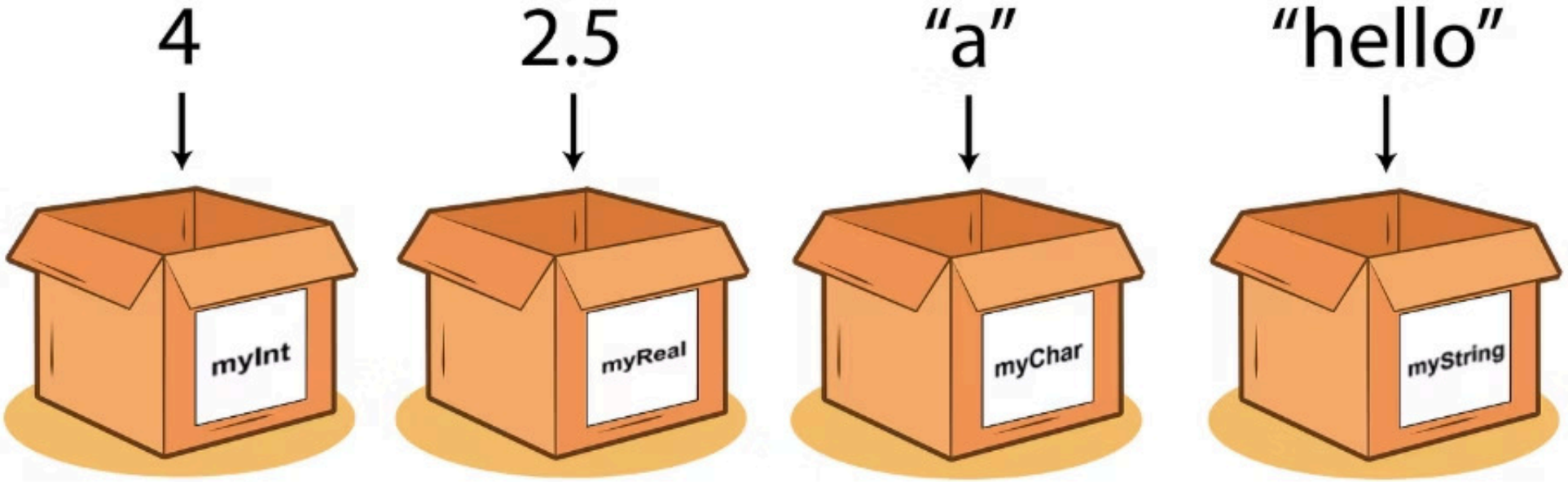
Python Class 2

Data Types, Type Casting, Operators & Variables

By the end of this session, you'll understand how Python handles different types of data and how to manipulate them effectively.

VARIABLES IN MEMORY



Primary Data Types

Python offers several built-in data types that help us store and work with different kinds of information:

- Numeric: `int`, `float`
- Text: `str` (strings)
- Boolean: `bool` (True/False)
- Set: `set`, `frozenset`

Python uses **dynamic typing**; variables can change types during execution.

This flexibility makes Python **beginner-friendly** but **requires attention** to type handling.

The `type()` Function

Checking Integer Type

```
>>> x = 42
>>> type(x)
```

Checking String Type

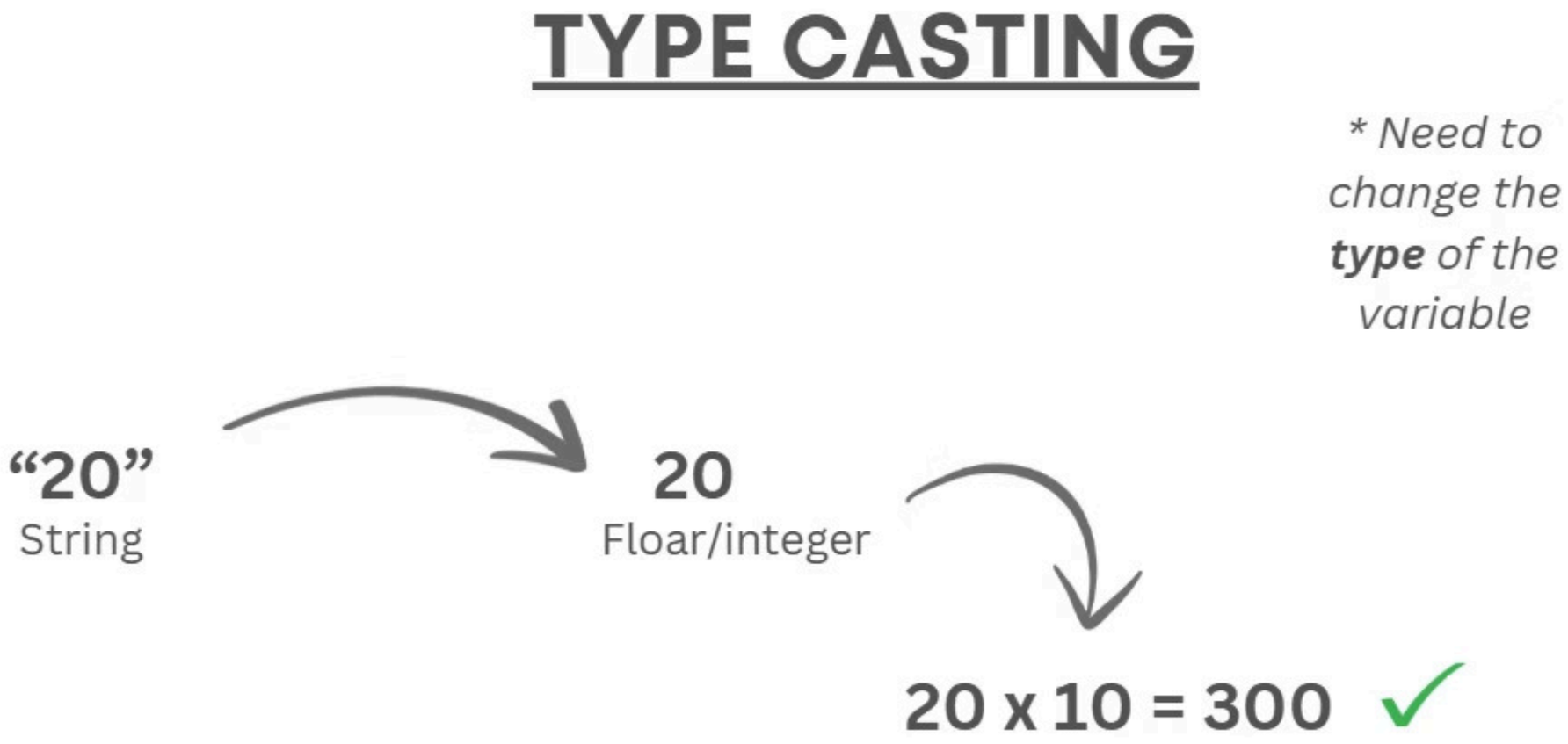
```
>>> name = "Python"
>>> type(name)
```

