Documento de Acompanhamento: Estruturas de Repetição em Python

1. Introdução às Estruturas de Repetição

Explicação Textual:
As estruturas de repetição (também chamadas de loops ou laços) permitem executar um mesmo bloco de código várias vezes. Imagine que você precisa imprimir números de 1 a 100 - sem repetições, você teria que escrever 100 linhas de código! Com loops, você escreve apenas algumas linhas e o Python repete o trabalho para você.

Lista de Exercícios Recomendada:
https://github.com/go4Mor4/Exericios-resolvidos_Estrutura-de-repeticao/tree/master

Por que usar repetições?

- Evitar código repetitivo: Escrever uma vez, executar muitas vezes
- Processar listas de dados: Percorrer todos os itens de uma coleção
- Executar até uma condição ser atendida: Tentativas, validações, etc.
- Automatizar tarefas repetitivas: Cálculos em série, processamento em lote

2. O Loop `for`: Para Quando Você Sabe Quantas Vezes Repetir

Explicação Textual:
O `for` é ideal quando sabemos quantas vezes queremos repetir algo ou quando queremos percorrer uma sequência (como uma lista, string ou uma sequência numérica).

Sintaxe básica:

```
python
for variável in sequência:
    # bloco de código que será repetido
    # para cada item da sequência
```

Exemplo 1: Contando de 1 a 5

```
python
for contador in range(1, 6):  # range(início, fim) - o fim NÃO é incluído
    print(contador)
Resultado:
text
1
2
3
4
5
```

Exemplo 2: Percorrendo uma lista

```
python
frutas = ["maçã", "banana", "laranja", "uva"]

for fruta in frutas:
    print(f"Eu gosto de {fruta}")
Resultado:
text
Eu gosto de maçã
Eu gosto de banana
Eu gosto de laranja
Eu gosto de uva
```

Exemplo 3: Percorrendo uma string

```
python
palavra = "Python"
```

```python
for letra in palavra:
    print(letra)
```

Resultado:

```
text
P
y
t
h
o
n
```

3. A Função `range()`: Nosso Aliado nos Loops

Explicação Textual:
A função `range()` cria uma sequência de números que é muito útil com o loop `for`.
Ela tem três formas principais:

Formas de usar `range()`:

```python
python
# 1. range(fim) - começa em 0, vai até fim-1
for i in range(5):      # 0, 1, 2, 3, 4
    print(i)

# 2. range(início, fim) - vai de início até fim-1
for i in range(2, 6):   # 2, 3, 4, 5
    print(i)

# 3. range(início, fim, passo) - pula de passo em passo
for i in range(0, 10, 2):  # 0, 2, 4, 6, 8
    print(i)

# Contagem regressiva
for i in range(5, 0, -1):  # 5, 4, 3, 2, 1
    print(i)
```

Exemplo prático: Tabuada

```
python
numero = int(input("Digite um número para ver sua tabuada: "))

print(f"Tabuada do {numero}:")
for i in range(1, 11):  # De 1 a 10
    resultado = numero * i
    print(f"{numero} x {i} = {resultado}")
```

4. O Loop `while`: Para Quando Você Não Sabe Quantas Vezes Repetir

Explicação Textual:
O `while` é usado quando não sabemos quantas vezes precisamos repetir, mas sabemos a condição que deve ser verdadeira para continuar repetindo. Ele executa ENQUANTO uma condição for verdadeira.

Sintaxe básica:

```
python
while condição:
    # bloco de código que será repetido
    # ENQUANTO a condição for True
```

Exemplo 1: Contador simples

```
python
contador = 1

while contador <= 5:
    print(contador)
    contador += 1  # contador = contador + 1
```

Resultado:

```
text
1
2
```

```
3
4
5
```

Exemplo 2: Validação de entrada

```
python
senha_correta = "python123"
senha = ""

while senha != senha_correta:
    senha = input("Digite a senha: ")
    if senha != senha_correta:
        print("Senha incorreta! Tente novamente.")

print("Acesso permitido!")
```

Exemplo 3: Menu interativo

```
python
opcao = 0

while opcao != 4:
    print("\n=== MENU ===")
    print("1. Ver saldo")
    print("2. Sacar")
    print("3. Depositar")
    print("4. Sair")

    opcao = int(input("Escolha uma opção: "))

    if opcao == 1:
        print("Saldo: R$ 1000,00")
    elif opcao == 2:
        print("Saque realizado!")
    elif opcao == 3:
        print("Depósito realizado!")
    elif opcao == 4:
        print("Saindo do sistema...")
    else:
        print("Opção inválida!")
```

5. Diferenças entre `for` e `while`

Quando usar `for`:

- Quando sabemos quantas vezes vamos repetir
- Para percorrer sequências (listas, strings, etc.)
- Quando queremos iterar sobre elementos específicos

Quando usar `while`:

- Quando não sabemos quantas vezes vamos repetir
- Quando dependemos de uma condição que muda durante a execução
- Para criar menus interativos
- Para validações de entrada

