**NAMESPACES: GLOBAL vs LOCAL SCOPE**

enemies = 1

def increase\_enemies():

enemies = 2

print(f"enemies inside function: {enemies}")

increase\_enemies()

print(f"enemies outside function: {enemies}")

enemies inside function: 2

enemies outside function: 1

Local Scope:

Exists within functions.

Namespace: Anything(variables) that we give name.

**Does Python have Block Scope?**

There is no such thing as block scope in Python. This means that new variables declared inside an if else/while loop/for loop… anything that has indentation, is available outside of it too. E.g.:

game\_level = 3

def create\_enemy():

enemies = [“Skeleton”, “Zombie”, “Alien”]

if game\_level < 3:

new\_enemy = enemies[0]

print(new\_enemy) #This is possible

However, within a function, there is local scope.

**How to modify a global variable:**

The global variable has to be explicitly defined.

enemies = 1

def increase\_enemies():

global enemies

enemies += 1

print(f"enemies inside function: {enemies}")

It is highly NOT recommended to modify global scope.

Here, we could do the following instead of modifying the global scope:

def increase\_enemies():

return enemies+1

**Python Constants & Global Scope:**

Global scopes can be incredibly useful, specially while defining constants.

e.g. pi = 3.14159

Usually, in order to differentiate between constants and the variables, the constants are defined in upper case.

e.g.: PI = 3.14159,

URL = /….

TWITTER\_HANDLE = “…”

Etc.