FICHA TRABALHO 1

MODALIDADE:	Aprendizagem +	Não aplicável	
CURSO:	Técnico/a Especialista em Gestão de Informação e Ciência dos Dados		
UFCD:	Limpeza e transformação de dados em Python	CÓDIGO UFCD:	10808
FORMADOR/A:	Bruno Silva	DATA:	

OBJETIVOS

- Introdução a biblioteca Pandas
- Instalação e configuração dos ambientes de trabalho Pandas e Jupyter
- Comandos básicos com a biblioteca Pandas

Parte 1 – Instalação e Configuração Pandas

Pandas é uma biblioteca para Ciência de Dados de código aberto (*open source*), construída sobre a linguagem Python.

Esta biblioteca providencia uma abordagem **rápida** e **flexível**, com estruturas robustas para se trabalhar com dados relacionais (ou rotulados), e tudo isso de maneira simples e intuitiva.

Pode ser utilizado para várias atividades e processos, entre eles: limpeza e tratamento de dados, análise exploratória de dados, suporte em atividades de Machine Learning, consultas e queries em base de dados relacionais, visualização de dados, webscraping e muito mais.

Além disso, também possui ótima integração com várias outras bibliotecas muito utilizadas em Ciência de Dados, tais como: Numpy, Scikit-Learn, Seaborn, Altair, Matplotlib, Plotly, Scipy e outros.

Podemos trabalhar os dados no terminal do Visual Studio Code ou também podemos instalar a aplicação Jupyter. Como tal, vamos ver como instalar a aplicação no vosso computador.

Parte 2 – Instalação e Configuração Jupyter

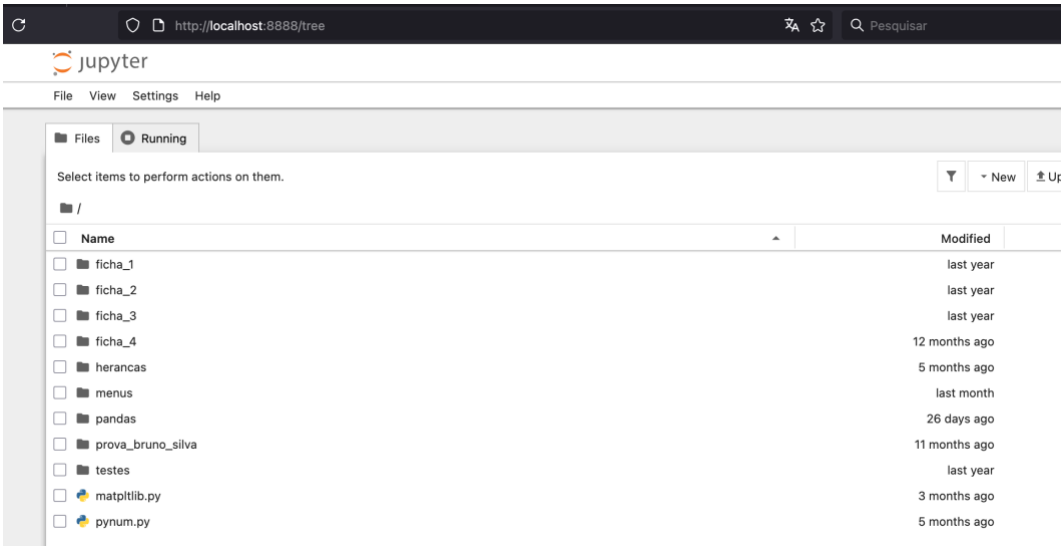
Comando instalação Jupyter:

- **Windows:** pip install jupyter
- **MacOS:** pip3 install jupyter

Para correr o servidor da aplicação basta digitar:

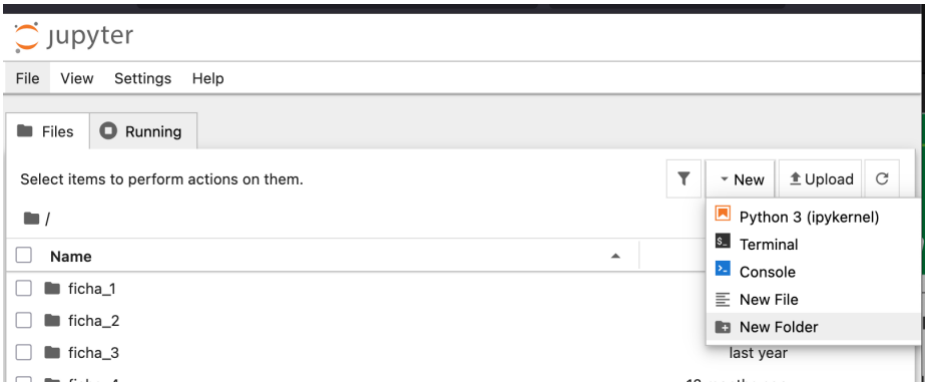
- **Windows:** python -m notebook
- **MacOS:** python3 -m notebook

Exemplo da aplicação a correr:



Por defeito, o servidor vai abrir na pasta do projeto que estiver a trabalhar.

