13.147 - Leaflet - Transferring variable function parameters

July 23, 2018

1 Variable function parameters

1.1 Receive variable function parameters

Sometimes you want to allow a function to accept a variable number of parameters.

You can use the * notation for this; the parameters then end up in a tuple. This function then accepts a variable number of parameters:

```
In [1]: def calculate_max(*params):
            print(params)
            current_max = params[0]
            for item in params:
                if item > current_max:
                    current_max = item
            return current_max

        calculate_max(1, 2, 3)
```

```
(1, 2, 3)
```

```
Out[1]: 3
```

In addition, you have the option of receiving several named parameters using the ** notation. These parameters then end up in a dictionary and you can access them from the function:

```
In [2]: def f(**args):
            print(args)

        f(key="value", key2="Value 2")
```

```
{'key': 'value', 'key2': 'Value 2'}
```

Of course, both can also be combined. It is important that the parameter with an asterisk (here: `*params` must precede the parameter with two asterisks `**args`).

All normal parameters now end up in the `*params` tuple. All named parameters end up in the dictionary `**args`.

```
In [3]: def f(*params, **args):
            print(params)
            print(args)

        f("A Value", "Another Value", key="value", key2="value2")
```

```
('A Value', 'Another Value')
{'key': 'value', 'key2': 'value2'}
```

1.2 Call function with variables parameters

Similar to how a * has combined several parameters in the function definition, we can also "unpack" several parameters.

Here in this case we have a list `l` and we want the first list element to be passed as parameter `a` and the second as parameter `b`:

```
In [4]: def f(a, b):
            print(a)
            print(b)

        l = [1, 2]
        f(*l)
```

```
1
2
```

The same works for a dictionary, of course. Here we need ** to unpack the parameters:

```
In [5]: def f(a, b):
            print(a)
            print(b)

        l = {"a": 1, "b": 2}
        f(**l)
```

```
1
2
```

1.2.1 Why are we doing this?

Sometimes we just want to "loop through" parameters, i.e. we don't want to receive them in large quantities, but simply pass them on to another function.

In this example, parameters are packed into a dictionary (`**plot_params`, line 4). This allows variable parameters to be transferred.

These parameters are then unpacked from the dictionary into `plt.plot([1, 2, 3],[5, 6, 5], **plot_params)` and converted into normal function parameters.

For example, we can be involved in such a process.

Example:

```
In [6]: %matplotlib inline
        import matplotlib.pyplot as plt

        def create_plot(**plot_params):
            print(plot_params)

            plt.plot([1, 2, 3], [5, 6, 5], **plot_params)
            plt.show()

        create_plot(color="r", linewidth=10, linestyle="dashed")
```

```
{'color': 'r', 'linewidth': 10, 'linestyle': 'dashed'}
```