Variables and Data Types

What are Variables?

A variable in Python represents a named location that refers to a value and whose values can be used and processed during the program run. In other words, variables are labels/names to which we can assign value and use them as a reference to that value throughout the code.

Variables are fundamental to programming for two reasons:

- **Variables keep values accessible:** For example, The result of a time-consuming operation can be assigned to a variable, so that the operation need not be performed each time we need the result.

- **Variables give values context:** For example, The number 56 could mean many different things, such as the number of students in a class, or the average weight of all students. Assigning the number 56 to a variable with a name like **num_students** would make more sense, to distinguish it from another variable **average_weight**, which would refer to the average weight of the students. This way we can have different variables pointing to different values.

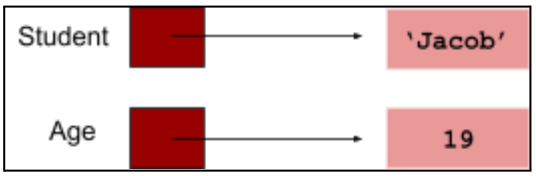
How are Values Assigned to A Variable?

Values are assigned to a variable using a special symbol "=", called the **assignment operator**. An operator is a symbol, like = or +, that performs some operation on one or more values. For example, the + operator takes two numbers, one to the left of the operator and one to the right, and adds them together. Likewise, the "=" operator takes a value to the right of the operator and assigns it to the name/label/variable on the left of the operator.

For Example: Now let us create a variable namely Student to hold a student's name and a variable Age to hold a student's age.

```
>>> Student = "Jacob"
>>> Age = 19
```

Python will internally create labels referring to these values as shown below:



Now, let us modify the first program we wrote.

```
greeting = "Hello, World!"
print(greeting)
```

Here, the Python program assigned the value of the string to a variable greeting, and then when we call print(greeting), it prints the value that the variable, greeting, points to i.e. **"Hello, World!"**
We get the output as:-

```
Hello, World!
```

Naming a Variable

You must keep the following points in your mind while naming a variable:-

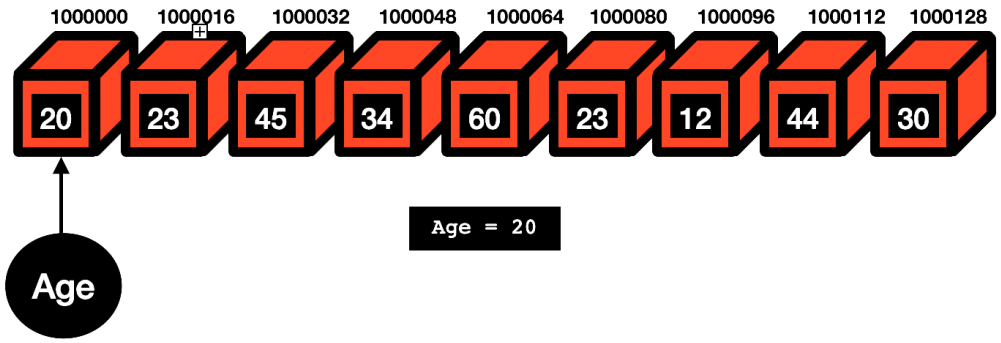
- Variable names can contain letters, numbers, and underscores.
- They cannot contain spaces.
- Variable names cannot start with a number.
- Variable names are case-sensitive. For example:- The variable names Temp and temp are different.
- While writing a program, creating self-explanatory variable names helps a lot in increasing the readability of the code. However, too long names can clutter up the program and make it difficult to read.

Python Variables in Memory

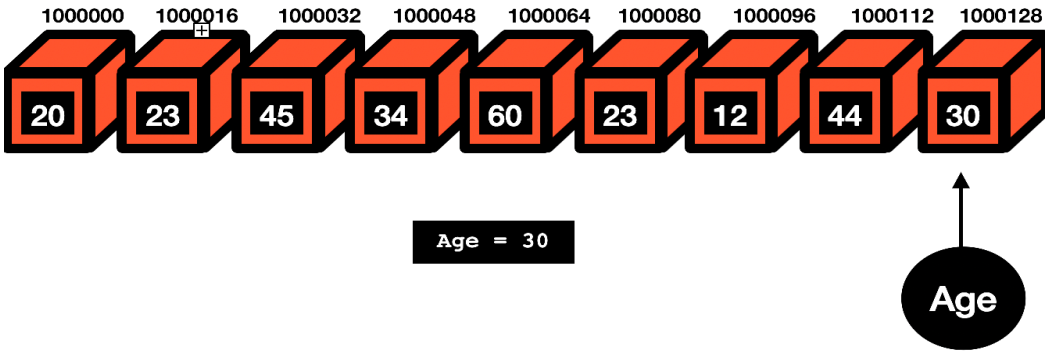
Python variables are not created in the form most other programming languages do. These variables do not have fixed locations, unlike other languages. The locations they refer/point to change every time their value changes.

