Math Code Lab

Level1

Week-2

Data Types and Operators

Recap

What is Programming?

1

Programming is the **process of giving instructions to a computer** to perform specific tasks.

2

These **instructions are organized into programs** - sets of precise directions that tell computers exactly what to do and how to do it.

3

Think of programming as having a **conversation with your computer** but using a language it understands.

4

You communicate what you want it to accomplish, and **it follows your directions precisely**.

Core Programming Constructs

Variables, Data Types & Operators

Variables are like labeled containers that store information.
Data types define what kind of information they hold (numbers, text, etc.).
Operators are **symbols or words** that let you perform actions on data.

```
name = "Alex"  # Text
(string)age = 25      # Number
(integer)
```

Input/Output & Conditionals

Programs communicate with users through input/output. Conditionals let programs make decisions based on conditions.

```
if temperature > 80:
print("It's hot today!")
```

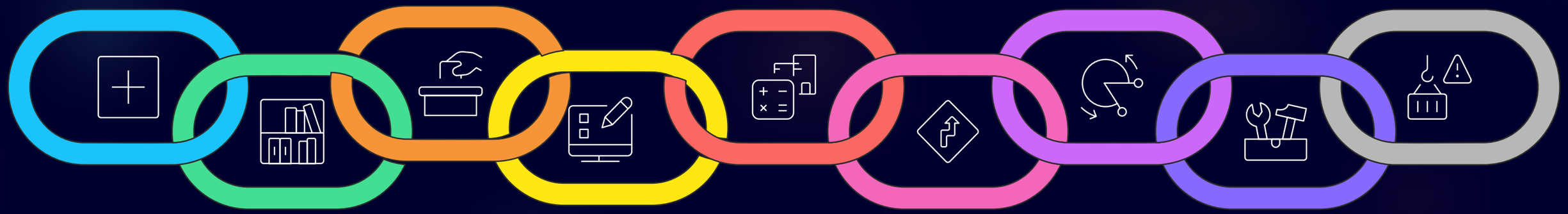
Loops & Functions

Loops repeat actions multiple times.
Functions group code into reusable blocks that perform specific tasks.

```
def greet(name): return "Hello,
" + name
```

Week 2

Core Programming Constructs



Variables

Store and manipulate data

Data Structures

Organize and manage data efficiently

Assignment

Assign values to variables

Input/Output

Interact with users

Operators

Perform calculations and comparisons

Conditionals

Make decisions based on conditions

Loops

Repeat code blocks

Functions

Group reusable code

Exception Handling

Manage runtime errors

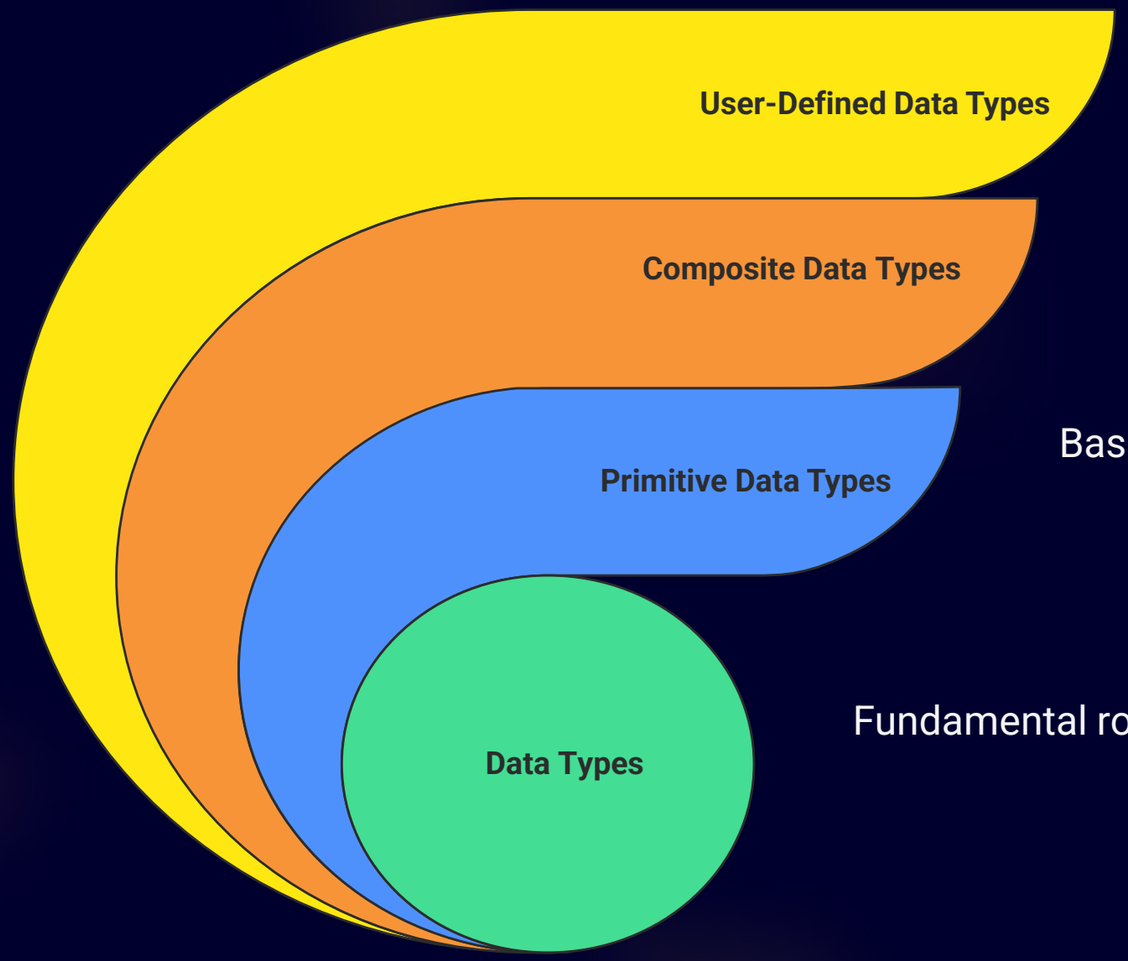
Data Types

What Are **Data Types**?

Think of **data types** as different kinds of **boxes** to store information.

The **type** tells the computer what kind of data it is, and what actions (**operators**) are allowed to be performed.

Hierarchy of Data Types in Programming



Custom data structures

Collections of data values

Basic building blocks of data

Fundamental role in data storage

Simple (Primitive) Data Types

These are the **basic building blocks**, like LEGO pieces you start with.

Data Type	What It Stores	Example	Analogy	Python Example
int	Whole numbers	7, -3, 100	🍎 Counting apples	age = 10
float	Decimal numbers	3.14, -0.5	🥛 Measuring water	price = 5.99
bool	True or False	True, False	🔆 Light switch ON/OFF	is_raining = False
char	Single character (in C/C++)	'A', 'z'	⌨️ One keyboard key	grade = 'A'
str	Text (words, sentences)	"hello"	📿 Beads on a string	name = "Leo"

⚠️ *Note:* Python doesn't have a separate char type — single characters are just strings of length 1.

Composite (Complex) Data Types

These store **multiple values** together like a container or toolbox.

🧱 A. Homogeneous (Same type inside) – a bag of oranges or apples

🥗 B. Heterogeneous (Mixed types) – a bag of vegetables, fruits and snacks

🧱 A. Homogeneous (Same type inside)

array

Fixed-size collection of same type (C/Python)

Example: [1, 2, 3]

Python Code: import array; nums = array.array('i', [1, 2, 3])

list

Flexible-size collection (can mix types)

Example: ["red", 5, True]

Python Code: colors = ["red", "blue"]

B. Heterogeneous (Mixed types)

tuple

Fixed-size group of items

Example: (12, "dog", True)

Python Code: pet = (12, "dog", True)

