Merkblatt - Variable Funktionsparameter uebergeben

May 7, 2020

1 Variable Funktionsparameter

1.1 Variable Funktionsparameter entgegennehmen

Manchmal möchtest du einer Funktion erlauben, eine variable Anzahl an Parametern entgegenzunehmen.

Dafür kannst du die *-Schreibweise verwenden; die Parameter landen dann in einem Tupel. Dadurch akzeptiert diese Funktion dann eine variable Anzahl an Parametern:

[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)
```

[1]:
```
3
```

Zudem hast du die Möglichkeit, über die **-Schreibweise mehrere, benannte Parameter entgegenzunehmen. Diese Parameter landen dann in einem Dictionary, und du kannst aus der Funktion darauf zugreifen:

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

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

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

Das beides funktioniert natürlich auch kombiniert. Wichtig ist hierbei, der Parameter mit einem Sternchen (hier: `*params` muss vor dem Parameter mit zwei Sternchen stehen `**args`).

Alle normalen Parameter landen jetzt in dem Tupel `*params`. Alle benannten Parameter landen im Dictionary ‘`**args`.

[7]:
```
def f(*params, **args):
    print(params)
    print(args)

f("Ein weiterer Wert", "Noch ein Wert", key="value", key2="value2")
```

```
('Ein weiterer Wert', 'Noch ein Wert')
{'key': 'value', 'key2': 'value2'}
```

1.2 Funktion mit Variablen Funktionsparametern aufrufen

Ähnlich wie ein `*` in der Funktionsdefinition mehrere Parameter zusammengefasst hat, können wir auch mehrere Parameter quasi "entpacken".

Hier in dem Fall haben wir eine Liste `l` und wir möchten, dass das erste Listenelement als Parameter `a` übergeben wird und das zweite als Parameter `b`:

[8]:
```
def f(a, b):
    print(a)
    print(b)

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

```
1
2
```

Gleiches funktioniert natürlich auch für ein Dictionary. Hier brauchen wir `**`, um die Parameter zu entpacken:

[10]:
```
def f(a, b):
    print(a)
    print(b)

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

```
1
2
```

1.2.1 Warum machen wir das?

Manchmal möchten wir Parameter einfach nur "durchschleifen", also gar nicht groß entgegennehmen, sondern einfach an eine andere Funktion weiterleiten.

Hier im Beispiel werden also Parameter in ein Dictionary gepackt (`**plot_params`, Zeile 4). Dadurch können variable Parameter übergeben werden.

Diese Parameter werden dann in `plt.plot([1, 2, 3], [5, 6, 5], **plot_params)` wieder aus dem Dictionary entpackt und in normale Funktionsparameter umgewandelt.

So können wir uns z. B. in einen solchen Prozess einklinken.

Beispiel:

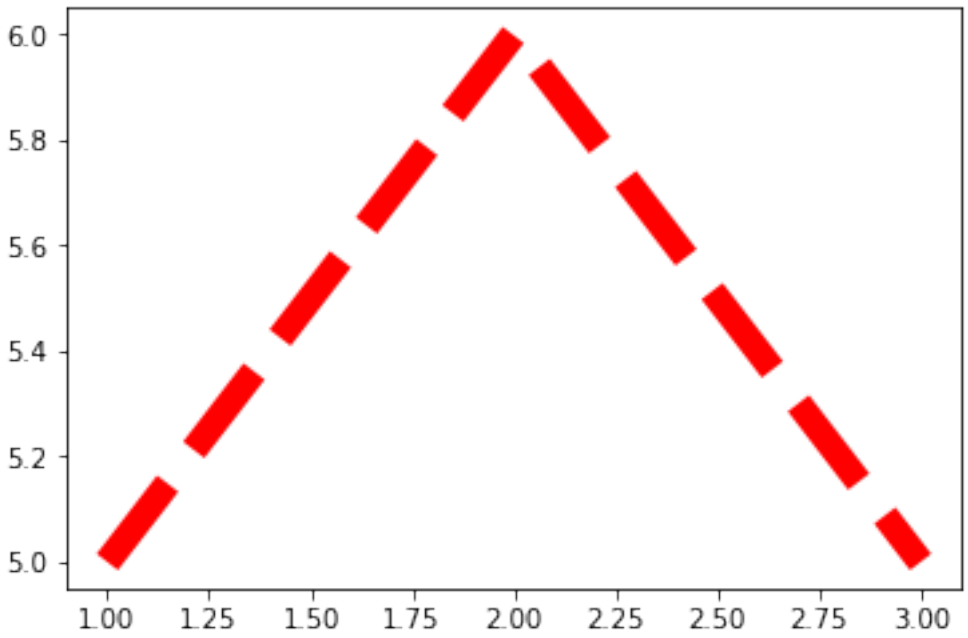
[11]:
```
%matplotlib inline
import matplotlib.pyplot as plt

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'}
```



[ ]: