

**Nome:** João Geiger Piza

**Repositório no Github:** <https://github.com/joaogpizza/Lipai>

**Códigos das Videoaulas:**

aula01.py:

```
""" Aula 01 - Introdução a funções """

# Função é um bloco que realiza uma tarefa específica
# Dividir o problema em pequenas partes
# Evitar duplicação de código

# 1. Standard Library Functions - built-in functions
# ex: print, len

print('Olá', 123, True)

nomes = ['João', 'Maria']
tamanho_lista = len(nomes)
print(nomes, tamanho_lista)

# 2. User Defined Functions
# Definidas pelo desenvolvedor(a)
# Fazerem parte da solução do problema

# Declaração
# nome: saudacoes
# parametros: nenhum
# retorno: nenhum
def saudacoes():
    print('Olá')

# Chamada
saudacoes()
saudacoes()
saudacoes()

# Declaração
# nome: saudacoes
# parametros: nome
# retorno: nenhum
def saudacoes(nome):
```

```
print(f'Olá {nome}')
```

# Chamada

# valores, variáveis, expressões => argumentos

# 'Maria' é um argumento passado para o parâmetro nome

```
saudacoes('Maria')
saudacoes('Pedro')
nome = 'Carlos'
saudacoes(nome)
```

# Declaração

# nome: calcular\_media

# parametros: nota1, nota2, nota3

# retorno: nenhum

```
def calcular_media(nota1, nota2, nota3):
    media = (nota1 + nota2 + nota3) / 3
    print(media)
```

# Chamada

# argumentos são literais

```
calcular_media(10.0, 3.0, 6.0)
```

n1 = 10.0

n2 = 3.0

n3 = 8.0

# argumentos são variáveis

```
calcular_media(n1, n2, n3)
```

# Declaração

# nome: calcular\_media

# parametros: nota1, nota2, nota3

# retorno: media

```
def calcular_media(nota1, nota2, nota3):
    media = (nota1 + nota2 + nota3) / 3
    return media
```

media = calcular\_media(10.0, 8.4, 3.2)

```
print('valor da média', media)
```

# enviar no email

# salvar no banco de dados

# salvar no arquivo

aula02.py:

```
""" Aula02 - Arguments: positional, keyword, default value """

# declara uma funcao que soma dois numeros
def somar(n1, n2):
    return n1 + n2

def dividir(dividendo, divisor):
    return dividendo / divisor

# argumentos posicionais
print(somar(10.0, 3.5))
print(somar(2.0, 6.5))
print(dividir(10.0, 2.0))

# unpack list e tuplas
numeros = (10.0, 5.5)
print('somar numeros de uma lista', somar(numeros[0], numeros[1]))
print('somar numeros de uma lista', somar(*numeros))

# argumentos nomeados (keyword)
print(somar(n1=3.0, n2=6.2))
print(somar(n2=5.0, n1=7.8))
print(dividir(divisor=3.0, dividendo=10.0))

# unpack dict
numeros = {
    "n1": 10.2,
    "n2": 5.3
}

print('somar numeros de um dict', somar(**numeros))

# Declaracao
# nome: saudacoes
# parametros: nome (obrigatorio), saudacao (opcional)
# retorno: string
def saudacoes(nome, saudacao='Oi'):
    return f'{saudacao} {nome}'

print(saudacoes('João', 'Olá'))
```

```
print(saudacoes('Maria', 'Bom dia'))
print(saudacoes('Pedro'))

print(saudacoes(saudacao='Oi Oi', nome='Marcio'))
print(saudacoes(nome='Karina'))
```

aula03.py:

```
""" Aula 03 - Entendendo *args e *kwargs """

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')
world_cup_titles('Espanha', '2010')

def calculate_price(value, **kwargs):
    tax_percentage = kwargs.get('tax_percentage')
    discount = kwargs.get('discount')
    if tax_percentage:
        value += value * (tax_percentage / 100)
    if discount:
        value -= discount
    return value

final_price = calculate_price(100.0)
print(final_price)

final_price = calculate_price(100.0, discount=5.0)
print(final_price)

final_price = calculate_price(100.0, tax_percentage=7)
print(final_price)

final_price = calculate_price(100.0, tax_percentage=7, discount=5.0)
print(final_price)

# Testando os dois juntos
def calcular_preco(*args, **kwargs):
    porcentagem_imposto = kwargs.get('porcentagem_imposto')
    desconto = kwargs.get('desconto')
```

```

    resultado = []
    for item in args:
        if porcentagem_imposto:
            item += item * (porcentagem_imposto / 100)
        if desconto:
            item -= desconto
        resultado.append(item)
    return resultado

print('----')
valores = calcular_preco(10.0, 23.0, 118.0, 23.7, desconto=3.0,
porcentagem_imposto=10.0)
print(valores)

