

# FIA/P

Graduação  
Inteligência Artificial & Chatbot



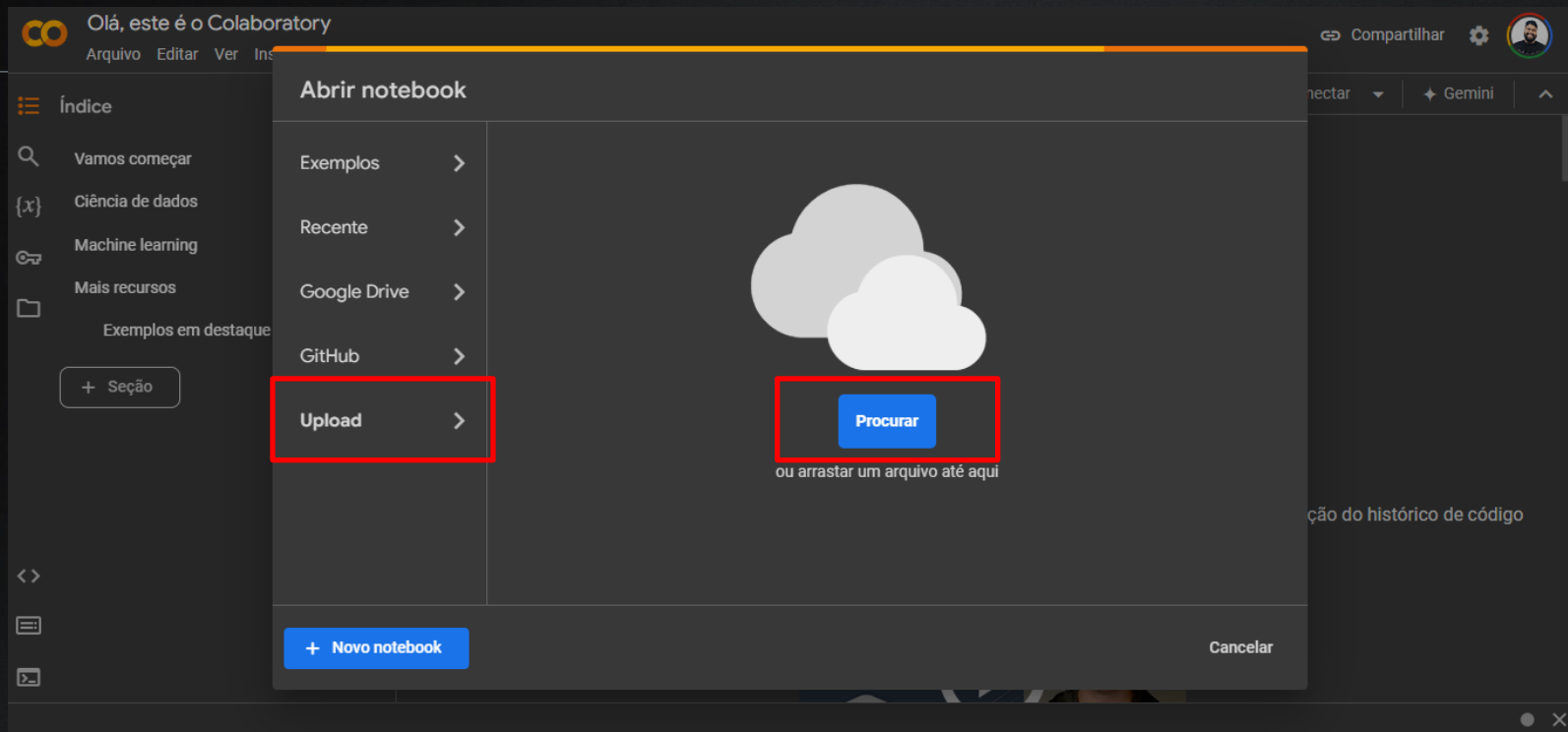
## numPy & Pandas

numPy e Pandas são bibliotecas essenciais para análise de dados em Python. numpy fornece suporte para arrays eficientes e operações matemáticas, enquanto pandas usa essa base para trabalhar com dados estruturados em tabelas, facilitando a manipulação, análise e limpeza dos dados.




