**Laboratório Virtual - Atividade 1**

|  |  |
| --- | --- |
| **Disciplina** | **RDP – Reconhecimento de Padrões** |

**Objetivos**

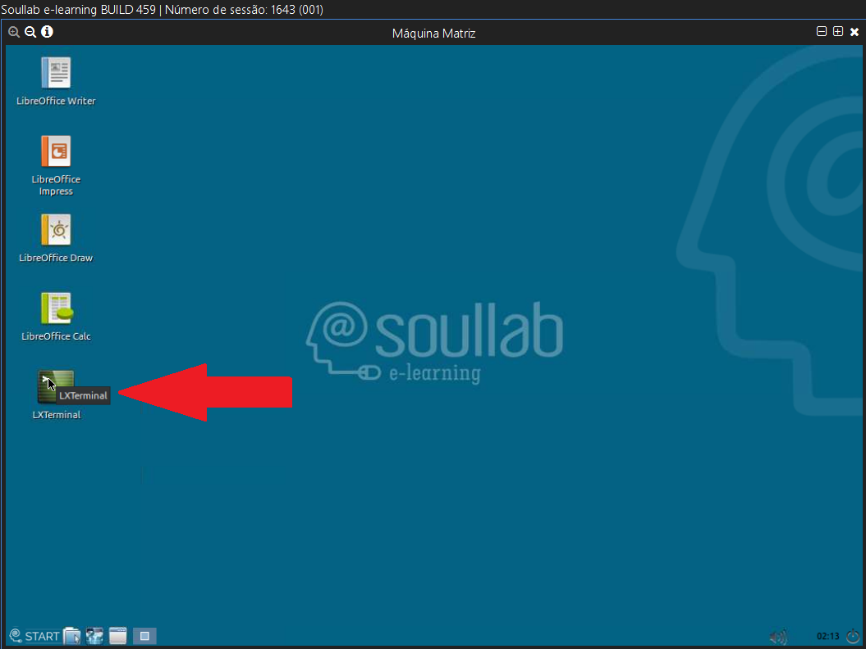
A primeira atividade prática em laboratório virtual possui como objetivos principais:

* Propiciar ao aluno a oportunidade de conhecer o laboratório virtual;
* Iniciar os serviços do Enthopy Canopy;
* Apresentar os principais conceitos do Jupyter;
* Executar um programa básico exemplo;
* Verificar os resultados e plotar alguns valores;

Ao final desta atividade, o aluno deverá ser capaz de executar um programa básico no Enthought Canopy e o no Jupyter utilizando a linguagem de programação Python.

**Acessando o ambiente virtual**

No momento de iniciar esta atividade é esperado que o aluno já tenha recebido seus dados de login para acesso ao laboratório virtual. Após executar o login no laboratório virtual, inicie o Terminal do Linux, conforme a imagem abaixo:



Foi criado, no ambiente virtual uma instância do Enthought Canopy[[1]](#footnote-1). O Canopy é um ambiente de Data Science criado para facilitar a produção de algoritmos de machine learning, deep learning, data mining utilizando ferramentas Python. Ele facilita a utilização de várias ferramentas uteis para cientistas de dados

**Inicializando o Canopy**

Como o ambiente já se encontra instalado então basta inicializa-lo. Para isso basta entrar no terminal e digitar

~/Canopy/canopy

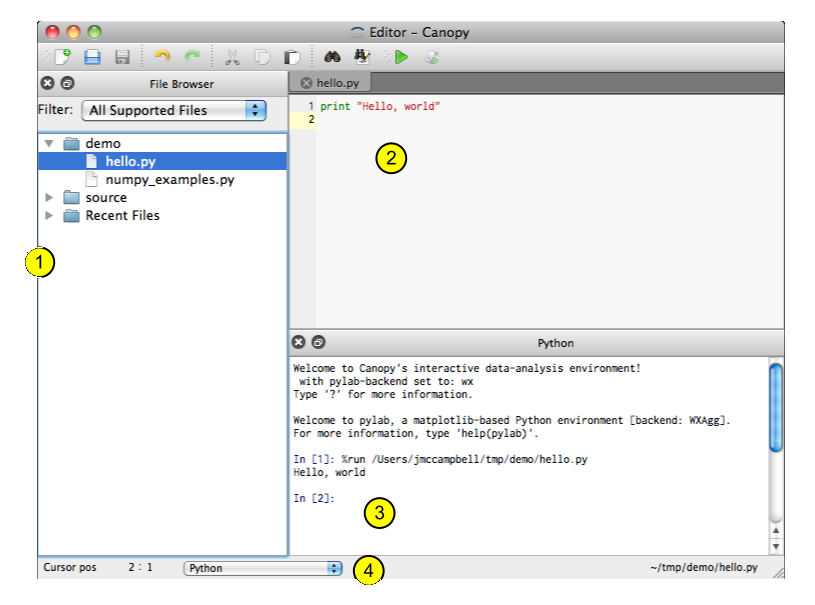
Isso abrirá a janela do Editor. O editor é formado de quatro janelas:

