IMP POINTS

* Strings are immutable. Once they are created, they cannot be changed.
* If you want to assign the same value to more than one variables then you can use the chained assignment:

myFirstVariable = mySecondVariable = 1

* myFirstVariable = 1  
  mySecondVariable = 2  
  myFirstVariable = "Hello You"

Assigning a value is known as **binding**in Python. In the example above, we have assigned the value of 1 to myFirstVariable.

Note how I assigned an integer value of 1 and then a string value of “Hello You” to the same myFirstVariable variable. **This is possible due to the fact that the data types are dynamically typed in python.**

**Local Scope**

* Variables declared within a function, as an example, can only exist within the block.
* Once the block exists, the variables also become inaccessible.

def some\_funcion():  
 TestMode = Falseprint(TestMode) <- Breaks as the variable doesn't exist outside

In Python, if-else and for/while loop block doesn’t create any local scope.

for i in range(1, 11):  
 test\_scope = "variable inside for loop"print(test\_scope)

Output:

variable inside for loop

With if-else block

is\_python\_awesome = Trueif is\_python\_awesome:  
 test\_scope = "Python is awesome"  
print(test\_scope)

Output:

Python is awesome

**GLOBAL SCOPE**

* You can declare a global variable outside of functions. It’s important to note that to assign a global variable a new value, you will have to use the “global” keyword:

TestMode = True  
def some\_function():  
 global TestMode  
 TestMode = Falsesome\_function()  
print(TestMode) <--Returns False

Removing the line “global TestMode” will only set the variable to False within the some\_function() function

## Set Operations

* A set is an unordered data collection without any duplicates. We can define a set variable as:

set = {9,1,-1,5,2,8,3, 8}  
print(set)

This will print: {1, 2, 3, 5, 8, 9, -1}

Note duplicates are removed.

Lambda can be used as anonymous function.

my\_lambda = lambda x,y,z : x - 100 + y - zmy\_lambda(100, 100, 100) # returns 0

**Syntax**:

variable = lambda arguments: expression

**HOW THE ASSIGNMENT WORK?**

def update\_variable(variable\_to\_update):

print(id(variable\_to\_update))

variable6 = 'Tushar'

update\_variable(variable6)

print('Variable6: ', id(variable6))

Here they are also pointing to each other.

140689528378544

Variable6: 140689528378544

Here u can see that variable\_to\_update in the method is pointing the variable named as variable6

List is ordered and mutable.

[] defines the list collection framework.

Tuple is ordered and immutable.

() defines the list collection framework.

Set is unordered and immutable.