Understanding Computer Science

Nithin Vadekkapat

March 7, 2025

What is Computer Science?

- Computer Science is not merely the study of computers.
- Computers are tools that aid in problem-solving.
- The focus is on problems, problem-solving, and algorithmic solutions.

The Role of Algorithms

- An algorithm is a step-by-step set of instructions to solve a problem.
- Some problems may not have a solution.
- Computer Science studies both solvable and unsolvable problems.

Computability in Computer Science

- A problem is **computable** if an algorithm exists to solve it.
- Computer Science studies both computable and non-computable problems.
- Solutions are independent of the machine used.

Abstraction in Computer Science

- Abstraction helps separate the logical and physical perspectives.
- Example: Driving a car.
- Users interact with the interface without needing to understand the mechanics.

Procedural Abstraction

- Users (clients) do not need to know implementation details.
- The interface provides a way to interact with complex implementations.
- Example: Python's `math` module.

```
import math
math.sqrt(16)
```

- We do not need to know how `sqrt` is implemented.
- We only need to know its name and how to use it.

Programming and Algorithms

- Programming is encoding an algorithm into a notation (programming language) for computer execution
- Without an algorithm, there can be no program
- Computer science is not just programming, but programming is an essential part
- Programming creates representations of our solutions

From Algorithms to Programs

- Algorithms describe solutions in terms of:
  - Data needed to represent the problem instance
  - Steps necessary to produce the intended result
- Programming languages must provide notation for both process and data
- Languages provide control constructs and data types

Control Constructs

- Control constructs allow algorithmic steps to be represented unambiguously
- Minimum requirements for algorithm representation:
  - Sequential processing
  - Selection for decision-making
  - Iteration for repetitive control
- Any language with these basic statements can represent algorithms

Data Types

- All data in computers are strings of binary digits
- Data types provide interpretation for binary data
- Built-in/primitive data types are building blocks for algorithm development
- Example: Integer data type
  - Binary digits interpreted as familiar numbers (23, 654, -19)
  - Supports operations like addition, subtraction, multiplication

Complexity Challenges

- Problems and solutions are often very complex
- Simple language-provided constructs and data types:
  - Are sufficient to represent complex solutions
  - But can be at a disadvantage during problem-solving
- Higher-level abstractions help manage this complexity

Abstraction in Problem-Solving

- Computer scientists use abstractions to focus on the "big picture"
- Creating models of the problem domain enables:
  - More efficient problem-solving process
  - More consistent description of data with respect to the problem
- Abstractions allow us to ignore implementation details temporarily

Data Abstraction

- Abstract Data Type (ADT): logical description of data and operations
- Focuses on *what* the data represents, not *how* it's constructed
- Creates encapsulation around data through information hiding
- Users interact with the interface only, not the implementation

ADT Implementation

- Implementation of an ADT is called a data structure
- Data structure provides the physical view of the data
- Uses programming constructs and primitive data types
- Different implementations can satisfy the same ADT
- Choice of implementation affects efficiency

Python Data Structures

- Python provides built-in data structures
- These structures are used to represent data in programs
- Common data structures include:
  - Lists
  - Tuples
  - Sets
  - Dictionaries

Objects

- Everything in Python is an object
- Objects have attributes and methods
- Objects can be created from classes
- Classes define the attributes and methods of objects

