



AULA DE REVISÃO PARA A AVALIAÇÃO FINAL

Algoritmos e Estrutura de Dados I

Prof. Dr. Dilermando Piva Jr

INTRODUÇÃO À COMPUTAÇÃO - PARTE 1: HISTÓRIA E EVOLUÇÃO DA COMPUTAÇÃO

Qual foi o principal objetivo do projeto de Charles Babbage para sua "Máquina Analítica"?

- a) Realizar cálculos aritméticos básicos
- b) Executar instruções programadas
- c) Armazenar e recuperar dados
- d) Resolver equações diferenciais
- e) Processar informações em formato binário

INTRODUÇÃO À COMPUTAÇÃO - PARTE 1: HISTÓRIA E EVOLUÇÃO DA COMPUTAÇÃO

Qual foi o principal objetivo do projeto de Charles Babbage para sua "Máquina Analítica"?

- a) Realizar cálculos aritméticos básicos
- b) Executar instruções programadas**
- c) Armazenar e recuperar dados
- d) Resolver equações diferenciais
- e) Processar informações em formato binário

INTRODUÇÃO À COMPUTAÇÃO - PARTE 2: A INFORMAÇÃO E SUA REPRESENTAÇÃO. CONVERSÃO ENTRE BASES

Qual é o resultado da conversão do número binário 10110101 para decimal?

- a) 181
- b) 149
- c) 213
- d) 341
- e) 85

INTRODUÇÃO À COMPUTAÇÃO - PARTE 2: A INFORMAÇÃO E SUA REPRESENTAÇÃO. CONVERSÃO ENTRE BASES

Qual é o resultado da conversão do número binário 10110101 para decimal?

a) 181

b) 149

c) 213

d) 341

e) 85

INTRODUÇÃO À COMPUTAÇÃO - PARTE 3: ÁLGEBRA BOOLEANA. SISTEMA COMPUTACIONAL: HARDWARE E SOFTWARE

Qual é o resultado da operação lógica $(A \text{ AND } B) \text{ OR } (\text{NOT } C)$ em Álgebra Booleana, sendo $A=1$, $B=0$ e $C=1$?

- a) 0
- b) 1
- c) A
- d) B
- e) C

INTRODUÇÃO À COMPUTAÇÃO - PARTE 3: ÁLGEBRA BOOLEANA. SISTEMA COMPUTACIONAL: HARDWARE E SOFTWARE

Qual é o resultado da operação lógica $(A \text{ AND } B) \text{ OR } (\text{NOT } C)$ em Álgebra Booleana, sendo $A=1$, $B=0$ e $C=1$?

a) 0

b) 1

c) A

d) B

e) C

A LÓGICA E OS ALGORITMOS

Qual é a diferença entre um algoritmo e um programa de computador?

- a) Não há diferença, são sinônimos.
- b) Um algoritmo é uma sequência de passos, enquanto um programa é a implementação desse algoritmo em uma linguagem de programação.
- c) Um algoritmo é executado por um computador, enquanto um programa é executado por um humano.
- d) Um algoritmo é uma estrutura de dados, enquanto um programa é uma sequência de instruções.
- e) Um algoritmo é uma abstração, enquanto um programa é uma realização concreta.

A LÓGICA E OS ALGORITMOS

Qual é a diferença entre um algoritmo e um programa de computador?

a) Não há diferença, são sinônimos.

b) Um algoritmo é uma sequência de passos, enquanto um programa é a implementação desse algoritmo em uma linguagem de programação.

c) Um algoritmo é executado por um computador, enquanto um programa é executado por um humano.

d) Um algoritmo é uma estrutura de dados, enquanto um programa é uma sequência de instruções.

e) Um algoritmo é uma abstração, enquanto um programa é uma realização concreta.

EXPRESSÕES MATEMÁTICAS E TIPOS DE DADOS

Qual tipo de dado em Python oferece maior precisão para cálculos envolvendo números decimais?

- a) int
- b) float
- c) complex
- d) decimal
- e) Todos oferecem a mesma precisão

EXPRESSÕES MATEMÁTICAS E TIPOS DE DADOS

Qual tipo de dado em Python oferece maior precisão para cálculos envolvendo números decimais?

a) int

b) float

c) complex

d) decimal

e) Todos oferecem a mesma precisão

EXPRESSÕES MATEMÁTICAS E TIPOS DE DADOS

Qual tipo de dado em Python oferece maior precisão para cálculos envolvendo números decimais?

a) int

b) float

c) complex

d) decimal

e) Todos oferecem a mesma precisão

ENTRETANTO....

```
from decimal import Decimal

# Criação de números Decimals
num1 = Decimal('0.1')
num2 = Decimal('0.2')

# Operações aritméticas com Decimals
result = num1 + num2

print(result) # Saída: 0.3
```

AMBIENTE DE PROGRAMAÇÃO E ESTRUTURAS SEQUENCIAIS

Qual é a principal diferença entre um compilador e um interpretador de linguagens de programação?

