F<BR/C>
DE SOFTWARE

# UC: PROGRAMAÇÃO ESTRUTURADA E ORIENTADA A OBJETOS

## PROFESSOR: THIAGO ALMEIDA

# EXERCÍCIOS FUNÇÕES

11 - Crie uma função que receba como argumento uma lista, com valores de qualquer tipo. A função deve imprimir todos os elementos da lista numerando-os. Exemplo:

1, Pera

2, Uva

3, Maça

4, Salada mista

## EXERCÍCIOS FUNÇÕES

12 - Crie uma função que receba uma lista como argumento, os valores da lista devem ser numéricos, por fim retorne a média desses valores.

## EXERCÍCIOS FUNÇÕES

13 – Crie uma função que retorne uma lista com valor padrão. Para esta função, consideraremos como argumentos de entrada a ***quantidade de elementos*** e o ***valor padrão*** a ser atribuído a todos eles. Exemplo de retorno:

[1,1,1,1,1,1,1,1]

[“A”,”A”,”A”,”A”]

## EXERCÍCIOS FUNÇÕES

14 – Crie uma função que receba como argumento a potência elétrica de determinado aparelho e o tempo ligado e retorne o consumo em KWh em seguida chame outra função para calcular a conta de energia, levando em consideração o  consumo e o valor do KWh como argumentos.

# FUNÇÕES COM MÚLTIPLOS ARGUMENTOS *ARGS **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)

# FUNÇÕES COM MÚLTIPLOS ARGUMENTOS *ARGS **KWARGS

```
def argumentos(arg, *args):
    print("Primeiro argumento:", arg)
    for item in args:
        print("Argumento de *args", item)

argumentos('primeiro_argumento', 'banana', 'maça',
'laranja')
```

## FUNÇÕES COM MÚLTIPLOS ARGUMENTOS

Argumentos não nomeados ***args:***

```
def world_cup_titles(country, *args):
    print('Country: ', country))
    for title in args:
        print('year: ', title)
```

# FUNÇÕES COM MÚLTIPLOS ARGUMENTOS

Argumentos não nomeados *args, o python recebe os argumentos em uma lista, na qual podemos acessar seu valor pela de acordo com posição arg[0] ou em um loop for:

```
def world_cup_titles(country, *args):
    print('Country: ', country))
    for title in args:
        print('year: ', title)
```

```
>>>world_cup_titles('Brasil', '1958', '1962', '1970', '1994', '2002')
```

# FUNÇÕES COM MÚLTIPLOS ARGUMENTOS

Argumentos nomeados *****kwargs:***

**kwargs faz um dicionário cujo conteúdo podemos acessar através do nome do argumento.

```
def local_estudo(**kwargs):
    print(kwargs)
    for key, value in kwargs.items():
        print ("%s == %s" %(key, value))
local_estudo(first ='Senac', second ='Hub', third='Academy')
```

# FUNÇÕES COM MÚLTIPLOS ARGUMENTOS

Argumentos nomeados *****kwargs:***

**kwargs faz um dicionário cujo conteúdo podemos acessar através do nome do argumento.

```
def calculate_tax(value, **kwargs):
    total = 0
    print(kwargs)
    if 'iss' in kwargs:
        total += value * kwargs['iss']
    if 'pis' in kwargs:
        total += value * kwargs['pis']
    return total
```

# **KWARGS

Chamada de função **kwargs:

A diferença da estrutura no código é que no *args usamos 1 asterisco e não damos o nome ao parâmetro e no **Kwargs usamos 2 asteriscos e damos nome ao parâmetro.

calculate_tax(1000, iss=0,05, pis=0.033)

O resultado do Kwargs é um dicionário com todos os parâmetros nele, por este motivo você tem a opção de criar condições para cada parâmetro usando o IF ou outras estruturas condicionais:

```
if ‘iss’ in kwargs:
        total += value * kwargs[‘iss’]
    if ‘pis’ in kwargs:
```

# **KWARGS

```
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!"))
```

## EXERCÍCIOS FUNÇÕES

15 – Crie uma função que receba múltiplos argumentos não nomeados, considere que a função receba números inteiros como argumentos e retorne a soma dos argumentos.

## EXERCÍCIOS FUNÇÕES

16 – Crie uma função que receba múltiplos argumentos não nomeados, considere que a função receba números flutuantes como argumentos e retorne a média dos argumentos.

# EXERCÍCIOS FUNÇÕES

17 – Crie uma função que armazene os dados de uma pessoa em um dicionário e imprima-os na tela, Utilize argumentos nomeados **kwargs, exemplo de saída:

```
Data type of argument: <class 'dict'>
Firstname is John
Lastname is Wood
Email is johnwood@nomail.com
Country is Wakanda
Age is 25
Phone is 9876543210
```

# EXERCÍCIOS FUNÇÕES

18 – Uma rede de churrascaria realiza promoções semanais e precisa automatizar os descontos de acordo com o dia da semana (terça-feira 10%, quarta-feira 15%, quinta-feira 20%). Crie uma função que calcule o preço final do consumo por pessoa. Considere a taxa de atendimento e o couvert, caso o cliente concorde com o pagamento. Utilize argumentos nomeados **kwargs, Exemplo de chamada da função:

desconto('quinta-feira',valor=99.90,taxa=0.10,couvert=15)

```
Conta S/ Taxas:  79.92
Conta C/ Taxas:
 Rodízio: 79.92
 Taxa Serviço: 7.99
 Couvert: 15
 TOTAL R$: 102.91
```