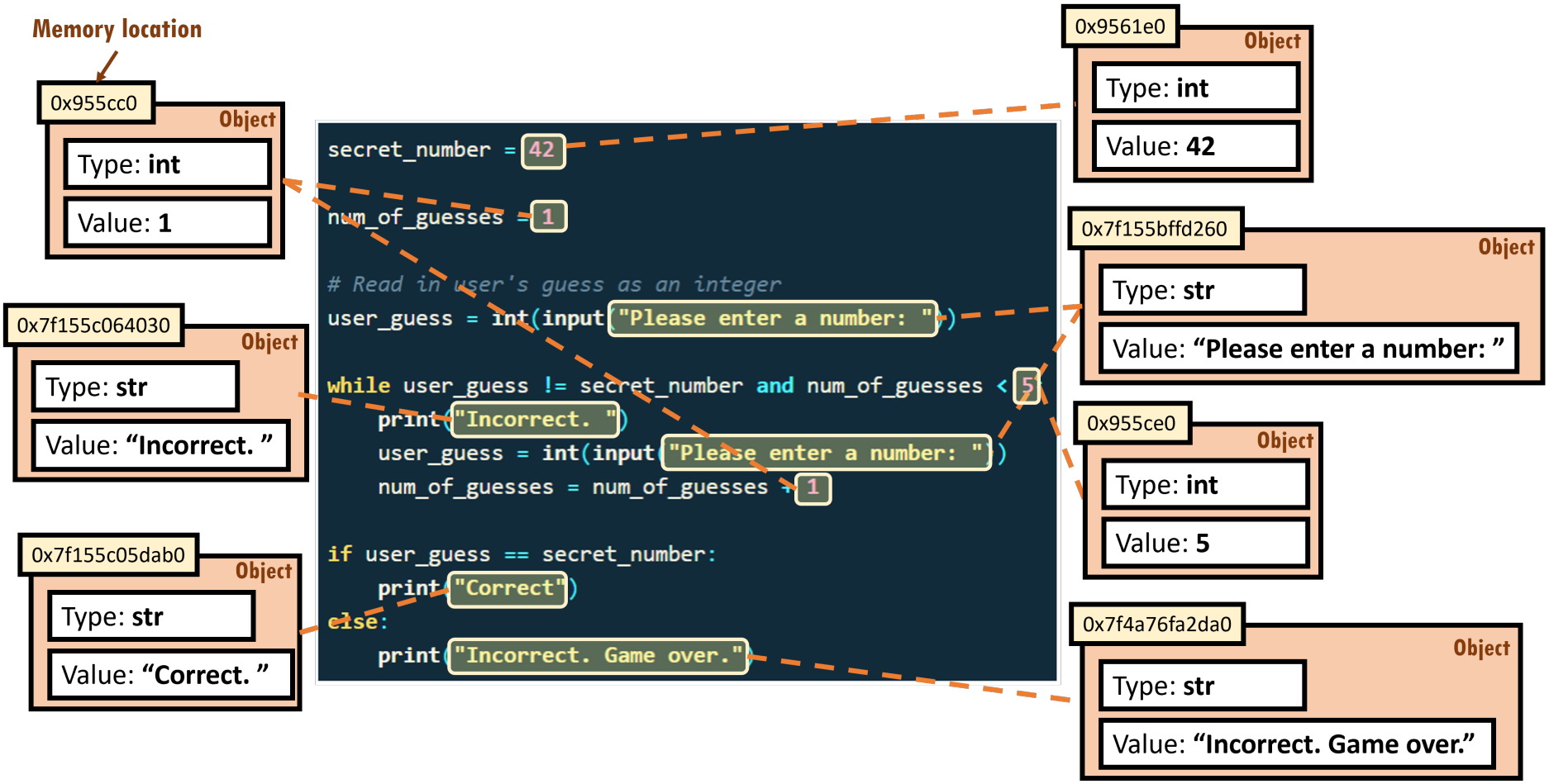

Figure: Objects in Python

Built in atomic datatypes

- Python provides built-in data types
- These types are used to represent data in programs
- Common data types include:
  - Integers
  - Floats
  - Strings
  - Booleans

Operators and Expressions

- Objects can be combined using operators
- Arithmrtic operators in python
  - + (addition)
  - - (subtraction)
  - * (multiplication)
  - / (division)
  - // (floor division, or integer part of division)
  - % (modulo, or remainder after division)
  - ** (exponential - raised to the power)

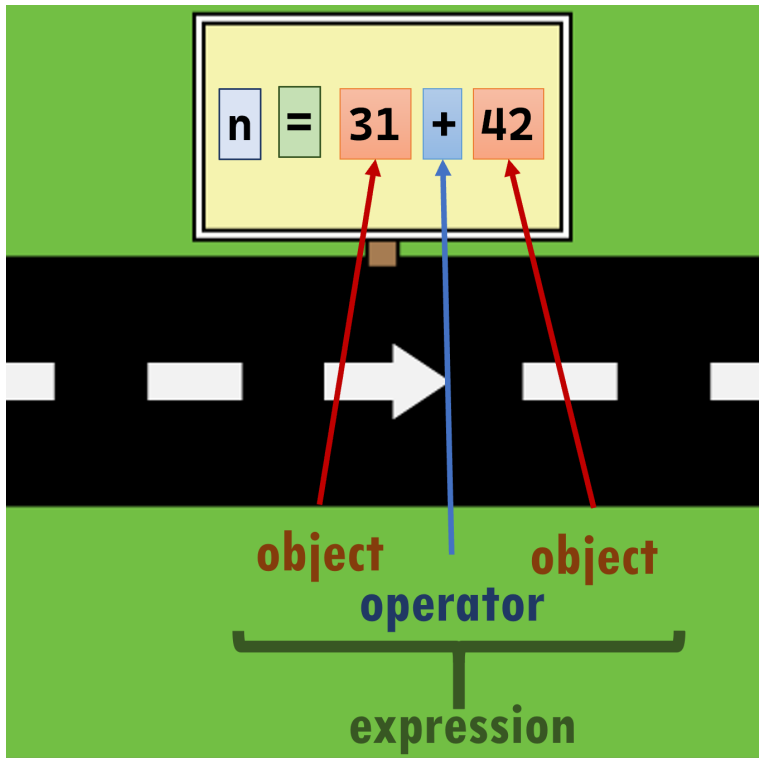

Figure: Operators in Python

Overloaded Operators

- Some operators are not limited to arithmetic operations with numbers.
- For example, + and * can be used for a different kind of operation when used with strings.

```
>>> 'hello' + 'world'
'helloworld'
>>> 'hello' * 3
'hellohellohello'
```

