Python Functions: Returning Multiple Variables and Function Scopes: Takeaways

Syntax

- Return **multiple variables**:

```
def sum_and_difference(a, b):
    a_sum = a + b
    difference = a - b
    return a_sum, difference
sum_1, diff_1 = sum_and_difference(15, 10)
```

- Lists versus tuples:

```
a_list = [1, 'a', 10.5]
a_tuple = (1, 'a', 10.5)
```

- Use return statements in function bodies:

```
def price(item, cost):
    return "The " + item + " costs $" + str(cost) + "."

print(price("chair", 40.99))
```

- Use print statements in function bodies:

```
def price(item, cost):
    print("The " + item + " costs $" + str(cost) + ".")

price("chair", 40.99)
```

Concepts

- Parameters and return statements aren't mandatory when we create a function.

```
def print_constant():
    x = 3.14
    print(x)
```

- The code inside a function definition executes only when we call the function.
- When we call a function, the variables defined inside the function definition are saved into a temporary memory that is erased immediately after the function finishes running. The temporary memory associated with a function is isolated from the memory associated with the main program (the main program is the part of the program outside function definitions).
- We call the part of a program where we can access a variable the scope. The variables defined in the main program are in the global scope, while the variables defined inside a function are in the local scope.
- Python searches the global scope if a variable isn't available in the local scope, but the reverse doesn't apply. Python won't search the local scope if it doesn't find a variable in the global scope. Even if it searched the local scope, the memory associated with a function is temporary, so the search would be pointless.

Resources

- Python official documentation
- Style guide for Python code