gc.collect()` can force this process immediately.

Q4:

What are the memory advantages of a WeakSet?

Ans:

A WeakSet stores objects with weak references. If the original object is deleted, it is automatically removed from the WeakSet, saving memory.

Q5:

Why is set lookup faster than list, and what's the time complexity?

Ans:

Set lookup is faster because it uses a hash table, giving average time complexity of O(1). A list requires checking each element, making it O(n).