https://printpython.pl/poczatki/zadanie-z-gwiazdka/

Często spotykamy się z funkcjami, które wśród argumentów zawierają „magiczne” wyrażenia *args i **kwargs. Jednak początkujący programiści Pythona miewają problemy ze zrozumieniem działania tego mechanizmu.

Zacznijmy od odpowiedzenia sobie na pytanie czym są te „argsy i kwargsy” i czemu poprzedzają je gwiazdki. Wyrażenie „args” bierze się z od słowa „arguments” czyli argumenty i jest to zazwyczaj zmienna, która zawiera tuple argumentów pozycyjnych. Natomiast „kwargs” bierze się od „keyword arguments” czyli argumenty nazwane i zmienna taka zawiera pary nazwa-wartość argumentu. Jednak należy zaznaczyć, że nazwy te nie są wcale najważniejsze. Największa magia kryje się w gwiazdkach je poprzedzających, a „args” i „kwargs” możemy zastąpić dowolnymi innymi nazwami zmiennych.

Umieszczając w definicji funkcji wyrażenie z jedną lub dwoma gwiazdkami pozwalamy na przekazywanie do niej argumentów w dowolniej liczbie i nazwie. Bez konieczności konkretnego określenia tych parametrów. Jest to przydatne w momencie gdy chcemy, aby nasza funkcja była nieco bardziej uniwersalna.

Po pierwsze: *args

Wyrażenia z jedną gwiazdką (*) używamy gdy do funkcji chcemy przekazać dowolną liczbę argumentów pozycyjnych. Czyli takich dla których przy wywołaniu funkcji nie podajemy ich nazwy, a przypisanie wartości bazuje na kolejności argumentów. Z tego powodu parametr *args umieszczamy na końcu listy parametrów w definicji funkcji.

```
def parametr_args(argument, *args):
    print("zawartość args: {}".format(args))
    print("argument nazwany: {}".format(argument))
    for arg in args:
        print("argument z *args: {}".format(arg))

parametr_args('python', 'spam', 'eggs', 'test')
```

Możesz zauważyć, że wywołując powyższą funkcję przekazaliśmy w jej wywołaniu więcej argumentów niż zadeklarowaliśmy. Nadmiarowe argumenty zostały umieszczone w tupli po której z łatwością możemy iterować, aby dostać się do tych argumentów.

Po drugie: **kwargs

W przypadku gdy do naszej funkcji chcemy przekazywać argumenty, które wyróżniają się nazwą możemy użyć parametru z dwoma gwiazdkami (**). Przekazane w ten sposób argumenty są dostępne w funkcji w postaci słownika. Jego pary klucz-wartość stanowią nazwa i wartość przekazanego argumentu.

```
def parametr_kwargs(argument, **kwargs):
    print("argument: {}".format(argument))
    print("zawartość kwargs: {}".format(kwargs))

parametr_kwargs(dodatkowy=48, nastepny=111, argument=12)
```

```
# argument: 12
# zawartość kwargs: {'dodatkowy': 48, 'nastepny': 111}
```

Jak widzisz w powyższym przykładzie argumenty, które nie zostały zdefiniowane w funkcji trafiły do słownika o nazwie kwargs.

Jak używać gwiazdek w *args i **kwargs?

Możesz również używać obu poznanych przed chwilą parametrów. W tym przypadku pamiętaj jedynie o kolejności – wyrażenie z dwoma gwiazdkami musi być na końcu, z jedną gwiazdką wcześniej, a pozostałe zdefiniowane argumenty na początku.

Gwiazdek możesz również używać w wywołaniu funkcji. Jest to tak zwane rozpakowywanie list i słowników z argumentami. Jeżeli funkcja przyjmuje kilka argumentów i posiadasz listę, która zawiera argumenty, które chcesz przekazać do tej funkcji wystarczy poprzedzić nazwę listy gwiazdką zamiast podawać kolejne argumenty ręcznie.

```
lista_argumentow = [1,3,5]
def rozpakowywanie(pierwszy, drugi, trzeci):
    print(pierwszy)
    print(drugi)
    print(trzeci)

rozpakowywanie(*lista_argumentow)

# 1
# 3
# 5
```

Przyznasz, że ten sposób wygląda dużo lepiej niż:

```
rozpakowywanie(lista_argumentow[0], lista_argumentow[1], lista_argumentow[2])
```

Mam nadzieję, że ten artykuł rozjaśnił Ci działanie gwiazdek w Pythonie. Jeżeli masz jeszcze jakieś pytanie dotyczące działania tego mechanizmu zapraszam do zostawienia komentarza.

Jeżeli dopiero zaczynasz swoją przygodę z Pythonem lub jesteś bardziej zaawansowany sprawdź jak może pomóc Ci **Jupyter Notebook**.

A może potrzebujesz pomocy w nauce programowania i nie wiesz do kogo zwrócić się z pytaniami? Zajrzyj na stronę **O Blogu** znajdziesz tam więcej szczegółów na temat prowadzonego przeze mnie mentoringu.

Spodobał Ci się ten artykuł, sprawdź też pozostałe wpisy na moim blogu:

- Magiczne metody – czyli szczypta magii w Pythonie
  Moim zdaniem rzeczą, która należy wymienić na początku wśród tych, które sprawiają, że Python jest tak przyjazny poczatkującym użytkownikom jest jego wszechobecna spójność. Już… Czytaj więcej: Magiczne metody – czyli szczypta magii w Pythonie
- Pytest – ujarzmij fixtury i mocki
  Jedną z najważniejszych rzeczy w trakcie pracy nad aplikacją jest zapewnienie tego, by miała ona jak najmniej błędów oraz żeby kolejne zmiany w kodzie… Czytaj więcej: Pytest – ujarzmij fixtury i mocki

- Co nowego w Python 3.8
  Obecnie najnowszą, oficjalną wersją języka jest wersja Python 3.8, wersja ta posiada kilka nowości względem poprzednich edycji Pythona. W tym wpisie przyjrzymy się pokrótce…
  Czytaj więcej: Co nowego w Python 3.8
