PYTHON ACADEMY

ARGS E KWARGS NO PYTHON

Nesse ebook você vai aprender o que são os argumentos não-nomeados *args e argumentos nomeados **kwargs das funções em Python.

PYTHONACADEMY.COM.BR

Salve salve querido Pythonista!

Nesse post veremos dois conceitos muito importantes, o conceito de `*args` (ou argumentos não nomeados) e `**kwargs` (ou argumentos nomeados)!

Ambos são geralmente utilizados no contexto da criação de **Funções** em Python.

Se ainda não domina Funções, clique aqui e veja nosso post completo sobre Funções no Python

Agora vamos desvendar o que essas duas palavrinhas querem dizer em Python! 😉

`*args` e `**kwargs`

*Args e **kwargs permitem que você passe um número não especificado de argumentos para uma função.

Dessa forma, ao escrever uma função, você não precisa definir quantos argumentos serão passados para sua função.

Observação importante: Escrever `*args` e `**kwargs` é apenas uma convenção: `*args` vem do inglês *"arguments"* (argumntos) e `**kwargs` vem do inglês *"keyword arguments"* (argumentos nomeados). Podemos, por exemplo, utilizar `*banana` e `**bananas` se quisermos 😆

Argumentos não nomeados `*args`

Vamos dar uma olhada em `*args` primeiro!

O `*args` é usado para receber uma lista de argumentos de comprimento variável sem a palavra-chave (*keyword*) na função.

Aqui está um exemplo pra facilitar o entendimento:

```
def função_com_argumentos_variaveis(arg, *args):
    print("Primeiro argumento:", arg)

    for agumento in args:
        print("Argumento de *args", argumento)

função_com_argumentos_variaveis('primeiro_arg', 'arg2', 'arg3', 'arg3')
```

Veja a saída:

```
Primeiro argumento: primeiro_arg
Argumento de *args: arg2
Argumento de *args: arg3
Argumento de *args: arg3
```

Fique atento a ordem!

Argumentos não nomeados (`*args`) vem sempre primeiro que os argumentos nomeados (`**kwargs`).

Veja a seguinte chamada de função:

```
funcao_com_argumentos_variaveis(arg='arg', 'arg2', 'arg3', 'arg3')
```

O seguinte erro será lançado:

```
File "<stdin>", line 1
SyntaxError: positional argument follows keyword argument
```

Agora vamos a um exemplo mais útil!

Uma função que soma os argumentos passados, independente do número de argumentos que seja passado, veja:

```
def adicao(*args):
    resultado = 0

    for argumento in args:
        resultado += argumento

    return resultado

print(adicao(1, 2))
print(adição(1, 2, 4, 6))
print(adição(1, 2, 4, 6, 8, 10))
```

Resultando em:

```
3
13
31
```

> 💡 *Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Te convido a conhecer!*

Argumentos nomeados `**kwargs`

O `**kwargs` possibilita verificarmos os parâmetros nomeados da função, isto é, aqueles parâmetros que são passados com um nome!

Eles estarão disponíveis como um dicionário (`{'chave': 'valor'}`) dentro da função.

Vamos ao exemplo:

```python
def concatena (**kwargs):
    print(f'Valores recebidos: {kwargs}')
    resultado = ""

    for valor in kwargs.values():
        resultado += f'{valor} '
    return resultado

print(concatena(a="Python", b="Academy",  c="Rules!"))
print()
print(concatena(a="Python", b="é", c="na", d="Python", e="Academy!"))
```

Veja a saída, perceba que recebemos um `dict` na variável `kwargs`:

```
Valores recebidos: {'a': 'Python', 'b': 'Academy', 'c': 'Rules!'}
Python Academy Rules!

Valores recebidos: {'a': 'Python', 'b': 'é', 'c': 'na', 'd': 'Python',
'e': 'Academy!'}
Python é na Python Academy!
```

Para melhor entender vamos juntar tudo e printar o que vem em cada um dos tipos de parâmetros.

