**Python Pandas**

**Curso 1 – Tratando e Analisando Dados:**

1. **Aula 1 – Conhecendo Jupyter:**
   1. O primeiro passo é baixar o Anaconda. Basta digitar no google e baixar, ou acesse o link: <https://www.anaconda.com>.
   2. Ambientes virtuais são muito importantes para setar e manter a versão e funcionalidades de todas as bibliotecas, linguagens etc, garantindo que caso queira rodar um código que você escreveu há um ano atrás, ele continue funcionando normalmente.
   3. Pip: gerenciador de pacotes padrão do python.
   4. Conda: gerenciador de pacotes padrão do Anaconda.
   5. Para criar esse ambiente virtual vamos utilizar o seguinte comando no cmd:

conda create --name alura\_pandas python=3.8.5 pandas=1.1.3

* + 1. Podemos alterar o nome, a versão do python e do pandas para quais quisermos.
    2. Caso o cmd esteja devolvendo um erro de HTTP, sugiro que utilize o cmd do próprio anaconda. Se fizer isso, vai criar o ambiente normalmente.
    3. Após o carregamento e criação do ambiente, ao entrar na parte de ambientes do anaconda veremos nosso ambiente lá:



* + 1. Para ativar o nosso ambiente e rolar ele no próprio cmd, precisamos colocar activate e o nome da pasta do nosso ambiente.
    2. Para desativar, basta digitar deactivate.
    3. Para visualizar os ambientes que temos usamos o comando “conda info –envs. Cmd retorna todos os ambientes que temos criados.
    4. Para destruir o ambiente: conda remove –name alura\_pandas –all.
  1. O que aprendemos:
     1. Anaconda é a principal distribuição para cientistas de dados que usam Python;
     2. Jupyter é a nossa ferramenta para executar código Python e visualizar os dados;
     3. Ambientes virtuais ajudam a isolar um projeto para definir a versão das bibliotecas e do próprio Python.

1. **Aula 2 – Importando Dados:**
   1. Estamos utilizando o notebook do jupyter para fazer nossos códigos agora, portanto os atalhos e comandos são meio diferentes.
      1. O jupyter é esse “app” disponível no anacondas:



* + 1. Ele executa e abre diretamente no navegador:



* + 1. A partir dele podemos criar e acessar pastas, arquivos, dentre outras coisas, como o explorer.
    2. Criamos uma pasta de projetos -> curso pandas -> dados, e fizemos o upload dos dados “alugueis” que vamos trabalhar.
    3. Criamos um notebook com o python 3 e começamos importando o pandas e lendo o arquivo:





* + 1. Caso tenha alguma dúvida do que passar como parâmetro e ou quais métodos/funções uma biblioteca tem, pode apertar SHIFT + TAB para ele exibir a ajuda.
  1. Atalhos jupyter:
     1. M: muda a célula selecionada para tipo markdown;
     2. Y: muda a célula para tipo código;
     3. A: cria uma célula acima da que está;
     4. B: cria uma célula abaixo da que está;
     5. SHIFT + ENTER: roda o código da clélula.
  2. Podemos criar um DataFrame a partir dos tipos de dados:



* + 1. Podemos passar outro parâmetro para o DataFrame fazendo ele dar nome às colunas:

pd.DataFrame(dados.dtypes, columns=['Tipos de Dados'])



* + 1. Outro jeito de fazer isso, mas na primeira coluna dessa vez:

tipos\_de\_dados.columns.name = 'Variáveis'



* 1. Extras:
     1. Aula extra de como importar dados de diversos tipos de extensão de arquivos. O primeiro foi json.
     2. Podemos abrir um arquivo e mandar printar a leitura dele para termos uma noção dos dados:



* + 1. Para importar os dados json é igual ao csv:



* + 1. Abertura e leitura de dados funcionam igual para todas as extensões:



* + 1. Para importar, usamos o table:



* + 1. No caso de arquivos excel não conseguimos ler, mas o import é assim:



* + 1. Pegar informação de html:



* + - 1. Note que ela nos devolve uma lista, para devolver bonitinho colocamos index 0:



* + 1. Caso não tenha salvo a pag web, basta passar o url dela:



* + - 1. Quando a página tem mais de uma tabela, recebemos todas elas no nosso DataFrame:



* + - 1. Nesse caso, para acessar cada uma das tabelas é só indexar de 0 (primeira) até a última delas, visualizando todas.
  1. O que aprendemos:
     1. Como importar a biblioteca (import pandas as pd);
     2. Como ler fontes de dados diferentes:
        1. Uma base CSV (pd.read\_csv(...));
        2. Uma base JSON (pd.read\_json(...));
        3. Uma base TXT (pd.read\_table(...));
        4. Um arquivo EXCEL (pd.read\_excel(...));
        5. Uma página HTML (pd.read\_html(...)).
     3. Vários métodos e atributos úteis de dataframes, como:
        1. info();
        2. head();
        3. dtypes;
        4. columns;
        5. shape.
     4. E sobre Jupyter, vimos como:
        1. Criar diferentes tipos de células dentro do Jupyter;
        2. Acessar a documentação;
        3. Como reexecutar todas as células.

1. **Aula 3 – Series e Index:**
   1. Drop\_duplicates(): Elimina todas as duplicatas de uma series/DataFrame mantendo o primeiro:





* + 1. Também possui o parâmetro inplace.
  1. Podemos mostrar todos os index de uma variável em uma lista:



* + 1. Se quisermos alterar e colocar em ordem crescente, podemos fazer com que o index da series receba o range do tamanho dela:



* 1. Extra: Criando DataFrames:
     1. Podemos criar series de maneira bem simples:



* + 1. Existe o parâmetro índice que podemos utilizar para colocar rótulos personalizados ao invés de numérico como default:



* + 1. Podemos usar a mesma estrutura acima para criar um dicionário:



* + 1. E com isso facilitar a criação do índice:



* + 1. Podemos fazer operações matemáticas com as series cujo elas alteram seus valores:



* 1. Extra 2: Concatenando DataFrames
     1. Aprendemos a criar DataFrames com listas, dicts, tuples e tudo mais.
     2. Nada que já não saibamos fazer, mas caso tenha alguma dúvida e/ou queira consultar: <http://localhost:8891/notebooks/Projetos%20Python/Curso%20Pandas/extras/Criando%20Estrutura%20de%20Dados.ipynb>.
     3. Podemos criar condições no index e mandar receber um valor diferente do que existe para substituir caso seja verdadeiro:

 

* + 1. Concat([dataframes]): Concatena listas de DataFrames a partir das colunas:



* + - 1. Se alterarmos o eixo pra 1 ele inverte:

