**Introduction to Variables**

Variables in programming are temporary storage spaces in a computer's memory that hold different data values.

They are essential in changing a program's state as their values change and are used to transfer data between functions.

Variables go through three states: creation (declaration), assignment (initialization), and change (execution).

In Python, variables are defined using the assignment character (=) and follow naming conventions for readability.

Variable names can include letters (upper and lower case), numbers, or underscores but should not start with a number or contain spaces.

Python is case-sensitive, and variable names cannot match Python keywords.

You can check Python's built-in attributes using the dir() function with the \_\_builtins\_\_ attribute.

Special methods, exceptions, and functions found in dir(\_\_builtins\_\_), dir(\_\_doc\_\_), dir(\_\_name\_\_), and dir(\_\_package\_\_) cannot be used as variable

Using variables

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).

A variable is assigned automatically to an appropriate data type. For example, Python automatically assigns a variable to a string datatype, if an input or value is given that contains letters or words.

Values of the same type can be manipulated together. Python finds a way to manipulate values into a common type by casting the values automatically.

example:"

Example 1: Automatic assignment of variables:

num1 = 15 #num1 is automatically assigned as an integer

#value

num2 = 30 #num2 is also assigned as an integer value

print (num1 + num2) #answer is printed

**Casting**

Implicit Casting: In implicit casting, the compiler automatically converts a value from one data type to another when it's confident there will be no data loss. For example, casting from an integer to a floating-point variable or from one integer to another.

Explicit Casting: Explicit casting is used when there might be data loss during the conversion, and extra code needs to be written to ensure that the value remains the same. An example is casting from a floating-point value to an integer value.

Consolidating Learnings

String (str):

A string is a data type used to represent a sequence of characters, typically used for text in programming. It can contain letters, numbers, symbols, spaces, and other characters.

In Python, strings are enclosed in either single (' ') or double (" ") quotation marks.

Example: "Hello, World!" is a string.

Integer (int):

An integer is a data type used to represent whole numbers (positive or negative) without any fractional or decimal part.

Integers are used for counting, indexing, and performing mathematical operations that don't involve fractions.

Examples: -5, 0, 42.

**Casting from Integer to String:**

To convert an integer to a string, you can use the str() function. This function changes the integer into a string, allowing it to be used as text.

Example:

integer\_num = 42 # An integer

string\_num = str(integer\_num) # Casting the integer to a string

print(string\_num) # Output will be a string: "42"

**Python's Naming Conventions:**

Python follows specific naming conventions to make code more readable and maintainable:

Variable Names: Use lowercase letters with underscores (snake\_case). For example, my\_variable, user\_count.

Function Names: Also use lowercase letters with underscores. For example, calculate\_average, print\_result.

Constant Names: Use all uppercase letters with underscores to represent constants. For example, PI, MAX\_LENGTH.

Class Names: Use CamelCase, where each word starts with a capital letter and there are no spaces or underscores. For example, MyClass, CarModel.

Module Names: Use lowercase letters with underscores to name modules (files containing code). For example, my\_module.py.

Package Names: Package names (folders containing modules) should also use lowercase letters. Avoid using underscores in package names. For example, mypackage.