Script com alguns exemplos apresentados em aula.

Importante! Para executar cada programa, copie apenas o conteúdo referente a ele em um arquivo .py e execute com seu interpretador python

```
# Base de dados para armazenar todos os cadastros
base_dados = []  # Inicializo minha lista geral


while True:
    # Menu principal
    print("\nMenu Principal:")
    print("c - Cadastrar novo usuário")
    print("e - Exibir todos os cadastros")
    print("s - Sair do programa")

    opcao = input("Escolha uma opção: ").lower()

    if opcao == 'c':
        # Cadastro de novo usuário
        print("\nNovo Cadastro:")
        var_cadastro = []

        primeiro_nome = input("Qual o seu primeiro nome? ")
        var_cadastro.append(primeiro_nome)
        ultimo_sobrenome = input("Qual o seu último sobrenome? ")
        var_cadastro.append(ultimo_sobrenome)
        idade = int(input("Qual a sua idade? "))
        var_cadastro.append(idade)
        altura = float(input("Qual a sua altura em metros? "))
        var_cadastro.append(altura)
        base_dados.append(var_cadastro)
        print("\nCadastro realizado com sucesso!")

    elif opcao == 'e':
        print("\nTodos os Cadastros:")
        if len(base_dados) == 0:
            print("Nenhum cadastro encontrado.")
        else:
            for i in range(len(base_dados)):
                cadastro = base_dados[i]
                print(f"\nCadastro {i+1}:")  # i+1 para começar do 1 em vez de 0
                print(f"Primeiro nome: {cadastro[0]}")
                print(f"Último sobrenome: {cadastro[1]}")
                print(f"Idade: {cadastro[2]} anos") # print("Idade: ", cadastro[2], " anos")
                print(f"Altura: {cadastro[3]} metros")
```

```
    elif opcao == 's':
        print("Encerrando o programa...")
        break

    else:
        print("Opção inválida! Por favor, escolha 'c', 'e' ou 's'.")




'''
tupla = (1, 3, 5)

lista = list(tupla)   # lista[1, 3, 5]
lista[0] = 2     # lista[2, 3, 5]
print(tupla[0], lista[0])    # 1  2




print(index(mat_3x3, 3.1))  # Output esperado: [1][0]


while True:
    # Passo 1: Ler o nome do atleta
    nome = input("Digite o nome do atleta (ou deixe em branco para encerrar): ")

    # Verifica se o nome está vazio (encerra o programa)
    if nome == "":
        print("Programa encerrado.")
        break

    # Passo 2: Ler os cinco saltos
    saltos = []
    # media = 0
    for i in range(5):
        salto = float(input("Digite a distância do ", i+1, "º salto (em metros): "))
        saltos.append(salto)
        # media = media + salto
    # Passo 3: Calcular a média
    media = sum(saltos) / 5    # media = (saltos[0] +saltos[1] + saltos[2] + saltos[3] +
saltos[4])/ 5

    # Passo 4: Exibir os resultados
    print("\nResultado do atleta:")
    print(f"Nome: {nome}")
```

```
    print(type(saltos))  # Output esperado: list
    for cont in saltos:
        print("Salto: ", cont+1, "eh: ", saltos[cont])
    print("Média dos saltos:", media, "\n")
```

list - listas
Matrix - Matrizes
Array - vetores
Tuples - Tuplas

Exercício: Verificador de Quadrado Mágico
Um quadrado mágico é uma matriz 3x3 onde a soma dos números de cada linha, cada coluna e ambas as diagonais são iguais.

Tarefa:
Dada uma matriz 3x3 (uma lista de listas) com números de 1 a 9, verifique se ela é um quadrado mágico.

Exemplo de entrada: matriz = [[2, 7, 6],[9, 5, 1],[4, 3, 8]]

```
matriz = [
    [2, 7, 6],
    [9, 5, 1],
    [4, 3, 8]
]
```

Saída esperada:
"É um quadrado mágico!" # Pois todas as linhas, colunas e diagonais somam 15.