- a) Compiladores geram código de máquina, enquanto interpretadores executam o código diretamente.
- b) Compiladores são mais lentos, enquanto interpretadores são mais rápidos.
- c) Compiladores são usados para linguagens de alto nível, enquanto interpretadores são usados para linguagens de baixo nível.
- d) Compiladores exigem menos memória, enquanto interpretadores exigem mais memória.
- e) Não há diferença significativa entre compiladores e interpretadores.

AMBIENTE DE PROGRAMAÇÃO E ESTRUTURAS SEQUENCIAIS

Qual é a principal diferença entre um compilador e um interpretador de linguagens de programação?

a) **Compiladores geram código de máquina, enquanto interpretadores executam o código diretamente.**

b) Compiladores são mais lentos, enquanto interpretadores são mais rápidos.

c) Compiladores são usados para linguagens de alto nível, enquanto interpretadores são usados para linguagens de baixo nível.

d) Compiladores exigem menos memória, enquanto interpretadores exigem mais memória.

e) Não há diferença significativa entre compiladores e interpretadores.

ESTRUTURA CONDICIONAL

Qual é a saída do seguinte código em Python?

- a) Par
- b) Ímpar
- c) Negativo
- d) Nenhuma saída
- e) Erro de sintaxe

```
x = 10
if x > 0:
    if x % 2 == 0:
        print("Par")
    else:
        print("Ímpar")
else:
    print("Negativo")
```

ESTRUTURA CONDICIONAL

Qual é a saída do seguinte código em Python?

a) Par

b) Ímpar

c) Negativo

d) Nenhuma saída

e) Erro de sintaxe

```
x = 10
if x > 0:
    if x % 2 == 0:
        print("Par")
    else:
        print("Ímpar")
else:
    print("Negativo")
```


ESTRUTURA DE REPETIÇÃO

Qual é a diferença entre os laços for e while em Python?

- a) O laço for é usado para iterar sobre uma sequência, enquanto o laço while é usado para repetir um bloco de código indefinidamente.
- b) O laço for é mais eficiente em termos de desempenho, enquanto o laço while é mais flexível.
- c) O laço for requer que a quantidade de iterações seja conhecida previamente, enquanto o laço while não tem essa restrição.
- d) Não há diferença entre os laços for e while em Python.
- e) O laço for é usado para estruturas de dados imutáveis, enquanto o laço while é usado para estruturas de dados mutáveis.

ESTRUTURA DE REPETIÇÃO

Qual é a diferença entre os laços for e while em Python?

- a) O laço for é usado para iterar sobre uma sequência, enquanto o laço while é usado para repetir um bloco de código indefinidamente.
- b) O laço for é mais eficiente em termos de desempenho, enquanto o laço while é mais flexível.
- c) O laço for requer que a quantidade de iterações seja conhecida previamente, enquanto o laço while não tem essa restrição.**
- d) Não há diferença entre os laços for e while em Python.
- e) O laço for é usado para estruturas de dados imutáveis, enquanto o laço while é usado para estruturas de dados mutáveis.

ESTRUTURA DE REPETIÇÃO

Resolva:

Imagine que você é responsável por monitorar o consumo de energia elétrica em uma empresa durante um mês. A empresa deseja saber a média de consumo diário de energia para implementar medidas de eficiência energética. Você recebe uma lista com o consumo diário de energia (em kWh) durante os 30 dias do mês e precisa calcular a média desse consumo. Para isso, você decide escrever um programa em Python que utilize comandos de repetição para somar os consumos diários e, em seguida, calcular a média.

```
consumo_diario = [ 50, 45, 55, 60, 48, 52, 47,  
49, 53, 51, 50, 46, 54, 61, 49, 53, 48, 50, 52,  
47, 56, 50, 55, 60, 48, 52, 47, 49, 53, 51 ]
```

