## Estrutura de Repetição: FOR e WHILE

No Python e em outras linguagens de programação, a repetição de código facilita o processo de executar tarefas que precisam se repetir várias vezes.

Para isso, usamos o for e o while.

A única diferença entre eles é o nome, pois a sintaxe é similar.

Ambos são usados para realizar repetições, mas o for é geralmente usado quando sabemos quantas vezes queremos repetir, e o while é usado quando a repetição depende de uma condição que será verificada durante a execução.

## Exemplo - Estrutura de Repetição: FOR (Conceito)

```
# Estrutura de um laço de repetição FOR:

for i in range(n):
    # Códigos que serão repetidos

i = # É o nome da variável que vai controlar a repetição (pode ser qualquer nome escolhido).
range(n) = # Define quantas vezes o código será repetido. O valor de n indica o número total de repetições (começando de 0 até n-1).
```

## Estrutura de Repetição: FOR

```
for nome in range (10):
    print('Leticia')
```

## Conceito - Estrutura de Repetição: WHILE (Conceito)

```python
# Estrutura de um laço de repetição While:

i = 0  # Inicializa a variável
while i < n:  # A condição que deve ser verdadeira para continuar a repetição
    # Códigos que serão repetidos
    i += 1  # Incrementa a variável para evitar um loop infinito
```

## Estrutura de Repetição: WHILE

A variável i controla a repetição e começa com um valor inicial (geralmente 0).

O laço while continuará repetindo enquanto a condição i < n for verdadeira. i += 1 incrementa a variável para garantir que a condição eventualmente se torne falsa e o laço termine.

Em resumo:

i += 1 aumenta i em 1 a cada vez que o laço é executado, garantindo que a condição do laço eventualmente se torne falsa e o laço termine.

## Estrutura de Repetição: WHILE

```
inicio = 0
final = 5
while inicio < final:
    print(inicio)
    inicio += 1
```

## Estrutura de Repetição: For

```python
# Lista range de rede

# Rede base
rede_base = '198.168.0.1'

# Imprimir a rede base
print('A rede é:', rede_base)
print('\nA lista de IPs é:')

# Gerar a lista de IPs
for sequencia in range(1, 11):  # Gera IPs de 198.168.0.1 até 198.168.0.10
    ip = f'198.168.0.{sequencia}'  # Cria o IP com a sequência
    print(ip)
```

## Estrutura de Repetição: For

```python
# Mostrar índices de uma lista durante 5 vezes em um print

# Listas de marcas e quantidades
marcas = ['Ford', 'Toyota', 'Ferrari', 'McLaren']
qtd_disponivel = [500, 2000, 6, 2]

# Verificando o comprimento das listas (quantidade de elementos)
qtd = len(marcas)

# Loop para mostrar a marca e a quantidade
for indice in range(qtd):
    print(f'A marca: {marcas[indice]}, teve {qtd_disponivel[indice]} fabricados.')
```

## Estrutura de Repetição: For

```python
# Verificar valores

vendas_mensais = [1000, 500, 1500, 3000, 600, 900, 8000, 6000, 100, 102]
meta = 1000

contador = 0
for maior_venda in vendas_mensais:
    if maior_venda >= meta:
        contador += 1
print('\n' + 'Quantidade de vendedores que atigiram a meta: ''\n')
print('Apenas', contador, 'conseguiram bater a meta')
```

## Estrutura de Repetição: For

```python
# Utilizando FOR para verificações em uma lista

qtd_estoque = [40, 50, 600, 700, 500]
produtos = ['Golf', 'Parati', 'Fusca', 'Vespa', 'Gol']
estoque_minimo = 500

# Iterando sobre a lista de quantidades de estoque
for i, qtd_atual in enumerate(qtd_estoque):
    if qtd_atual < estoque_minimo:
        # Usando f-string para formatar a mensagem de forma mais legível
        print(f'O produto {produtos[i]} está abaixo do mínimo, com {qtd_atual} unidades.')
```

## Estrutura de Repetição: For

```python
# Exemplo de lista para cadastro (cada quarto tem nome e CPF do hóspede)
quartos = [
    ['Gabriel', 'CPF: 456.563.002-00'],
    ['Leticia', 'CPF: 456.563.002-01'],
    ['Sidnei', 'CPF: 456.563.002-02']
]

# Perguntar quantas pessoas vão se cadastrar
qtd_pessoas = int(input('Quantas pessoas você quer cadastrar? '))

# Laço para fazer o cadastro das pessoas
for i in range(qtd_pessoas):
    # Perguntar o nome do hóspede
    nome = input('Nome do hóspede: ')

    # Perguntar o CPF do hóspede
    cpf = input('CPF do hóspede: ')

    # Adicionar o nome e CPF na lista de quartos
    quartos.append([nome, f'CPF: {cpf}'])

# Mostrar os cadastros realizados
print('\nCadastro dos hóspedes:')
for hospede in quartos:
    print(f'{hospede[0]} - {hospede[1]}')
```

