Matéria Adiantada dia 22/10

Semana 29 - Aula 1

Semana 29 - Aula 1

Tópico Principal da Aula: Criação de programas: prática

Subtítulo/Tema Específico: Estruturas Condicionais (if, elif, else)

Código da aula: [SIS]ANO1C1B4S29A1

Objetivos da Aula:

- Compreender os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas simples de operação.
- 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: Slide 06 - Estruturas Condicionais Simples (`if`)

- **Definição:** As estruturas condicionais permitem que um programa realize a **tomada de decisão**, executando um bloco de código específico somente se uma condição for avaliada como **verdadeira** (`True`). A instrução fundamental para isso é o comando **`if`**.
- **Aprofundamento/Complemento (se necessário):** Em Python, a condição é uma expressão booleana que utiliza operadores relacionais (`==`, `>`). O bloco de código a ser executado sob a condição é definido pela **indentação** (geralmente 4 espaços) após o comando `if` seguido de dois pontos (`:`). A correta indentação é crucial para o fluxo de execução.
- **Exemplo Prático:** Verificar se um usuário atingiu a idade mínima para acesso.
- Python

```
idade = 18
if idade >= 18:
    print("Acesso concedido. Bem-vindo.")
```

-
-
- **Vídeos Sugeridos:**
  - HASHTAG PROGRAMAÇÃO. 3 estruturas básicas de Python em menos de 10 minutos.
  - NA PRÁTICA: como IF, ELIF e ELSE funcionam em PYTHON? - YouTube

Referência do Slide: Slide 07 - Estruturas Condicionais Compostas (if-else)

- **Definição:** A estrutura **if-else** é usada para definir uma ação alternativa caso a condição do if seja **falsa** (False). Isso garante que o programa sempre siga um de dois caminhos mutuamente exclusivos, cobrindo o caso positivo e o caso negativo.
- **Aprofundamento/Complemento (se necessário):** O bloco **else** não recebe uma condição. Ele atua como o **caminho padrão** ou de exceção, sendo executado apenas quando todas as condições anteriores (o if) falham. É fundamental para criar um fluxo de decisão binária completo.
- **Exemplo Prático:** Classificar a nota de um aluno como aprovado ou reprovado.
- Python

```
nota = 6.5
if nota >= 7.0:
    print("Aprovado!")
else:
    print("Reprovado.") # Este bloco seria executado.
```

-
-
- **Vídeos Sugeridos:**
  - HASHTAG PROGRAMAÇÃO. 3 estruturas básicas de Python em menos de 10 minutos.
  - NA PRÁTICA: como IF, ELIF e ELSE funcionam em PYTHON? - YouTube

Referência do Slide: Slide 08 - Estruturas Condicionais Encadeadas (if-elif-else)

- **Definição:** Utiliza-se a estrutura **if-elif-else** quando é necessário testar **múltiplas condições distintas e sequenciais**. O programa testa as condições em ordem, e apenas o bloco de código da **primeira condição verdadeira** encontrada é executado, ignorando as demais.
- **Aprofundamento/Complemento (se necessário):** O comando **elif** (abreviação de "else if") permite adicionar quantas condições intermediárias forem necessárias. A ordem de avaliação é crítica: se a condição do if inicial for falsa, o programa passa para o primeiro elif, e assim sucessivamente. O bloco final else é opcional, mas serve para capturar qualquer caso que não se enquadre em nenhuma das condições anteriores.
- **Exemplo Prático:** Classificar o conceito de um produto com base em sua pontuação de qualidade.
- Python

```
pontuacao = 88
if pontuacao >= 90:
    print("Conceito: Excelente (A)")
elif pontuacao >= 80:
    print("Conceito: Bom (B)") # Este bloco seria executado.
else:
```

```
print("Conceito: Regular (C)")
```

-
-
- **Vídeos Sugeridos:**
    - HASHTAG PROGRAMAÇÃO. 3 estruturas básicas de Python em menos de 10 minutos.
    - NA PRÁTICA: como IF, ELIF e ELSE funcionam em PYTHON? - YouTube

---

Semana 29 - Aula 2

Semana 29 - Aula 2

Tópico Principal da Aula: Criação de programas: prática

Subtítulo/Tema Específico: Estruturas de Repetição (For e While)

Código da aula: [SIS]ANO1C1B4S29A2

Objetivos da Aula:

- Compreender os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas simples de operação.
- 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: Slide 06 - Estrutura de Repetição `For`

- **Definição:** O laço **`for`** é utilizado para **iteração definida**, ou seja, para percorrer (iterar) os elementos de uma sequência (como listas ou *strings*) ou executar um bloco de código por um **número predefinido e conhecido** de vezes (utilizando a função `range()`).
- **Aprofundamento/Complemento (se necessário):** O `for` é a escolha ideal quando se sabe exatamente quantas vezes a repetição deve ocorrer ou quando se quer processar cada item de uma coleção. A variável de iteração assume o valor de cada elemento da sequência a cada *loop*.
- **Exemplo Prático:** Calcular a tabuada de multiplicação de um número (repetindo 10 vezes).
- Python

```
numero = 5
```