ESTRUTURA DE REPETIÇÃO

Possível resolução....

```
# Lista com o consumo diário de energia em kWh durante 30 dias
consumo_diario = [
    50, 45, 55, 60, 48, 52, 47, 49, 53, 51,
    50, 46, 54, 61, 49, 53, 48, 50, 52, 47,
    56, 50, 55, 60, 48, 52, 47, 49, 53, 51
]

# Inicializa a variável para armazenar o total de consumo
total_consumo = 0

# Loop para somar o consumo diário
for consumo in consumo_diario:
    total_consumo += consumo

# Calcula a média de consumo diário
media_consumo = total_consumo / len(consumo_diario)

# Exibe a média de consumo diário
print(f'A média de consumo diário de energia é: {media_consumo:.2f} kWh')
```

STRINGS

Qual é o resultado da seguinte operação em Python?

```
s = "Python"
```

```
print(s[1:4:2])
```

a) ytn

b) Pyt

c) yh

d) hon

e) yt

STRINGS

Qual é o resultado da seguinte operação em Python?

```
s = "Python"
```

```
print(s[1:4:2])
```

a) ytn

b) Pyt

c) yh

d) hon

e) yt

LISTAS, VETORES E MATRIZES

Qual é a principal desvantagem de se usar listas em Python em comparação com arrays numéricos?

- a) Listas podem armazenar dados de tipos diferentes, enquanto arrays numéricos são homogêneos.
- b) Listas são mais eficientes em termos de uso de memória.
- c) Operações básicas, como acesso a elementos, são mais lentas em listas.
- d) Listas não permitem o uso de operações vetoriais.
- e) Listas não têm métodos embutidos, diferente dos arrays numéricos.

LISTAS, VETORES E MATRIZES

Qual é a principal desvantagem de se usar listas em Python em comparação com arrays numéricos?

- a) Listas podem armazenar dados de tipos diferentes, enquanto arrays numéricos são homogêneos.**
- b) Listas são mais eficientes em termos de uso de memória.
- c) Operações básicas, como acesso a elementos, são mais lentas em listas.
- d) Listas não permitem o uso de operações vetoriais.
- e) Listas não têm métodos embutidos, diferente dos arrays numéricos.

LISTAS, VETORES E MATRIZES

Resolva:

Imagine que você trabalha no departamento de recursos humanos de uma empresa e precisa analisar a produtividade dos funcionários ao longo de uma semana. Você tem uma matriz onde cada linha representa um funcionário e cada coluna representa o número de tarefas completadas em um dia da semana (de segunda a sexta-feira). Seu objetivo é calcular a produtividade semanal de cada funcionário (soma das tarefas completadas durante a semana) e identificar o funcionário mais produtivo.

```
produtividade = [ [5, 7, 6, 8, 7], # Funcionário 1  
                  [6, 5, 6, 7, 6], # Funcionário 2  
                  [8, 9, 7, 8, 9], # Funcionário 3  
                  [4, 6, 5, 5, 6], # Funcionário 4  
                  [7, 8, 7, 8, 7] ]# Funcionário 5
```

LISTAS, VETORES E MATRIZES

Uma possível resolução...

Imagine que você trabalha no departamento de recursos humanos de uma empresa e precisa analisar a produtividade dos funcionários ao longo de uma semana. Você tem uma matriz onde cada linha representa um funcionário e cada coluna representa o número de tarefas completadas em um dia da semana (de segunda a sexta-feira). Seu objetivo é calcular a produtividade semanal de cada funcionário (soma das tarefas completadas durante a semana) e identificar o funcionário mais produtivo.

```
produtividade = [  
    [5, 7, 6, 8, 7], # Funcionário 1  
    [6, 5, 6, 7, 6], # Funcionário 2  
    [8, 9, 7, 8, 9], # Funcionário 3  
    [4, 6, 5, 5, 6], # Funcionário 4  
    [7, 8, 7, 8, 7]  # Funcionário 5  
]  
  
# Lista para armazenar a produtividade semanal de cada funcionário  
produtividade_semanal = []  
  
# Calcula a produtividade semanal de cada funcionário  
for funcionario in produtividade:  
    total_tarefas = sum(funcionario)  
    produtividade_semanal.append(total_tarefas)  
  
# Identifica o funcionário mais produtivo  
mais_produtivo = max(produtividade_semanal)  
indice_mais_produtivo = produtividade_semanal.index(mais_produtivo)  
  
# Exibe a produtividade semanal de cada funcionário  
for i, total in enumerate(produtividade_semanal, start=1):  
    print(f'Produtividade semanal do Funcionário {i}: {total} tarefas')  
  
# Exibe o funcionário mais produtivo  
print(f'O Funcionário mais produtivo é o Funcionário {indice_mais_produtivo + 1} com  
{mais_produtivo} tarefas completadas na semana')
```

OUTRAS ESTRUTURAS DE DADOS SIMPLES: TUPLAS, DICIONÁRIOS E CONJUNTOS

Qual das seguintes estruturas de dados em Python é imutável?