"Nao eh um quadrado magico!
Vamos lá:
Alternativa1: matriz = [[2, 7, 6],[9, 5, 1],[4, 3, 8]]

```
# Passo 1:
Alternativa2: Ler a matriz 3x3 do usuário
matriz = []
print("Digite os números da matriz 3x3, linha por linha:")

for i in range(3):
    linha = []
    for j in range(3):
        num = int(input(f"Digite o número da posição [{i+1}][{j+1}]: "))
        linha.append(num)
```

```
    matriz.append(linha)



# Passo 2: Calcular a soma esperada (primeira linha)
soma_esperada = 0
for num in matriz[0]:
    soma_esperada += num

# Passo 3: Verificar linhas
eh_magico = True

# Verifica linhas
for i in range(3):
    soma_linha = 0
    for j in range(3):
        soma_linha += matriz[i][j]
    if soma_linha != soma_esperada:
        eh_magico = False
        break

# Verifica colunas (se as linhas forem válidas)
if eh_magico:
    for j in range(3):
        soma_coluna = 0
        for i in range(3):
            soma_coluna += matriz[i][j]
        if soma_coluna != soma_esperada:
            eh_magico = False
            break

# Verifica diagonais (se linhas e colunas forem válidas)
if eh_magico:
    diagonal_principal = matriz[0][0] + matriz[1][1] + matriz[2][2]
    diagonal_secundaria = matriz[0][2] + matriz[1][1] + matriz[2][0]
    if diagonal_principal != soma_esperada or diagonal_secundaria != soma_esperada:
        eh_magico = False

# Passo 4: Exibir resultado
if eh_magico:
    print("É um quadrado mágico! Soma mágica =", soma_esperada)
else:
    print("Não é um quadrado mágico.")'"
```

```
# Tamanho fixo da matriz (3x3)
TAMANHO = 3


# Inicializa uma matriz vazia
matriz = []


print(f"Digite os elementos da matriz {TAMANHO}x{TAMANHO}, um por vez:")

# Preenche a matriz
for i in range(TAMANHO):
    linha = []  # Cria uma nova linha
    for j in range(TAMANHO):
        # Pede cada elemento individualmente
        while True:
            try:
                valor = float(input(f"Elemento [{i+1}][{j+1}]: "))
                linha.append(valor)
                break
            except ValueError:
                print("Por favor, digite um número válido.")
    matriz.append(linha)  # Adiciona a linha completa à matriz

# Mostra a matriz formatada
print("\nMatriz criada:")
for linha in matriz:
    print(linha)


# Passo 1: Usar o código do Exercício 1 para criar a matriz
print("Crie a matriz 2x2:")
n = 2
matriz = []

for i in range(n):
    while True:
        linha_input = input(f"Linha {i+1}: ").split()
        if len(linha_input) == n:
            try:
                linha = [float(elemento) for elemento in linha_input]
                matriz.append(linha)
                break
            except ValueError:
                print("Erro: Digite apenas números separados por espaços")
        else:
```

```
            print(f"Você deve digitar exatamente {n} elementos")

# Passo 2: Mostrar a matriz
print("\nMatriz informada:")
for linha in matriz:
    print(linha)

# Passo 3: Calcular o determinante
# Para matriz [[a, b], [c, d]]: determinante = (a*d) - (b*c)
a = matriz[0][0]
b = matriz[0][1]
c = matriz[1][0]
d = matriz[1][1]

determinante = (a * d) - (b * c)

# Passo 4: Mostrar o resultado
print(f"\nO determinante da matriz é: {determinante:.2f}")



# ========================================
# print() input()   importar uma biblioteca(import math pow() sqrt())

MAX = 8

# Seguindo com as funções
def saudacao(nome, mensagem="Bom dia"):
    print(f"{mensagem}, {nome}!") # Variavel dentro de uma funcao é de escopo local

def f(a, b, c):
    return (a + b * (3 ** c))

saudacao("Diego")
x = 1
y = 7
z = 6
print(z, y, x)
w = f(c=x, b=y, a=z)
print(w)



#Argumentos nomeados (keyword arguments) - Ordem não importa
def calcular_imc(peso, altura):
    return peso / (altura ** 2)

# inicio
```

```
alt = 1.90
w = 70
res = calcular_imc(w, alt)
print(res)
print(calcular_imc(56, 1.45))
print(calcular_imc(altura=1.75, peso=68))



#Calculadora

def mult(numero1, numero2):
    return numero1 * numero2
def divisao(op, a, b):
    return a/b

def calculadora_interativa():