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

-
-
- **Vídeos Sugeridos:**
    - HASHTAG PROGRAMAÇÃO. Estrutura de Repetição FOR no Python - Criando um Loop.
    - Curso de Python | Estrutura de Repetição Loops For e While - YouTube

Referência do Slide: Slide 07 - Estrutura de Repetição While

- **Definição:** O laço **while** é usado para **iteração indefinida**, executando um bloco de código **enquanto uma condição específica for verdadeira**. O número de repetições não é conhecido no início e depende do estado da condição de parada.
- **Aprofundamento/Complemento (se necessário):** O comando while exige que o programador garanta que, em algum momento, a condição de teste se torne falsa. Caso contrário, o programa entrará em um **loop infinito**, consumindo recursos e travando o fluxo de execução. É comumente usado em cenários como menus interativos ou leitura de dados até que um valor sentinela seja inserido.
- **Exemplo Prático:** Simular um *login* simples, solicitando a senha até que ela esteja correta.
- Python

```
senha_correta = "1234"
senha_digitada = ""
while senha_digitada != senha_correta:
    senha_digitada = input("Digite a senha: ")
print("Acesso liberado!")
```

-
-
- **Vídeos Sugeridos:**
    - HASHTAG PROGRAMAÇÃO. Estrutura de Repetição FOR no Python - Criando um Loop.
    - Curso de Python | Estrutura de Repetição Loops For e While - YouTube

---

Semana 29 - Aula 3

Semana 29 - Aula 3

Tópico Principal da Aula: Criação de programas: prática

Subtítulo/Tema Específico: Listas (Vetores)

Código da aula: [SIS]ANO1C1B4S29A3

Objetivos da Aula:

- Compreender os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas simples de operação.
- 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: Slide 06 - Introdução a Listas (Estruturas de Dados)

- **Definição:** Uma **Lista** (equivalente a um **Vetor** em termos de armazenamento sequencial) é uma estrutura de dados fundamental em Python, definida por colchetes []. Ela é uma coleção **ordenada** de itens, que podem ser de tipos variados, e é **mutável**, permitindo a alteração, adição ou remoção de elementos após sua criação.
- **Aprofundamento/Complemento (se necessário):** O principal modo de acesso e manipulação dos elementos da lista é através do **índice**, que sempre se inicia em **zero (0)**. A mutabilidade é um ponto chave: ela permite que a lista se adapte dinamicamente às necessidades do programa, sendo diferente de coleções imutáveis como as tuplas.
- **Exemplo Prático:** Gerenciamento da fila de atendimento de uma clínica.
- Python

```
fila_atendimento = ["João", "Ana", "Pedro"]
print(f"Próximo a ser atendido: {fila_atendimento[0]}") # Acessa "João"

fila_atendimento.append("Maria") # Adiciona Maria no final
print(fila_atendimento) # Saída: ['João', 'Ana', 'Pedro', 'Maria']
```

-
-
- **Vídeos Sugeridos:**
  - DANKI CODE. Aprenda Python na prática (listas, tuplas, dicionários) || Tutorial completo.
  - Aula 23 - Listas (vetores) em Python - YouTube

Semana 29 - Aula 4

Semana 29 - Aula 4

Tópico Principal da Aula: Criação de programas: prática

Subtítulo/Tema Específico: Dicionários (Mapas)

Código da aula: [SIS]ANO1C1B4S29A4

Objetivos da Aula:

- Compreender os conceitos sobre o desenvolvimento e a execução prática de programas computacionais utilizando estruturas simples de operação.
- 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: Slide 06 - Dicionários (Estruturas de Dados)

- **Definição: Dicionários** (ou **Mapas**) são estruturas de dados que armazenam informações na forma de pares **chave:valor**. Diferente das listas, que usam índices numéricos, os dicionários são indexados por **chaves literais**, que devem ser únicas e imutáveis (geralmente *strings*).
- **Aprofundamento/Complemento (se necessário):** Dicionários são definidos por chaves `{}` e são ideais para modelar objetos ou registros, onde o acesso aos dados é feito pelo nome do campo (a chave). Eles são estruturas **mutáveis** e fornecem acesso e recuperação de dados extremamente rápidos, sendo amplamente utilizados em interações com APIs e arquivos JSON.
- **Exemplo Prático:** Representar os dados cadastrais de um funcionário.
- Python

```
funcionario = {
   "id": 101,
   "nome": "Carla Silva",
   "cargo": "Analista Júnior"
}

# Acessando o valor pela chave
print(f"O cargo de Carla é: {funcionario['cargo']}")

# Adicionando um novo par chave:valor
funcionario["salario"] = 4500.00
print(funcionario)
```

-
-
- **Vídeos Sugeridos:**
  - CODE BY DUDA. Entendendo estrutura de dados!

- Dicionários em Python (Curso Python para machine learning - Aula 5) - YouTube