- a) Lista
- b) Dicionário
- c) Conjunto
- d) Tupla
- e) Todas as estruturas de dados em Python são mutáveis

OUTRAS ESTRUTURAS DE DADOS SIMPLES: TUPLAS, DICIONÁRIOS E CONJUNTOS

Qual das seguintes estruturas de dados em Python é imutável?

a) Lista

b) Dicionário

c) Conjunto

d) Tupla

e) Todas as estruturas de dados em Python são mutáveis

MODULARIZAÇÃO DE ALGORITMOS (FUNÇÕES)

Qual é o propósito principal do uso de docstrings em funções Python?

- a) Documentar o código-fonte da função.
- b) Fornecer uma descrição textual da função.
- c) Definir os parâmetros obrigatórios da função.
- d) Especificar o tipo de retorno da função.
- e) Todas as alternativas acima.

MODULARIZAÇÃO DE ALGORITMOS (FUNÇÕES)

Qual é o propósito principal do uso de docstrings em funções Python?

- a) Documentar o código-fonte da função.
- b) Fornecer uma descrição textual da função.
- c) Definir os parâmetros obrigatórios da função.
- d) Especificar o tipo de retorno da função.
- e) Todas as alternativas acima.**

MODULARIZAÇÃO DE ALGORITMOS (FUNÇÕES)

Resolva:

Imagine que você é um desenvolvedor de software para uma empresa de eventos. A empresa organiza diversos eventos ao longo do ano, e cada evento tem uma lista de participantes. A empresa deseja uma funcionalidade que permita verificar se um determinado participante está inscrito em um evento específico. Você precisa criar uma função que receba a lista de participantes de um evento e o nome de um participante e retorne um valor booleano indicando se o participante está inscrito ou não.

```
participantes_evento = ['Ana', 'Bruno', 'Carlos', 'Diana', 'Eduardo']  
nome = 'Carlos'
```

```
def verificar_participante(participantes, nome):
```

MODULARIZAÇÃO DE ALGORITMOS (FUNÇÕES)

Uma possível resolução...

Imagine que você é um desenvolvedor de software para uma empresa de eventos. A empresa organiza diversos eventos ao longo do ano, e cada evento tem uma lista de participantes. A empresa deseja uma funcionalidade que permita verificar se um determinado participante está inscrito em um evento específico. Você precisa criar uma função que receba a lista de participantes de um evento e o nome de um participante e retorne um valor booleano indicando se o participante está inscrito ou não.

```
def verificar_participante(participantes, nome):  
    """  
    Verifica se um participante está na lista de participantes de um evento.  
    :param participantes: Lista de nomes dos participantes  
    :param nome: Nome do participante a ser verificado  
    :return: True se o participante estiver na lista, False caso contrário  
    """  
  
    for participante in participantes:  
        if participante.lower() == nome.lower():  
            return True  
    return False  
  
# Exemplo de uso  
participantes_evento = ['Ana', 'Bruno', 'Carlos', 'Diana', 'Eduardo']  
nome_participante = 'Carlos'  
  
# Verifica se o participante está inscrito no evento  
esta_inscrito = verificar_participante(participantes_evento, nome_participante)  
if esta_inscrito:  
    print(f'{nome_participante} está inscrito no evento.')  
else:  
    print(f'{nome_participante} não está inscrito no evento.')
```


MÓDULOS. NUMPY

Qual é uma das principais vantagens do uso da biblioteca NumPy em comparação com o uso de listas padrão do Python?

- a) Maior flexibilidade na manipulação de dados.
- b) Melhor desempenho em operações matemáticas e vetoriais.
- c) Suporte a processamento de dados multidimensionais.
- d) Integração mais fácil com outras bibliotecas de análise de dados.
- e) Todas as alternativas acima.

MÓDULOS. NUMPY

Qual é uma das principais vantagens do uso da biblioteca NumPy em comparação com o uso de listas padrão do Python?

