Nome: André Machado Silva

Repositório no GitHub:
https://github.com/andrems1/onboarding-LIPAI/tree/main/Semana%202/Atividade_04

Código das Videoaulas

```
def saudacao(nome):
    print(f"Olá, {nome}!")

def somar(a, b):
    return a + b

saudacao("André")
resultado = somar(10, 5)
print(f"Resultado da soma: {resultado}")

#-------------------------------------

def exibir_info(nome, idade=18, cidade="Desconhecida"):
    print(f"Nome: {nome}, Idade: {idade}, Cidade: {cidade}")

# Chamada posicional
exibir_info("Maria", 25, "São Paulo")

# Chamada nomeada
exibir_info(cidade="Rio de Janeiro", nome="João", idade=30)

# valor padrão
exibir_info("Pedro")

#-------------------------------------

# múltiplos argumentos como uma tupla
def somar_todos(*args):
    return sum(args)

print(somar_todos(1, 2, 3, 4, 5))

# múltiplos argumentos nomeados como dicionário
def apresentar_dados(**kwargs):
    for chave, valor in kwargs.items():
        print(f"{chave}: {valor}")

apresentar_dados(nome="Ana", curso="Python", nivel="Iniciante")
```

Exercícios
ex01

```
def soma_imprime(n1, n2, n3):
    res = n1 + n2 + n3
    print(f"A soma é: {res}")


soma_imprime(10, 20, 30)
```

ex02

```
def soma_retorna(n1, n2, n3):
    return n1 + n2 + n3

resultado = soma_retorna(10, 20, 30)
print(f"O retorno foi: {resultado}")
```

ex03

```
def soma_tupla(numeros):
    soma = 0
    for num in numeros:
        soma += num
    return soma

valores = (5, 10, 15)
print(f"Soma da tupla: {soma_tupla(valores)}")
```

ex04

```
def soma_args(*args):
    return sum(args)

print(f"Soma args: {soma_args(1, 2, 3, 4, 5)}")
```

ex05

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

def obter_classificacao(imc):
    """Retorna a classificação com base no IMC."""
    if imc < 18.5:
        return "Baixo peso"
```

```
    elif 18.5 <= imc <= 24.9:
        return "Peso normal"
    elif 25.0 <= imc <= 29.9:
        return "Excesso de peso"
    elif 30.0 <= imc <= 34.9:
        return "Obesidade de Classe 1"
    elif 35.0 <= imc <= 39.9:
        return "Obesidade de Classe 2"
    else:
        return "Obesidade de Classe 3"

def situacao_individuo(imc):
    """Retorna a situação ('normal', 'perder peso', 'ganhar peso')."""
    if 18.5 <= imc <= 24.9:
        return "Normal"
    elif imc < 18.5:
        return "Ganhar peso"
    else:
        return "Perder peso"

# Programa principal
p = float(input("Digite o peso (kg): "))
a = float(input("Digite a altura (m): "))

pessoa = {'peso': p, 'altura': a}

imc_calc = calcular_imc(pessoa)
classificacao = obter_classificacao(imc_calc)
situacao = situacao_individuo(imc_calc)

print(f"\nIMC: {imc_calc:.2f}")
print(f"Classificação: {classificacao}")
print(f"Recomendação: {situacao}")
```

ex06

```
def calcular_volume(medidas):
    v = (medidas['comprimento'] * medidas['altura'] * medidas['largura']) / 1000
    return v

def calcular_potencia_termostato(volume, t_desejada, t_ambiente):
    return volume * 0.05 * (t_desejada - t_ambiente)
```

```
def calcular_filtragem(volume):
    minimo = 2 * volume
    maximo = 3 * volume
    return minimo, maximo

# Entrada
dados_aquario = {
    'comprimento': float(input("Comprimento (cm): ")),
    'altura': float(input("Altura (cm): ")),
    'largura': float(input("Largura (cm): "))
}
temp_amb = float(input("Temperatura ambiente (°C): "))
temp_des = float(input("Temperatura desejada (°C): "))

# Processamento
vol = calcular_volume(dados_aquario)
potencia = calcular_potencia_termostato(vol, temp_des, temp_amb)
filt_min, filt_max = calcular_filtragem(vol)

# Saída
print(f"\n--- Resultados ---")
print(f"Volume do aquário: {vol:.2f} Litros")
print(f"Potência do termostato: {potencia:.2f} W")
print(f"Filtragem necessária: entre {filt_min:.1f} e {filt_max:.1f} Litros/hora")
```