Python Fundamentals

Variables and Data Types in Python

What are Variables?

- Variables are used to store data that can be used and manipulated in a program.
- A variable is created when you assign a value to it using the `=` operator.
- Example:

```
name = "Alice"
age = 25
height = 5.6
```

Variable Naming Rules

- Variable names can contain letters, numbers, and underscores.
- Variable names must start with a letter or underscore.
- Variable names are case-sensitive.
- Avoid using Python keywords as variable names (e.g., `print`, `if`, `else`).

Best Practices

- Use descriptive names that reflect the purpose of the variable.
- Use lowercase letters for variable names.
- Separate words using underscores for readability (e.g., `first_name`, `total_amount`).

Data Types in Python

Python supports several built-in data types:

- **Integers (`int`)**: Whole numbers (e.g., `10` , `-5`).
- **Floats (`float`)**: Decimal numbers (e.g., `3.14` , `-0.001`).
- **Strings (`str`)**: Text data enclosed in quotes (e.g., `"Hello"` , `'Python'`).
- **Booleans (`bool`)**: Represents `True` or `False` .
- **Lists**: Ordered, mutable collections (e.g., `[1, 2, 3]`).
- **Tuples**: Ordered, immutable collections (e.g., `(1, 2, 3)`).
- **Sets**: Unordered collections of unique elements (e.g., `{1, 2, 3}`).
- **Dictionaries**: Key-value pairs (e.g., `{"name": "Alice", "age": 25}`).

Checking Data Types

- Use the `type()` function to check the data type of a variable.

  ```python
  print(type(10))       # Output: <class 'int'>
  print(type("Hello"))  # Output: <class 'str'>
  ```

Typecasting in Python

What is Typecasting?

- Typecasting is the process of converting one data type to another.
- Python provides built-in functions for typecasting:
  - `int()` : Converts to integer.
  - `float()` : Converts to float.
  - `str()` : Converts to string.
  - `bool()` : Converts to boolean.

Examples:

```
# Convert string to integer
num_str = "10"
num_int = int(num_str)
print(num_int)  # Output: 10

# Convert integer to string
num = 25
num_str = str(num)
print(num_str)  # Output: "25"

# Convert float to integer
pi = 3.14
pi_int = int(pi)
print(pi_int)   # Output: 3
```

Taking User Input in Python

Using the `input()` Function

- The `input()` function allows you to take user input from the keyboard.
- By default, `input()` returns a string. You can convert it to other data types as needed.
- Example:

  ```
  name = input("Enter your name: ")
  age = int(input("Enter your age: "))
  print(f"Hello {name}, you are {age} years old.")
  ```

Comments, Escape Sequences & Print Statement

Comments

- Comments are used to explain code and are ignored by the Python interpreter.
- Single-line comments start with `#`.
- Multi-line comments are enclosed in `'''` or `"""`.

  ```python
  # This is a single-line comment
  '''
  This is a
  multi-line comment
  '''
  ```

Escape Sequences

- Escape sequences are used to include special characters in strings.
- Common escape sequences:
  - `\n` : Newline
  - `\t` : Tab
  - `\\` : Backslash
  - `\"` : Double quote
  - `\'` : Single quote
- Example:

  ```python
  print("Hello\nWorld!")
  print("This is a tab\tcharacter.")
  ```

Print Statement

- The `print()` function is used to display output.

- You can use `sep` and `end` parameters to customize the output.

```
print("Hello", "World", sep=", ", end="!\n")
```

Operators in Python

Types of Operators

1. **Arithmetic Operators**:
    1. `+` (Addition), `-` (Subtraction), `*` (Multiplication), `/` (Division), `%` (Modulus), `**` (Exponentiation), `//` (Floor Division).
    2. Example:

    ```
    print(10 + 5)   # Output: 15
    print(10 ** 2)  # Output: 100
    ```

2. **Comparison Operators**:
    1. `==` (Equal), `!=` (Not Equal), `>` (Greater Than), `<` (Less Than), `>=` (Greater Than or Equal), `<=` (Less Than or Equal).
    2. Example:

    ```
    print(10 > 5)   # Output: True
    print(10 == 5)  # Output: False
    ```

3. **Logical Operators**:
    1. `and`, `or`, `not`.
    2. Example:

    ```
    print(True and False)  # Output: False
    print(True or False)   # Output: True
    print(not True)        # Output: False
    ```

4. **Assignment Operators**:

    1. `=`, `+=`, `-=`, `*=`, `/=`, `%=`, `**=`, `//=`.
    2. Example:

    ```
    x = 10
    x += 5  # Equivalent to x = x + 5
    print(x)  # Output: 15
    ```

5. **Membership Operators**:

    1. `in`, `not in`.
    2. Example:

    ```
    fruits = ["apple", "banana", "cherry"]
    print("banana" in fruits)  # Output: True
    ```

6. **Identity Operators**:

    1. `is`, `is not`.
    2. Example:

    ```
    x = 10
    y = 10
    print(x is y)  # Output: True
    ```

Summary

- Variables store data, and Python supports multiple data types.
- Typecasting allows you to convert between data types.
- Use `input()` to take user input and `print()` to display output.
- Comments and escape sequences help make your code more readable.
- Python provides a variety of operators for performing operations on data.