



# Engenharia de Software

## Computacional thinking with Python

Prof. Dr. Francisco Elânio

# Objetivo

- **Trabalhar com listas em Python**
- **Função zip, enumerate, zip longest**
- **Apresentar Exemplos/Exercícios**

# Listas em Python

## Combinando listas

```
lista1 = [1, 2, 3]
```

```
lista2 = [4, 5, 6]
```

```
lista_combinada = lista1 + lista2
```

```
print(lista_combinada)
```

## Somando listas

```
lista1 = [1, 2, 3]
```

```
lista2 = [4, 5, 6]
```

```
soma_elementos = [a + b for a, b in zip(lista1, lista2)]
```

```
print(soma_elementos)
```

# Função zip com listas

É usada para combinar dois ou mais listas, em que elementos das posições correspondentes são emparelhados. O iterável resultante contém tuplas, onde o primeiro elemento de cada lista é emparelhado, o segundo elemento de cada lista é emparelhado e assim por diante.

```
carro = ["Honda", "Mitsubishi", "Volvo", "BMW"]
```

```
potencia = [120, 134, 180, 300]
```

```
combine = zip(carro, potencia)
```

```
print(set(combine))
```



# Função zip e enumerate()

A função enumerar ( ) é usada para iterar sobre uma lista e retornar o índice e o elemento correspondente nesse índice.

```
nomes = ['Messi', 'Zidane', 'Ronaldo Fenômeno']
```

```
idades = [36, 51, 47]
```

```
for i, (nome, idade) in enumerate(zip(nomes, idades)):
```

```
    print(i, nome, idade)
```

# Função zip com múltiplas iterações

Pode fazer diversas iterações como entrada e retornar uma iteração de tuplas, onde cada tupla contém elementos das posições correspondentes das iteráveis de entrada.

```
lista1 = [1, 2, 3]
```

```
lista2 = ['a', 'b', 'c']
```

```
lista3 = ['Peixe', 'Pássaro', 'Répteis']
```

```
zipped = zip(lista1, lista2, lista3)
```

```
result = list(zipped)
```

```
print(result)
```

## *Tupla em Python*

Sequência de valores de qualquer tipo.

# Filtrando elementos de uma lista baseada em outra

## Filtrando elementos de uma lista baseada em outra

```
lista1 = [1, 2, 3, 4, 5]
```

```
lista2 = ['a', 'b', 'c', 'd', 'e']
```

```
pares = [b for a, b in zip(lista1, lista2) if a % 2 != 0]
```

```
print(pares)
```

### *Função zip*

Itera simultaneamente sobre os elementos correspondentes de lista1 e lista2.

### *Variáveis a e b*

**a** receberá os elementos de lista1 e **b** receberá os elementos de lista2.

### *Lista de Compreensão*

Cria uma nova lista (pares) com os elementos de b (lista2) para os quais a condição  $a \% 2 \neq 0$  é verdadeira.

# Filtrando elementos de uma lista baseada em outra

```
lista1 = [1, 2, 3, 4, 5]
lista2 = ['a', 'b', 'c', 'd', 'e']
combinação = list(zip(lista1, lista2))
for valor, letra in combinação:
    if valor > 3:
        print(letra)
```

## *Valor e Letra*

Valor, Letra desempacota a tupla em duas variáveis.

Valor irá conter o elemento de lista1, Letra que irá conter o elemento de lista2.



# Copiando elementos de uma lista para outras duas listas

```
V = [9, 8, 7, 12, 0, 13, 21]
```

```
P = []
```

```
I = []
```

```
for e in V:
```

```
    if e % 2 == 0:
```

```
        P.append(e)
```

```
    else:
```

```
        I.append(e)
```

```
print( "Pares: ", P)
```

```
print("Impares: ", I)
```

Lista vazias

Percorre toda lista

Verifica se valores da lista são divisíveis por 2 e inseri em uma nova lista.

Caso não seja divisível por 2, será criado uma nova lista.

# Exercício 01

Para uma lista de pacientes, crie um algoritmo em Python que mostre os pacientes que possuem diabetes. Utilize a função zip para desenvolver seu código.

Pacientes	Idades	Condição
João	45	Diabetes
Maria	60	Hipertensão
Pedro	30	Diabetes
Ana	50	asma

# Exercício 01

```
pacientes = ['João', 'Maria', 'Rafael', 'Ana']  
idades = [45, 60, 30, 50]  
condicao = ['diabetes', 'hipertensão', 'diabetes', 'asma']
```

```
for paciente, idade, cond in zip(pacientes, idades, condicao):  
    if cond.lower() == 'diabetes':  
        print(f'Paciente: {paciente} - Idade: {idade} anos')
```

# Exercício 02

Utilizando a função zip, append, for, if, elif, else, crie um algoritmo para separar os pacientes por faixa de idade: < 18 jovem, >18 e <= 65 adulto, >65 idoso. Após a classificação dos pacientes jovens, adultos e idosos, utilize a função append para passar os pacientes para uma lista chamada fase vida e imprima o resultado com a classificação de pacientes.

Pacientes	Idades	Condição
João	45	Diabetes
Maria	60	Hipertensão
Pedro	30	Diabetes
Ana	50	asma



# Operações com Listas de tamanhos diferentes

```
lista1 = [1, 2, 3, 4, 5]
```

```
lista2 = [10, 20, 30]
```

```
from itertools import zip_longest
```

```
combinação = list(zip_longest(lista1, lista2, fillvalue=0))
```

```
print(combinação)
```

## *zip\_longest*

Preenche com determinados valores a lista 2, que possui comprimento diferente de lista 1.

# Exercício 03

**Calcular o preço total de compras em um supermercado com base nas duas listas:**

`precos = [2.50, 1.20, 3.00, 2.00, 4.50]`

`quantidades = [10, 5, 2, 3, 6]`

# Exercício 03

```
precos = [2.50, 1.20, 3.00, 2.00, 4.50]
```

```
quantidades = [10, 5, 2, 3, 6]
```

```
valor_total = sum([preco * quantidade for preco, quantidade in zip(precos,  
quantidades)])
```

```
print(f"O valor total das compras é R${valor_total:.2f}")
```

# Exercício 04

**Calcular o preço total de compras em um supermercado com base nas duas listas e apresentar o total para os Produtos A, B, C, D e E.**

```
precos = [2.50, 1.20, 3.00, 2.00, 4.50]
```

```
quantidades = [10, 5, 2, 3, 6]
```

```
produtos = ['Produto A', 'Produto B', 'Produto C', 'Produto D',  
'Produto E']
```



# Exercício 04

```
precos = [2.50, 1.20, 3.00, 2.00, 4.50]
```

```
quantidades = [10, 5, 2, 3, 6]
```

```
produtos = ['Produto A', 'Produto B', 'Produto C', 'Produto D', 'Produto E']
```

```
valor_total = 0
```

```
for preco, quantidade, produto in zip(precos, quantidades, produtos):
```

```
    total_produto = preco * quantidade
```

```
    print(f"O total para {produto} é R${total_produto:.2f}")
```

```
    valor_total += total_produto
```

```
print(f"\nO valor total das compras é R${valor_total:.2f}")
```

*O importante é não parar  
de questionar (Einstein)*

# Referências

**ASCENCIO, A. F. G, CAMPOS, E. A. V. Fundamentos da Programação de Computadores: algoritmos, Pascal, C/C++ e Java, 2ª Edição, São Paulo: Pearson 2007.**

**SALVETTI, Dirceu Douglas; BARBOSA, Lisbete Madson. Algoritmos. São Paulo: Pearson, 2004.**