## Dados estruturados

Vá até: <https://colab.research.google.com/>. Faça o Login utilizando sua conta google. E Realize o upload do notebook **Introdução a numPy e Pandas - Inicio de Aula.ipynb**



# Dados estruturados









## Notebook Introdução a numPy e Pandas - Início de Aula.ipynb

 Introdução a numPy e Pandas - Início de Aula.ipynb ☆

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda [Todas as alterações foram salvas](#)

+ Código + Texto

Conectar Gemini

O numpy é uma biblioteca fundamental para computação numérica em Python. Ela fornece suporte para arrays multidimensionais, que permitem armazenar e manipular grandes volumes de dados de forma eficiente. Além disso, oferece funções matemáticas de alto desempenho para cálculos científicos e análises complexas.

### ▼ Criação de um ndarray

Um ndarray (N-dimensional array) é o principal objeto da biblioteca numpy. Ele é uma estrutura de dados que armazena elementos de forma multidimensional em um array de tamanho fixo, onde todos os elementos devem ser do mesmo tipo (como inteiros, floats, etc.). O ndarray é altamente eficiente para operações matemáticas e computacionais.

```
[ ] #Vamos criar um nd array simples
```

### ▼ Operações Matemáticas

```
[ ] #Multiplicação do ndarray por 2
```

## Import das bibliotecas

Vamos começar importando as bibliotecas que são essenciais para a análise de dados em Python.

### ▼ Import das Bibliotecas

```
[1] import numpy as np  
import pandas as pd
```

### 1 – numPy

- Ao importar numpy com `import numpy as np`, você está criando um alias `np` que é uma forma comum e prática de abreviar o nome da biblioteca para facilitar o uso em código. Isso é útil porque o nome completo da biblioteca (`numpy`) pode ser longo e usar `np` torna o código mais limpo e fácil de ler.

### 2 – Pandas

- Ao importar pandas com `import pandas as pd`, você está criando um alias `pd` que é uma convenção padrão na comunidade Python para usar pandas. Esse alias simplifica a escrita e leitura do código.



## numPy e o ndarray

O numpy é uma biblioteca fundamental para computação numérica em Python. Ela fornece suporte para arrays multidimensionais, que permitem armazenar e manipular grandes volumes de dados de forma eficiente. Além disso, oferece funções matemáticas de alto desempenho para cálculos científicos e análises complexas.

Vamos começar criando um ndarray. Um ndarray (N-dimensional array) é o principal objeto da biblioteca numpy. Ele é uma estrutura de dados que armazena elementos de forma multidimensional em um array de tamanho fixo, onde todos os elementos devem ser do mesmo tipo (como inteiros, floats, etc.). O ndarray é altamente eficiente para operações matemáticas e computacionais.

```
[71] #Vamos criar um nd array simples
arr = np.array([
    [5, 6, 7, 8],
    [8, 8, 8, 9],
    [1, 1, 1, 1]])
arr
```

```
array([[5, 6, 7, 8],
       [8, 8, 8, 9],
       [1, 1, 1, 1]])
```

## Operações matemáticas simples com numPy

O NumPy é amplamente utilizado para realizar operações matemáticas rápidas e eficientes em arrays. Aqui estão alguns exemplos e operações matemáticas simples que você pode realizar com o NumPy:

```
[72] #Multiplicação do ndarray por 2  
arr * 2
```

```
↗ array([[10, 12, 14, 16],  
        [16, 16, 16, 18],  
        [ 2,  2,  2,  2]])
```

```
[73] #Subtração do ndarray por 5  
arr - 5
```

```
↗ array([[ 0,  1,  2,  3],  
        [ 3,  3,  3,  4],  
        [-4, -4, -4, -4]])
```

```
[74] #Multiplicação de dois ndarray  
arr * arr
```

```
↗ array([[25, 36, 49, 64],  
        [64, 64, 64, 81],  
        [ 1,  1,  1,  1]])
```



## Seleção, Indexação e Fatiamento com o numPy

Vamos explorar os conceitos de Seleção, Indexação e Fatiamento (Slicing) em NumPy, que são fundamentais para manipular e acessar dados em arrays (ndarray).