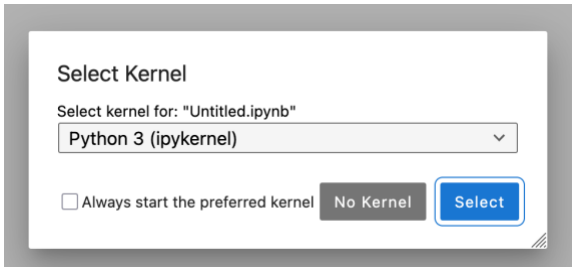
Para trabalhar nos projetos da aplicação, vamos criar uma pasta dedicada para isso, bastando fazer os seguintes passos:



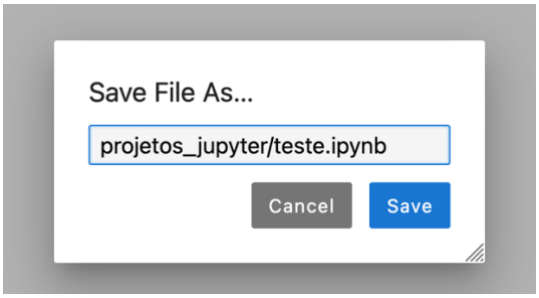
1. Vamos dar o nome da pasta: **projetos_jupyter**

2. Entre dentro da pasta e crie o primeiro ficheiro jupyter:

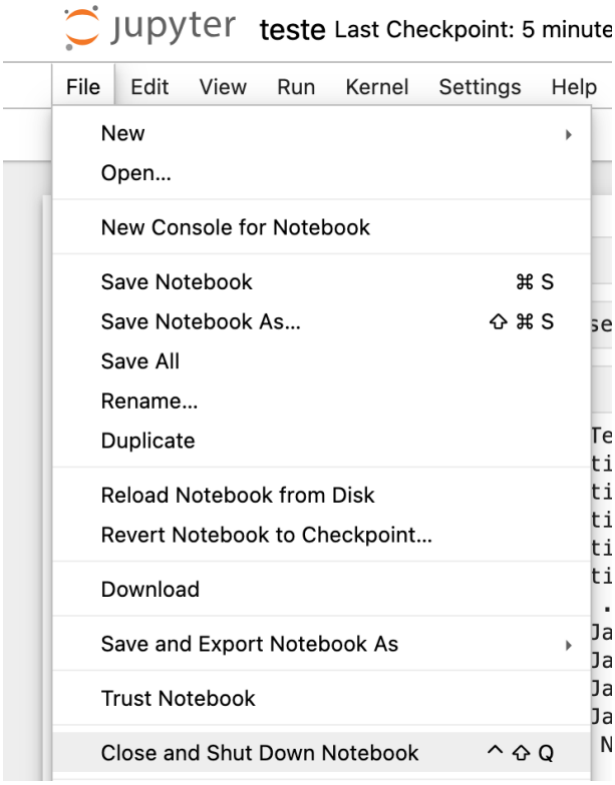
3. Confirmar o kernel que vem por defeito:



4. Criar um novo ficheiro:



Para desligar o servidor da aplicação jupyter basta selecionar o menu File e escolher a opção “Close and Shut Down Notebook”:



Parte 3 – Introdução ao Pandas

Parte 3.1 – Series VS Dataframes

Dentro do Pandas, temos dois objetos primários importantes:

- **Series**: são objetos de tipo vetor unidimensional, identificados pelo índice (index), que é responsável por identificar cada registo.

```
0      1.4
1      1.4
2      1.3
3      1.5
4      1.4
      ...
145    5.2
146    5.0
147    5.2
148    5.4
149    5.1
Name: PetalLengthCm, Length: 150, dtype: float64
```

- **DataFrames**: são objetos bidimensionais, de tamanho variável.

  O seu formato é de uma tabela, onde os dados são organizados em linhas e colunas. Além disso, podemos pensar a Series como uma única coluna e o DataFrame seria uma união de várias Series sob um mesmo índice.

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica

Parte 3.2 – Ficheiros para trabalhar com a biblioteca Pandas

Podemos trabalhar com a criação de cada uma dessas estruturas usando os métodos do Pandas (pandas.DataFrame e pandas.Series) sobre estruturas nativas do Python (como listas, arrays e dicionários).

