Essa é uma versão dos Guias de Estudo desenvolvida para auxiliar o processo de impressão e para facilitar a leitura dos guias por programas que fornecem a leitura automatizada para suporte a todos os alunos que necessitem. Dessa forma, não apresentaremos ilustrações nesse arquivo.* *#ParaTodosLerem****.*

Módulo #11

Guias de estudo

Módulo #11

Nos módulos anteriores você aprendeu sobre programação e aprendeu a utilizar a linguagem Python para desenvolver os seus programas. Neste e nos próximos módulos você vai aplicar todo este conhecimentos em diferentes áreas. No módulo 9 você aprendeu a utilizar o Python para desenvolver programas voltados para analisar e visualizar dados, o que possui muita relação com a área de ciência de dados, area esta que vem ganhando um grande destaque no mercado.

Há algumas décadas pessoas e empresas têm gerado uma quantidade significativa de dados, os quais ficam armazenados nas bases de dados das empresas. Estes dados, quando bem explorados, podem explicar e ser transformados em muitas coisas boas. Por exemplo, é possível descobrir o porquê de uma empresa ter vendido menos em um determinado mês; qual produto determinados clientes gostariam de comprar; quantos produtos manter em estoque; etc.. Extrair informações relevantes a partir de um conjunto de dados é uma tarefa da área de ciência de dados e uma das maneiras de fazer isso é por meio de análise e visualização dos dados.

Existem algumas ferramentas já implementadas em Python que podemos utilizar para nos auxiliar no processo de análise e visualização de dados, como por exemplo Numpy, MatlibPlot e Pandas. Vamos ver alguns exemplos de como usar?

Suponha que você é professor e possui como dados as notas dos seus alunos. Caso queira armazenar apenas as notas você pode utilizar um Numpy array de 1 dimensão:

```
#ARRAY 1D

notasAluno = np.array([5,4,3.1,2,5,8])
notasAluno
```

No exemplo acima armazenamos 6 notas de um aluno em um array chamado `notasAluno`.

O tamanho de um array informa quantos elementos ele carrega e para obter esta informação você pode utilizar a função shape:

```
#Veja o tamanho
notasAluno.shape
```

Quando criamos o array chamado `notasAluno` *ele carregava apenas as notas de um único aluno. Então você pode estar se perguntando, como faço para armazenar a nota de vários alunos? Bom, para isso podemos utilizar um array com 2 dimensões:*

```
notasPorAlunos =
np.array([[10,20,3.5,10,5,9],[10,20,30,5,10,9],[20,20,40,9,1,10]])
notasPorAlunos
```

Observe na estrutura criada que temos vários arrays dentro de um único array. Deste modo, cada linha será um aluno e nas colunas teremos as 6 notas deles :

```
array([[10. , 20. ,  3.5, 10. ,  5. ,  9. ],
       [10. , 20. , 30. ,  5. , 10. ,  9. ],
       [20. , 20. , 40. ,  9. ,  1. , 10. ]])
```

E se quisermos acompanhar os alunos desta turma durante 2 anos, como podemos armazenar estes dados? Para isso, podemos utilizar um array de 3 dimensões:

```
notasPorAlunosPorAno=np.array([[[9,3,3,10,1,9],[5,20,3.5,10,5,8],[2,
5,9,10,7,9]] , [[10,7,4,10,5,9],[9,20,7,10,5,9],[3,8,9,10,4,1]]])
notasPorAlunosPorAno
```

A estrutura fica da seguinte maneira:

```
array([[[ 9. ,  3. ,  3. , 10. ,  1. ,  9. ],
        [ 5. , 20. ,  3.5, 10. ,  5. ,  8. ],
        [ 2. ,  5. ,  9. , 10. ,  7. ,  9. ]],

       [[10. ,  7. ,  4. , 10. ,  5. ,  9. ],
        [ 9. , 20. ,  7. , 10. ,  5. ,  9. ],
        [ 3. ,  8. ,  9. , 10. ,  4. ,  1. ]]])
```

O primeiro conjunto com as 6 notas dos 3 alunos diz respeito a um determinado ano e o segundo conjunto a um outro ano.

