#23_Variables_and_Functions

03/06/2025

Example 1

```
1  my_var = 1
2
3  def my_func():
4      print(my_var) # Output: 1
5
6  my_func()
7
```

my_var on line 1 is the same as my_var on line 4

Example 2

```
1  my_var = 1
2
3  def my_func():
4      my_var = 2
5
6  my_func()
7  print(my_var) # Output: 1
```

Global variable

Local variable

Refers to the global variable

Assigning a variable inside a function, line 4, creates a new local variable with the function. Hence, my_var on line 4 differs from my_var on line 1 and 7.

09/06/2025

Example 3

```
my_var = [1]

def my_func():
    my_var[0] = 2

my_func()
print(my_var) # Output: [2]
```

Mutating my_var by Reassigning [0] with 2

Global variable

Reassigning an element in a collection mutates the collection.

Example 4

```
1 def my_func():
2     my_var = 1
3
4 my_func()
5 print(my_var) # NameError: name 'my_var' is not defined
6
```

As my_var is only visible inside the function in which it is defined, try to print my_var on line 5 results in an error

09/06/2025

Example 5

```
1  my_var = [1]
2
3  def my_func(my_var):
4      my_var.append(2)
5
6  my_func(my_var)
7  print(my_var) # Output: [1, 2]
```

my_var on lines 3 and 4 is a different variable to the variable on lines 1, 6 and 7. Although they do point to the global my_var = [1], therefore, append mutates the object assigned to my_var on line 1.

Example 6

```
1  my_var = [1]
2
3  def my_func(my_var):
4      my_var = [2]
5
6  my_func(my_var)
7  print(my_var) # Output: [1]
```

my_var on line 4 references a new list, [2]. While the global my_var on line 7 continues to reference the original list, [1]

09/06/2025

Example 7

```
1  my_var = "Hello"
2
3  def my_func():
4      print(my_var + " world") # Output: "Hello world"
5
6  my_func()
7
```

Note the space

my_var on line 4 is the same as line 1, so assignment to my_var on line 4, hence it accesses the global my_var on line 1.

Example 8

```
1  my_var = "Hello"
2
3  def my_func():
4      return my_var + " world"
5
6  my_func()
7  print(my_var) # Output: "Hello"
```

The calling code on line 6 does not capture the return of line 4. Therefore, my_var has its original value when printed on line 7.

Concatenation of "Hello" + " world"

09/06/2025

Example 9

```
1  my_var = "Hello"
2
3  def my_func():
4      my_var = my_var + " world"
5      # UnboundLocalError: local variable 'my_var' referenced
6        before assignment
7      return my_var
8
9  my_func()
10 print(my_var)
```

my_var creates a new local variable inside my_func. That variable is initially undefined, hence why an error occurs when the code attempts to concatenate my_var and " world".

Regardless, line 10 would only print "Hello" as line 9 does not capture the eventual return of my_var. Therefore, it refers to the global my_var on line 1.

Summary

· Different concepts covered in Example 1 to 9 resulting in different outputs; These include:

- variable scope
- mutability
- variables as references
- passing arguments to functions

} Covered earlier in course and important so will be covered in future assignments.