==

==

1️⃣ How does Python handle memory management?

✅ Python uses Automatic Memory Management with:

- Reference Counting: Each object has a count of references. When it drops to zero, Python deletes the object.
- Garbage Collection: Python detects cyclic references (objects referring to each other) and removes them.
- Memory Pools: Python uses memory blocks efficiently with a private heap.

Example:

```
import sys
a = [1, 2, 3]
print(sys.getrefcount(a))  # Shows reference count
```

===

2 What is List Comprehension? Give an Example.

✅ List comprehension is a concise way to create lists using a single line of code instead of loops.

Example:

```
# Without list comprehension
squares = []
for i in range(5):
    squares.append(i**2)

# With list comprehension
squares = [i**2 for i in range(5)]
print(squares)  # Output: [0, 1, 4, 9, 16]
```

===

3 What's the difference between copy() and deepcopy()?

✅ **copy()** creates a **shallow copy** (changes in nested objects affect both).
✅ **deepcopy()** creates a **deep copy** (fully independent copy).

Example:

```python
import copy

a = [[1, 2], [3, 4]]
b = copy.copy(a)     # Shallow copy
c = copy.deepcopy(a)  # Deep copy

a[0][0] = 99
print(b)  # Affected: [[99, 2], [3, 4]]
print(c)  # Not affected: [[1, 2], [3, 4]]
```

==

4️⃣ How is Exception Handling done in Python?

✅ Python uses try-except blocks to handle errors gracefully.
✅ finally ensures cleanup code runs no matter what.

Example:

```python
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print(result)
except ZeroDivisionError:
    print("You can't divide by zero!")
except ValueError:
    print("Invalid input! Enter a number.")
finally:
    print("Execution completed.")
```

==

5️⃣ What is a lambda function and how does it differ from a regular function?

✅ **Lambda functions** are anonymous, one-liner functions used for quick calculations.
✅ Regular functions use def and can have multiple statements.

One-liner
expression :
x+y

Example:

```python
# Regular function
def add(x, y):
    return x + y

# Lambda function
add_lambda = lambda x, y: x + y

print(add(5, 3))  # Output: 8
print(add_lambda(5, 3))  # Output: 8
```

==