Veja o exemplo:

```python
def funcao(arg_1, arg_2, *args, arg_kw, **kwargs):
    print(f'Parâmetro 1: {arg_1}')
    print(f'Parâmetro 2: {arg_2}')
    print(f'*args: {args}')
    print(f'Argumento nomeado: {arg_kw}')
    print(f'**kwargs: {kwargs}')

# Chamada da Função
funcao(
    1,          # Parâmetro 1
    'a',        # Parâmetro 2
    'arg1',     # *args
    'arg2',     # *args
    2,          # *args
    arg_kw='KW',# arg_kw
    banana='B', # **kwargs
    bananas='A' # **kwargs
)
```

A saída deste código será:

```
Parâmetro 1: 1
Parâmetro 2: a
*args: ('arg1', 'arg2', 2)
Argumento nomeado: KW
**kwargs: {'banana': 'B', 'bananas': 'A'}
```

Conclusão

Agora você entende de onde vem esse tal de `*args` e `**kwargs`!

Se tiver qualquer dúvida, manda aqui embaixo no box de comentários 😉

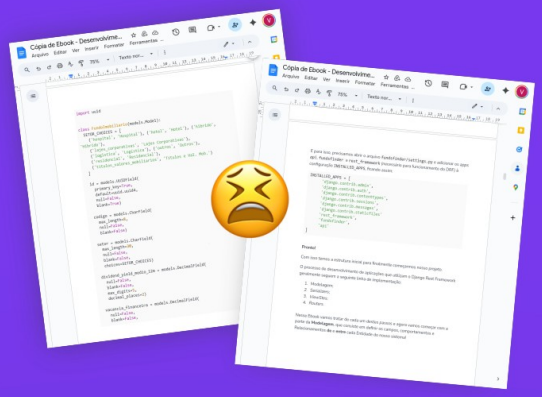
Até a próxima!

Não se esqueça de conferir!

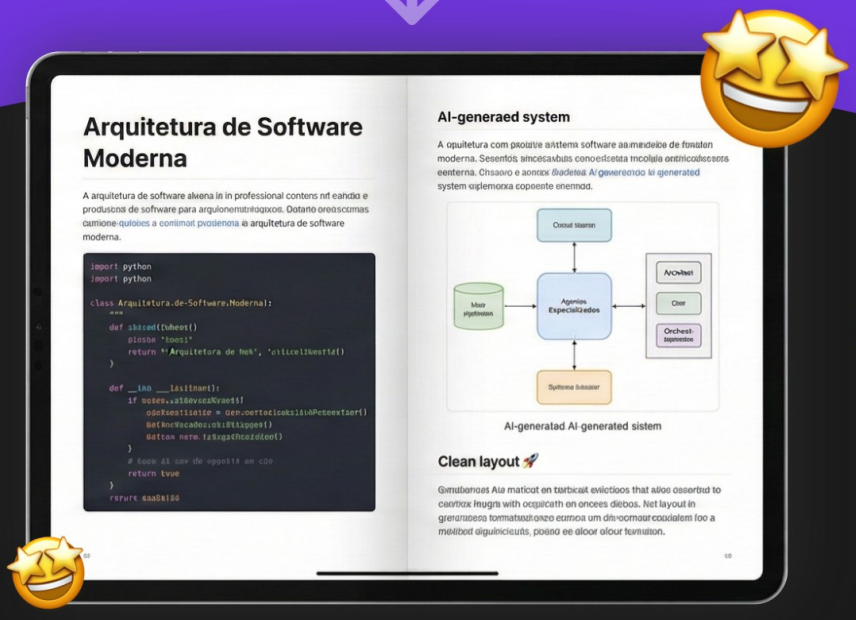
DevBook

Crie Ebooks técnicos em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código**!



Chega de formatar código no Google Docs



Syntax Highlight

Infográficos feitos por IA

Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

Edite em Markdown em Tempo Real

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS