Data Structures in Python: Atomic Data Types

John Karuitha, PhD

February 27, 2025

1 Data Structures in Python: Integers, Floats, Booleans, and Strings

In Python, **data structures** refer to how data is stored, organized, and manipulated. Python has several built-in data types that help you work with different kinds of information. In this lesson, we will focus on four fundamental types: **integers, floats, booleans, and strings**.

---

2 1. Integers (`int`)

2.1 Definition:

Integers are whole numbers that can be positive, negative, or zero, without any decimal points. They belong to the `int` data type.

2.2 Example:

```
age = 25   # Positive integer
temperature = -10  # Negative integer
count = 0   # Zero
```

2.3 Methods and Operations for Integers:

- **Arithmetic Operations:** Python supports basic arithmetic operations with integers such as addition (`+`), subtraction (`-`), multiplication (`*`), division (`/`), and more.

```
a = 10
b = 5
print(a + b)  # Output: 15 (addition)
print(a - b)  # Output: 5 (subtraction)
print(a * b)  # Output: 50 (multiplication)
print(a / b)  # Output: 2.0 (division, note the result is a float)
```

- **`abs()` Function:** Returns the absolute value of an integer (removes the negative sign).

```
print(abs(-5))  # Output: 5
```

- **`pow()` Function:** Raises an integer to a certain power (exponentiation).

```
print(pow(2, 3))  # Output: 8 (2 to the power of 3)
```

- **Type Conversion Methods:** You can convert an integer to other types using functions like `float()` or `str()`.

```
num = 10
print(float(num))  # Output: 10.0 (converted to a float)
print(str(num))    # Output: '10' (converted to a string)
```

---

3 2. Floats (`float`)

3.1 Definition:

Floats (floating-point numbers) are numbers that contain decimal points. They are used when more precision is required, such as in measurements or financial data.

3.2 Example:

```
price = 19.99  # Float value
height = 5.7   # Float value
```

3.3 Methods and Operations for Floats:

- **Arithmetic Operations:** Just like integers, floats can be added, subtracted, multiplied, or divided.

  ```
  x = 5.5
  y = 2.0
  print(x + y)  # Output: 7.5
  print(x * y)  # Output: 11.0
  ```

- **`round()` Function:** This rounds a float to a specified number of decimal places.

  ```
  value = 3.14159
  print(round(value, 2))  # Output: 3.14 (rounded to 2 decimal places)
  ```

- **`int()` Function:** Converts a float to an integer, but this truncates the decimal part.

  ```
  print(int(4.9))  # Output: 4
  ```

---

4 3. Booleans (`bool`)

4.1 Definition:

Booleans represent two values: `True` or `False`. They are commonly used in conditions and comparisons.

4.2 Example:

```
is_active = True
is_student = False
```

4.3 Boolean Expressions and Methods:

- **Comparison Operators:** Booleans often result from comparisons, using operators like `==`, `!=`, `>`, `<`, `>=`, `<=`.

```
x = 10
y = 5
print(x > y)  # Output: True
print(x == y) # Output: False
```

- **Logical Operators:** You can combine boolean values using logical operators like `and`, `or`, and `not`.

```
a = True
b = False
print(a and b)  # Output: False (both must be True)
print(a or b)   # Output: True (one of them is True)
print(not a)    # Output: False (negates True to False)
```

- `bool()` **Function:** Converts other data types into a boolean. Any non-zero value or non-empty string is `True`; zero, empty lists, or `None` are `False`.

```
print(bool(1))       # Output: True
print(bool(0))       # Output: False
print(bool("hello")) # Output: True
print(bool(""))      # Output: False
```

---

5 4. Strings (`str`)

5.1 Definition:

Strings are sequences of characters enclosed in single (`'...'`) or double quotes (`"..."`). They are used for representing text.

5.2 Example:

```
name = "John"
greeting = 'Hello, World!'
```

5.3 Common String Methods:

- **`len()` Function:** Returns the length of the string (number of characters).

```
print(len("Python"))  # Output: 6
```

- **`lower()` and `upper()` Methods:** Convert the string to lowercase or uppercase.

```
text = "Hello"
print(text.lower())  # Output: "hello"
print(text.upper())  # Output: "HELLO"
```

- **`strip()` Method:** Removes leading and trailing spaces from the string.

```
message = "  Hello!  "
print(message.strip())  # Output: "Hello!"
```

- **String Concatenation:** You can combine two or more strings using the `+` operator.

```
first_name = "John"
last_name = "Doe"
full_name = first_name + " " + last_name
print(full_name)  # Output: John Doe
```

- **`replace()` Method:** Replaces a part of the string with another substring.

```
text = "Hello, World!"
print(text.replace("World", "Python"))  # Output: Hello, Python!
```

- **`split()` Method:** Splits the string into a list of substrings based on a delimiter (default is space).

```
sentence = "Python is fun"
words = sentence.split()
print(words)  # Output: ['Python', 'is', 'fun']
```

- **String Formatting:** You can format strings using the `format()` method or f-strings.

```
name = "Alice"
age = 25
print("My name is {} and I am {} years old.".format(name, age))
# Output: My name is Alice and I am 25 years old.

print(f"My name is {name} and I am {age} years old.")
# Output: My name is Alice and I am 25 years old.
```

---

6 Conclusion

Understanding the basic data types in Python—integers, floats, booleans, and strings—is critical as they are the foundation for building more complex programs. Each data type has its own set of methods and functions that allow you to manipulate and interact with the data. As you progress, you will frequently encounter these types and their operations, so practicing them is essential for mastering Python programming (Downey 2024; McKinney 2022).

References

Downey, Allen B. 2024. *Think Python*. 3rd ed. Green Tea Press.
McKinney, Wes. 2022. *Python for Data Analysis*. " O'Reilly Media, Inc.".