Comparação direta:

```
python
# Mesma tarefa com for e while

# Com FOR (sabemos que são 5 repetições)
print("Contagem com FOR:")
for i in range(1, 6):
    print(i)

# Com WHILE (condição: enquanto i <= 5)
print("\nContagem com WHILE:")
i = 1
while i <= 5:
    print(i)
    i += 1
```

6. CUIDADO: Loop Infinito!

Explicação Textual:
Um loop infinito acontece quando a condição do `while` nunca se torna falsa. O programa fica preso no loop para sempre!

```python
# EXEMPLO DE LOOP INFINITO - NÃO EXECUTE!
contador = 1
while contador <= 5:
    print(contador)
    # Esqueci de incrementar o contador!
    # contador += 1  # SEM ESTA LINHA, contador SEMPRE será 1
```

Como evitar loops infinitos:

1. Sempre verifique se a condição pode se tornar falsa
2. Atualize as variáveis que fazem parte da condição
3. Use `break` para sair quando necessário (veremos a seguir)

7. Comandos Especiais: `break` e `continue`

`break`: Interrompe o Loop Imediatamente

Explicação Textual:
O `break` faz o loop parar completamente, mesmo que a condição ainda seja verdadeira. É como um "botão de emergência" para sair do loop.

```python
# Exemplo com for
print("Números até encontrar o 5:")
for numero in range(1, 11):
    if numero == 5:
        break  # Para o loop quando encontrar 5
    print(numero)
# Resultado: 1 2 3 4

# Exemplo com while
print("\nContagem regressiva com break:")
contador = 10
```

```
while contador > 0:
    print(contador)
    if contador == 5:
        break  # Para quando chegar em 5
    contador -= 1
# Resultado: 10 9 8 7 6 5
```

`continue`: Pula para a Próxima Iteração

Explicação Textual:
O `continue` não para o loop, mas pula o restante do código da iteração atual e vai direto para a próxima.

```
python
# Imprimir apenas números pares
print("Números pares de 1 a 10:")
for numero in range(1, 11):
    if numero % 2 != 0:  # Se for ímpar
        continue  # Pula para o próximo número
    print(numero)
# Resultado: 2 4 6 8 10

# Exemplo: processar apenas nomes não vazios
nomes = ["Ana", "", "Carlos", "   ", "Maria", "João"]
print("\nNomes não vazios:")
for nome in nomes:
    if nome.strip() == "":  # Se nome estiver vazio ou só espaços
        continue  # Pula este nome
    print(f"Olá, {nome}!")
# Resultado: Olá, Ana! Olá, Carlos! Olá, Maria! Olá, João!
```

8. Loops Aninhados: Um Loop Dentro do Outro

Explicação Textual:
Podemos colocar um loop dentro de outro. Isso é útil para trabalhar com tabelas, matrizes ou combinações.

```
python
# Tabuada completa
print("Tabuadas de 1 a 5:")
```

```
for i in range(1, 6):  # Loop externo: números de 1 a 5
    print(f"\nTabuada do {i}:")
    for j in range(1, 11):  # Loop interno: multiplicadores de 1 a 10
        print(f"{i} x {j} = {i*j}")
python
# Desenhar um retângulo com asteriscos
linhas = 4
colunas = 6

print("Retângulo:")
for i in range(linhas):      # Para cada linha
    for j in range(colunas): # Para cada coluna
        print("*", end=" ")  # end=" " para não pular linha
    print()  # Pula linha após cada linha do retângulo
# Resultado:
# * * * * * *
# * * * * * *
# * * * * * *
# * * * * * *
```

9. Exemplos Práticos Detalhados

Exemplo 1: Sistema de Tentativas de Login

```
python
senha_correta = "python123"
tentativas = 3

while tentativas > 0:
    senha = input(f"Digite a senha ({tentativas} tentativa(s) restante(s)): ")

    if senha == senha_correta:
        print("Login bem-sucedido!")
        break  # Sai do loop se acertar
    else:
        print("Senha incorreta!")
        tentativas -= 1

    if tentativas == 0:
        print("Conta bloqueada! Entre em contato com o suporte.")
```

Exemplo 2: Calculadora de Média com Quantidade Desconhecida de Notas

```
python
print("Calculadora de Média")
print("Digite -1 para terminar")

soma = 0
quantidade = 0

while True:  # Loop infinito controlado
    nota = float(input("Digite uma nota (-1 para sair): "))

    if nota == -1:
        break  # Sai do loop quando usuário digitar -1

    if nota < 0 or nota > 10:
        print("Nota inválida! Digite uma nota entre 0 e 10.")
        continue  # Pula para a próxima iteração

    soma += nota
    quantidade += 1

if quantidade > 0:
    media = soma / quantidade
    print(f"Média das {quantidade} notas: {media:.2f}")
else:
    print("Nenhuma nota foi informada.")
```

