**Listas.py**

# Criação da lista mesclada

lista\_mesclada = [1, 2, 3, "Olá, Python", True, 12.6]

print("Conteúdo da lista:", lista\_mesclada)

# Adicionando uma lista aninhada usando o método append

lista\_mesclada.append(["Lista aninhada"])

print("Conteúdo da lista após append:", lista\_mesclada)

# Inserindo o número 5 na posição 4

lista\_mesclada.insert(4, 5)

print("Conteúdo da lista após insert:", lista\_mesclada)

# Imprimindo o tamanho atual da lista

print("Tamanho da lista:", len(lista\_mesclada))

# Removendo o item da posição 1

lista\_mesclada.pop(1)

print("Conteúdo da lista após remover o item da posição 1:", lista\_mesclada)

# Criando uma nova lista com os primeiros 4 elementos

nova\_lista\_mesclada = lista\_mesclada[:4]

print("Conteúdo da nova lista mesclada:", nova\_lista\_mesclada)

**Tuplas.py**

# Criação da tupla

primeira\_tupla = (1, 2, 3, 4, "Olá, tupla")

# Imprimindo o conteúdo da tupla

print("Conteúdo da tupla:", primeira\_tupla)

# Imprimindo o índice do elemento 4

indice\_do\_quatro = primeira\_tupla.index(4)

print("Índice do elemento 4:", indice\_do\_quatro)

# Verificando se a tupla contém o elemento 3

contem\_tres = 3 in primeira\_tupla

print("A tupla contém o elemento 3?", contem\_tres)

# Verificando se a tupla contém o elemento 33

contem\_trinta\_tres = 33 in primeira\_tupla

print("A tupla contém o elemento 33?", contem\_trinta\_tres)

**Sets.py**

# Criação do set inicial

set\_inicial = {11, 12, 13, 14}

print("Conteúdo do set inicial:", set\_inicial)

# Adicionando o elemento 15 ao set

set\_inicial.add(15)

print("Conteúdo do set após adicionar 15:", set\_inicial)

# Atualizando o set com novos elementos

set\_inicial.update([1, 2, 3, 4, 5])

print("Conteúdo do set após a atualização:", set\_inicial)

# Removendo o elemento 13 do set

set\_inicial.discard(13)

print("Conteúdo do set após remover o 13:", set\_inicial)

# Criação de um novo set com o método set()

novo\_set = set([20, 21, 23, 1, 2])

print("Conteúdo do novo set:", novo\_set)

# União dos dois sets

uniao\_sets = set\_inicial.union(novo\_set)

print("União dos dois sets:", uniao\_sets)

# Interseção dos dois sets

intersecao\_sets = set\_inicial.intersection(novo\_set)

print("Interseção dos dois sets:", intersecao\_sets)

# Diferença entre os dois sets

diferenca\_sets = set\_inicial.difference(novo\_set)

print("Diferença entre os dois sets:", diferenca\_sets)

# Diferença simétrica entre os dois sets

diferenca\_simetrica\_sets = set\_inicial.symmetric\_difference(novo\_set)

print("Diferença simétrica entre os dois sets:", diferenca\_simetrica\_sets)

**Dicionários.py**

# Criação do dicionário meu\_dicionario

meu\_dicionario = {

    "codigo\_1": "Python",

    "codigo\_2": "Java",

    "codigo\_3": "PHP"

}

# Imprimindo o conteúdo do dicionário

print("Conteúdo do dicionário:", meu\_dicionario)

# Imprimindo o tipo de dados do dicionário

print("Tipo de dados do dicionário:", type(meu\_dicionario))

# Utilizando o método get para imprimir o valor da chave "linguagem"

linguagem = meu\_dicionario.get("codigo\_1")

print("Valor da chave 'codigo\_1':", linguagem)

# Imprimindo o tamanho do dicionário

print("Tamanho do dicionário:", len(meu\_dicionario))

# Criação de um novo dicionário aninhado chamado dicionario\_frutas

dicionario\_frutas = {

    1: {"nome": "limão", "tipo": "ácida"},

    2: {"nome": "laranja", "tipo": "ácida"},

    3: {"nome": "manga", "tipo": "semiácida"},

    4: {"nome": "maçã", "tipo": "semiácida"},

    5: {"nome": "banana", "tipo": "doce"},

    6: {"nome": "mamão", "tipo": "doce"}

}

# Imprimindo o valor das chaves "nome" e "tipo" da chave 1

print("Chave 1 - Nome:", dicionario\_frutas[1]["nome"], "- Tipo:", dicionario\_frutas[1]["tipo"])