    print("\nCalculadora Interativa")
    print("1. Somar")
    print("2. Subtrair")
    print("3. Multiplicar")
    print("4. Dividir")
    print("0. Sair")
    while True:
        opcao = input("\nEscolha uma operação (0-4): ")
        if opcao == "0":
            print("Calculadora encerrada.")
            break
        num1 = float(input('Digite o primeiro número: '))
        num2 = float(input("Digite o segundo número: "))
        if opcao == "1":
            print("Resultado: ", num1+num2)
            #resultado = calculadora(somar, num1, num2)
            #print(f"Resultado: {num1} + {num2} = {resultado}")
        elif opcao == '2':
            resultado = num1 - num2
            print(f"Resultado: {resultado}") #print("Resultado: ", resultado)
        elif opcao == '3':
            r = mult(num1, num2)
            print(f"Resultado: {r}")
        elif opcao == '4':
            operador = "/"
            print(divisao(operador, num1, num2))

# Inicio do meu programa
calculadora_interativa()
```

```
# Número variável de argumentos (*args e **kwargs)
def soma(*numeros):
    print(numeros)   # OUT: [1, 2, 3, 4, 5]
    total = 0
    for n in numeros:
        total += n
    return total

def soma_alt(a,b,e,r,v):
    lista = a,b,e,r,v
    total = 0
    for n in lista:
        total += n
    return total

print(soma(1, 2, 3, 4, 5))  # Pode receber qualquer quantidade
def mostrar_info(**kwargs):
    print(kwargs)   # OUT:[nome:"Diego", idade:35, cidade:"Curitiba"]
    for i, j in kwargs.items():
        print(f"{i}: {j}")

print("Inicio do meu programa aqui")
mostrar_info(nome="Diego", idade=35, cidade="Curitiba")



# Funções como Objetos de Primeira Classe
def quadrado(x):
    return x ** 2

def hello():
    print("Ola")

# Funções como argumentos
def aplicar_operacao(func, valor):
    #return quadrado(valor)
    return func(valor)  # return quadrado(4)

f = quadrado  # f agora é uma referência à função
print(f(5))  # Saída: 25

print("inicio do Programa!!")
print(aplicar_operacao(quadrado, 4))  # Saída: 16
```

```
[[a, a, a, a],
 [a, a, a, a]]


# RECURSÃO
def fatorial(n):
    if n == 1:
        return 1
    else:
        return n * fatorial(n - 1)

print(fatorial(5))  # Saída: 120


# ============================================
# Funções que retornam funções
def criar_multiplicador(n):
    print("comando da função hierarquicamente superior")
    def multiplicador(x):
        print("Comando dentro da funcao multiplicador")
        if..
        return x * n  # Funções podem retornar constantes, variaveis ou operações (lógica ou
aritimética)
    return multiplicador


# Inicio
dobro = criar_multiplicador(2)
triplo = criar_multiplicador(3)

print(dobro(5))  # Saída: 10
print(triplo(5))  # Saída: 15




# ============================================
# Funções Lambda (Anônimas)
# Função lambda simples
quadrado = lambda x: x ** 2   # lambda n: pow(n, 2)
print(quadrado(4))  # Saída: 16

# Uso com filter()
numeros = [1, 2, 3, 4]
quadrados = list(filter(lambda x: x**2, numeros))
print(quadrados)  # Saída: [1, 4, 9, 16]

pares = list(filter(lambda x: x % 2 == 0, numeros))
```

```
print(pares)  # Saída: [2, 4]



# Fibonacci
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)


# Inicio do programa
print([fibonacci(i) for i in range(10)])  # [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]







# Processamento de Matriz (Lista de Listas)
# Função para processar/transformar uma matriz
def processar_matriz(matriz, operacao):
    nova_matriz = []
    for linha in matriz:
        nova_linha = []
        for elemento in linha:
            novo_elemento = operacao(elemento)
            nova_linha.append(novo_elemento)
        nova_matriz.append(nova_linha)
    return nova_matriz

# Funções de operação para exemplos
def dobrar(num):
    return num * 2

def quadrado(num):
    return num ** 2

def eh_par(num):
    return 1 if num % 2 == 0 else 0

# Exemplo de uso
matriz_original = [
```

```
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]


# Aplicando diferentes operações
matriz_dobrada = processar_matriz(matriz_original, dobrar)
print("\nMatriz com elementos dobrados:")
for linha in matriz_dobrada:
    print(linha)

matriz_quadrados = processar_matriz(matriz_original, quadrado)
print("\nMatriz com quadrados dos elementos:")
for linha in matriz_quadrados:
    print(linha)

matriz_par_impar = processar_matriz(matriz_original, eh_par)
print("\nMatriz indicando pares (1) e ímpares (0):")
for linha in matriz_par_impar:
    print(linha)



# EXEMPLO 2... AINDA COM MATRIZ
def media_vizinhos(matriz, i, j):
    soma = 0
    cont = 0
    linhas = len(matriz)
    colunas = len(matriz[0]) if linhas > 0 else 0

    # Verifica todas as posições vizinhas
    for x in [i-1, i, i+1]:
        for y in [j-1, j, j+1]:
            # Ignora a própria posição e posições fora da matriz
            if (x != i or y != j) and 0 <= x < linhas and 0 <= y < colunas:
                soma += matriz[x][y]
                cont += 1
    return soma / cont if cont > 0 else 0    # Aqui temos condicional no retorno. Coisa nova

# Processamento especial: criar matriz de médias dos vizinhos
def criar_matriz_medias(matriz):
    nova_matriz = []
    for i in range(len(matriz)):
        nova_linha = []
        for j in range(len(matriz[i])):
            media = media_vizinhos(matriz, i, j)
            nova_linha.append(media)
```

```
        nova_matriz.append(nova_linha)
    return nova_matriz

matriz_original = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

# Inicio da funcao Main
matriz_medias = criar_matriz_medias(matriz_original)
print("\nMatriz com médias dos vizinhos:")
for linha in matriz_medias:
    print(linha)



#Inicio do programa!
print("Cálculo do determinante de matriz 3x3 usando a Regra de Sarrus\n")

# matrizFixa = [[2, 3, 4],[45, 4, 1],[1, 3, 1]]
# Entrada da matriz 3x3
print("Digite os elementos da matriz linha por linha:")
matriz = []
for i in range(3):
    linha = []
    for j in range(3):
        elemento = float(input("Elemento : [", i+1, "][", j+1, "]"))
        linha.append(elemento)
    matriz.append(linha)

# Mostra a matriz digitada
print("\nMatriz inserida:")
for linha in matriz:
    print(linha)

# Aplica a Regra de Sarrus
# Repete as duas primeiras colunas ao lado direito da matriz
#print("matriz[0]", matriz[0])
#a = matriz[0][0]
a, b, c = matriz[0]
d, e, f = matriz[1]
g, h, i = matriz[2]
#matriz = [[a, b, c] [a][b]
#          [d, e, f],[d][e]
#          [g, h, i]][g][h]

#matriz[0].append = matriz[0][0]
```

```
#matriz[0].append = matriz[0][1]
#         [[a, b, c, a, b]      (b * f * g)
#         [d, e, f, d, e]
#         [g, h, i]][g][h]



# Calcula as diagonais principais e secundárias
diagonal_principal = (a * e * i) + (b * f * g) + (c * d * h)
diagonal_secundaria = (c * e * g) + (a * f * h) + (b * d * i)

# Determinante é a diferença entre as diagonais
det = diagonal_principal - diagonal_secundaria

print(f"\nO determinante da matriz é: {det}")




def ler_matriz():
    """Lê uma matriz 3x3 do usuário"""
    print("Digite os elementos da matriz 3x3 linha por linha:")
    matriz = []
    for i in range(3):
        linha = []
        for j in range(3):
            elemento = float(input(f"Elemento [{i+1}][{j+1}]: "))
            linha.append(elemento)
        matriz.append(linha)
    return matriz

def mostrar_matriz(x):
    """Mostra a matriz formatada"""
    print("\nMatriz:")
    for y in x:
        print(y)

def calcular_determinante(matriz):
    """Calcula o determinante usando Sarrus"""
    # metodo de lagrange:  A_11 = (-1)^1+1 * D_11
    a, b, c = matriz[0]
    d, e, f = matriz[1]
    g, h, i = matriz[2]

    # Calcula as diagonais
    diagonal_principal = (a * e * i) + (b * f * g) + (c * d * h)
    diagonal_secundaria = (c * e * g) + (a * f * h) + (b * d * i)
    d = diagonal_principal - diagonal_secundaria
```

```
    return d

# Programa principal
#print("Digite o tamanho da Matriz")
#tamanho = int(input())
mat = ler_matriz()
mostrar_matriz(mat)
det = calcular_determinante(mat)
print("A determinante da matriz eh: ", det)
```