**1.** **Painel do navegador de arquivos:** mostra um ou mais diretórios e arquivos abertos recentemente. Clique duas vezes em um arquivo para abri-lo no Editor de Códigos.

**2.** **Editor de código:** um editor de texto de uso geral com recursos adicionais especificamente para edição de código Python.

**3. Painel Python:** integra um prompt IPython (Interactive Python) que permite testar código rapidamente, experimentar ideias e executar código diretamente do editor.

**4. Barra de status do editor:** mostra informações sobre o arquivo atualmente exibido no editor de código: linha e coluna (1 e 22, respectivamente, na imagem acima), tipo de arquivo e caminho e nome do arquivo.



**Criando nosso primeiro projeto[[2]](#footnote-2)**

O Jupyter Notebook

O Jupyter Notebook é uma iniciativa que veio para facilitar o processo de confecção de programas em Python de forma interativa. Através de uma interface simples, a aplicação permite que você divida seu código em blocos independentes chamados células — do inglês Cell — e até mesmo inclua elementos HTML em sua apresentação.

**Passo 1.** Primeiro, inicie o Jupyter Notebook. Você pode acessá-lo através do menu iniciar, ou ao digitar “jupyter notebook” no prompt de comando. Feito isso, será aberta uma janela no seu navegador padrão, mas você poderá acessar a aplicação de qualquer browser através do endereço: “localhost:8888/tree”;

**Passo 2.** Após acessar o Jupyter Notebook, você deve notar que a estrutura de pastas criadas é semelhante ao encontrado em seu Window Explorer. Escolha ou crie uma para salvar seus projetos;

**Passo 3.** Feito isso, selecione o botão “New”. Dentre as opções mostradas, você poderá criar uma pasta (Folder), arquivo de texto (Text file) ou um Notebook. Selecione “Python3". Uma nova janela será aberta com seu novo Notebook.

Quando você cria o seu notebook, uma tela semelhante a Figura 2 é mostrada. Inicialmente, a interface do Jupyter Notebook é composta pelos seguintes elementos:

1. Nome do notebook e a informação de quando foi salvo pela última vez. Um notebook é qualquer arquivo de apresentação criado no Jupyter com a extensão .ipynb.
2. Barra de menus. Permite acesso às principais funções da ferramenta. Nela você poderá criar novas células
3. Barra de botões. Atalhos para algumas das rotinas mais utilizadas do Jupyter.
4. Célula — ou Cell, em inglês. É aqui que construimos nosso programa.