```

## Exercícios:

ex01.py:

```

""" Exercício 01 S2.A4 """

def soma(n1, n2, n3):
    print(n1 + n2 + n3)

soma(3, 4, 7)

```

ex02.py:

```

""" Exercício 02 S2.A4 """

def soma(n1, n2, n3):
    return n1 + n2 + n3

RESULTADO = soma(1, 2, 3)
print(RESULTADO)

```

ex03.py:

```

""" Exercício 03 S2.A4 """

def soma(numeros):
    resultado = 0
    for numero in numeros:
        resultado += numero
    return resultado

RESULTADO = soma((1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

```

```
print(RESULTADO)
```

ex04.py:

```
""" Exercício 04 S2.A4 """

def soma(*args):
    resultado = 0
    for numero in args:
        resultado += numero
    return resultado

RESULTADO = soma(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
print(RESULTADO)
```

ex05.py:

```
""" Exercício 05 S2.A4 """

def calcular_imc(individuo):
    """Retorna o IMC de um indivíduo com base na sua altura e peso."""
    return individuo['peso'] / (individuo['altura'] ** 2)

def obter_classificacao(imc):
    """Retorna a classificação com base no IMC."""
    if imc < 18.5:
        return 'Baixo peso'
    elif imc < 25.0:
        return 'Peso normal'
    elif imc < 30.0:
        return 'Excesso de peso'
    elif imc < 35.0:
        return 'Obesidade de Classe 1'
    elif imc < 40:
        return 'Obesidade de Classe 2'
    return 'Obesidade de Classe 3'

def situacao_individuo(imc):
    """Retorna a situação ('normal', 'perder peso', 'ganhar peso') com
base no IMC"""
    if imc < 18.5:
        return 'ganhar peso'
    elif imc < 25.0:
        return 'normal'
    return 'perder peso'
```

```

PESO = float(input('Digite seu peso (kg): '))
ALTURA = float(input('Digite sua altura (m): '))

INDIVIDUO = {
    'peso': PESO,
    'altura': ALTURA
}

IMC = calcular_imc(INDIVIDUO)
CLASSIFICACAO = obter_classificacao(IMC)
SITUACAO = situacao_individuo(IMC)

print(f'IMC = {IMC}\n' +
      f'Classificação = {CLASSIFICACAO}\n' +
      f'Situação = {SITUACAO}')

```

ex06.py:

```

""" Exercício 06 S2.A4 """

def calcular_volume(informacoes):
    """ Calcula o volume por meio das informacoes fornecidas """
    return (informacoes['comprimento'] * informacoes['altura'] *
            informacoes['largura'] / 1000)

def potencia_termostato(informacoes):
    """ Calcula a potência do termostato por meio das informacoes
    fornecidas """
    volume = calcular_volume(informacoes)
    return volume * 0.05 * (informacoes['temperatura desejada'] -
                            informacoes['temperatura ambiente'])

def faixa_filtragem(informacoes):
    """ Calcula a faixa mínima e máxima de filtragem recomendada """
    volume = calcular_volume(informacoes)
    return (2*volume, 3*volume)

COMPRIMENTO = float(input('Digite o comprimento do aquário, em cm: '))
ALTURA = float(input('Digite a altura do aquário, em cm: '))
LARGURA = float(input('Digite a largura do aquário, em cm: '))
TEMP_DESEJADA = float(input('Digite a temperatura desejada: '))
TEMP_AMBIENTE = float(input('Digite a temperatura ambiente: '))

```

```
INFOS = {
    'comprimento': COMPRIMENTO,
    'altura': ALTURA,
    'largura': LARGURA,
    'temperatura desejada': TEMP_DESEJADA,
    'temperatura ambiente': TEMP_AMBIENTE
}

VOLUME = calcular_volume(INFOS)
POTENCIA = potencia_termostato(INFOS)
FAIXA_FILTRAGEM = faixa_filtragem(INFOS)

print(f'Volume: {VOLUME} Litros\n' +
      f'Potência requerida do termostato: {POTENCIA}\n' +
      f'Filtragem mínima: {FAIXA_FILTRAGEM[0]}\n' +
      f'Filtragem máxima: {FAIXA_FILTRAGEM[1]}\n')
```

### Reflexão sobre funções:

- 1) A função do ex02 é mais flexível que a do ex01, pois ela permite realizar outras operações além da impressão com o valor calculado dentro dela.
- 2) A abordagem do ex03/ex04 é mais flexível que a do ex02, pois permite um número qualquer de parâmetros, ao invés de requerer exatamente 3.
- 3) Prefiro usar tuplas quando ou os chamadores vão possuir tuplas ou quando quiser deixar claro que os dados passados como parâmetros não serão modificados. Prefiro \*args quando os chamadores não possuem uma estrutura com as variáveis que serão passadas como parâmetro.