Com o numpy você pode estruturar seus dados com a quantidade de dimensões desejada. Vale lembrar que o numpy é uma ferramenta para estruturar os dados e você pode conhecer mais sobre ela na documentação original: https://numpy.org/doc/

O Pandas é uma ferramenta bastante utilizada para análise de dados e estruturas de dados. Com esta ferramenta é possível estruturar os dados de duas maneiras:

Serie: Uma Series é como um array de 1 dimensão. Toda Serie possui um índice e os valores são relacionados aos índices.

Vamos então criar uma série com as notas de um aluno:

```
notasAlunosSeries = pd.Series([5,4,3.1,2,5,8])
notasAlunosSeries
```

Veja o resultado:

```
0    5.0
1    4.0
2    3.1
3    2.0
4    5.0
5    8.0
```

Podemos alterar os índices 0,1,2,3,4 e 5 para o título das notas (prova1, prova2, prova3, prova4, prova5 e prova6)

```
notasAlunosSeriesIndex = pd.Series([5,4,3.1,2,5,8], index=["prova 1", "prova 2", "prova 3", "prova 4", "prova 5", "prova 6"])
notasAlunosSeriesIndex
```

Podemos recuperar todos os valores:

```
#pega somente os valores
notasAlunosSeries.values
```

e também todos os índices:

```
#analisa os índices. Começa em 0, tem 6 posições e as posições caminham de 1 em 1
notasAlunosSeries.index
```

Quanto o aluno tirou na prova1?

```
#recupera a nota relacionada ao indice prova 1
notasAlunosSeriesIndex["prova 1"]
```

Para recuperar este valor basta acessar o índice `"prova 1"` da série `notasAlunosSeriesIndex`.

Você já aprendeu como estruturar e recuperar dados de uma série. Agora vamos ver como podemos trabalhar com dataframes. Vale ressaltar que o dataframe permite criar uma tabela com observações e atributos (linhas e colunas).

Vamos ver um exemplo:

```
#cria o dataframe
df = pd.DataFrame({'Aluno' : ["João", "Maria", "Pedro", "Julia",
"José"],
                   'Faltas' : [3,4,2,1,4],
                   'Nota Final' : [80.0,75.8,50.1,100.0,60.5]})
df
```

Nesta estrutura temos as chaves `'Aluno'`, `'Faltas'` e `'Nota Final'` e os valores associados a elas. Vamos visualizar o resultado:

	Aluno	Faltas	Nota Final
0	João	3	80.0
1	Maria	4	75.8
2	Pedro	2	50.1
3	Julia	1	100.0
4	José	4	60.5

Para recuperar os índices, os quais ficam dispostos nas colunas, basta acessar a função columns:

```
df.columns
```

Se quiser ver os valores (nomes) de todos os alunos basta acessar o índice Alunos:

```
df["Aluno"]
```

Resultado:

```
0     João
1    Maria
2    Pedro
3    Julia
4     José
```

Muitas vezes você receberá os dados em um arquivo no formato csv, excel, html e outros. Quando isso acontecer basta você carregar os dados deste aquivo para uma dessas estruturas. Se for uma tabela é mais comum carrega-los em um dataframe. Vamos ver um exemplo:

```
#como ler um arquivo csv, e carrega-lo para m dataframe:
df = pd.read_csv("NomeAquivo.csv")
df
```

Veja outras opções para carregar os dados de um arquivo em um dataframe na documentação do pandas:
https://pandas.pydata.org/docs/user_guide/io.html

Veja a documentação do matlibplot para que você possa visualizar os dados no formato de gráficos:
https://matplotlib.org

Lembre-se

Quando quiser analisar e visualizar grande quantidade de dados, identifique primeiro qual a melhor estrutura para armazenar os dados e quais análises você quer fazer sobre eles. Os dados que envolvem cálculos estatísticos como média, soma e outros devem ser feitos com dados do tipo número, nunca com strings.