Chapter 1: Python Fundamentals

Lesson 2: Variables and Data types

---

1. What Are Variables?

Variables are like labeled containers in Python that store data values. Python is dynamically typed, meaning you don't need to declare a variable's type explicitly—it figures it out based on the value you assign.

Declaring a Variable:

- Use a name (e.g., `age`, `name`) followed by an equals sign (`=`) and a value.
- Example:

```
age = 25
name = "Alice"
```

Rules for Variable Names:

- Must start with a letter (a-z, A-Z) or an underscore (_).
- Can contain letters, numbers (0-9), and underscores.
- Case-sensitive (`Age` and `age` are different).
- Avoid using Python keywords (e.g., `if`, `for`, `class`).

---

2. Data Types in Python

Python has several built-in data types that define the kind of data a variable can hold. Let's explore the most common ones:

1. **Integer (`int`)**
    - Whole numbers, positive or negative, without decimals.
    - Example:

```
x = 10
y = -5
print(type(x))  # Output: <class 'int'>
```

2. **Float (`float`)**
    - Numbers with decimal points or in scientific notation.
    - Example:

```
pi = 3.14
temp = -2.5
sci_notation = 1.2e3  # 1200.0
print(type(pi))  # Output: <class 'float'>
```

3. **Complex (`complex`)**
    - Numbers with a real and imaginary part (used in advanced math).
    - Example:

```
z = 3 + 4j
print(type(z))  # Output: <class 'complex'>
```

4. **String (`str`)**
    - A sequence of characters enclosed in single (`'`) or double (`"`) quotes.
    - Example:

```
name = "Bob"
greeting = 'Hello, world!'
multiline = """This is
a multi-line
string."""
print(type(name))  # Output: <class 'str'>
```

5. **Boolean (`bool`)**
    - Represents `True` or `False`, often used in conditions.
    - Example:

```
is_active = True
is_empty = False
print(type(is_active))  # Output: <class 'bool'>
```

6. **NoneType (`None`)**
    - Represents the absence of a value or null.
    - Example:

```
result = None
print(type(result))  # Output: <class 'NoneType'>
```

---

3. Checking and Converting Data Types

- **Check Type:** Use `type()` to identify a variable's data type (as shown in examples above).

- **Convert Types:** Python allows type casting using functions like `int()`, `float()`, `str()`, etc.
    - Example:

```
num_str = "123"
num_int = int(num_str)  # Convert string to integer
print(num_int + 5)  # Output: 128

float_num = float("3.14")  # Convert string to float
print(float_num)  # Output: 3.14

num = 10
num_str = str(num)  # Convert integer to string
print("Value: " + num_str)  # Output: Value: 10
```

---

4. Practical Examples

Let's tie this together with a small program:

```
# Variables and mixed data types
name = "Charlie"  # String
age = 30  # Integer
height = 5.9  # Float
is_student = True  # Boolean

# Display information
print("Name: " + name + ", Type: " + str(type(name)))
print("Age: " + str(age) + ", Type: " + str(type(age)))
print("Height: " + str(height) + ", Type: " + str(type(height)))
print("Is Student? " + str(is_student) + ", Type: " + str(type(is_student)))
```

Output:

```
Name: Charlie, Type: <class 'str'>
Age: 30, Type: <class 'int'>
Height: 5.9, Type: <class 'float'>
Is Student? True, Type: <class 'bool'>
```

---

5. Practice Exercises

1. Create three variables: `first_name` (a string), `age` (an integer), and `height` (a float). Assign them values and print each one.
2. Assign your favorite number to a variable called `fav_number` as an integer. Then convert it to a string and print: "My favorite number is: " followed by the number.
3. Create a variable `temp` with a float value (e.g., 23.5). Print it with the message: "Today's temperature is: " followed by the value.

4. Create a variable `is_raining` and set it to `True`. Print "It is raining: " followed by the value of `is_raining`. Then change it to `False` and print again.
5. Assign these values to variables: `x = 15`, `y = 3.14`, `z = "Python"`. Use `type()` to check and print the data type of each variable.
6. Assign `price = 19.99` (a float). Convert it to an integer and print both the original and converted values with labels like "Original: " and "Converted: ".
7. Create variables: `name = "Alex"`, `score = 85`, `active = True`. Print a sentence using concatenation, e.g., "Alex has a score of 85 and active status is True".
8. Create a variable `data = None`. Print its value and its type using `type()` with a message like "Data value: " and "Data type: ".