Python preloads some commonly used values in an area of memory. This memory space has values/literals at defined memory locations and all these locations have different addresses.

When we give the command, **Age = 20,** the variable Age is created as a label pointing to a memory location where 20 is already stored. If 20 is not present in any of the memory locations, then 20 is stored in an empty memory location with a unique address and then the Age is made to point to that memory location.



Now, when we give the second command, **Age = 30**, the label Age will not have the same location as earlier. Now it will point to a memory location where 30 is stored. So this time the memory location for the label Age is changed.



Data Types

Data types are the classification or categorization of data items. Data types represent a kind of value that determines what operations can be performed on that data. Numeric, non-numeric, and Boolean (true/false) data are the most used data types. However, each programming language has its classification largely reflecting its programming philosophy. Python offers the following built-in data types:

- Numbers
  - Integers
  - Floating Point Numbers
  - Complex Numbers
- Strings
- Boolean Values
- List, Tuple, and Dictionary (To be covered later in the course)

Type Code	Description	Default Size (In Bytes)
`int`	Integers	4
`float`	Floating Point Numbers	4
`bool`	Boolean Values	1

Note:- If a variable has been assigned a value of some data type. It can be reassigned as a value belonging to some other Data Type in the future.

```python
a= "Raw" # String Data Type
a= 10 # Integer Data Type
a= 5.6 # Floating Point Number Data Type
a= 1+8j # Complex Number
a= True # Boolean Value
```

What is Literals

In Python, literals are notations used to represent fixed values in code. They are the actual data themselves, not variables or expressions. Essentially, literals are the raw data that your program works with.

Here are some common types of literals in Python:

1. **Numeric literals:** These represent numbers. For example:
   - Integer literals: `42`, `-10`, `0`
   - Floating-point literals: `3.14`, `-0.001`, `2.0`

2. **String literals:** These represent text. They are enclosed in either single quotes (`'`) or double quotes (`"`). For example:
   - Single-quoted string: `'hello'`, `'Python'`
   - Double-quoted string: `"world"`, `"is fun"`

3. **Boolean literals:** These represent the two truth values, `True` and `False`.

Literals provide a straightforward and concise way to represent data within your Python code.

Taking User Input

Developers often need to interact with users, either to get data or to provide some sort of result.

How to take User Input?

To get the input from the user interactively, we can use the built-in function, input(). This function is used in the following manner:

```
variable_to_hold_the_input_value = input(<Prompt to be displayed>)
```

```
In[] : age = input("What is your age?")
```

The above statement will display the prompt as:-

```
What is your age?_________ ←{User input here}
```

We will get the following interactive output:

```
In[] : name = input("Enter your name: ")
Enter your name: Rishabh #User Input
In[] : age = input("Enter your age: ")
Enter your age: 20 #User Input
In[] : name
Out[] : 'Rishabh'
In[] : age
Out[] : '19'
```

Note:- input() function always returns a value of the String type. Notice that in the above script the output for both name and age, Python has enclosed the output in quotes, like 'Rishabh' and '19', which implies that it is of String type. This is just because, whatever the user inputs in the input() function, is treated as a String. This would mean that even if we input an integer value like 20, it will be treated like a string '19' and not an integer. Now, we will see how to read Numbers in the next section.

Reading Numbers

Python offers two functions **int() and float()** to be used with the input() function to convert the values received through input() into the respective numeric types integer and floating-point numbers.

The steps will be:-

1. Use the input() function to read the user input.
2. Use the int() and float() functions to convert the value read into integers and floating-point numbers, respectively. This process is called **Type Casting**

Let us take an example:-

```
In[] : age= int(input("Enter Your Age: "))
Enter Your Age: 19
In[] : age
Out[] : 19
```

Here, the output will be 19 and not '19', i.e. the output is an Integer and not a String.

Arithmetic Operators in Python

The Arithmetic Operators are used in Python in the same way as they are used in Mathematics.

OPERATOR	DESCRIPTION
+	Add two operands
-	Subtracts second operand from the first

*	Multiplies two operands
/	Divides numerator by denominator (Floating Point Division)
//	Divides numerator by denominator (Floor Division) - Acts as a floor function
**	**Exponent Operator**- The first operand raised to the power of the second operand
%	**Modulo Operator**- Calculates remainder left after dividing first by second

Let us see how these operators work:-

```
In[] : print(5 + 2) # Addition
Out[] : 7
In[] : print(5 - 2) # Subtraction
Out[] : 3
In[] : print(5 * 2) # Multiplication
Out[] : 10
In[] : print(5 / 2) # Floating Point Division
Out[] : 2.5
In[] : print(5 // 2) # Floor Division
Out[] : 2 # 5 divided by 2 gives 2.5 and value of floor(2.5) is 2
In[] : print(5 ** 2) # Calculate Exponent
Out[] : 25 # 5 raised to the power of 2 is 25
In[] : print(5 % 2) # Modulus
Out[] : 1 # Remainder 1 is left after dividing 5 by 2
```