DAY 12 SCOPE AND NUMBER GUESSING GAME

EXAMPLE:

enemies = 1 #GLOBAL  
  
def increase\_enemies():  
 enemies = 2 #LOCAL  
 print(f"enemies inside function: {enemies}")  
  
increase\_enemies()  
print(f"enemies outside function: {enemies}")

OUTPUT:

enemies inside function: 2

enemies outside function: 1

EXAMPLE: [LOCAL SCOPE] or [FUNCTION SCOPE]

def my\_function():

my\_local\_var = 2

# This will cause a NameErrorr

print(my\_local\_var)

EXAMPLE: [GLOAL SCOPE]

my\_global\_var = 3

def my\_function():

# This works no problems

print(my\_global\_var)

BLOCK SCOPE:

EXAMPLE:

# Accessible anywhere  
my\_global\_var = 1  
def my\_function():  
 # Only accessible within my\_function()  
 my\_local\_var = 2  
for \_ in range(10):  
 # Accessible anywhere  
 my\_block\_var = 3

GLOBAL :

enemies = 1  
def increase\_enemies():  
 global enemies  
 enemies += 1  
 print(f"enemies inside function: {enemies}")  
increase\_enemies()  
print(f"enemies outside function: {enemies}")

The reason you need to declare enemies as global inside the function is because, by default, Python treats variables inside a function as **local** to that function. This means that without the global keyword, any changes made to enemies inside the function would create a new, local variable instead of modifying the global one.

Here’s how it works:

* When you declare enemies at the top (outside the function), it is a **global variable**. It can be accessed from anywhere in your code.
* Inside the increase\_enemies() function, if you want to **modify** the global enemies variable, you need to explicitly tell Python that you want to refer to the global one, not create a new local variable.

If you didn’t use the global keyword, Python would assume you are trying to create a new local variable called enemies within the function, and changes would not affect the global enemies.