dict

Key–value pair structure

Example: {"name": "Leo", "age": 10}

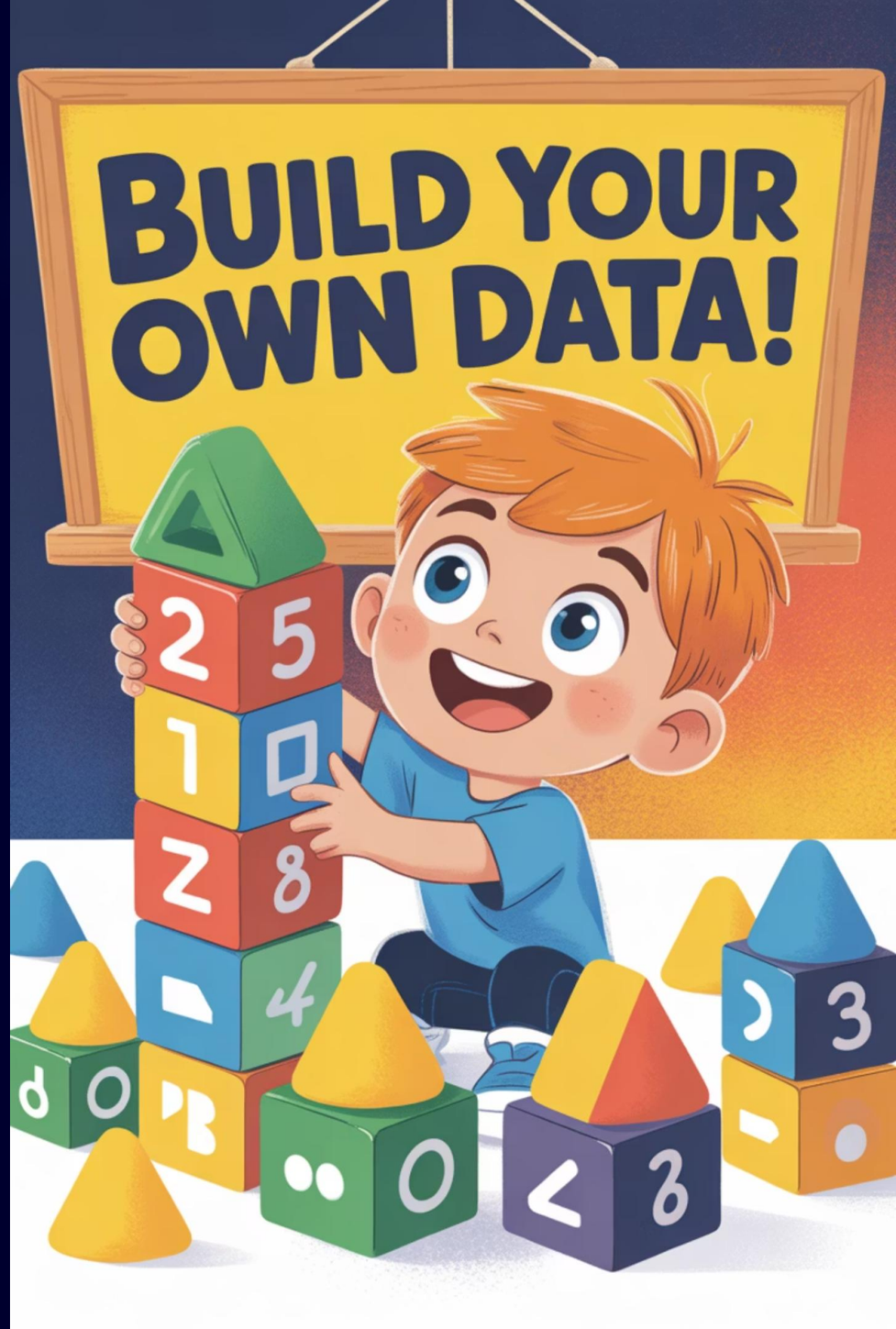
Python Code: student = {"name": "Leo", "age": 10}

User-Defined Types

These let you **design your own** data structures.

🧊 Structure / Class

```
class Student: def __init__(self, name, age): self.name = name
self.age = ageleo = Student("Leo", 10)print(leo.name) # Output:
Leo
```



What Are **Operators**?

Operators are **symbols or words** that let you perform actions on data.

🧮 Arithmetic Operators (Used with int, float)

Operator	Name	Example	Output	Meaning
+	Addition	5 + 2	7	Add values
-	Subtraction	10 - 4	6	Subtract values
*	Multiplication	3 * 4	12	Multiply
/	Division	8 / 2	4.0	Divide (float result)
//	Floor Division	9 // 2	4	Whole number only
%	Modulus	9 % 2	1	Remainder
**	Power	2 ** 3	8	Exponent (power)

⚖️ **Comparison Operators** (Used with int, float, str)

They return **True or False**.

1. Equal to (==)
   Example: 5 == 5
   Result: True

2. Not equal to (!=)
   Example: 5 != 3
   Result: True

3. Greater than (>)
   Example: 7 > 3
   Result: True

4. Less than (<)
   Example: 2 < 4
   Result: True

5. Greater or equal (>=)
   Example: 4 >= 4
   Result: True

6. Less or equal (<=)
   Example: 3 <= 5
   Result: True

💡 Logical Operators (Used with bool)

and

Meaning: Both must be true

Example: True and False

Output: False

or

Meaning: Either one is true

Example: True or False

Output: True

not

Meaning: Opposite value

Example: not True

Output: False

📝 String Operators (Used with str)

+ (Concatenation)

Meaning: Join strings together

Example: "Hi" + "!"

