Class 2 — Meaning, Objects, and Representation

These notes summarize the key ideas from today's class. They are designed to help you reason about Python programs before running them.

Big Idea

Python works because code has stable meaning. That stability comes from how Python represents information, not from guessing or trial-and-error.

Everything in Python Is an Object

Every value in Python is an object. An object has three essential parts:

1. Type — what kind of thing it is (int, float, str, bool, etc.)
2. Value — what information it represents
3. Interface — what operations are allowed on it

Objects can exist even if we do not give them a name.

Variables Are Names (Bindings)

A variable is a name bound to an object. Variables do not store values like boxes. Reassigning a variable changes the binding, not the object.

Expressions vs. Statements

Expressions are evaluated and produce values. They do not change program state by themselves.

Statements are executed and perform actions. Some statements change state; others produce side effects.

All state changes occur inside statements, but not all statements change state.

Arithmetic Operators and Precedence

An important invariant of expressions is that they have a single, unambiguous meaning. Operator precedence and associativity exist to guarantee this.

Parentheses change the structure of an expression and therefore its meaning. They always override default precedence rules.

Primitive Data Types (This Course)

1. int — exact whole numbers
2. float — approximate real numbers

3 str — ordered sequences of characters

4 bool — logical values: True or False

Python also has a complex type, but we will not use it in this course.

Immutability and State Change

There are only two ways state can change in a program:

1 Replacement — rebinding a name to a different object

2 Modification — mutating an existing object

All primitive types used in this course are **immutable**. They cannot be modified after creation.

Strings

Strings are immutable, ordered sequences of characters. String methods always return new string objects rather than changing existing ones.

Floating-Point Representation

Computers store numbers in base 2. Many decimal fractions (such as 0.1) do not have an exact finite representation in base 2.

The key guarantee of floats is **consistent approximation**, not exact decimal accuracy.

Representation Invariants

A representation invariant is a rule that forbids invalid states. These rules ensure that objects continue to represent what they claim to represent.

Examples include: integers being whole numbers, strings remaining ordered and immutable, and expressions having a single meaning.

Key takeaway: Programming works because representations come with guarantees. Understanding those guarantees allows you to reason about code before running it.