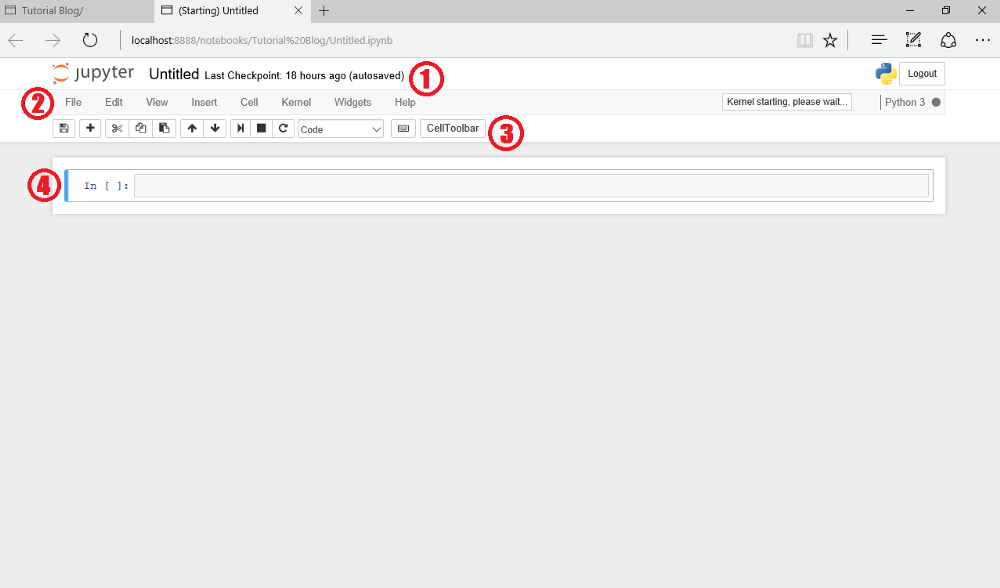


Figura 2 – Jupyter Notebook

**Entendo o nosso dataset**

Aqui começa a nossa jornada. Temos nosso primeiro desafio: entender nosso dataset. Independentemente do projeto, é crucial que você saiba que tipo de dados está usando. Um bom início é entender o que cada coluna do seu arquivo significa. Abaixo você pode conferir as colunas encontradas no arquivo “vgsales.csv”:

* Rank: posição no ranking de vendas
* Name: nome do jogo
* Platform: plataforma em que o jogo foi liberado (PC, PS4, Xbox, etc.)
* Year: ano de lançamento do game
* Genre: gênero do jogo
* Publisher: empresa que publicou o jogo
* NA\_Sales: vendas na América do Norte (em milhões de dólares)
* EU\_Sales: vendas na Europa (em milhões de dólares)
* JP\_Sales: vendas no Japão (em milhões de dólares)
* Other\_Sales: vendas no restante do mundo (em milhões de dólares)
* Global\_Sales: total de vendas no mundo inteiro

**Estrutura do Projeto**

Nosso projeto está estruturado em duas pastas. A primeira pasta (data) contém os dados que utilizaremos nessa prática. A segunda pasta contém o arquivo de código-fonte que precisamos utilizar (src).

projeto

└ data

└ vgsales.csv

└ src

└ VideoGames.ipynb

**Importando as Bibliotecas**

Para começar nosso projeto precisamos identificar quais as bibliotecas utilizaremos. Para come

|  |
| --- |
| %matplotlib inline import pandas as pd import matplotlib.pyplot as plt |

A linha %matplotlib inline serve para fazer com que os gráficos gerados por nosso código sejam mostrados dentro do notebook, e não em uma nova janela — como é feito caso essa cláusula não seja adicionada. O comando import faz com que o Python importe alguma biblioteca para ser utilizada em seu código. O nome da biblioteca deve surgir após o import. Em nosso caso, pandas e matplotlib são as bibliotecas que utilizaremos.

Provavelmente você percebeu a palavra as depois do nome da biblioteca. Esse comando adiciona um “apelido” a ela, para que possa ser referenciada depois. Nesse caso, ao precisarmos utilizar uma função do pandas, por exemplo, ao invés de digitarmos pandas.nome\_funcao() podemos digitar pd.nome\_funcao().

**Leitura do dataset**

Após a importação das bibliotecas, criaremos uma nova célula para realizar a leitura do nosso dataset. Utilizando a função pd.read\_csv('local\_do\_arquivo.extensao') carregamos o dataset para a memória e o armazenamos dentro da variável “games”.

O próximo passo do nosso código é utilizar a função head() para visualizarmos como nossa tabela é apresentada pelo pandas. A função head() irá exibir X quantidades de linhas iniciais de nosso dataset. Se nenhum valor for colocado entre os parênteses, o pandas retornará as cinco primeiras linhas da tabela.

|  |
| --- |
| gamedata = pd.read\_csv('../data/vgsales.csv')  gamedata.head(10) |

Todo código do nosso primeiro será feito em um arquivo. Nesse primeiro programa precisaremos que o arquivo cap1-tweets.txt seja colocado dentro do diretório /data logo abaixo do diretório padrão do projeto.

**Renomeando as colunas**

Outra coisa interessante que podemos fazer em uma apresentação é mostrar dados em um idioma que a audiência compreenda. Imagine que você irá mostrar os resultados de seu trabalho para pessoas que não entendem inglês. Eles não teriam ideia do que significa Genre, por exemplo.

