Qual a forma correta de executar a função: def criar\_carro(\*, modelo, ano, placa, marca, motor, combustivel) criar\_carro(modelo="Palio", ano=1999, placa="ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina") criar\_carro("Palio", 1999, "ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina")

**O ChatGPT disse:**

ChatGPT

A função que você definiu usa o parâmetro \* antes dos outros parâmetros, o que significa que todos os parâmetros a seguir devem ser nomeados ao chamá-la. Isso impede que você passe argumentos posicionais para esses parâmetros. Portanto, a forma correta de chamar a função é utilizando argumentos nomeados.

Dessa forma, a primeira chamada está correta:

python

Copiar código

criar\_carro(modelo="Palio", ano=1999, placa="ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina")

A segunda chamada:

python

Copiar código

criar\_carro("Palio", 1999, "ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina")

não está correta, porque os primeiros três argumentos não têm nomes e a função exige que todos os parâmetros sejam passados com nome.

Portanto, sempre utilize a sintaxe com argumentos nomeados ao chamar essa função.

4o mini

**Você disse:**

Em Python podemos atribuir funções a variáveis, passá-las como parâmetro para outras funções e usar como valor de retorno para uma função. VERDADEIRO FALSO

**O ChatGPT disse:**

ChatGPT

**VERDADEIRO.**

Em Python, funções são objetos de primeira classe, o que significa que você pode:

1. Atribuir funções a variáveis.
2. Passar funções como parâmetros para outras funções.
3. Retornar funções de outras funções.

Aqui está um exemplo simples para ilustrar isso:

python

Copiar código

# Atribuindo uma função a uma variável

def saudacao():

return "Olá!"

minha\_funcao = saudacao # Atribuindo a função a uma variável

print(minha\_funcao()) # Chamando a função através da variável

# Passando uma função como parâmetro

def executa\_funcao(func):

return func()

print(executa\_funcao(saudacao)) # Passando a função saudacao como parâmetro

# Usando uma função como valor de retorno

def cria\_funcao():

def nova\_funcao():

return "Nova função criada!"

return nova\_funcao

funcao\_criada = cria\_funcao() # Retornando a função

print(funcao\_criada()) # Chamando a nova função

Assim, a afirmação é verdadeira!

4o mini

**Você disse:**

É possível utilizar variáveis globais no escopo da função sem receber a variável por argumento? Sim, utilizando a palavra reservada global Não, Python não permite o uso de variáveis que estão fora do escopo da função

**O ChatGPT disse:**

ChatGPT

**Sim, utilizando a palavra reservada global.**

Em Python, você pode usar a palavra-chave global dentro de uma função para indicar que você está se referindo a uma variável global, em vez de criar uma nova variável local com o mesmo nome. Aqui está um exemplo:

python

Copiar código

x = 10 # Variável global

def funcao():

global x # Indica que estamos usando a variável global x

x += 5 # Modifica a variável global

print(x)

funcao() # Saída: 15

print(x) # Saída: 15

Sem a palavra-chave global, qualquer atribuição a x dentro da função criaria uma nova variável local x, sem afetar a variável global. Portanto, a resposta correta é a primeira: sim, utilizando a palavra reservada global.

4o mini

**Você disse:**

Funções em Python podem retornar mais de um valor. FALSO VERDADEIRO

**O ChatGPT disse:**

ChatGPT

**VERDADEIRO.**

Em Python, funções podem retornar múltiplos valores usando uma tupla. Você pode simplesmente separar os valores com vírgulas na instrução return. Aqui está um exemplo:

python

Copiar código

def retornar\_multiplos\_valores():

return 1, 2, 3 # Retorna uma tupla com três valores

a, b, c = retornar\_multiplos\_valores() # Desempacotando os valores

print(a, b, c) # Saída: 1 2 3

No exemplo acima, a função retornar\_multiplos\_valores retorna três valores, que podem ser facilmente desempacotados em variáveis separadas. Portanto, a afirmação é verdadeira!

4o mini

**Você disse:**

Qual a forma correta de executar a função: def criar\_carro(modelo, ano, placa, /, marca, motor, combustivel) criar\_carro("Palio", 1999, "ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina") criar\_carro(modelo="Palio", ano=1999, placa="ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina")

**O ChatGPT disse:**

ChatGPT

A definição da função criar\_carro(modelo, ano, placa, /, marca, motor, combustivel) utiliza o caractere / para indicar que os três primeiros parâmetros (modelo, ano e placa) devem ser passados como argumentos posicionais, enquanto os outros três (marca, motor e combustivel) podem ser passados como argumentos nomeados.

Diante disso, a forma correta de chamar a função é a primeira:

python

Copiar código

criar\_carro("Palio", 1999, "ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina")

A segunda chamada:

python

Copiar código

criar\_carro(modelo="Palio", ano=1999, placa="ABC-1234", marca="Fiat", motor="1.0", combustivel="Gasolina")

não está correta, pois está tentando passar os primeiros três argumentos como nomeados, o que não é permitido devido à presença do /.

Portanto, apenas a primeira chamada é válida.

4o mini

**Você disse:**

Dado o código: def funcao(\*args, \*\*kw):, qual será o valor de args e kw ao executar: funcao("python", 2022, curso="dio") O código é inválido, irá acontecer um erro durante a execução do programa. args = ["python", 2022] e kw = {"curso": "dio"} args = ["python", 2022, "curso", "dio"] e kw = {} args = [] e kw = {"python": 2022, "curso": "dio"} args = ("python", 2022) e kw = {"curso": "dio"}

**O ChatGPT disse:**

ChatGPT

A declaração da função def funcao(\*args, \*\*kw): permite que você capture argumentos posicionais em args e argumentos nomeados em kw.

Ao chamar a função com funcao("python", 2022, curso="dio"), a execução ocorrerá da seguinte forma:

* Os valores "python" e 2022 serão capturados em args como uma tupla.
* O argumento nomeado curso="dio" será capturado em kw como um dicionário.

Portanto, os valores serão:

* args = ("python", 2022)
* kw = {"curso": "dio"}

Assim, a resposta correta é:

**args = ("python", 2022) e kw = {"curso": "dio"}.**

4o mini

**Você disse:**

Como podemos declarar uma função Python que recebe 2 argumentos? func minha\_funcao(a, b): def minha\_funcao(a, b) {} def minha\_funcao(a, b): function minha\_funcao(a, b): func minha\_funcao(a, b) {}

**O ChatGPT disse:**

ChatGPT

A forma correta de declarar uma função em Python que recebe dois argumentos é:

python

Copiar código

def minha\_funcao(a, b):

Portanto, a opção correta entre as que você forneceu é:

**def minha\_funcao(a, b):**

As outras opções estão incorretas por diferentes razões:

* func minha\_funcao(a, b): e function minha\_funcao(a, b): não são sintaxes válidas em Python.
* def minha\_funcao(a, b) {} utiliza chaves {}, que não são usadas para definir blocos de código em Python (ao invés disso, utilizamos dois pontos : e identação).

4o mini

Parte superior do formulário



Parte inferior do formulário