Python Variables and Data Types

Variables

- A variable in Python is like a container where we store data. It's essentially a name that refers to a value.
- Every piece of data stored in a variable has a specific type, known as a data type (e.g., `int` for integers, `str` for strings).

Example:

```
a = 10
```

In this example:

- `10` is an integer data type.
- `a` is a variable that stores the value `10`.

How It Works:

- When you assign `10` to `a`, Python creates an object to store the value `10` in the heap memory.
- The variable `a` is a reference to this object and is stored in the stack memory.

Data Types

Common data types in Python include:

- **Integer (`int`)**: Whole numbers, e.g., `10`, `-5`.
- **String (`str`)**: Sequence of characters, e.g., `"Hello"`, `'Python'`.
- **Float (`float`)**: Numbers with decimal points, e.g., `10.5`, `-3.14`.
- **Boolean (`bool`)**: Represents `True` or `False`.
- **List (`list`)**: Ordered collection of items, e.g., `[1, 2, 3]`.
- **Dictionary (`dict`)**: Collection of key-value pairs, e.g., `{"name": "John", "age": 30}`.

Operations on Variables

Variables in Python can be used to perform various operations. Here are some basic arithmetic operations:

1. **Addition (`+`)**:

   ```
   x = 5
   y = 3
   result = x + y  # result is 8
   ```

2. **Subtraction (`-`)**:

   ```
   x = 5
   y = 3
   result = x - y  # result is 2
   ```

3. **Multiplication (`*`)**:

```python
x = 5
y = 3
result = x * y  # result is 15
```

4. **Division (`/`)**:

```python
x = 5
y = 3
result = x / y  # result is approximately 1.67
```

Key Points to Remember:

- Variables are like labels that point to values stored in memory.
- Data types define the kind of data a variable can hold and the operations that can be performed on it.
- Python automatically manages memory allocation for variables.
- Understanding data types and operations is fundamental to working effectively with Python.

Example Code:

Here's a simple Python script demonstrating variables and basic operations:

```python
# Variable assignment
a = 10
b = 5

# Arithmetic operations
addition = a + b
subtraction = a - b
multiplication = a * b
division = a / b

# Printing results
print("Addition:", addition)
print("Subtraction:", subtraction)
print("Multiplication:", multiplication)
print("Division:", division)
```

Practice Exercise:

Try creating variables of different data types and perform operations on them:

1. Create two integer variables and add them.
2. Check Interger is EVEN or ODD
3. Swap two number, without taking 3rd variable

Most Asked Interview Questions

1. **Is `int` mutable or immutable and why?**

    - **Answer:** `int` is immutable. This means that once an integer object is created, its value cannot be changed. If you perform any operation that changes the integer, a new integer object is created and the reference is updated.

2. **Explain `int` variable memory management in Python.**

    - **Answer:** In Python, integers are stored as objects in the heap memory. When you create an integer variable, an object is created in the heap and the variable holds a reference to this object in the stack. Python uses a technique called "integer interning" for small integers (typically -5 to 256), where these integers are pre-allocated and reused to save memory and improve performance.

3. **If we have two `int` variables having the same value, how many objects will be created internally?**

    - **Answer:** If the integer value is within the range of interned integers (usually -5 to 256), then only one object will be created and both variables will reference the same object. For values outside this range, two separate objects will be created.

4. **How to print the memory location of an `int` variable?**

    - **Answer:** You can use the `id()` function to get the memory address of an integer variable.

        ```
        a = 10
        print(id(a))  # prints the memory address of the variable 'a'
        ```

5. **How to check the data type of a variable?**

    - **Answer:** You can use the `type()` function to check the data type of a variable.

        ```
        a = 10
        print(type(a))  # prints <class 'int'>
        ```