Checking Boolean Type

```
>>> is_fun = True
>>> type(is_fun)
```

Understanding a variable's type is crucial because it determines what operations you can perform. For example, you can multiply integers but not strings by other strings.

Type Casting: Converting Data Types



Implicit Conversion (Automatic)

```
>>> x = 10    # int
>>> y = 3.14  # float
>>> z = x + y # float (10.0 + 3.14)
>>> type(z)
```

Python automatically converts the integer to a float to perform the addition.

Explicit Conversion (Manual)

```
>>> num_str = "42"  # string
>>> num_int = int(num_str)  # convert to int
>>> num_float = float(num_str)  # convert to float
>>> str(3.14)  # convert float to string
'3.14'
```

Watch out for invalid conversions!

```
>>> int("hello") # ValueError!
```

Python automatically converts the integer to a float to perform the addition.

input() statement

input() #result for input is always a str

int(input()) #int

float(input()) #float

Operator Precedence in Python

1. Parentheses ()

Always evaluated first: `(2 + 3) * 4 = 20`

2. Exponentiation **

Power operations: `2 ** 3 = 8`

3. Multiplication *, Division /, Floor Division //, Modulus %

These operations have equal precedence: `10 * 2 / 5 = 4.0`

4. Addition +, Subtraction -

Lowest precedence: `5 + 2 * 3 = 11`

When in doubt, use parentheses to make your code more readable and ensure operations happen in the intended order.

Variable Naming Conventions & Rules

Rules (Must Follow)

- Must start with letter (a-z, A-Z) or underscore (_)
- Remaining characters can be letters, numbers, or underscores
- Case-sensitive (age ≠ Age ≠ AGE)
- Cannot use reserved keywords (`if`, `for`, etc.)

Conventions (Best Practices)

- Use snake_case for variables and functions (all lowercase with underscores)
- Choose descriptive names: `user_age` not `ua`
- Constants in ALL_CAPS: `MAX_ATTEMPTS = 3`

Assignment Operator =

Basic Assignment

```
name = "Python"
age = 30
is_programming = True
```

Multiple Assignment

```
# Assign same value to multiple variables
x = y = z = 0

# Assign different values simultaneously
width, height = 640, 480
```

Augmented Assignment

```
counter = 0
counter += 1  # Same as:
counter = counter + 1
```

Other operators: -=, *=, /=, %=, etc.

Remember that = is assignment, not equality testing. For equality comparison, use ==.

Quick Recap: Key Concepts Covered

1 Data Types

Python has built-in types like `int`, `float`, `str`, and `bool`. Use `type()` to identify any value's type.

2 Type Casting

Convert between types using functions like `int()`, `float()`, and `str()`. Be careful with invalid conversions.

3 Operator Precedence

Python follows PEMDAS (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction). Use parentheses to control order.

4 Variables

Follow naming rules, use descriptive names, and assign values with the `=` operator.

Exercise: Practice Your Skills

Your Task:

1. Calculate: `5 + 3 * 2 - (4 / 2)`
2. Ask the user to input temperature in **Celsius**, convert it to **Fahrenheit** using the formula:

$$\text{Fahrenheit} = \left(\text{Celsius} \times \frac{9}{5}\right) + 32$$

Wrap-Up & Homework

Review Notes

Go through today's examples and make sure you understand each concept. Experiment with code in the Python interpreter.

Prepare Questions

Note any questions or confusion that arise while practicing, and we'll address them at the beginning of our next class.

Remember: these fundamentals form the foundation of all Python programming. Mastering them now will make future concepts much easier to learn!