Output: "Hi!"

* (Repetition)

Meaning: Repeat string multiple times

Example: "ha" * 3

Output: "hahaha"

Assignment Operators (Used with any data type)

= (Assign value)
Example: x = 10
Result: x becomes 10

+= (Add and assign)
Example: x += 2
Result: x = x + 2

-= (Subtract and assign)
Example: x -= 1
Result: x = x - 1

*= (Multiply and assign)
Example: x *= 3
Result: x = x * 3

/= (Divide and assign)
Example: x /= 2
Result: x = x / 2

🔢 Understanding the Integer (`int`) Data Type

What it is:

Integers (`int`) are used to store **whole numbers** – that means numbers without any decimal points. They can be positive, negative, or zero.

Example Use:

- Counting items (e.g., number of apples 🍎)
- Representing age (e.g., 12 years old 🎂)
- Scores in a game (e.g., 100 points 🏆)
- Temperature without decimals (e.g., -3 degrees ❄️)

Integers are fundamental for calculations that involve discrete quantities.

Python Example:

```
# Integer examples
age = 12
apples = 5
temperature = -3

# Operations with integers
total = apples + 10
print("Total apples:", total)
```

🔢 Numbers with Decimals (`float`) Data Type

What it is:

Floats (`float`) are used to store **numbers that have a decimal point**. They can represent fractions, measurements, or any value that requires precision beyond whole numbers.

Example Use:

- Measuring quantities (e.g., 📏 height, ⚖️ weight)
- Financial calculations (e.g., 💸 prices, interest rates)
- Scientific measurements (e.g., 🌡️ temperature, 🚀 speed)
- Averages (e.g., 📊 average score)

Python Example:

```
# Float examples
height = 4.5
feetprice = 12.99
temperature = -2.3

# Operations with floats
new_price = price * 2
print("Double price:", new_price)
```

Floats are essential when dealing with continuous or fractional values in your programs.

🔡 Text and Words (`str`) Data Type

What it is:

A string (`str`) is used to store **sequences of characters**. Think of it as a chain of individual letters, numbers, spaces, and symbols that form words, sentences, or even entire documents.

Example Use:

- Storing names (e.g., "Alice") 👩‍💻
- Holding messages (e.g., "Hello, world!") ✉️
- Representing addresses or URLs (e.g., "www.example.com") 🌐
- Displaying book titles or product descriptions 📖

Python Example:

```
# String examples
name = "Alice“
greeting = "Hello, " + name
print(greeting)

String slicing (getting parts of a string)
first_letter = name[0]
print("First letter:", first_letter)

String repetition (making copies)
laugh = "ha " * 3
print(laugh) # ha ha ha
```

Strings are fundamental for handling all forms of text data in your programs.

✅ Boolean (`bool`) Data Type

What it is:

A Boolean (`bool`) data type stores only two possible values: `True` or `False`. It's like a simple switch that can only be ON or OFF, representing logical states.

Example Use:

- Answering a "Yes/No" question (e.g., Is it raining? ☔)
- Checking if a user is logged in (e.g., `is_logged_in = True` 🔑)
- Determining if a condition is met (e.g., `age > 18` ✅)
- Controlling the flow of your program with decisions (e.g., `if/else` statements 🚦)

Booleans are crucial for making decisions and controlling the logic of your programs.

Python Example:

```python
# Boolean variable examples
is_student = True
is_winter = False

# Using booleans in conditional statements
if is_student:
    print("You get a student discount!")

if is_winter:
    print("Don't forget your coat!")

# Example: Combining booleans to make decisions
age = 20
is_citizen = True

# You can vote if you are at least 18 years old and a citizen
can_vote = (age >= 18) and is_citizen
print("Can vote:", can_vote)
```

5. NoneType (None) Data Type

What it is:

`NoneType` has only one possible value: `None`. It signifies the **absence of a value**, or that a variable or function result is intentionally empty or has not been assigned yet.

Example Use:

- Initializing a variable when you don't have a value yet.
- Representing a function that doesn't return anything meaningful.
- Indicating that a search or operation found no results.
- Default value for optional parameters in functions.

