**Day 2 - Interview Questions**

1. **What is the difference between '==' and 'is' when comparing variables in Python?**

In Python, == and is are both used for comparison, but they serve different purposes:

1. **== (Equality Operator)**:
   1. **Purpose**: It checks if the values or contents of two variables are the same.
   2. **Example**:

python

Copy code

a = [1, 2, 3]

b = [1, 2, 3]

print(a == b) # Output: True

* 1. In the example, both a and b have the same values [1, 2, 3], so a == b returns True, even though a and b are stored in different memory locations.

1. **is (Identity Operator)**:
   1. **Purpose**: It checks if two variables refer to the **same object** in memory. This is about object identity, not value.
   2. **Example**:

python

Copy code

a = [1, 2, 3]

b = [1, 2, 3]

print(a is b) # Output: False

* 1. In this example, even though a and b have the same values, they are different objects in memory, so a is b returns False.

1. **What is operator precedence in Python, and how does it affect expressions?**

Operator precedence determines the order in which operators are evaluated in an expression. Operators with higher precedence are evaluated first. Python follows the standard operator precedence rules, like multiplication and division before addition and subtraction.

1. **Explain the purpose of the // operator in Python.**
2. In Python, the // operator is used for **floor division**. It divides two numbers and returns the **integer quotient**, rounding down to the nearest whole number, which means it discards the decimal part (if any).
3. **How can you use the 'in' and 'not in' operators to check for membership in Python?**

In Python, the in and not in operators are used to check **membership** in a sequence (such as a list, tuple, string, or dictionary).

**Purpose:**

* **in operator**: Checks if a value **exists** in a sequence.
* **not in operator**: Checks if a value **does not exist** in a sequence.

### Examples:

1. **Using in to check if an element is present**:

python

Copy code

my\_list = [1, 2, 3, 4, 5]

print(3 in my\_list) # Output: True

print(6 in my\_list) # Output: False

1. **Using not in to check if an element is absent**:

python

Copy code

my\_list = ['apple', 'banana', 'cherry']

print('banana' not in my\_list) # Output: False

print('grape' not in my\_list) # Output: True

1. **What is a variable in Python, and how are variables defined?**
2. A **variable** in Python is a name that stores a value. It allows you to store data that can be used or modified later in the program.

### Example:

1. python
2. Copy code
3. age = 25 # 'age' is a variable storing the value 25
4. name = "Alice" # 'name' is a variable storing the value "Alice"
5. Here, age and name are variables that hold values, which can be changed during the program's execution.
6. **Explain the difference between mutable and immutable variables in Python.** Mutable variables can be changed or modified after creation. Lists and dictionaries are examples of mutable types.

Immutable variables cannot be changed after creation. Integers, strings, and tuples are examples of immutable types.

**Are there constants in Python, and if so, how are they typically defined?** In Python, there are **no built-in constant types** like in some other programming languages such as Java or C++. However, **constants are often represented by using uppercase variable names** as a convention to indicate that their values should not be changed. This is a naming convention rather than a language feature.

For example:

python

Copy code

PI = 3.14159

MAX\_USERS = 100

DATABASE\_URL = "https://mydb.example.com"

While Python doesn't enforce immutability for these variables, using uppercase names helps signal to other developers that these values are intended to be constants.

Additionally, Python's standard library does have some **immutable constants**, such as:

* None
* True
* False
* Constants in the math module like math.pi and math.e

1. **Explain the difference between / and // operators in Python.**

/ performs standard division, resulting in a floating-point number.

// performs floor division, which rounds the result down to the nearest integer.

1. **Explain the use of the += operator in Python with an example.**
2. The += operator in Python is used for **addition assignment**. It adds the value on the right-hand side to the variable on the left and then updates the left-hand variable with the new value.
3. **Example:**
4. python
5. Copy code
6. x = 5
7. x += 3 # Equivalent to x = x + 3
8. print(x) # Output: 8
9. In this example, x += 3 is the same as saying x = x + 3. The value of x is updated to 8.

**10.What are logical operators in Python?**

Comparison operators in Python are used to compare values and return a Boolean result (True or False). They include:

* **==** (Equal to): Checks if two values are equal.
* **!=** (Not equal to): Checks if two values are not equal.
* **>** (Greater than): Checks if the left value is greater than the right value.