Functions: Intermediate: Takeaways ⤴

Syntax

- Initiating parameters with **default arguments**:

```python
def add_value(x, constant=3.14):

    return x + constant
```

- Using **multiple return statements**:

```python
def sum_or_difference(a, b, do_sum):

    if do_sum:

        return a + b

    return a - b
```

- Returning **multiple variables**:

```python
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)
```

Concepts

- We need to avoid using the name of a built-in function to name a function or a variable because this overwrites the built-in function.
- Each built-in function is well documented in the official Python **documentation**.
- Parameters and return statements are not mandatory when we create a function.

```python
def print_constant():

    x = 3.14

    print(x)
```

- The code inside a function definition is executed only when the function is called.

- When a function is called, 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).

- The part of a program where a variable can be accessed is often called scope. The variables defined in the main program are said to be in the global scope, while the variables defined inside a function are in the local scope.

- Python searches the global scope if a variable is not 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](#)