Variable

Kittikun Jitpairod

Introduction

In computer programming, variables act as containers that store data in a program. They allow for easy management of information throughout a program's run. Python uses a system called dynamic typing, which figures out the type of data in a variable based on what is stored in it. This feature helps make Python easy for new programmers to learn.

Variable Naming Rules

When creating variables in Python, certain naming rules must be followed:

1. The first character must be a letter (a-z, A-Z) or an underscore (_).
2. After the first character, the name can include letters, numbers, and underscores.
3. Variable names are case-sensitive. This means `age`, `Age`, and `AGE` are three different variables.
4. Python keywords, which are special words used by Python, cannot be used as variable names.

- **Examples of valid variable names:** `name`, `age`, `student_count`, `_private_var`, `total2`
- **Examples of invalid variable names:** `2total` (starts with a number), `my-var` (has a hyphen), `for` (Python keyword)

While Python allows variables to start with an underscore, this is generally avoided unless there's a specific reason to do so.

Variable Assignment

Storing data in variables is called assignment. In Python, this is done using the equals sign (=). The basic format is:

```
variable_name = value
```

Here are some examples:

```
x = 5
name = "Alice"
pi = 3.14159
```

These examples create three variables:

- x is given the integer value 5
- name is given the string value “Alice”
- pi is given the decimal value 3.14159

Multiple Assignment

Python allows assigning values to multiple variables in one line of code. This can be done in two main ways:

```
a, b, c = 1, 2, 3
print(f"a = {a}, b = {b}, c = {c}")
```

```
a = 1, b = 2, c = 3
```

This gives a the value 1, b the value 2, and c the value 3.

Another way is to give the same value to multiple variables at once:

```
a = b = c = 0
print(f"a = {a}, b = {b}, c = {c}")
```

```
a = 0, b = 0, c = 0
```

This sets x, y, and z all equal to 0.

While multiple assignment can make code shorter, it should be used carefully to keep the code easy to read.

Variable Types

Python is a dynamically typed language, which means you don't need to declare the type of a variable when you create it. The interpreter infers the type based on the value assigned. However, understanding the basic types is crucial for effective programming.

1. Numeric Types:
    - int: Integers → `5`, `-3`, `0`
    - float: Floating-point numbers → `3.14`, `-0.001`, `2.0e-4`
    - complex: Complex numbers → `3+4j`
2. Sequence Types:
    - str: Strings → `"Hello"`, `'Python'`
    - list: Mutable sequences → `[1, 2, 3]`
    - tuple: Immutable sequences → `(1, 2, 3)`

Python is a dynamically typed language, which means you don't need to declare the type of a variable when you create it. The interpreter infers the type based on the value assigned. However, understanding the basic types is crucial for effective programming.

3. Mapping Type:
    - dict: Key-value pairs → `{"name": "John", "age": 30}`
4. Set Types:
    - set: Unordered collection of unique elements → `{1, 2, 3}`
    - frozenset: Immutable version of set
5. Boolean Type:
    - bool: `True` or `False`
6. None Type:
    - None: Represents the absence of a value

Check Variable Types

You can check the type of a variable using the `type()` function:

```
age = 25          # integer (int)
height = 1.75     # floating-point number (float)
name = "Bob"      # string (str)
is_student = True # boolean (bool)

print(type(age))
print(type(height))
print(type(name))
print(type(is_student))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

Variable Types Conversion

Python also allows type conversion between compatible types:

```
a = int(3.14)    # Converts float to int: 3
b = float(5)     # Converts int to float: 5.0
c = str(42)      # Converts int to str: "42"
d = bool(1)      # Converts int to bool: True

print(type(a))
print(type(b))
print(type(c))
print(type(d))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

Variable Reassignment

A useful feature of Python's dynamic typing is the ability to change both the value and the type of a variable. For example:

```
x = 5
print(x)
x = "Hello"
print(x)
```

```
5
Hello
```

Here, x starts as an integer but is then changed to a string. While this flexibility can be useful, it should be used thoughtfully to avoid confusion.

Variable Type Annotation

Variable type annotation, introduced in Python 3.5, allows programmers to clearly state the expected types of variables. This practice makes code easier to read, improves documentation, and helps find potential errors before the program runs.

Examples of type annotations in Python:

```
name: str = "Alice"
age: int = 30
height: float = 1.75
is_student: bool = True
print(f"{age}: type {type(age)}")
```

```
30: type <class 'int'>
```

The annotation has nothing to do with the data type assigned.

```
age: int = 30.5
print(f"{age}: type {type(age)}")
```

```
30.5: type <class 'float'>
```