**Indexação** é o processo de acessar elementos individuais em um array:

```
[131] #Retorno o primeiro índice com o numpy  
arr[0]
```

```
↩ array([5, 6, 7, 8])
```

**Seleção** refere-se a acessar um subconjunto de elementos com base em uma condição ou uma lista de índices.

```
[78] #Selecionar todos os valores menores do que 3  
arr[arr < 3]
```

```
↩ array([1, 1, 1, 1])
```

**Fatiamento** é o processo de acessar uma fatia ou um subarray de um array.

```
[87] #retornar os três primeiros valores do primeiro array  
arr[0:1,0:3]
```

```
↩ array([[5, 6, 7]])
```

## Funções embutidas com o numPy

O NumPy oferece uma vasta gama de funções embutidas que são altamente otimizadas para operar em arrays e realizar cálculos matemáticos e estatísticos. Aqui estão algumas das funções embutidas mais comuns e úteis:

### Soma

```
[91] #Função embutida para somar os valores de um ndarray  
np.sum(arr)
```

63

### Média

```
[93] #Função embutida para realizar a média dos valores de um ndarray  
np.mean(arr[1])
```

8.25

### Max

```
[94] #Função embutida para retornar o valor máximo de um ndarray  
arr.max()
```

9

## Pandas

Pandas é uma biblioteca de código aberto para análise de dados em Python. Ela é usada principalmente para manipulação, limpeza e análise de dados estruturados, como tabelas (similares às planilhas do Excel). Pandas fornece estruturas de dados eficientes e flexíveis, chamadas de Series (colunas unidimensionais) e DataFrames (tabelas bidimensionais).

Vamos começar explorando um Dataframe:

```
#Dataframe nota dos alunos
data = {
    'Aluno': ['Robb', 'Jon', 'Arya'],
    'Matemática': [85, 90, 95],
    'História': [88, 92, 80],
    'Ciências': [90, 85, 88]
}
```

Um DataFrame no pandas é uma estrutura de dados bidimensional que se assemelha a uma tabela ou planilha, onde você pode armazenar e manipular dados de forma organizada. Cada coluna no DataFrame pode ser de um tipo de dado diferente (inteiro, float, string, etc.), e cada linha representa uma entrada ou observação no conjunto de dados.

# Criação de um DataFrame no Pandas

Vamos criar um novo DataFrame a partir dos dados da variável data com o comando pd.DataFrame.

```
#realizar a leitura de um Dataframe
df_notas = pd.DataFrame(data)
#Exibir os 5 primeiros registros
df_notas.head()
```



	Aluno	Matemática	História	Ciências
0	Robb	85	88	90
1	Jon	90	92	85
2	Arya	95	80	88

## 1. Criar um DataFrame

- `pd.DataFrame(data)`: Este comando cria um DataFrame `df_notas` a partir de uma variável `data`.
- `data`: Esta variável deve ser uma estrutura de dados que o pandas pode converter em um DataFrame, como um dicionário, uma lista de listas, ou uma lista de dicionários.
- DataFrame: É uma tabela bidimensional, onde os dados são organizados em linhas e colunas.

