**Python Syntax: Variables**

Variables are a temporary storage space in a computer’s memory. When a variable’s value changes the program’s current state also changes. A variable acts as a container to hold a different number of data items or values. All programming languages use variables, as they are among the most important elements in programming, and that is why a good understanding of variables will only make your job easier when writing programs. Variables are also used to move data between functions; this will be discussed later.

Every variable is created with an initial value. A variable can be in three states:

* Variable creation (Declaration)
* Variable assignment (Initialization)
* Variable changed (Execution)

Once the code which created the variable has finished executing, the variable is destroyed.

In Python, variables are defined in a standard way, by using the **assignment** character (=). This changes the value of the variable. Naming conventions specify the way in which variables should be named. This standard is used to make code more readable, and thus easier to understand.

The rules include the start and continuation characters. Variable names may contain any upper or lower case letter (A–Z, a–z), a number, or the underscore character. They may not begin with a number or contain spaces. Continuation characters are any characters except whitespace characters like tab and space.

Here are a few examples of **valid** variable names:

* c
* ref\_number
* admin
* aVeryLongName

Here are a few examples of **invalid** variable names:

* True
* $name
* 12Graph

In Python identifiers are case sensitive, so for example, firstName, FirstName, FIRSTNAME, and firstname are four different identifiers. A second rule is that variables cannot have the same name as Python’s keywords. We can find out what keywords are in Python, by using the function called dir(). If this function is called with the \_\_builtins\_\_ attribute, it returns a list of Python’s built-in attributes.

The \_\_builtins\_\_ module contains all Python’s built-in attributes, which can be used with the dir()function. The ones that are returned are identified with the following characteristics:

* Python’s built-in exceptions start with a capital letter.
* The rest are either functions or data type names.
* Identifiers that start and end with one or two underscores are special methods.

All variables have to be assigned to a data type like a string (a series of characters) or an integer (positive and negative whole numbers). There are others, some of which will be discussed at a later stage.

Python has a powerful feature regarding the assignment. A variable is assigned automatically to an appropriate data type. For example, Python automatically assigns a variable to a string data-type, if an input or value is given that contains letters or words. Values of the same type can be manipulated together. Sometimes Python finds a way to manipulate values into a common type by casting the values automatically. There are cases where values need to be cast explicitly.

Casting can be done in two ways:

* Implicitly**:** The compiler automatically casts a value from one data type to another when assured that there will be no data loss.
  + - * Forexample**.** casting from an integer variable to a floating-point variable or casting from an integer variable to another integer variable
* Explicitly**:** A value cannot be automatically cast from one data type to another if it will result in data loss. Extra code has to be written to ensure that the value stays the same and only the data type changes.
  + - * Forexample**,** casting from a floating-point value to an integer value

In Python, variables are used to store and manage data. They act as symbolic names for values or objects. Here are some key points about Python variables:

1. Variable Naming Rules:

- Variable names can contain letters, numbers, and underscores.

- Variable names must start with a letter or an underscore, followed by letters, numbers, or underscores.

- Variable names are case-sensitive, so `myVar` and `myvar` are treated as different variables.

2. Assignment:

- You assign a value to a variable using the `=` operator.

- For example: `my\_variable = 42` assigns the value `42` to the variable `my\_variable`.

3. Data Types:

- Python is dynamically typed, meaning you don't need to declare the data type of a variable explicitly.

- The data type of a variable is determined automatically based on the value assigned to it.

- Common data types include integers, floats, strings, lists, dictionaries, and more.

4. Variable Reassignment:

- You can reassign a variable to a different value of a different data type at any time.

- For example: `my\_variable = "Hello, World"` is a valid reassignment.

5. Variable Scope:

- Variables have different scopes, which define where in your code the variable is accessible.

- The main scopes are local, enclosing (non-local), and global. You can use the `global` keyword to declare a global variable.

6. Variable Names and Conventions:

- It's a good practice to use descriptive variable names to make your code more readable.

- By convention, variable names are usually in lowercase with words separated by underscores (snake\_case).

- For constants, they are typically in uppercase (e.g., `PI = 3.14159`).

Here are some examples of working with variables in Python:

python

# Variable assignment

my\_variable = 42

text = "Hello, World"

pi = 3.14159

# Variable reassignment

my\_variable = "Python is fun"

# Data type is determined automatically

type(my\_variable) str (string)

# Variable naming conventions

user\_name = "John Doe"

num\_of\_items = 10

MAX\_VALUE = 100

Variables are fundamental in Python and are used to store and manipulate data, making Python a versatile and dynamic language for various programming tasks.