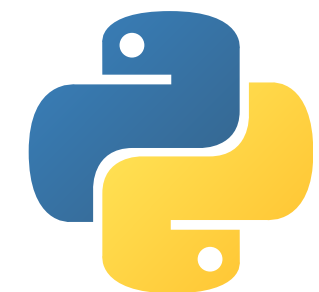







DATA SCIENCE



Programação Dinâmica








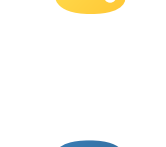


INICIO

-  **verificar se há python na máquina, instalar ou atualizar** atento ao sistema operacional
-  **atualizar o pip** atento ao sistema operacional, sempre verificar o comando na documentação
-  **instalar o pandas** atento ao s.o., olhar a documentação, e já vem com numpy
-  **instalar o scipy e após o matplotlib** quase tudo pronto
-  **ao final instalar o jupyter notebook** pois é nele que o código será escrito e rodará

LEITURA

-  **`pandas.read_csv("local do arquivo", encoding = "UTF-8", sep = ";")`** retorna um dataframe
-  **`pandas.read_excel("local do arquivo.xlsx", ...)`** transforma o arquivo em um dataframe

COMANDOS

-  ***dataframe.head()*** retorna as colunas e as 5 primeiras linhas
-  ***dataframe.shape*** retorna a quantidade de linhas e colunas
-  ***dataframe.sheet_names*** retorna o nome das abas do arquivo
-  ***dataframe.columns.values*** retorna um vetor com nome de todas as colunas
-  ***dataframe.filter(items="nome.da.coluna"/array de colunas)***
retorna uma serie só com a coluna selecionada, ou dataframe com as colunas do array
-  ***dataframe[linha inicial : linha final]*** retorna as linhas pelo número da inicial a final -1
-  ***series.value_counts()*** retorna a letra/palavra e a quantidade de aparições
-  ***dataframe/series.sort_index()*** retorna a letra/palavra e a quantidade de aparições

 **`variável.dropna()`** retira as linhas que tem NaN

 **`variável.groupby("nome.da.coluna").algumacoisa`**

retorna a primeira coluna como principal e faz algumacoisa com a segunda

 **`groupby("nome.da.coluna").count()`**

conta quantas colunas estão preenchidas com qualquer coisa da primeira

 **`groupby("nome.da.coluna").value_counts()`**

conta quantas vezes se repete cada elemento da coluna

 **`groupby("nome.da.coluna").max() / .min()`** retorna o valor máximo ou o mínimo

 **`dataframe["nome.da.nova.coluna"] = dicionário[variável] for var in dataframe.coluna`**

cria uma nova coluna com o valor correspondente a chave do dicionário para cada item da coluna do dataframe que seja igual a chave do dicionário (substitui bem o replace)

 **`variavel['coluna1', 'coluna2'].sort_values(by = 'coluna2', ascending = False)`**










retorna/exibe os valores da coluna 2 como parâmetros ordenados do maior para o menor

 **`variável.where(variável.coluna == 'texto'/número)`** filtra pelo texto ou número



variável.sort_values(by=['coluna'], ascending = bool) organiza a coluna escolhida

MATPLOTLIB

-  ***variável.hist()*** monta um histograma dos dados numéricos
-  ***variável.hist(bins = número)*** controla o espaçamento da linha X
-  ***variável.plot()*** monta um histograma dos dados numéricos
-  ***variável.unstack().plot()*** separa os dados por grupos no gráfico
-  ***variável.plot().legend()*** pra dar nome a cada resultado do gráfico
-  ***variável.plot(figsize = x , y)*** numerando x e y mexe no tamanho do gráfico
-  ***variável.plot(fontsize = 'número')*** altera o tamanho das letras em x e y
-  ***ax = variável.plot()*** retorna o gráfico para usar os métodos da classe axis
-  ***ax.set_title('Título', fontsize = número)*** dá título ao gráfico



ax.legend(bbox_to_anchor = (x,y), loc = número/letra, borderaxespad = 0)

bbox diz onde a caixa ficará de 0,0 a 1,1 - loc é localização na caixa



variável.plot(colormap = 'texto') altera as cores das linhas do gráfico