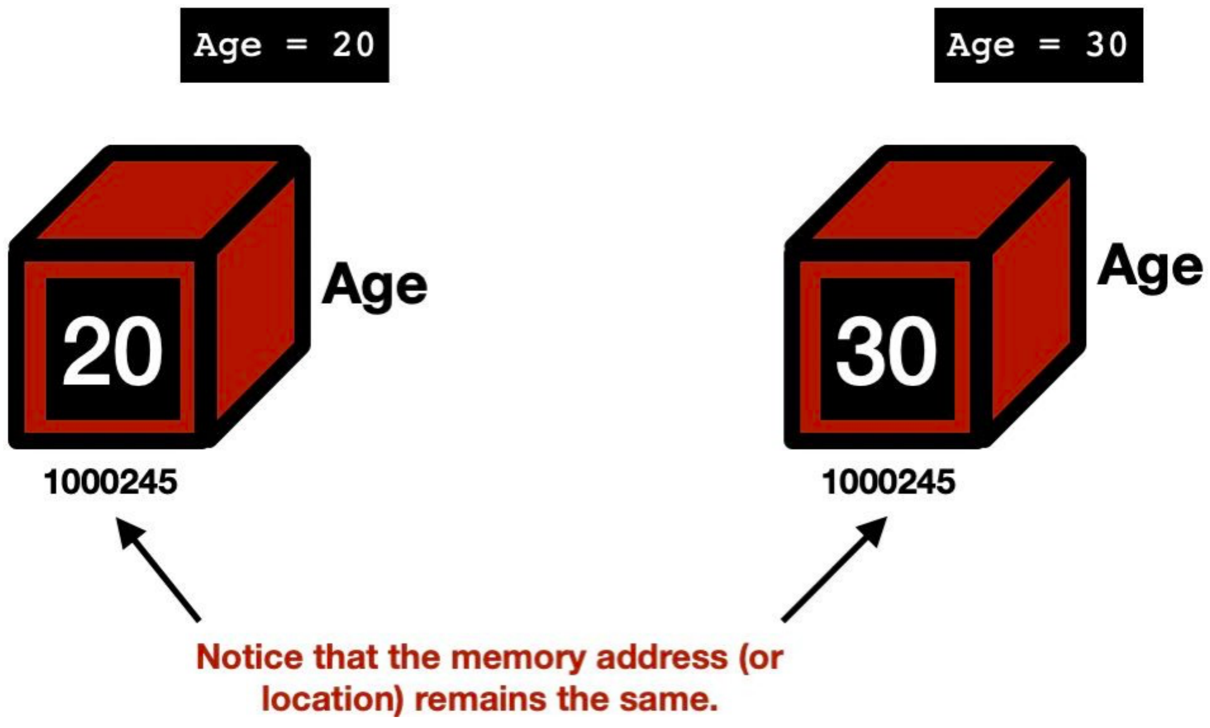
Traditional Programming Languages’ Variables in Memory

Let us study how variables and the values they are assigned, are represented in memory, in traditional programming languages like C, C++, Java, etc. In these languages, variables are like storage containers. They are like named storage locations that store some value. In such cases, whenever we declare a new variable, a new storage location is given to that name/label and the value is stored at that named location. Now, whenever a new value is reassigned to that variable, the storage location remains the same. However, the value stored in the storage location is updated. This can be shown from the following illustration.

Consider the following script:

```
Age = 20

Age = 30 # Re-assigning a different value to the same variable
```


Age = 20
Age = 30
20
Age
30
Age
1000245
1000245
Notice that the memory address (or location) remains the same.