# Imprimindo o valor das chaves "nome" e "tipo" da chave 2

print("Chave 2 - Nome:", dicionario\_frutas[2]["nome"], "- Tipo:", dicionario\_frutas[2]["tipo"])

# Iterando no dicionário dicionario\_frutas e imprimindo os valores das chaves "nome" e "tipo"

for chave, valor in dicionario\_frutas.items():

    print(f"Chave {chave} - Nome: {valor['nome']} - Tipo: {valor['tipo']}")

**Dicionários2.py**

# Criação do dicionário inicial

meu\_dicionario = {

    1: {'nome': 'Maria', 'idade': 26, 'nacionalidade': 'brasileira'}

}

# Utilizando o método update para adicionar novos elementos

meu\_dicionario.update({

    2: {'nome': 'João', 'idade': 30, 'nacionalidade': 'português'},

    3: {'nome': 'Ana', 'idade': 22, 'nacionalidade': 'espanhola'}

})

# Imprimindo o dicionário atualizado

print("Dicionário atualizado:", meu\_dicionario)

# Criando uma cópia do dicionário

copia\_dicionario = meu\_dicionario.copy()

# Removendo um elemento com pop e imprimindo o conteúdo

elemento\_removido = meu\_dicionario.pop(2)

print("Elemento removido (chave 2):", elemento\_removido)

print("Dicionário após remoção com pop:", meu\_dicionario)

# Removendo o último elemento com popitem e imprimindo o conteúdo

elemento\_removido\_final = meu\_dicionario.popitem()

print("Último elemento removido com popitem:", elemento\_removido\_final)

print("Dicionário após remoção com popitem:", meu\_dicionario)

# Removendo todos os elementos dos dicionários

meu\_dicionario.clear()

copia\_dicionario.clear()

print("Dicionário original após clear:", meu\_dicionario)

print("Cópia do dicionário após clear:", copia\_dicionario)

# Usando o método fromKeys para criar um novo dicionário

chaves = ['a', 'b', 'c']

novo\_dicionario = dict.fromkeys(chaves, 'valor\_padrao')

# Imprimindo o conteúdo do novo dicionário usando o método items

print("Conteúdo do novo dicionário (items):", novo\_dicionario.items())

# Imprimindo apenas as chaves do novo dicionário usando o método keys

print("Chaves do novo dicionário:", novo\_dicionario.keys())

# Imprimindo apenas os valores do novo dicionário usando o método values

print("Valores do novo dicionário:", novo\_dicionario.values())

**Operacoes.py**

def calcular\_media(notas):

    """

    Calcula a média das notas recebidas.

    :param notas: Lista com as notas dos 4 bimestres.

    :return: Média das notas.

    """

    return sum(notas) / len(notas)

def verificar\_reprovacao(media):

    """

    Verifica se o aluno foi reprovado com base na média.

    :param media: Média das notas do aluno.

    :return: True se a média for inferior a 6 (reprovado), False caso contrário.

    """

    return media < 6

def alunos\_reprovados(dados\_alunos, matriculas\_reprovados):

    """

    Retorna a lista de alunos reprovados.

    :param dados\_alunos: Dicionário com os dados dos alunos.

    :param matriculas\_reprovados: Lista com as matrículas dos alunos reprovados.

    :return: Mensagem com os alunos reprovados.

    """

    for matricula in matriculas\_reprovados:

        aluno = dados\_alunos[matricula]

        media = calcular\_media(aluno["notas"])

        print(f'Aluno Reprovado: {aluno["nome"]} – Matrícula: {matricula} – Média Final: {media:.2f}')

**Main.py**

from operacoes import calcular\_media, verificar\_reprovacao, alunos\_reprovados

# Dados dos alunos

dados\_alunos = {

    26: {"nome": "Maria", "notas": [8, 7, 5, 9]},

    101: {"nome": "Ana", "notas": [9, 9, 8, 9]},

    13: {"nome": "João", "notas": [6, 5, 5, 5]},

    37: {"nome": "Ágatha", "notas": [8, 6, 7.5, 9]},

    72: {"nome": "Joaquim", "notas": [6, 5.5, 5, 7]},

    5: {"nome": "Félix", "notas": [10, 8, 8, 8]}

}

# Lista de matrículas dos alunos reprovados

matriculas\_reprovados = []

# Verificação da média e aprovação/reprovação de cada aluno

for matricula, dados in dados\_alunos.items():

    media = calcular\_media(dados["notas"])

    if verificar\_reprovacao(media):

        matriculas\_reprovados.append(matricula)

# Exibição dos alunos reprovados

alunos\_reprovados(dados\_alunos, matriculas\_reprovados)