Matéria Adiantada dia 04/11

Semana 31 - Aula 1

Tópico Principal da Aula: Estruturas de Repetição

Subtítulo/Tema Específico: Loop for

Código da aula: [SIS]ANO1C1B4S31A1

Objetivos da Aula:

- Conhecer os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas de repetição .
- Desenvolver sistemas computacionais utilizando ambiente de desenvolvimento.
- Migrar sistemas implementando rotinas e estruturas de dados mais eficazes .

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Introdução ao Loop for

- **Definição:** O laço de repetição **for** é uma estrutura de controle de fluxo utilizada para **iterar sobre uma sequência**. Uma sequência pode ser uma
- **lista**, **tupla**, **dicionário**, **conjunto** ou até mesmo uma *string*.
- **Aprofundamento/Complemento (se necessário):** O for executa um bloco de código para cada item na sequência. Ele é a escolha mais adequada quando o
- **número de iterações é conhecido** ou pode ser determinado pelo tamanho da coleção a ser percorrida. Em Python, frequentemente usamos a função
- range() dentro do for para iterar um número específico de vezes.
- Exemplo Prático (Soma de Elementos de uma Lista):
  O loop for permite que um programa calcule a soma de todos os elementos de uma lista de forma simples e eficiente, sem a necessidade de somar item por item manualmente.
- Python

```
numeros = [10, 20, 30]
soma = 0
for numero in numeros:
    soma += numero
print(soma) # Saída: 60
```

-
-
- **Link de Vídeo:**

1. HASHTAG PROGRAMAÇÃO. Estrutura de repetição FOR no Python - Criando um loop

---

Semana 31 - Aula 2

Tópico Principal da Aula: Estruturas de Repetição

Subtítulo/Tema Específico: Loop while

Código da aula: [SIS]ANO1C1B4S31A2

Objetivos da Aula:

- Conhecer os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas de repetição.
- Desenvolver sistemas computacionais utilizando ambiente de desenvolvimento .
- Migrar sistemas implementando rotinas e estruturas de dados mais eficazes.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Introdução ao Loop while

- **Definição:** O laço de repetição **while** é utilizado para repetir um bloco de código **enquanto uma condição lógica for avaliada como verdadeira**.
- **Aprofundamento/Complemento (se necessário):** O while é ideal quando o **número de iterações não é conhecido** antecipadamente. A condição é verificada
- **antes de cada execução** do bloco de código. É fundamental que a condição se torne falsa em algum momento para evitar um
- *loop* infinito (que nunca termina).
- Exemplo Prático (Loop Controlado por Usuário):
  É comumente usado em menus interativos ou na leitura de dados cuja finalização depende de uma entrada específica (sentinela), como a sugestão de calcular a média de números até que o zero seja digitado17.
- Python

```
tentativas = 3
while tentativas > 0: # Repete enquanto a condição for verdadeira
    print(f"Tentativas restantes: {tentativas}")
    tentativas -= 1 # Reduz o contador para garantir que o loop termine
# Saída: Tentativas restantes: 3, 2, 1
```

-
-
- **Link de Vídeo:**
    1. LET'S DATA. While (estruturas de repetição) | Python em 30 minutos 18

Semana 31 - Aula 3

Tópico Principal da Aula: Estruturas de Repetição

Subtítulo/Tema Específico: Uso de break e continue

Código da aula: [SIS]ANO1C1B4S31A3

Objetivos da Aula:

- Conhecer os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas de repetição.
- Desenvolver sistemas computacionais utilizando ambiente de desenvolvimento.
- Migrar sistemas implementando rotinas e estruturas de dados mais eficazes.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Controle de Fluxo com break e continue

- **Definição:** Os comandos **break** e **continue** são comandos de controle utilizados para modificar o fluxo de execução dos laços de repetição (for ou while).
- **Aprofundamento/Complemento (se necessário):**
    1. **break:** Encerra o loop **imediatamente** e de forma incondicional. O programa salta para a próxima linha de código que segue a estrutura de repetição.
    2. **continue:** Interrompe apenas a iteração atual do loop e **pula para a próxima iteração**. Qualquer código após o
    3. continue no bloco atual é ignorado.
- Exemplo Prático (Filtragem e Interrupção):
  O uso de
  break é útil em algoritmos de busca (por exemplo, ao encontrar um número primo 27), e o
- continue é ideal para ignorar dados inválidos.
- Python

```
# Uso do continue para pular a impressão do número 5
for x in range(1, 10):
    if x == 5:
        continue # O restante do bloco (print) é ignorado para x=5
    print(x) # Saída: 1, 2, 3, 4, 6, 7, 8, 9
```

-
-
- **Link de Vídeo:**
    1. HASHTAG PROGRAMAÇÃO. Break e Continue no Python Ferramentas da estrutura de repetição
       28

Semana 31 - Aula 4

Tópico Principal da Aula: Estruturas de Repetição e Coleções

Subtítulo/Tema Específico: Compreensão de Listas (list comprehension)

Código da aula: [SIS]ANO1C1B4S31A4

Objetivos da Aula:

- Conhecer os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas de repetição .
- Desenvolver sistemas computacionais utilizando ambiente de desenvolvimento.
- Migrar sistemas implementando rotinas e estruturas de dados mais eficazes.

Recursos Adicionais (Sugestão, pode ser adaptado):

- Caderno para anotações;
- Acesso ao laboratório de informática e/ou internet.

Exposição do Conteúdo:

Referência do Slide: Introdução à Compreensão de Listas

- **Definição:** A **Compreensão de Listas** (List Comprehension) é uma funcionalidade poderosa e **concisa** do Python que permite a criação de listas de maneira eficiente[33].
- **Aprofundamento/Complemento (se necessário):** Ela viabiliza a criação de listas em uma **única linha de código**, combinando uma expressão (o que se quer inserir na lista) e uma sequência iterável (usualmente um loop `for`)[34]. A sintaxe compacta é considerada mais legível e idiomática (
- *Pythonic*) do que o uso do laço `for` tradicional seguido de `append()`[35]. A sintaxe básica é

- [expressão for item in iterável.
- Exemplo Prático (Quadrados com Filtro):
  É possível criar listas de forma condicional, usando cláusulas if.
- Python

```
# Criar uma lista com o quadrado apenas dos números pares de 0 a 9
quadrados_pares = [x**2 for x in range(10) if x % 2 == 0]
print(quadrados_pares) # Saída: [0, 4, 16, 36, 64]
```

-
-
- **Link de Vídeo:**
  1. HASHTAG PROGRAMAÇÃO. O que é, como usar e para que serve o List Comprehension no Python?