Variables

In Python, a variable is simply a name or a label that points to an object instance in memory

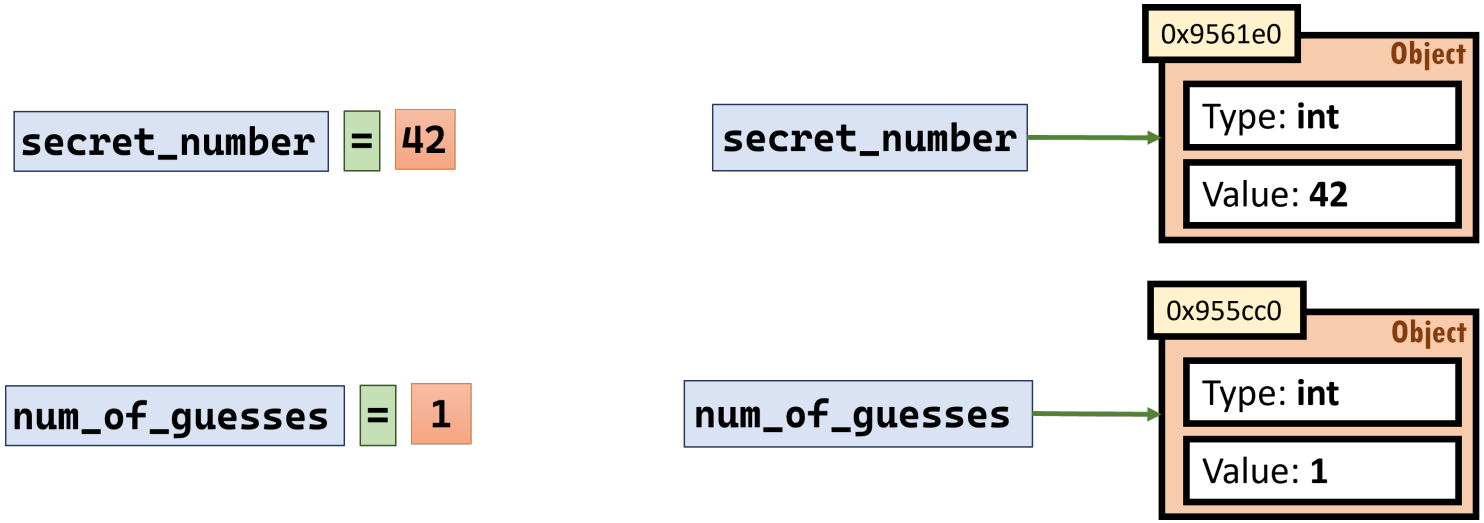

Figure: Variables in Python

Variable Assignments

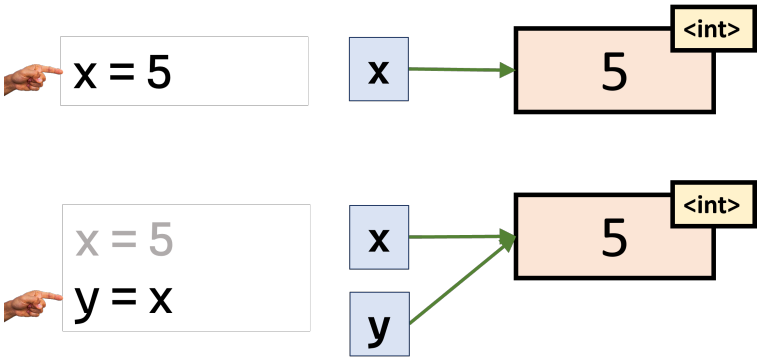

Figure: Variable Assignment

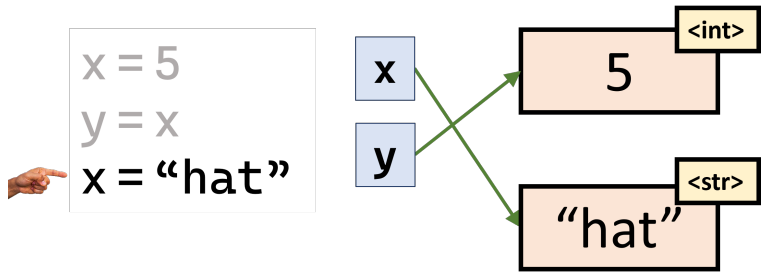

Figure: Variable Assignment

Variable Assignment

```
>>> x = 20
>>> y = x
>>> x = "hello"
>>> print(y)
20
>>> print(x)
hello
```

Naming Variables

- Variable names must start with a letter or an underscore
- The rest of the name can contain letters, numbers, and underscores
- Variable names are case-sensitive
- Variable names should be descriptive

Reserved Keywords