Exemplo 3: Jogo de Adivinhação Aprimorado

```
python
import random

numero_secreto = random.randint(1, 100)
tentativas = 0
max_tentativas = 7

print("=== Jogo de Adivinhação ===")
print("Tente adivinhar o número entre 1 e 100")
print(f"Você tem {max_tentativas} tentativas")
```

```python
while tentativas < max_tentativas:
    tentativas += 1
    palpite = int(input(f"\nTentativa {tentativas}/{max_tentativas}: "))

    if palpite < 1 or palpite > 100:
        print("Digite um número entre 1 e 100!")
        continue

    if palpite == numero_secreto:
        print(f"Parabéns! Você acertou em {tentativas} tentativa(s)!")
        break
    elif palpite < numero_secreto:
        print("Tente um número MAIOR")
    else:
        print("Tente um número MENOR")

    # Dica extra nas últimas tentativas
    if tentativas == max_tentativas - 1:
        if numero_secreto % 2 == 0:
            print("DICA: O número é PAR")
        else:
            print("DICA: O número é ÍMPAR")

if tentativas >= max_tentativas and palpite != numero_secreto:
    print(f"\nGame Over! O número era {numero_secreto}")
```

10. Exercícios Práticos

Exercício 1: Contagem Regressiva

```
python
"""
Faça um programa que faça uma contagem regressiva de 10 a 0,
e no final imprima "FOGO!"
"""
```

Exercício 2: Soma de Números Pares

```
python
"""
```

```
Calcule a soma de todos os números pares de 1 a 100.
Dica: Use range() com passo 2
"""
```

Exercício 3: Tabuada Interativa

```
python
"""
Peça um número ao usuário e mostre sua tabuada de 1 a 10.
Após mostrar, pergunte se quer ver outra tabuada.
Continue até que o usuário digite 'não'.
"""
```

Exercício 4: Validador de Senha

```
python
"""
Crie um sistema de cadastro que:
1. Peça uma senha (mínimo 6 caracteres)
2. Peça para confirmar a senha
3. Se as senhas não forem iguais, peça novamente
4. Continue pedindo até que as senhas coincidam
"""
```

Exercício 5: Calculadora de Fatorial

```
python
"""
Peça um número inteiro positivo e calcule seu fatorial.
Fatorial de 5 = 5 × 4 × 3 × 2 × 1 = 120
Dica: Use um loop para multiplicar os números
"""
```

Exercício 6: Números Primos

```
python
"""
```

```
Peça um número e verifique se ele é primo.
Um número é primo se é divisível apenas por 1 e por ele mesmo.
Dica: Use um loop para testar divisores de 2 até n-1
"""
```

Exercício 7: Padrão de Asteriscos

```
python
"""
Use loops aninhados para imprimir:
*
**
***
****
*****

Depois, modifique para imprimir:
*****
****
***
**
*
"""
```

Exercício 8: Média de Notas com Flag

```
python
"""
Peça notas ao usuário até que ele digite -1.
Calcule e mostre a média das notas.
Ignore notas inválidas (<0 ou >10).
"""
```

Exercício 9: Fibonacci Simples

```
python
"""
Imprima os primeiros 10 números da sequência de Fibonacci:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

```
Cada número é a soma dos dois anteriores.
"""
```

Exercício 10: Jogo do Par ou Ímpar

```
python
"""
Crie um jogo onde:
1. O computador escolhe um número de 0 a 10
2. O usuário diz se é par ou ímpar
3. O usuário tenta adivinhar o número
4. O usuário tem 3 tentativas
5. A cada erro, dê uma dica (maior/menor)
"""
```

11. Erros Comuns de Iniciantes

```
python
# ERRO: Esquecer de atualizar a variável de controle no while
contador = 0
while contador < 5:
    print(contador)
    # Falta: contador += 1  (loop infinito!)

# ERRO: Usar = em vez de == na condição
if numero = 10:  # ERRADO! = é atribuição
if numero == 10: # CORRETO! == é comparação

# ERRO: Não usar identação correta
for i in range(5):
print(i)  # ERRO! Falta identação

# ERRO: range() com início maior que fim e passo positivo
for i in range(5, 1):  # Não executa nenhuma vez!
    print(i)

# Correto para contagem regressiva:
for i in range(5, 0, -1):  # Passo -1
    print(i)
```

12. Boas Práticas

1. Escolha o loop certo: `for` para sequências conhecidas, `while` para condições
2. Evite loops infinitos: Sempre tenha uma forma de sair do `while`
3. Use nomes descritivos: `for aluno in turma:` em vez de `for x in y:`
4. Comente loops complexos: Explique o que o loop está fazendo
5. Teste casos limites: O que acontece na primeira e última iteração?
6. Use `break` e `continue` com moderação: Muitos podem tornar o código confuso

---

Dica Final: Comece sempre com loops simples e vá aumentando a complexidade gradualmente. A prática é essencial para entender quando e como usar cada tipo de loop!