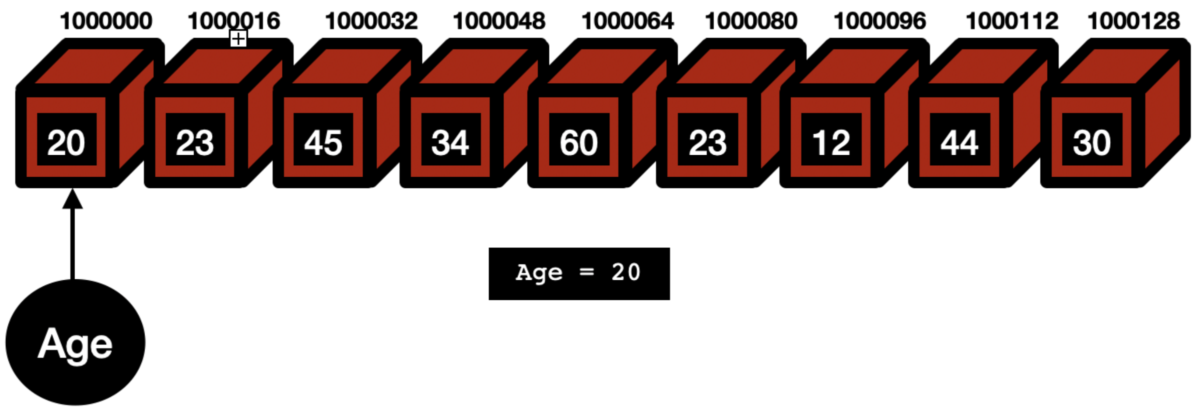
In the above script, when we declare a new variable Age, a container box/ Memory Location is named Age and the value 20 is stored in the memory address 1000245 with the name/label, Age. Now, on reassigning the value 30 to Age, the value 30 is stored in the same memory location. This is how the variables behave in Traditional programming languages.

Python Variables in Memory

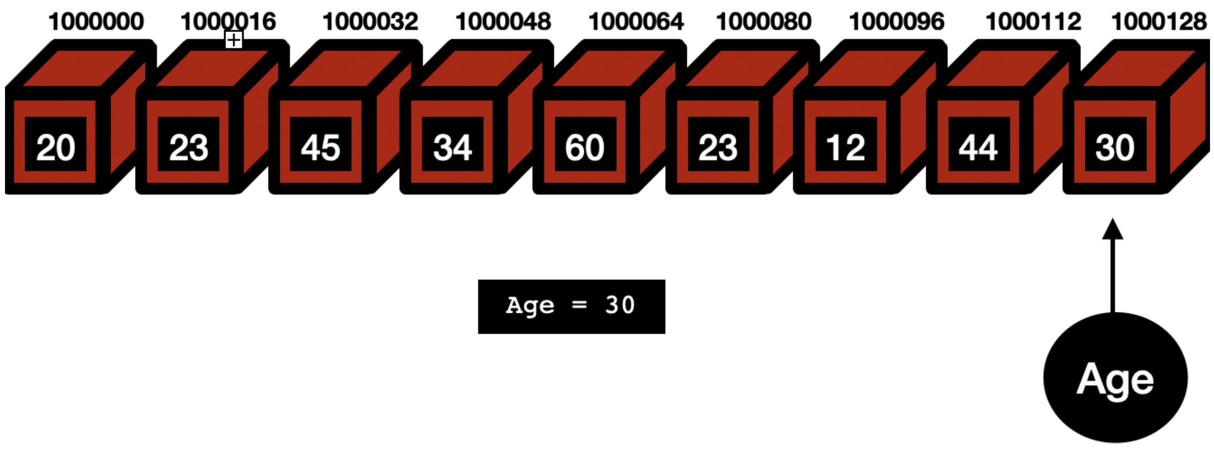
Python variables are not created in the form most other programming languages do. These variables do not have fixed locations, unlike other languages. The locations they refer/point to changes every time their value changes.

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.



One interesting thing to note while working with Python variables is that when we create a variable, we actually create an object somewhere in the memory with a unique mapping or ID/Address. We can see this unique ID generated against each object using ***id().***

Example:

```
a = 5          #An object, "a" is created that has a unique ID/memory block which stores 5 as a variable
b = 10         #An object, "b" is created that has a unique ID/memory block which stores 10 as a variable
print(id(a))   #Printing the unique ID for "a"
print(id(b))   #Printing the unique ID for "b"
```

output:

```
4425656800
4425656960
```