

Regras para parâmetros posicionais/nomeados

Para utilizar parâmetros posicionais e/ou nomeados é **mandatório** que os parâmetros **posicionais** estejam sempre à **esquerda** e os parâmetros **nomeados** estejam sempre à **direita**

Regras para chamada de funções

Chamando (executando) a função
passando parâmetros na forma
posicional

nome_da_funcao(1, 2)

def nome_da_funcao(parametro_um: int, parametro_dois: int):

...

Parâmetros nomeados podem ser passados em qualquer ordem!

nome_da_funcao(parametro_um=1, parametro_dois=2)

Chamando (executando) a função
passando parâmetros na forma
nomeada

Conceitos de execução de função (def)

Regras para parâmetros obrigatórios/opcionais

Para utilizar parâmetros obrigatórios e/ou opcionais é **mandatório** que os parâmetros obrigatórios estejam sempre à **esquerda** e os parâmetros opcionais estejam sempre à **direita**

Regras para criação de funções

**Parâmetro
Obrigatório**

**Parâmetro
Opcional**

`def nome_da_funcao(parametro_um: int, parametro_dois: int = 1):`
comandos_da_funcao
return

Vírgula para separar os parâmetros

**Anotando
Variável
(Parâmetro)**

Toda função retorna alguma coisa, mesmo que seja um objeto None.
Quando uma função não possui o comando 'return', esta função
retorna None.

O parâmetro obrigatório **deverá** ser passado no momento da chamada da função. O parâmetro opcional **poderá** ser passado na chamada desta; se não for passado, este parâmetro será atribuído pelo int 1 para ocorrer a execução da função.

Conceitos de criação de função (def)

nome_da_funcao(1, parametro_dois=2)

Posicional

Nomeado

No momento de **execução**
da função

é possível usar parâmetros posicionais **e/ou** nomeados
para isto,
é necessário passar os posicionais **SEMPRE**
à esquerda
e os nomeados **SEMPRE**
à direita

Do contrário, vai dar um erro (Exception)

Parâmetro que **obriga** que os parâmetros à **esquerda** sejam utilizados como **posicionais**

```
def nome_da_funcao(parametro_um: int, /, parametro_dois, *, parametro_tres: int = 1):  
    comandos_da_funcao  
    return
```

Desta forma, o **parâmetro_dois** poderá ser usado tanto como **posicional** ou **nomeado**

Parâmetro que **obriga** que os parâmetros à **direita** sejam utilizados como **nomeados**

Obrigando o usuário passar parâmetros posicionais/nomeados

Retornando uma tupla com três posições explicitamente

```
def nome_da_funcao(parametro_um: int, parametro_dois: int = 1):  
    valor_um = parametro_um + parametro_dois  
    valor_dois = parametro_um - parametro_dois  
    valor_tres = parametro_um * parametro_dois  
    return (valor_um, valor_dois, valor_tres)
```



```
def nome_da_funcao(parametro_um: int, parametro_dois: int = 1):  
    valor_um = parametro_um + parametro_dois  
    valor_dois = parametro_um - parametro_dois  
    valor_tres = parametro_um * parametro_dois  
    return valor_um, valor_dois, valor_tres
```

Retornando uma tupla com três posições implicitamente

Agrupando parâmetros

O parâmetro `args`
por possuir o `*`
vai **agrupar** todos os
parâmetros **posicionais**
passados na **execução**

O parâmetro `kwargs`
por possuir o `**`
vai **agrupar** todos os
parâmetros **nomeados**
passados na **execução**

```
def nome_da_funcao(*args, **kwargs):
```

```
    print(args) ← Vai imprimir uma tupla: (1, 2, 3, 4)
```

```
    print(kwargs) ← Vai imprimir um dicionário: {"parametro_um": 7, "parametro_dois": 9}
```

```
    ← Vai retornar None, pois a função termina sem o comando return
```

```
nome_da_funcao(1, 2, 3, 4, parametro_um=7, parametro_dois=9)
```

Execução (chamada) da função **nome_da_funcao**

1. A tupla **args** e o dicionário **kwargs** são apenas os nomes dos parâmetros dentro da função. É possível escolher outros nomes.
2. A sintaxe `*` e `**` permite que a função possua uma quantidade de parâmetros variável para parâmetros posicionais e nomeados.
3. É possível usar apenas `*` ou apenas `**` dependendo das necessidades da aplicação.

Posicionais e Nomeados

Explodindo parâmetros

Parâmetros que serão “**abastecidos**”
por uma tupla/lista “**explodida**”

Parâmetros que serão “**abastecidos**”
por um dicionário “**explodido**”

```
def nome_da_funcao(parametro_um, parametro_dois, parametro_tres, parametro_quatro):  
    print(parametro_um, parametro_dois) ← Vai imprimir 1 e 2 (devido aos valores que serão passados)  
    print(parametro_tres, parametro_quatro) ← Vai imprimir 5 e 8 (devido aos valores que serão passados)  
    ← Vai retornar None, pois a função termina sem o comando return
```

tupla = (1, 2)

dicionario = {"parametro_tres": 5, "parametro_quatro": 8}

nome_da_funcao(*tupla, **dicionario) ← Execução (chamada) da função **nome_da_funcao**

A tupla vai ser
“**explodida**”

e a **primeira** posição da tupla
vai ser usada no **parametro_um**
(por ser a primeira posição da função)
e a **segunda** posição da tupla
vai ser usada no **parametro_dois**
(por ser a segunda posição da função)

O dicionário vai ser
“**explodido**”

e a **primeira** chave do dicionário
vai ser usada no **parametro_tres**
(por possuir o mesmo nome dentro da função)
e a **segunda** chave do dicionário
vai ser usada no **parametro_quatro**
(por possuir o mesmo nome dentro da função)

Posicionais e Nomeados