## Estrutura de Repetição: For

```python
# Listas de estoques e empresas
estoque = [3000, 1000, 5000, 10000, 300]
empresas = ['Microsoft', 'Apple', 'Samsung', 'LG', 'Nokia']

# Estoque mínimo
estoque_minimo = 8000

# Percorre os estoques e verifica se cada um está abaixo ou acima do estoque mínimo
for i, qtd_estoque in enumerate(estoque):
    if qtd_estoque < estoque_minimo:
        # Se o estoque estiver abaixo do mínimo, imprime a empresa correspondente
        print(f'Empresa abaixo do mínimo: {empresas[i]}')
    else:
        # Caso contrário, imprime a empresa com estoque suficiente
        print(f'Empresa com estoque suficiente: {empresas[i]}')
```

## Estrutura de Repetição: For

Podemos parar o for, no momento que ele encontrar uma informação, como no Slide a seguir:

## Estrutura de Repetição: For

```python
# Listas de estoques e empresas
estoque = [8000, 8000, 5000, 10000, 300]
empresas = ['Microsoft', 'Apple', 'Samsung', 'LG', 'Nokia']

# Estoque mínimo
estoque_minimo = 8000

# Percorre os estoques e verifica se cada um está abaixo ou acima do estoque mínimo
for i, qtd_estoque in enumerate(estoque):
    if qtd_estoque < estoque_minimo:
        # Se o estoque estiver abaixo do mínimo, imprime a empresa e para o loop
        print(f'Empresa abaixo do mínimo: {empresas[i]}')
        break  # Interrompe o loop assim que encontrar o primeiro estoque abaixo do mínimo
else:
    # Se não encontrar nenhuma empresa abaixo do mínimo, exibe isso
    print('Todas as empresas têm estoque suficiente.')
```

## Estrutura de Repetição: While

Vamos analisar exemplos com o comando *While a partir do próximo Slide:*

## Estrutura de Repetição: While

```python
soma = 0

while True:  # O loop vai continuar até que o usuário decida parar
    numero = input("Digite um número (ou 'sair' para encerrar): ")

    if numero.lower() == 'sair':
        break  # Encerra o loop se o usuário digitar 'sair'

    try:
        numero = float(numero)  # Tenta converter a entrada para número
        soma += numero  # Adiciona o número à soma
    except ValueError:
        print("Por favor, digite um número válido.")

print(f"A soma dos números digitados é: {soma}")
```

## Estrutura de Repetição: While

```python
while True:
    numero = int(input("Digite um número (0 para sair): "))

    if numero == 0:
        print("Programa encerrado.")
        break  # Encerra o loop se o número for 0

    if numero % 2 == 0:
        print(f"{numero} é um número par.")
    else:
        print(f"{numero} é um número ímpar.")
```

Bônus: Arvore de natal!

```python
# Altura da árvore
altura = 6

# Parte da copa da árvore
for i in range(altura):
    # Imprime espaços para centralizar a árvore
    print(" " * (altura - i - 1) + "*" * (2 * i + 1))

# Parte do tronco da árvore
for i in range(2):
    print(" " * (altura - 1) + "|")
```

Desafio prático:

Qualidade do Produto:

Temos uma lista de produtos e suas avaliações. O objetivo é analisar a qualidade de cada produto.

Se a avaliação for **acima ou igual a 4.5**, o produto será considerado **“Excelente"**. Se for **acima ou igual a 3.5**, será **“Bom "**. Se for **abaixo de 3.5**, o produto será **"precisa de melhorias"**.

Faça um programa que mostre a avaliação de cada produto.

Desafio prático:

Faixa de preço:

Temos uma lista de produtos e seus preços. Classifique os produtos de acordo com o valor de seu preço:

Econômico: se o preço for menor que R$ 100.
Médio: se o preço for menor que R$ 500 mas maior ou igual a R$ 100.
Premium: se o preço for maior ou igual a R$ 500.

Faça um programa que exiba a categoria de cada produto com base no preço.

Desafio prático:

Crie uma lista de vendedores e as vendas realizadas por cada um.

A meta de vendas de cada vendedor é de R$ 20.000.

Se um vendedor vender duas vezes ou mais a meta (acima de R$ 40.000), ele terá excelente desempenho.

Se ele atingir a meta de R$ 20.000 ou mais, será considerado bom desempenho. Caso contrário, o desempenho será abaixo da meta.

Faça um programa que informe o desempenho de cada vendedor com base no valor das suas vendas.

Desafio prático:

Crie uma lista de produtos e seus estoques.

O estoque mínimo necessário para cada produto é de 100 unidades.

Se o estoque de um produto for menor que 100, ele está abaixo do mínimo.

Se o estoque for maior ou igual a 100, ele está acima do mínimo.

Faça um programa que conte quantos produtos estão acima e quantos estão abaixo do estoque mínimo.

Desafio - While:

Sistema de Pontuação de Jogos:

Vamos criar um sistema simples de pontuação para um jogo. Os jogadores recebem pontos de acordo com a seguinte regra:

Se o jogador completar uma fase extra, ele ganha 100 pontos adicionais.
Se o jogador completar uma fase difícil, ele ganha 50 pontos adicionais.
Se o jogador completar uma fase normal, ele ganha 20 pontos adicionais.

No entanto, se um jogador completar três ou mais fases extras, ele ganha um bônus de 200 pontos no final. Caso o jogador complete somente fases difíceis ou somente fases normais, o total de pontos será multiplicado por 1,5.

Faça um programa que calcule a pontuação final de um jogador, com base nas fases que ele completou. O programa deve exibir a pontuação final e o tipo de bonificação (caso o jogador tenha recebido alguma).

# Dúvidas