## 2. Exibir os 5 primeiros registros

- `df_notas.head()`: Esta função exibe as primeiras 5 linhas do DataFrame `df_notas`.
- Uso: É útil para obter uma visão rápida dos dados e verificar se o DataFrame foi carregado corretamente e se os dados estão conforme esperado.
- Parâmetro opcional: Você pode passar um número inteiro para `head()` para exibir um número específico de linhas. Por exemplo, `df_notas.head(10)` exibirá as primeiras 10 linhas.

## Operações Matemáticas com Pandas

O pandas oferece uma ampla gama de operações matemáticas e estatísticas para manipulação e análise de dados em DataFrames e Series. Essas operações podem ser realizadas diretamente sobre colunas de dados e são otimizadas para lidar com grandes conjuntos de dados de forma eficiente.

### Multiplicação

```
[96] #Multiplicação de uma coluna de um Dataframe pandas  
df_notas['Matemática'] * 3
```



Matemática

0	255
1	270
2	285

dtype: int64

### Divisão

```
[97] #Divisão de uma coluna de um Dataframe pandas  
df_notas['Matemática'] / 2
```



Matemática

0	42.5
1	45.0
2	47.5

dtype: float64

## Filtros e Seleções com Pandas

Filtros e seleções de colunas são operações fundamentais no pandas para manipular e analisar dados. Vou detalhar como você pode realizar essas operações em um DataFrame.

### Seleção de Colunas

Selecionar uma única coluna: Você pode acessar uma única coluna usando o nome da coluna como uma chave.

```
[99] #Selecionar dados de uma coluna  
df_notas['Matemática']
```



Matemática

0	85
1	90
2	95

dtype: int64

```
[100] #Selecionar duas colunas  
df_notas[['Matemática', 'História']]
```



Matemática História

0	85	88
1	90	92
2	95	80



# Filtros e Seleções com Pandas

## Seleção de Linhas / Índices

Selecionar por índice: Use `iloc` para selecionar linhas por índice numérico.

O `iloc` é um método do pandas usado para acessar dados de um DataFrame ou Series com base na posição das linhas e colunas, ou seja, por índices numéricos. O nome "iloc" vem de integer location, o que significa que ele utiliza números inteiros para identificar a localização dos dados

```
[101] #Selecione a primeira posição das linhas  
df_notas.iloc[0]
```



0	
Aluno	Robb
Matemática	85
História	88
Ciências	90

dtype: object

# Filtros e Seleções com Pandas

## Filtro por condições

Filtrar com várias condições: Use operadores lógicos para combinar condições.

```
[105] #Selecione as notas de um Aluno  
df_notas[df_notas['Aluno'] == 'Jon']
```



	Aluno	Matemática	História	Ciências
1	Jon	90	92	85

## Filtro com alteração de valores

O loc é um método do pandas utilizado para acessar dados em um DataFrame ou Series com base no rótulo das linhas e colunas, ou seja, por nomes e rótulos, ao contrário do iloc que usa posições numéricas.

```
[104] #Selecione as notas de um Aluno com loc  
df_notas.loc[df_notas['Aluno'] == 'Jon']
```



	Aluno	Matemática	História	Ciências
1	Jon	90	92	85

## Funções embutidas com Pandas

No pandas, funções embutidas são métodos e funções que vêm integradas na biblioteca para facilitar a manipulação e análise de dados. Estas funções são otimizadas para trabalhar com estruturas de dados do pandas, como DataFrames e Series, e ajudam a realizar tarefas comuns de forma eficiente. Vamos explorar algumas das principais funções embutidas em pandas.

### Média

```
[130] #Qual a média da turma na matéria de Matemática?  
df_notas['Matemática'].mean()
```

89.0

```
[129] #Qual a média geral do Aluno chamado Robb  
df_notas[  
    (df_notas['Aluno'] == 'Robb')  
].iloc[:,1:4].mean(axis=1)
```

0

0 87.666667

dtype: float64

**Copyright © 2025**

**Apresentação criada pelo prof. Dr Gustavo Molina.**

**Adaptada do material do prof. Rodolfo Moreira.**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

FIAP