`None` is unique and is often used as a placeholder or a signal for "nothing" in programming logic.

Python Example:

```
# NoneType example
data = None
result = None

# Checking if a variable is None
if data is None:
    print("No data found!")  # ✅ This will print

# A function that may return None
def get_user_profile(user_id):
    if user_id == 123:
        return {"name": "Alice", "age": 30}
    else:
        return None  # User not found

# Trying to fetch a profile that doesn't exist
profile = get_user_profile(456)

if profile is None:
    print("User profile not found.")  # ✅ This will print
```

Mini Exercise: Meet Your Variables!

It's time to put your new knowledge of data types to practice!

Your Mission:

Create a Python program that stores the following information about yourself:

- Your Name (**string**)
- Your Age (**integer**)
- Your Height (**float**)
- Your Student Status (**boolean**)
- Your Favorite Color (**string**)

Finally, combine and print all these details in a single, friendly sentence!

This exercise helps you understand how different data types are used to represent various kinds of information in real-world scenarios.

Solution: Meet Your Variables!

Here's one way you could solve the mini exercise. This program assigns different types of information to variables and then combines them into a single output string.

```
# 📝 Assign your variables
my_name = "Alice"         # String: name
my_age = 10               # Integer: age
my_height = 4.5           # Float: height in feet
is_student = True         # Boolean: student status
fav_color = "Blue"        # String: favorite color

# 📢 Print a formatted introduction
print(f"Hi, I'm {my_name}, {my_age} years old, and {my_height} feet tall.")
print(f"It's {is_student} that I'm a student, and my favorite color is {fav_color}.")
```

This example demonstrates how each data type (string, integer, float, boolean) can store specific kinds of information, making your programs versatile!

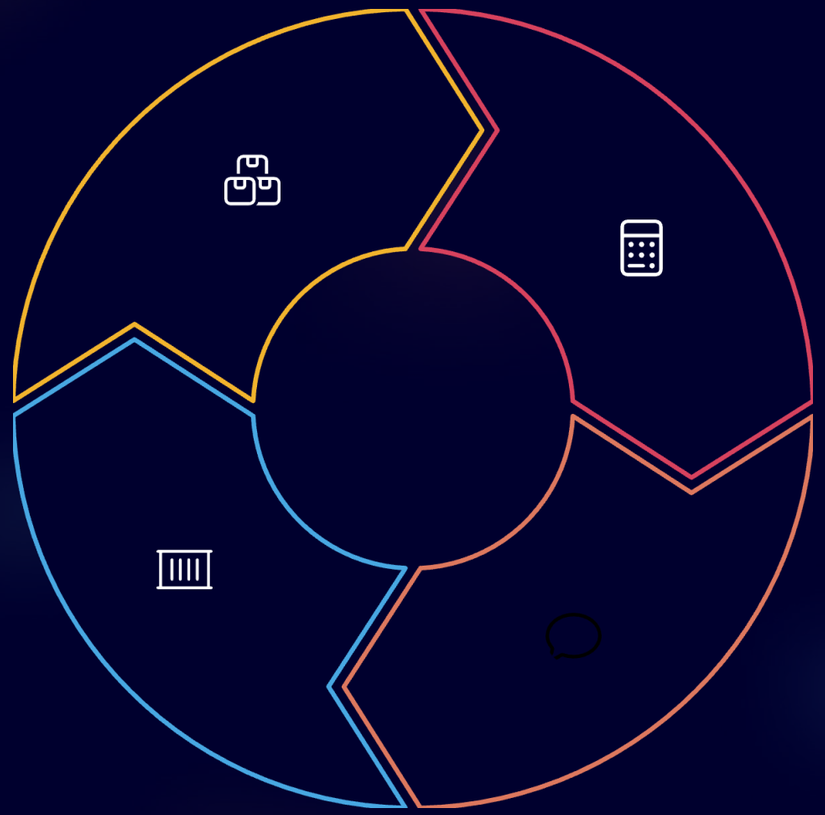
Let's Review What We Learned!

Data Types

Different containers for storing information

Operators

Symbols that perform actions on data



Complex Types

Collections like lists, tuples, and dictionaries

Primitive Types

Basic building blocks like int, float, bool, str

Practice Makes Perfect!

Try creating your own variables with different data types and use operators to see what happens!

Thank You!

Happy coding! 🚀