- a) Maior flexibilidade na manipulação de dados.
- b) Melhor desempenho em operações matemáticas e vetoriais.
- c) Suporte a processamento de dados multidimensionais.
- d) Integração mais fácil com outras bibliotecas de análise de dados.
- e) Todas as alternativas acima.**

MÓDULOS. PANDAS

Qual é a principal diferença entre as estruturas de dados Series e DataFrame da biblioteca Pandas?

- a) Series armazena uma única coluna de dados, enquanto DataFrame armazena múltiplas colunas.
- b) Series é uma estrutura unidimensional, enquanto DataFrame é bidimensional.
- c) Series é mais eficiente em termos de uso de memória, enquanto DataFrame é mais flexível.
- d) Series é imutável, enquanto DataFrame é mutável.
- e) Não há diferença significativa entre as estruturas Series e DataFrame.

MÓDULOS. PANDAS

Qual é a principal diferença entre as estruturas de dados Series e DataFrame da biblioteca Pandas?

a) Series armazena uma única coluna de dados, enquanto DataFrame armazena múltiplas colunas.

b) Series é uma estrutura unidimensional, enquanto DataFrame é bidimensional.

c) Series é mais eficiente em termos de uso de memória, enquanto DataFrame é mais flexível.

d) Series é imutável, enquanto DataFrame é mutável.

e) Não há diferença significativa entre as estruturas Series e DataFrame.

MÓDULOS. PANDAS

Resolva:

Você trabalha como analista de dados em uma pequena loja de conveniência que deseja entender melhor suas vendas diárias. A loja coleta informações sobre as vendas de produtos em um DataFrame, incluindo colunas como data, produto, quantidade_vendida e receita. Seu objetivo é utilizar o pacote Pandas em Python para realizar algumas análises simples, como calcular a receita total, identificar o produto mais vendido e calcular a média de vendas diárias.

```
dados_vendas = pd.DataFrame({ 'data': ['2024-06-01', '2024-06-01', '2024-06-02', '2024-06-02', '2024-06-03'],  
    'produto': ['Camiseta', 'Calça', 'Camiseta', 'Tênis', 'Calça'], 'quantidade_vendida': [5, 2, 3, 4, 1],  
    'receita': [100.00, 200.00, 60.00, 400.00, 100.00] })
```

PANDAS... RESOLUÇÃO

```
import pandas as pd

# Cria um DataFrame com os dados de vendas
dados_vendas = pd.DataFrame({
    'data': ['2024-06-01', '2024-06-01', '2024-06-02', '2024-06-02', '2024-06-03'],
    'produto': ['Camiseta', 'Calça', 'Camiseta', 'Tênis', 'Calça'],
    'quantidade_vendida': [5, 2, 3, 4, 1],
    'receita': [100.00, 200.00, 60.00, 400.00, 100.00]
})

# Exibe o DataFrame para verificar os dados
print("Dados de Vendas:")
print(dados_vendas)

# Calcula a receita total
receita_total = dados_vendas['receita'].sum()
print(f"\nReceita total: R${receita_total:.2f}")
```

PANDAS... RESOLUÇÃO

```
# Identifica o produto mais vendido (em quantidade)
produto_quantidades = {}

for i, produto in enumerate(dados_vendas['produto']):
    if produto in produto_quantidades:
        produto_quantidades[produto] += dados_vendas['quantidade_vendida'][i]
    else:
        produto_quantidades[produto] = dados_vendas['quantidade_vendida'][i]

produto_mais_vendido = max(produto_quantidades, key=produto_quantidades.get)
quantidade_mais_vendida = produto_quantidades[produto_mais_vendido]

print(f"Produto mais vendido: {produto_mais_vendido} ({quantidade_mais_vendida} unidades)")
```

PANDAS... RESOLUÇÃO

```
# Calcula a média de vendas por dia

# Converte a coluna 'data' para o tipo datetime
dados_vendas['data'] = pd.to_datetime(dados_vendas['data'])

dias_unicos = dados_vendas['data'].dt.date.unique()

media_vendas_por_dia = {}

for dia in dias_unicos:
    vendas_no_dia = dados_vendas[dados_vendas['data'].dt.date == dia]['quantidade_vendida'].sum()
    media_vendas_por_dia[dia] = vendas_no_dia

media_vendas = sum(media_vendas_por_dia.values()) / len(media_vendas_por_dia)

print(f"\nMédia de vendas por dia: {media_vendas:.2f} unidades")
```




BONS ESTUDOS!!

E ATÉ A NOSSA AVALIAÇÃO, DIA 11/06/2024