Learn Python Programming from Scratch

Topic: Variables in Python

1. What is a Variable?

In simple terms, a **variable** is a named storage location that holds a value. Think of it like a container or a box with a label on it. The label is the variable's name, and what's inside the box is its value. Python is a dynamically typed language, which means you don't need to declare the variable type explicitly. Python figures it out for you!

2. Creating Variables

Creating a variable is a straightforward process. You simply write the variable's name, followed by an equals sign (`=`), and then the value you want to assign to it.

```
# Creating a variable named 'x' and assigning it the integer value 10
x = 10

# Creating a variable named 'greeting' and assigning it the string "Hello, Python!"
greeting = "Hello, Python!"

# Creating a variable named 'pi' and assigning it the float value 3.14
pi = 3.14

# You can also assign multiple variables in one line
a, b, c = 1, 2, 3

print(f"The value of x is: {x}")
print(f"The greeting is: {greeting}")
print(f"The value of pi is: {pi}")
print(f"The values of a, b, and c are: {a}, {b}, {c}")
```

3. Variable Naming Rules

Follow these rules to create valid variable names:

- **Start with a letter or an underscore (`_`)**: A variable name cannot start with a number.
- **Contain only alphanumeric characters and underscores**: This includes letters (A-Z, a-z), numbers (0-9), and the underscore (`_`).
- **Case-sensitive**: `my_variable` and `My_Variable` are two different variables. It's a common practice to use **snake_case** (all lowercase words separated by underscores) for variable names.

- **Avoid reserved keywords**: Don't use Python's built-in keywords (like `if`, `for`, `while`, `class`, etc.) as variable names.

```
# Valid variable names
my_name = "Alice"
_private_var = 100
score_2024 = 95

# Invalid variable names (will raise a SyntaxError)
# 1variable = "invalid"
# my-variable = "invalid"
# for = "invalid" # 'for' is a reserved keyword
```

4. Reassigning Variables

You can change the value of a variable at any time after it's been created. This is known as **reassignment**.

```
# Initial assignment
counter = 10
print(f"Initial value of counter: {counter}")

# Reassigning the variable with a new value
counter = counter + 5
print(f"New value after reassignment: {counter}")

# You can also change the data type of the variable
counter = "Now I'm a string"
print(f"After changing data type: {counter}")
```

5. Checking Variable Type

Python provides the built-in `type()` function to check the data type of a variable. This is especially useful for debugging.

```
my_number = 42
my_string = "Data Scientist"
my_list = [1, 2, 3]

print(f"Type of my_number: {type(my_number)}")
print(f"Type of my_string: {type(my_string)}")
print(f"Type of my_list: {type(my_list)}")
```

6. Deleting a Variable

You can remove a variable from memory using the `del` keyword.

```
my_variable_to_delete = 99
print(f"Variable exists: {my_variable_to_delete}")
```

```
# Now, let's delete it
del my_variable_to_delete

# This will now raise a NameError because the variable no longer
exists
# print(my_variable_to_delete)
```

Exercises

1. Create a variable `name` and assign your name to it.
2. Create a variable `age` and assign your age to it.
3. Print a sentence combining both variables, like "My name is [name] and I am [age] years old."
4. Reassign the `age` variable to your age next year.
5. Check the type of both variables using the `type()` function.

Practical Examples

Let's explore some practical examples of working with variables in Python. These examples demonstrate the fundamental concepts we've discussed.

Variable Assignment Examples

Here are some basic examples of creating and assigning values to variables. Python automatically determines the data type based on the value you assign.

In [1]:
```
# Basic variable assignments demonstrating different data types

# String variable - stores text data
name = "Alice"

# Integer variable - stores whole numbers
age = 30

# Float variable - stores decimal numbers
height = 5.6

# Boolean variable - stores True/False values
is_student = True

# Display the variables and their values
print("Name:", name)
print("Age:", age)
print("Height:", height)
print("Is student:", is_student)
```

```
Name: Alice
Age: 30
Height: 5.6
Is student: True
```

Key Variable Rules to Remember

Let's review the important rules and best practices for working with variables:

- You don't need to declare variable types explicitly - Python infers them automatically
- Use the equals sign (=) to assign values to variables
- Variable names should start with a letter or underscore (_)
- Variable names can contain letters, numbers, and underscores
- Variable names are case-sensitive ('age' and 'Age' are different)
- Avoid using Python reserved keywords as variable names
- Make variable names descriptive for better code readability
- Use underscores to separate words in variable names (snake_case)
- Use lowercase letters for variable names (convention)
- Avoid single-character names except for counters/iterators in loops
- You can reassign variables to new values at any time

In [2]:
```
# Examples of good variable naming:
student_name = "John"
total_score = 95
is_passed = True
user_email = "john@example.com"

print("Student Name:", student_name)
print("Total Score:", total_score)
print("Passed:", is_passed)
print("Email:", user_email)
```

```
Student Name: John
Total Score: 95
Passed: True
Email: john@example.com
```

Course Information

Learn Python Programming from Scratch

Author: Prakash Ukhalkar

Topic: Python Fundamentals - Variables and Data Types

Built with ❤️ for the Python learning community