ch3

April 2, 2021

Python Variables

Python variables are references to python objects

In languages like ‘C’ when a variable is created, a portion in memory is allocated, and the value is stored at that location. when the variable is assigned a new value the memory is overwritten with the new value

int a=1

Subsequently a=2 will overwrite the same location

A statement like shown below will another memory location and copy the value of variable a to the newly created variable

int b=a

In python, a statement given below, in turn creates an Integer Object with value 1 and the reference of the object is assigned to the varible a. Instance is given a tag named ‘a’

```
<p>a=1</p>
<p><img src="variable4.jpg" width="50" height="20"></p>
<p>and a statment given below, in turn creates another integer object with value 2 and the<br>
ref reference of the object is assigned to the varible a. The previous integer object with<br>
value 1 is still there in the memory, may or may not be deleted immediately</p>
<p>a=2</p>
<p><img src="variable5.jpg" width="50" height="20"></p>
```

A statement given below, in turn assigns the reference of integer object with value 2 to both the variables ‘a’ and ‘b’

```
<p>b=a</p>
<p>this assignment will not create another, as how it happens in other languages like C and c+
```

Essentially, variables are only references to python objects

[2]:
```python
"""Let us understand the above concepts. Python has a built-in function called␣
↪'id' which
returns the reference of an object in decimal form"""
a=1
print("the decimal form of reference of a with value 1:",id(a))
a=2
print("the decimal form of reference of a with value 2:",id(a))
b=a
print("the decimal form of reference of b:",id(b)) # references of id of a and␣
↪b are same. which means they point to the same object
c=1
print("the decimal form of reference of c:",id(c)) # this show the integer␣
↪object with value '1' is still there in the memory
```

```
the decimal form of reference of a with value 1: 94500549886400
the decimal form of reference of a with value 2: 94500549886432
the decimal form of reference of b: 94500549886432
```

Types of python objects

Immutable objects

Mutable objects

Immutable Objects

Python objects, for which the reference changes on modification

Integer,Float,String, tuple fall under immutable objects

Mutable Objects

List, Dictionary etc., fall under mutable objects

Python objects, for which the reference does not change on modification

[ ]:
```
"""Examples to explain the above concepts"""
a = 5

print("value of a:",a)
print("reference of a before modification:",id(a))

a += 1

print("value of a:",a)
print("reference of a after modification:",id(a)) # a gets a new reference.␣
 ↪changing the value of a create a new object

"""observe the changes made to list object"""
list1 = ['a','b','c','d']

print("value of list:",list1)
print("reference of list1 before modification:",id(list1))

list1[2]='g'

print("value of list:",list1)
print("reference of list1 after modification:",id(list1)) # list reference␣
 ↪remains same, even after modification of list elements
```