**Variable and datatypes in Python**

**Python Variables**

In Computer programming, a variable acts as a storage unit designed to hold data.

**For example,**

```python

age = 25

print(age) # Output: 25

age = 30

print(age) # Output: 30

```

In this scenario, "age" acts as our variable, storing different values (25 and 30) at different times. Variables, like containers, hold data that can change over time, making them indispensable in programming.

**Best Practices for Naming Python Variables**

Here are some guidelines to follow when naming your variables:

* **Use Alphanumeric Characters and Underscores:** Variables can consist of letters (both uppercase and lowercase), digits, and underscores. For instance, you can name a variable my\_variable, temperature, or user\_age.
* **Choose Meaningful Names:** Select names that accurately describe the data or functionality they represent.

For example, num\_students is more descriptive than n, making your code easier to understand.

* + **Follow Naming Conventions:** Python offers several naming conventions, including:
    - **snake\_case:** Words are separated by underscores, e.g., my\_variable.
    - **MACRO\_CASE:** All letters are uppercase, with words separated by underscores, often used for constants, e.g., MAX\_VALUE.
    - **camelCase:** Words are concatenated without spaces, with each word's initial letter capitalized except for the first, e.g., myVariable.
    - **CapWords (or PascalCase):** Similar to camelCase but with the first letter of each word capitalized, often used for class names, e.g., MyClass.
  + **Be Consistent:** Stick to a consistent naming style throughout your codebase. Consistency enhances readability and makes it easier for others to understand and maintain your code.
  + **Avoid Reserved Keywords:** Python has reserved keywords like if, True, class, etc., which have predefined meanings in the language. Avoid using these as variable names to prevent conflicts and confusion.
  + **Be Case-Sensitive:** Remember that Python is case-sensitive. Variables num and Num refer to different entities. Therefore, ensure consistency in letter casing when referring to variables within your code.

**Python Constants**

Constants, on the other hand, remain steadfast, holding values that never waver. Let's consider the mathematical constants pi (π) and gravity (g), assigning them immutable values:

```python

PI = 3.14159

GRAVITY = 9.8

print(PI) # Output: 3.14159

print(GRAVITY) # Output: 9.8

```

Although Python doesn't enforce immutability, naming constants in all capital letters signifies their unchanging nature—a convention widely followed in the Python community.

```python

# Numeric Literals

quantity = 10

price = 29.99

complex\_number = 5 + 2j

print(quantity) # Output: 10

print(price) # Output: 29.99

print(complex\_number) # Output: (5+2j)

# Boolean Literals

is\_active = True

is\_valid = False

print(is\_active) # Output: True

print(is\_valid) # Output: False

# String Literals

name = 'Python'

message = "Hello, World!"

print(name) # Output: Python

print(message) # Output: Hello, World!

# Special Literal

null\_value = None

print(null\_value) # Output: None

```

Each literal type—numeric, boolean, string, and special—serves a unique purpose, facilitating data representation and manipulation in Python programs.

## **Python Data Types**

In programming, data types specify the variable type of data which can be stored in.

| Data Types | Classes | Description |
| --- | --- | --- |
| Numeric | int, float, complex | holds numeric values |
| String | str | holds a sequence of characters |
| Sequence | list, tuple, range | holds a collection of items |
| Mapping | dict | holds data in key-value pair form |
| Boolean | bool | holds either True or False |
| Set | set, frozenset | hold a collection of unique items |

**Numeric Data Types:**

Python offers several numeric data types to handle numerical values:

- **int:** Represents signed integers of unlimited length.

- float: Stores floating-point decimal numbers, accurate up to 15 decimal places.

- complex: Deals with complex numbers in the form of `a + bj`, where `a` is the real part and `b` is the imaginary part.

**Example**:

```python

num1 = 5

num2 = 2.0

num3 = 1 + 2j

```

**Sequence Data Types:**

Sequences are ordered collections of items, categorized into lists, tuples, and ranges:

- list: Mutable collection of items enclosed in square brackets.

- tuple: Immutable sequence of items enclosed in parentheses.

- range: Represents an immutable sequence of numbers.

**Example:**

```python

languages = ["Python", "Java", "C"]

product = ('Laptop', 999.99)

nums = range(1, 10)

```

**String Data Type:**

Strings are sequences of characters enclosed in single or double quotes:

**Example:**

```python

name = 'Python'

message = "Hello, World!"

```

**Set Data Type:**

Sets are unordered collections of unique items, represented by curly braces:

**Example:**

```python

unique\_numbers = {1, 2, 3, 4, 5}

```

**Dictionary Data Type**:

Dictionaries are key-value pairs, where each key is associated with a value:

**Example:**

```python

student\_info = {'name': 'Alice', 'age': 25, 'major': 'Computer Science'}

```

**Constants and Variables:**

In Python, variables are containers that hold data, while constants are unchangeable values. When naming variables and constants, adhere to these guidelines:

- Use alphanumeric characters and underscores.

- Choose names that make sense and follow a consistent naming convention (e.g., snake\_case).

- Avoid using reserved keywords as variable names.

**Example:**

```python

num\_students = 50

MAX\_VALUE = 100

snake\_case\_variable = "example"

```

In Python, variables serve as containers for storing data, while data types specify the nature of this data, including integers, floats, strings, and more. By adhering to naming conventions and choosing appropriate data types, developers can write clear and efficient code. Constants ensure that specific values remain unchanged throughout the program, enhancing code readability and maintainability. Understanding variables, data types, and constants is crucial for building robust and effective Python applications.