**Python Core Concepts**

**1. What are Python's key features?**

Python's key features include:

* **Interpreted Language**: Python code is executed line-by-line, without requiring compilation.
* **Dynamically Typed**: Variable types are determined at runtime, so no need for explicit type declarations.
* **Garbage Collection**: Python has automatic memory management through garbage collection.
* **Object-Oriented**: Supports OOP concepts like inheritance, polymorphism, encapsulation, and abstraction.
* **High-level Language**: It has an easy-to-read syntax, making it more human-readable.
* **Extensive Library Support**: Includes libraries like numpy, pandas, matplotlib, scikit-learn, etc.
* **Cross-platform**: Python runs on multiple platforms (Windows, Linux, macOS).

**2. Explain Python's memory management.**

Python's memory management is handled automatically and involves the following:

* **Dynamic Memory Allocation**: Python objects and data structures are allocated memory dynamically.
* **Garbage Collection**: Python has an in-built garbage collector that automatically deallocates memory of unused objects.
* **Reference Counting**: Memory is managed by reference counts. If the reference count of an object drops to zero, it is garbage collected.
* **Private Heap Space**: All Python objects are stored in a private heap, and the Python memory manager handles memory allocation.

**3. What is the difference between is and ==?**

* **is Operator**: Compares the identity of two objects, i.e., whether they point to the same memory location.  
  Example:

a = [1, 2, 3]

b = a

print(a is b) # True, because both point to the same object.

* **== Operator**: Compares the values of two objects.  
  Example:

x = [1, 2, 3]

y = [1, 2, 3]

print(x == y) # True, because the values are the same.

print(x is y) # False, because they are different objects in memory.

**4. Explain mutable and immutable data types in Python.**

* **Mutable Data Types**: Can be modified after creation.  
  Examples: list, set, dictionary.

lst = [1, 2, 3]

lst[0] = 100 # Modified, no new object created.

* **Immutable Data Types**: Cannot be modified after creation. Any modification creates a new object.  
  Examples: int, float, tuple, str.

s = "hello"

s[0] = 'H' # Error, strings are immutable.

**5. What are Python decorators?**

A **decorator** is a function that modifies the behavior of another function or class. It allows you to add functionality to existing functions without modifying their code. Decorators are often used for logging, enforcing access control, or timing function execution.

* **Example of a decorator:**

def decorator\_function(func):

def wrapper():

print("Something before the function is called.")

func()

print("Something after the function is called.")

return wrapper

@decorator\_function # Applying the decorator

def say\_hello():

print("Hello!")

say\_hello()

**Output:**

Something before the function is called.

Hello!

Something after the function is called.

================================================================