# LU05a - Verwendung von *args und **kwargs in Python

In Python gibt es spezielle Syntaxelemente, um eine beliebige Anzahl von Positional-Argumenten (`*args`) oder Keyword-Argumenten (`**kwargs`) in einer Funktion zu akzeptieren. Diese werden oft in Funktionen und Decorators verwendet, um sie flexibler und wiederverwendbarer zu gestalten.

## Grundlagen

- `*args`: Erlaubt es, eine variable Anzahl von Positional-Argumenten als Tupel zu akzeptieren.
- `**kwargs`: Erlaubt es, eine variable Anzahl von Keyword-Argumenten als Dictionary zu akzeptieren.

Beide können in einer Funktion kombiniert werden, müssen jedoch in der richtigen Reihenfolge angegeben werden: erst normale Parameter, dann `*args` und zuletzt `**kwargs`.

## Beispiel für *args

Ein einfaches Beispiel für die Verwendung von `*args`:

```
def sum_all(*args):
    return sum(args)

print(sum_all(1, 2, 3, 4))  # Output: 10
```

In diesem Beispiel akzeptiert die Funktion `sum_all` eine beliebige Anzahl von Positional-Argumenten und summiert sie auf.

## Beispiel für **kwargs

Ein einfaches Beispiel für die Verwendung von `**kwargs`:

```
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="Alice", age=30)
# Output:
# name: Alice
# age: 30
```

In diesem Beispiel akzeptiert die Funktion `print_info` eine beliebige Anzahl von Keyword-Argumenten und druckt sie aus.

# Kombinierte Verwendung

Beide können in einer Funktion kombiniert werden:

```
def combined_example(arg1, arg2, *args, kwarg1=None, **kwargs):
    print(arg1, arg2)
    print(args)
    print(kwarg1)
    print(kwargs)

combined_example(1, 2, 3, 4, 5, kwarg1="six", kwarg2="seven")
# Output:
# 1 2
# (3, 4, 5)
# six
# {'kwarg2': 'seven'}
```

In diesem Beispiel nimmt die Funktion `combined_example` zwei normale Argumente, eine variable Anzahl von Positional-Argumenten (`*args`), ein Keyword-Argument (`kwarg1`) und eine variable Anzahl von weiteren Keyword-Argumenten (`**kwargs`).

Diese Flexibilität macht `*args` und `**kwargs` zu einem leistungsfähigen Werkzeug für die Erstellung vielseitiger und wiederverwendbarer Funktionen in Python.

Es ist wichtig zu beachten, dass die Namen `args` und `kwargs` nicht festgelegt sind. Die eigentliche Funktionalität wird durch die * und ** Operatoren gesteuert. Daher könnten die Variablen auch anders benannt werden, solange sie den Operatoren * und ** folgen.

```
def example_function(*vars, **keyvars):
    print("Positional arguments:", vars)
    print("Keyword arguments:", keyvars)

example_function(1, 2, x=3, y=4)
# Output:
# Positional arguments: (1, 2)
# Keyword arguments: {'x': 3, 'y': 4}
```

In diesem Beispiel wurde `*args` durch `*vars` und `**kwargs` durch `**keyvars` ersetzt, und die Funktion verhält sich immer noch wie erwartet.

---

M323-LU05