Python Fundamentals

Learning Objectives

By the end of this section, you will be able to:

- Create and use variables with different data types
- Perform operations using Python operators
- Accept user input and display output
- Write effective comments and documentation
- Convert between different data types

Variables and Data Types

In Python, variables are containers for storing data values. Unlike some programming languages, Python has no command for declaring a variable—a variable is created when you first assign a value to it.

Creating Variables

```python
# Creating variables of different types
name = "Alice"          # String
age = 25                # Integer
height = 1.75           # Float
is_student = True       # Boolean
```

Variables in Python don't need explicit type declarations. Python determines the variable type based on the assigned value.

Naming Rules

Follow these rules when naming variables:

- Names can contain letters, numbers, and underscores
- Names must start with a letter or underscore
- Names are case-sensitive (age and Age are different variables)
- Names cannot be Python keywords (like `if`, `for`, `class`, etc.)

```python
# Valid variable names
user_name = "John"
_count = 1
score1 = 95.5

# Invalid variable names
# 1user = "John"    # Cannot start with a number
# if = "keyword"    # Cannot use a Python keyword
```

Common Data Types

1. Integers

Whole numbers without a decimal point.

```
age = 25
count = -10
zero = 0

# Large integers are handled automatically
population = 7800000000
```

2. Floating-Point Numbers (Floats)

Numbers with a decimal point.

```
height = 1.75
temperature = -2.5
pi_approximate = 3.14159

# Scientific notation
electron_mass = 9.1e-31  # 9.1 × 10^-31
```

3. Strings

Sequences of characters, enclosed in single or double quotes.

```python
name = "Alice"
message = 'Hello, Python!'

# Multi-line strings use triple quotes
address = """123 Main Street
Anytown, USA
12345"""
```

4. Booleans

Represent truth values: `True` or `False`.

```
is_active = True
is_completed = False

# Boolean expressions
is_adult = age >= 18
is_valid = name != ""
```

5. None Type

Represents the absence of a value.

```python
result = None  # Used when you need a placeholder
```

Checking Data Types

Use the `type()` function to check the data type of a variable:

```python
name = "Alice"
age = 25
height = 1.75
is_student = True

print(type(name))       # <class 'str'>
print(type(age))        # <class 'int'>
print(type(height))     # <class 'float'>
print(type(is_student))  # <class 'bool'>
```

Basic Operators and Expressions

Arithmetic Operators

```
a = 10
b = 3

sum_result = a + b           # Addition: 13
difference = a - b           # Subtraction: 7
product = a * b              # Multiplication: 30
quotient = a / b             # Division: 3.3333... (returns float)
integer_division = a // b    # Floor division: 3 (returns integer)
remainder = a % b            # Modulus (remainder): 1
power = a ** b               # Exponentiation: 1000 (10^3)

# Multiple operations
result = 5 + 3 * 2           # 11 (multiplication has precedence)
result = (5 + 3) * 2         # 16 (parentheses change precedence)
```

Assignment Operators

```
x = 10                      # Basic assignment

# Combined assignment operators
x += 5                      # x = x + 5 (x is now 15)
x -= 3                      # x = x - 3 (x is now 12)
x *= 2                      # x = x * 2 (x is now 24)
x /= 4                      # x = x / 4 (x is now 6.0)
x %= 4                      # x = x % 4 (x is now 2.0)
x **= 3                     # x = x ** 3 (x is now 8.0)
x //= 3                     # x = x // 3 (x is now 2.0)
```

Comparison Operators

```
a = 10
b = 5

print(a == b)  # Equal to: False
print(a != b)  # Not equal to: True
print(a > b)   # Greater than: True
print(a < b)   # Less than: False
print(a >= b)  # Greater than or equal to: True
print(a <= b)  # Less than or equal to: False
```

Logical Operators

```python
x = True
y = False

print(x and y)  # Logical AND: False (both must be True)
print(x or y)   # Logical OR: True (at least one must be True)
print(not x)    # Logical NOT: False (inverts the value)

# Complex conditions
age = 25
income = 50000
is_eligible = (age > 18) and (income > 30000)  # True
```

Identity Operators

```python
a = [1, 2, 3]
b = [1, 2, 3]
c = a

print(a is c)     # True (a and c refer to the same object)
print(a is b)     # False (a and b are equal but different objects)
print(a is not b) # True
```

Membership Operators

```python
fruits = ["apple", "banana", "cherry"]

print("apple" in fruits)        # True
print("orange" in fruits)       # False
print("orange" not in fruits)  # True
```

Input and Output Operations

Output with `print()`

```python
# Basic print
print("Hello, Python!")

# Printing multiple items
name = "Alice"
age = 25
print("Name:", name, "Age:", age)

# Printing with end parameter (default is newline)
print("Hello", end=" ")
print("World")  # Prints "Hello World" on one line

# Formatting with f-strings (Python 3.6+)
print(f"My name is {name} and I am {age} years old.")
```

```python
# Formatting with format() method
print("My name is {} and I am {} years old.".format(name, age))

# Formatting with positional arguments
print("My name is {0} and I am {1} years old.".format(name, age))

# Formatting with named arguments
print("My name is {n} and I am {a} years old.".format(n=name, a=age))

# Older style string formatting (still works)
print("My name is %s and I am %d years old." % (name, age))
```

Input with `input()`

The `input()` function reads a line from the user and returns it as a string.

```python
# Basic input
name = input("Enter your name: ")
print(f"Hello, {name}!")

# Getting numeric input (remember to convert the string)
age_str = input("Enter your age: ")
age = int(age_str)
print(f"In five years, you'll be {age + 5} years old.")

# Combined in one step
height = float(input("Enter your height in meters: "))
print(f"Your height is {height} meters.")
```

Comments and Documentation

Single-line Comments

```python
# This is a single-line comment
x = 10  # This comment is after code
```

Multi-line Comments

```python
# This is a multi-line comment using multiple single-line comments
# Python doesn't have a specific multi-line comment syntax
# So we just use multiple hash signs

"""
This is also used for multi-line comments
Although technically it's a string that isn't assigned to a variable
This style is often used for multi-line comments
"""
```

Docstrings

Docstrings are strings used to document functions, classes, and modules.

```python
def calculate_area(length, width):
    """
    Calculate the area of a rectangle.

    Parameters:
        length (float): The length of the rectangle
        width (float): The width of the rectangle

    Returns:
        float: The area of the rectangle
    """
    return length * width

# You can access docstrings using the __doc__ attribute
print(calculate_area.__doc__)
```

Type Conversion

Python allows you to convert between different data types using built-in functions.

Common Type Conversion Functions

```python
# String to integer
age_str = "25"
age = int(age_str)
print(type(age))  # <class 'int'>

# String to float
height_str = "1.75"
height = float(height_str)
print(type(height))  # <class 'float'>

# Integer/float to string
score = 95
score_str = str(score)
print(type(score_str))  # <class 'str'>

# Integer to float
count = 10
count_float = float(count)
print(count_float)  # 10.0

# Float to integer (truncates decimal part)
price = 29.95
price_int = int(price)
print(price_int)  # 29 (decimal part is truncated, not rounded)

# To boolean
# Zero values (0, 0.0, ""), empty containers, and None convert to False
```

Type Conversion Pitfalls

```python
# Invalid conversions raise errors
# int("hello")    # ValueError: invalid literal for int()
# float("text")   # ValueError: could not convert string to float

# Be careful with float precision
result = 0.1 + 0.2
print(result)          # 0.30000000000000004 (not exactly 0.3 due to float representation)
print(round(result, 2))  # 0.3 (rounded to 2 decimal places)

# Converting to integer truncates (doesn't round)
print(int(9.9))        # 9 (not 10)
```

Practice Exercises

Exercise 1: Variable Creation and Data Types

Create a script that:

1. Defines variables for your name, age, height (in meters), and whether you are a student
2. Prints each variable and its type
3. Creates a multi-line string with your favorite quote

```python
# Exercise 1 template
name = "Your Name"
age = 0  # Replace with your age
height = 0.0  # Replace with your height
is_student = False  # Change to True if you are a student

# Print variables and their types
print("Name:", name, "Type:", type(name))
```