3. Variables, Data Types & Keywords

Made By:

Yadneyesh (Curious Coder)
CodWithCurious.com

3.1 Variables in Python

- In Python, variables are used to store values that can be used later in a program. You can think of variables as containers that hold data. What makes Python unique is that you don't need to explicitly declare the type of a variable. You simply assign a value to a variable using the "=" operator.

- For example, you can create a variable called "name" and assign it a value like this:

```
name = "Yadnyesh"
```

Here, "name" is the variable name, and "Yadnyesh" is the value assigned to it. Python will automatically determine the type of the variable based on the value assigned to it. In this case, the type of the variable "name" is a string.

Variables in Python can hold different types of data, such as numbers, strings, lists, or even more complex objects. You can change the value of a variable at any time by assigning a new value to it. For instance:

```
age = 25
age = 26  # Updating the value of 'age' variable
```

Python also allows you to perform operations on variables. For example, you can add, subtract, multiply, or divide variables containing numbers. You can even combine variables of different types using operators. For instance:

```
x = 5
y = 3
z = x + y  # The value of 'z' will be 8

greeting = "Hello"
name = "John"
message = greeting + " " + name  # The value of 'message' will be "Hello John"
```

Variables provide a way to store and manipulate data in Python, making it easier to work with information throughout your program. By giving meaningful names to variables, you can make your code more readable and understandable.

3.1.1 Identifier in Python

In Python, an identifier is a name used to identify a variable, function, class, module, or any other user-defined object. An identifier can be made up of letters (both uppercase and lowercase), digits, and underscores (_). However, it must start with a letter or an underscore.

Here are some important rules to keep in mind when working with identifiers in Python:

- **Valid Characters:** An identifier can contain letters (a-z, A-Z), digits (0-9), and underscores (_). It cannot contain spaces or special characters like @, #, or $.

- **Case Sensitivity:** Python is case-sensitive, meaning uppercase and lowercase letters are considered different. So, "myVar" and "myvar" are treated as two different identifiers.

- **Reserved Words:** Python has reserved words, also known as keywords, that have predefined meanings in the language. These words cannot be used as identifiers. Examples of reserved words include "if," "while," and "def."

- Length: Identifiers can be of any length. However, it is recommended to use meaningful and descriptive names that are not excessively long.

- Readability: It is good practice to choose descriptive names for identifiers that convey their purpose or meaning. This helps make the code more readable and understandable.

Here are some examples of valid identifiers in Python:

- my_variable
- count
- total_sum
- PI
- MyClass

And here are some examples of invalid identifiers:

- 123abc (starts with a digit)
- my-variable (contains a hyphen)
- if (a reserved word)
- my var (contains a space)

3.2 Data Types in Python

Data types in Python refer to the different kinds of values that can be assigned to variables. They determine the nature of the data and the operations that can be performed on them. Python provides several built-in data types, including:

1. Numeric:

Python supports different numerical data types, including integers (whole numbers), floating-point numbers (decimal numbers), and complex numbers (numbers with real and imaginary parts).

a. **Integers (int):** Integers represent whole numbers without any fractional part. For example, age = 25.
b. **Floating-Point Numbers (float):** Floating-point numbers represent numbers with decimal points or fractions. For example, pi = 3.14.
c. **Complex Numbers (complex):** Complex numbers have a real and imaginary part. They are denoted by a combination of a real and imaginary number, suffixed with j or J. For example, z = 2 + 3j.

2. Dictionary:

Dictionaries are key-value pairs enclosed in curly braces. Each value is associated with a unique key, allowing for efficient lookup and retrieval. For example, person = {'name': 'John', 'age': 25, 'city': 'New York'}.

3. Boolean:

- Boolean (bool): Booleans represent truth values, either True or False. They are used for logical operations and conditions. For example, is_valid = True.

4. Set:

Sets (set): Sets are unordered collections of unique elements enclosed in curly braces. They are useful for mathematical operations such as union, intersection, and difference. For example, fruits = {'apple', 'banana', 'orange'}.

5. Sequence Type:

- Sequences represent a collection of elements and include data types like strings, lists, and tuples. Strings are used to store textual data, while lists and tuples are used to store ordered collections of items.
  - **Strings (str):** Strings represent sequences of characters enclosed within single or double quotes. For example, name = "John".
  - **Lists (list):** Lists are ordered sequences of elements enclosed in square brackets. Each element can be of any data type. For example, numbers = [1, 2, 3, 4].
  - **Tuples (tuple):** Tuples are similar to lists but are immutable, meaning their elements cannot be changed once defined. They are enclosed in parentheses. For example, coordinates = (3, 4).

3.3 Keywords in Python

- Keywords in Python are special words that have specific meanings and purposes within the Python language. They are reserved and cannot be used as variable names or identifiers.

- Keywords play a crucial role in defining the structure and behavior of Python programs.

- Keywords are like building blocks that allow us to create conditional statements, loops, functions, classes, handle errors, and perform other important operations.

- They help in controlling the flow of the program and specify how different parts of the code should behave.

- For example, the if keyword is used to check conditions and perform specific actions based on those conditions. The for and while keywords are used to create loops that repeat a block of code multiple times.

- The def keyword is used to define functions, which are reusable blocks of code that perform specific tasks.

List of all the keywords in Python:

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```