# Variables, Datatypes and Operators in Python

By

Prof. Muhammad Iqbal Bhat

Government Degree College Beerwah

# Topics

VARIABLES IN PYTHON

DATA TYPES IN PYTHON:

OPERATORS IN PYTHON

# Variables in Python:

Variables are essential components of programming languages. They are used to store data in a program.

Python is a dynamically typed language, which means that variables are not bound to a specific data type.

# Variable Naming Rules

Variable names can only contain letters (Capital and Small letters), numbers(0-9), and underscores (_).

They must start with a letter or an underscore.

Variable names are case-sensitive.

Variable names should be descriptive and not start with a capital letter (unless it's class).

# Variable Naming Rules (Examples):

| Rule | Example |
|---|---|
| Variable names can only contain letters, numbers, and underscores. | my_var, my_var2, my_var_3 |
| They must start with a letter or an underscore. | my_var, _my_var, MyVar |
| Variable names are case sensitive. | my_var, My_Var, MY_VAR |
| Variable names should be descriptive and not start with a capital letter (unless it's a class). | age, my_age, person_age |
| Variable names should not be a Python keyword. | person_name, my_list, my_tuple |

| Example | Reason for Incorrectness |
|---|---|
| 2_numbers | Starts with a number |
| my-var | Contains a hyphen |
| My_Var | Starts with an uppercase letter |
| class | Uses a Python keyword as the variable name |

| Example | Reason for Correctness |
|---|---|
| age | Only contains letters |
| _my_var | Starts with an underscore |
| my_list | Descriptive and does not start with an uppercase letter |
| person_name | Descriptive and does not use a Python keyword |

# Data types in Python:

# Data types in Python:

Integers: whole numbers without decimal points.

Floating-point numbers: numbers with decimal points.

Strings: text enclosed in quotation marks.

Boolean values: True or False.

# Data types in Python

| Data Type | Description | Example |
|---|---|---|
| Integer | Whole numbers, positive or negative. | 1, -5, 1000 |
| Float | Decimal numbers. | 1.5, -3.14, 0.0001 |
| String | A sequence of characters enclosed in quotes. | "Hello", 'world', "123" |
| Boolean | A value that is either True or False. | True, False |
| List | An ordered collection of items, enclosed in square brackets and separated by commas. | [1, 2, 3], ['apple', 'banana', 'orange'], [True, False, True] |
| Tuple | Similar to a list, but enclosed in parentheses and cannot be modified once created. | (1, 2, 3), ('apple', 'banana', 'orange'), (True, False, True) |
| Dictionary | An unordered collection of key-value pairs, enclosed in curly braces and separated by commas. | {'name': 'John', 'age': 25, 'city': 'New York'} |

# Operators in Python:

# Operators in Python:

Operators are used to performing operations on variables and values.

Python has several types of operators, including arithmetic, comparison, and logical operators.

Arithmetic Operators

Arithmetic operators are used to perform mathematical operations.

Examples include addition (+), subtraction (-), multiplication (*), division (/), modulo (%), and exponentiation (**).

Comparison Operators:

Comparison operators are used to compare two values.

Examples include equal to (==), not equal to (!=), greater than (>), less than (<), greater than or equal to (>=), and less than or equal to (<=).

Logical Operators:

Logical operators are used to perform logical operations.

Examples include AND (and), OR (or), and NOT (not).

Assignment Operators

Assignment operators are used to assign a value to a variable.

Examples include assign (=), add and assign (+=), subtract and assign (-=), multiply and assign (*=), divide and assign (/=), modulo and assign (%=), and exponentiate and assign (**=).

# Operators in Python:

| Operator | Description | Example |
|---|---|---|
| Arithmetic Operators | Used to perform mathematical operations. | +, -, *, /, %, **, // |
| Comparison Operators | Used to compare two values. | ==, !=, <, >, <=, >= |
| Assignment Operators | Used to assign values to variables. | =, +=, -=, *=, /=, %=, **=, //= |
| Logical Operators | Used to combine multiple conditions. | and, or, not |
| Identity Operators | Used to compare the memory locations of two objects. | is, is not |
| Membership Operators | Used to check if a value is a member of a sequence. | in, not in |
| Bitwise Operators | Used to perform bitwise operations on integers. | &,\|, ` |

# Operator Precedence:

- Python follows a specific order of operations when evaluating expressions.
- Parentheses can be used to override the default order of operations.
- PEMDAS (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction) is a useful mnemonic for remembering the order of operations.

# Operator Precedence:

| Precedence | Operator | Description | Example |
|---|---|---|---|
| 1 | ( ) | Parentheses (grouping) | (3 + 4) * 2 evaluates to 14 |
| 2 | ** | Exponentiation | 3 ** 2 evaluates to 9 |
| 3 | *, /, //, % | Multiplication, division, floor division, modulus | 5 * 2, 10 / 3, 10 // 3, 10 % 3 |
| 4 | +, - | Addition, subtraction | 3 + 4, 5 - 2 |
| 5 | <, <=, >, >= | Comparison operators | 3 < 5, 4 >= 3 |
| 6 | ==, != | Equality operators | 3 == 3, 4 != 3 |
| 7 | not | Logical NOT | not True evaluates to False |
| 8 | and | Logical AND | True and False evaluates to False |
| 9 | or | Logical OR | True or False evaluates to True |

# Questions?