A biblioteca Pandas vem com um conjunto de funcionalidades de alto nível para ler e escrever informação de vários tipos de ficheiros. Vejamos alguns:

Funções de leitura e escrita da biblioteca Pandas

Tipo de Dados	Formato	Função de Leitura	Função de Escrita
Texto	CSV	`read_csv`	`to_csv`
	TXT de largura fixa	`read_fwf`	–
	JSON	`read_json`	`to_json`
	HTML	`read_html`	`to_html`
	LaTeX	–	`to_latex`
	XML	`read_xml`	`to_xml`
	clipboard	`read_clipboard`	`to_clipboard`

Tipo de Dados	Formato	Função de Leitura	Função de Escrita
Binário	Excel	`read_excel`	`to_excel`
	ODF	`read_excel`	–
	HDF5	`read_hdf`	`to_hdf`
	Feather	`read_feather`	`to_feather`
	Parquet	`read_parquet`	`to_parquet`
	ORC	`read_orc`	`to_orc`
	Stata	`read_stata`	`to_stata`
	SAS	`read_sas`	–
	SPSS	`read_spss`	–
	Python Pickle	`read_pickle`	`to_pickle`
SQL	SQL	`read_sql`	`to_sql`
	Google BigQuery	`read_gbq`	`to_gbq`

Parte 3.3 – Importação dos ficheiros, Delimitadores e Localizações dos ficheiros

Passo 1: Para começar a trabalhar com o pandas, deve sempre importar o pacote Python pandas, conforme mostrado abaixo. Ao importar a biblioteca do pandas, o alias (acrónimo) mais comum para pandas é o pd.

```
1   #importar a biblioteca pandas
2   import pandas as pd
```

Passo 2: Descarregue o ficheiro diabetes.csv (que está junto com a documentação). Use a função ***read_csv()*** com o caminho da localização do ficheiro CSV para ler um arquivo de valores separados por vírgula:

```
4   #ler o ficheiro na localizacao
5   data = pd.read_csv("diabetes.csv")
```

Esta função vai carregar o arquivo CSV diabetes.csv para gerar um objeto Dataframe do pandas df.

Nota Importante:

Nos ficheiros CSV para separar a informação das colunas é utilizado o símbolo , (virgula), como podem ver mais abaixo:

```
pandas > diabetes.csv
1   Pregnancies,Glucose,BloodPressure,SkinThick
2   6,148,72,35,0,33.6,0.627,50,1
3   1,85,66,29,0,26.6,0.351,31,0
4   8,183,64,0,0,23.3,0.672,32,1
5   1,89,66,23,94,28.1,0.167,21,0
```

Passo 3: A leitura de arquivos de texto é semelhante à de arquivos CSV.

A única diferença entre os vários ficheiros é que precisa especificar qual é o Símbolo que separada a informação das colunas com o **parâmetro sep.**

Como tal, temos de complementar o código anterior e indicar qual o símbolo que separa as informações (ficheiros CSV pode ser o símbolo , (virgula) ou ; (ponto e vírgula ou outro qualquer símbolo delimitador):

```
4   #ler o ficheiro na localizacao
5   data = pd.read_csv("diabetes.csv", sep=",")
```

Nota Importante 1: Delimitadores

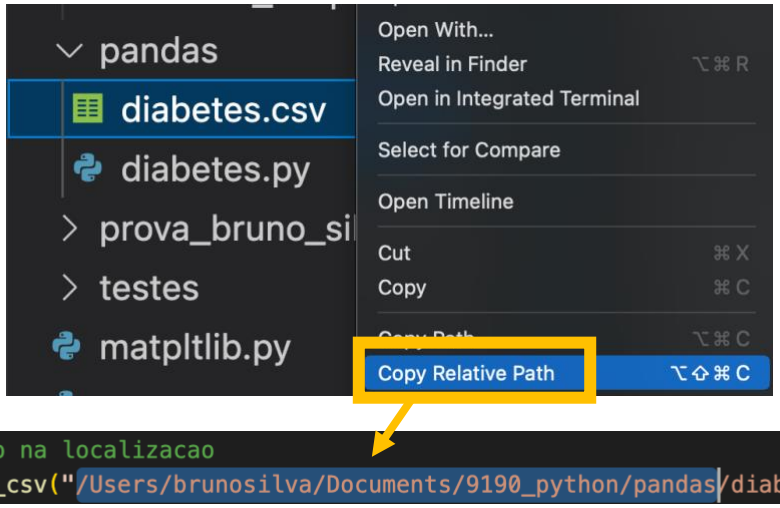
O argumento separador refere-se ao símbolo usado para separar as linhas em um DataFrame.

Vírgula (sep = ","), Ponto e Vírgula (sep = ";"), espaço em branco (sep = "\s"), tabulação (sep = "\t") e dois pontos (sep = ":") são os separadores comumente usados. Aqui, \s representa um único caractere de espaço em branco.

Nota Importante 2: Caminho do ficheiro (Localizações do sistema operativo)

Por vezes o sistema operativo pode ter dificuldades para descobrir o caminho relativo até ao ficheiro.

Após colocar o ficheiro dentro da pasta onde está a trabalhar, clique no lado direito do rato em cima do ficheiro onde estão os dados (neste caso o ficheiro csv) e selecione a opção "Copiar caminho relativo" e cole antes do nome do ficheiro:




```
4   #ler o ficheiro na localizacao
5   data = pd.read_csv("/Users/brunosilva/Documents/9190_python/pandas/diabetes.csv", sep=",")
```