Python les-materialen

# \*args en \*\*kwargs

Werk lang genoeg met Python en uiteindelijk zul je \*args en \*\*kwargs tegenkomen. Deze vreemde termen verschijnen als parameters in functiedefinities. Wat doen ze? Laten we een simpele functie bekijken:

def myfunc(a,b):  
 return sum((a,b))\*.05  
  
myfunc(40,60)  
  
# TODO delete this later

5.0

Deze functie retourneert 5% van de som van **a** en **b**. In dit voorbeeld zijn **a** en **b** *positionele* argumenten; dat wil zeggen, 40 wordt toegewezen aan **a** omdat dit het eerste argument is, en 60 aan **b**. Merk ook op dat om met meerdere positionele argumenten in de sum()-functie te werken, we ze als een tuple moesten doorgeven.

Wat als we met meer dan twee getallen willen werken? Een manier zou zijn om **te veel** parameters toe te wijzen en elke parameter een standaardwaarde te geven.

def myfunc(a=0,b=0,c=0,d=0,e=0):  
 return sum((a,b,c,d,e))\*.05  
  
myfunc(40,60,20)

6.0

Het is duidelijk dat dit geen erg efficiënte oplossing is, en dat is waar \*args om de hoek komt kijken.

## \*args

Wanneer een functieparameter begint met een asterisk, staat het een *arbitrary number (willekeurig aantal)* argumenten toe, en de functie neemt ze op als een tuple van waarden. Herschrijven van de bovenstaande functie:

def myfunc(\*args):  
 return sum(args)\*.05  
  
myfunc(40,60,20)

6.0

Merk op hoe het doorgeven van het trefwoord “args” aan de functie sum() hetzelfde deed als een reeks argumenten.

Het is vermeldenswaard dat het woord “args” zelf willekeurig is - elk woord is geschikt zolang het wordt voorafgegaan (preceded) door een asterisk. Om dit aan te tonen:

def myfunc(\*spam):  
 return sum(spam)\*.05  
  
myfunc(40,60,20)

6.0

## \*\*kwargs

Evenzo biedt Python een manier om willekeurige aantallen *keyworded*-argumenten af te handelen. In plaats van een tuple van waarden te creëren, bouwt \*\*kwargs een woordenboek van sleutel/waarde-paren. Bijvoorbeeld:

def myfunc(\*\*kwargs):  
 if 'fruit' in kwargs:  
 print(f"My favorite fruit is {kwargs['fruit']}") # review String Formatting and f-strings if this syntax is unfamiliar  
 else:  
 print("I don't like fruit")  
   
myfunc(fruit='pineapple')

My favorite fruit is pineapple

myfunc()

I don't like fruit

## \*args en \*\*kwargs gecombineerd

U kunt \*args en \*\*kwargs in dezelfde functie doorgeven, maar \*args moeten vóór \*\*kwargs verschijnen.

def myfunc(\*args, \*\*kwargs):  
 if 'fruit' and 'juice' in kwargs:  
 print(f"I like {' and '.join(args)} and my favorite fruit is {kwargs['fruit']}")  
 print(f"May I have some {kwargs['juice']} juice?")  
 else:  
 pass  
   
myfunc('eggs','spam',fruit='cherries',juice='orange')

I like eggs and spam and my favorite fruit is cherries  
May I have some orange juice?

Het plaatsen van keyworded-argumenten vóór positionele argumenten levert een uitzondering op:

myfunc(fruit='cherries',juice='orange','eggs','spam')

File "<ipython-input-8-fc6ff65addcc>", line 1  
 myfunc(fruit='cherries',juice='orange','eggs','spam')  
 ^  
SyntaxError: positional argument follows keyword argument

Net als bij “args”, kunt u elke gewenste naam gebruiken voor argumenten met trefwoorden - “kwargs” is slechts een populaire conventie.

Dat is het! Nu zou je moeten begrijpen hoe \*args en \*\*kwargs de flexibiliteit bieden om met een willekeurig aantal argumenten te werken!