🌟 Data Types in Python

In Python, a data type defines the type of a value that a variable can hold. Python has several built-in data types that allow you to store and manipulate different kinds of data. Understanding these data types is fundamental to writing effective and efficient code.

🚀 Why Use Variables?

- **Accuracy**: Ensures data is stored in the correct format.
- **Efficiency**: Operations on data are faster and more efficient when the correct data type is used.
- **Readability**: Makes code more readable and understandable.

🚀Common Data Types in Python

- Int: Number without decimal point.
- Float: Number with decimal point.
- String: Text
    - [Methods: capitalize, upper, lower,]
- List[]: ordered and changeable. Allows duplicate items.
    - [Methods: append, pop, clear, sort]
- Tuple(): ordered and unchangeable. Allows duplicate items.
- Set{}: unordered, unchangeable, and unindexed. No duplicate items.
- Dict{}: Key Value Pairs
- boolean
- none

🎉 Data Type Examples

Example 1: Basic Operation

```
a = 10
b = 5
print(f"Addition: {a + b}")  # 15
print(f"Subtraction: {a - b}")  # 5
print(f"Multiplication: {a * b}")  # 50
print(f"Division: {a / b}")  # 2.0
```

Example 2: Shopping List

```
# Creating a shopping list
shopping_list = ["milk", "eggs", "bread", "butter"]

# Adding items to the list
shopping_list.append("cheese")
shopping_list.append("fruits")

# Removing an item from the list
shopping_list.remove("bread")

# Displaying the updated list
print(shopping_list)  # Output: ['milk', 'eggs', 'butter', 'cheese',
'fruits']
```

Example 3: GPS Coordinates

```
# Creating a tuple of GPS coordinates
home_coordinates = (27.7172, 85.3240)  # Kathmandu, Nepal
office_coordinates = (28.6139, 77.2090)  # New Delhi, India

# Accessing the tuple elements
print("Home Coordinates:", home_coordinates)
print("Office Coordinates:", office_coordinates)

# Tuples are immutable, so the following line would raise an error
# home_coordinates[0] = 28.0000  # TypeError: 'tuple' object does not
support item assignment
```

Example 4: Unique Email Addresses

```
# Creating a set of email addresses
email_addresses = {"john@example.com", "jane@example.com",
```

```
"alice@example.com"}

# Adding new email addresses (duplicates are ignored)
email_addresses.add("bob@example.com")
email_addresses.add("john@example.com")  # Duplicate, will not be added

# Displaying the unique email addresses
print(email_addresses)  # Output: {'alice@example.com',
'john@example.com', 'jane@example.com', 'bob@example.com'}

# Checking if an email is already in the set
print("jane@example.com" in email_addresses)  # Output: True
```

Example 5: Dictionary of Student Grades

```
grades = {
    "Ram": "A",
    "Rahul": "B",
    "Suman": "C",
    "David": "B",
}

# Adding a new student's grade
grades["Suraj"] = "A"

# Accessing a student's grade
print(f"Ram's grade: {grades[Ram]}")

# Updating a student's grade
grades["Suman"] = "A"
```

💡 Challenge 3: Your Finance Manager

Write down what you spend each day from Sunday to Saturday using a dictionary, add them up to find the total for the week, and figure out the average amount spent per day.