Values and Data Types

- A **value** (or object) is data manipulated by programs (e.g., `5`, `"Hello, World!"`).
- **Data types**: integers (`int`), floating-point numbers (`float`), and strings (`str`).
- Use `type()` to check a value's data type.
- Strings can be enclosed in single (`' '`), double (`" "`), or triple quotes (`''' '''` or `""" """`).
- Strings in quotes that look like numbers (e.g., `"17"`) are still **strings**.

Variables

- A **variable** is a name referring to a value (e.g., `n = 17`).
- Assignment (`=`) links a name to a value (not equality).
- Variables can change to reference new values of different types.
- In math, variables are fixed once assigned, but in programming, they are **mutable**.

Variable Names and Keywords

- Variable names:
    - Must start with a letter/underscore.
    - Can contain letters and digits, but no spaces or symbols like `$`.
    - Are case-sensitive (`Bruce` ≠ `bruce`).
- Cannot use Python **keywords** (e.g., `class`, `if`, `for`, `while`).

Statements and Expressions

- A **statement** is an instruction Python executes (e.g., assignment, loops).
- An **expression** combines values, variables, operators, or functions and **produces a value**.
- Example: `1 + 1` or `len("hello")`.
- Assignment statements don't return values, but evaluating a variable gives its current value.

Operators and Operands

- **Operators**: `+`, `-`, `*`, `/`, `//`, `%`, `**`.
- **Operands**: the values operators act on.
- `/` → floating-point division, `//` → integer division (truncates), `%` → modulus (remainder).
- `%` is useful for divisibility tests and extracting digits.

Input

- `input()` collects user input as a **string**.

Example:

```
name = input("Enter your name: ")
print("Hello", name)
```

- To work with numbers, convert input using `int()` or `float()`.
- Example: converting seconds input into hours, minutes, seconds.