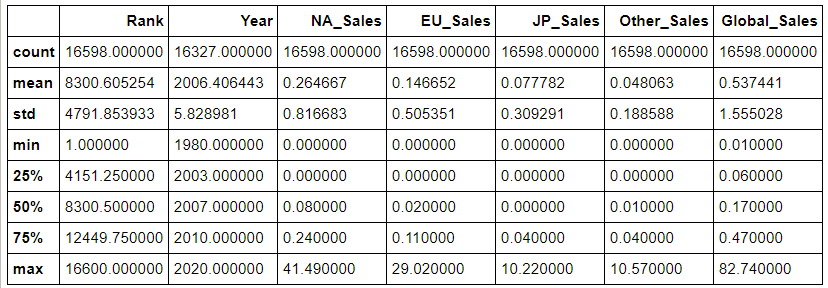
|  |
| --- |
| gamedata.columns = ['Ranking', 'Nome', 'Plataforma', 'Ano',  'Gênero', 'Editora', 'Vendas América do Norte', 'Vendas EUA', 'Vendas Japão', 'Outras vendas', 'Vendas Global']  gamedata.head(10) |

Só que mais uma vez o pandas pode nos ajudar nisso. Todo Dataframe possui o atributo columns, que nada mais é do que o nome das colunas da tabela. Para alterar o nome delas, basta informar os novos em uma lista do Python e armazená-los como se fosse uma variável.

**Gerando nossas primeiras estatísticas**

O pandas permite a criação de resumos estatísticos dos dados em memória. Para termos um resumo dos dados podemos utilizar a função describe. Essa função permite que se realize o cálculo da média, desvio padrão, mínimos, máximos e os quartis de cada uma das colunas.

|  |
| --- |
| gamedata.describe() |



Se quisermos apenas algumas linhas e colunas do nosso dataset podemos identifica-las através de colchetes. Se quisermos acessar uma coluna podemos identifica-la pelo seu nome.

|  |
| --- |
| gamedata['Nome'] |

Se quisermos identificar uma faixa dos nossos dados basta colocar, dentro do colchete, o número inicial e final desejado separados por ':'.

|  |
| --- |
| gamedata['Ranking'][2:10] |

**Tratando dados nulos**

Quando estamos realizando análises em dados, frequentemente encontraremos a necessidade de lidar com valores nulos. Tratam-se de campos que não tiveram um valor informado e, para que não fique um espaço vazio no seu DataFrame e, consequentemente gere um erro, o pandas insere o valor NaN — significa Not a Number, ou Não é Número, em tradução livre. Utilizando a função isnull() do pandas nos permite trabalhar com valores nulos encontrados em nosso DataFrame. Para exemplificar, no nosso caso filtraremos todos os registros que possuem o campo Ano nulo.

|  |
| --- |
| nullrows = gamedata['Ano'].isnull()  gamedata[nullrows].head() |

**Agregando dados**

O pandas possui várias funções que permite a agregação de dados. Podemos, por exemplo realizar a contagem de dados de uma determinada coluna. Para isso, utilizamos a função value\_counts() à partir do filtro da coluna aplicado sobre o dataset. Imagine que você deseja saber rapidamente quantos jogos foram lançados para cada plataforma. Para isso:

|  |
| --- |
| gamedata['Plataforma'].value\_counts() |

Se quisermos, por exemplo, somar todos os valores de uma determinada coluna, basta invocar a função sum() identificando qual a coluna desejada. No caso, estamos somando todas as Vendas na América do Norte.

|  |
| --- |
| gamedata['Vendas América do Norte'].sum() |

**Ordenando dados**

Se desejamos ordenar os dados, basta informar os nomes das colunas as quais gostaríamos de ordenar. Nesse caso, a ordem com que as colunas são informadas influenciam na ordenação: a ordenação é feita da primeira para a última coluna.

|  |
| --- |
| gamedata .sort\_values(['Nome',"Ano"]) |

**Criando gráficos**

Se desejamos criar gráficos o Python possui uma biblioteca chamada pyplot[[3]](#footnote-3). Nas próximas atividades verificaremos mais aprofundadamente cada uma das formas de se realizar isso. No entanto, o pandas possui uma forma simples de visualizar os dados com a função plot().

Após a realização de uma agregação ou ordenação basta adicionar a chamada da função para que se crie uma visualização dos dados.

|  |
| --- |
| gamedata['Plataforma'].value\_counts().plot() |

1. O ambiente Enthought Canopy (<https://www.enthought.com/product/canopy/>) e sua instalação de uma nova instância em ambiente Linux pode ser consultada na página <http://docs.enthought.com/canopy/quick-start/install_linux.html> [↑](#footnote-ref-1)
2. Prática baseada no artigo feito por Paulo Vasconcellos que pode ser vista em: <https://paulovasconcellos.com.br/como-criar-seu-primeiro-projeto-de-data-science-parte-2-de-2-cb9a2fe05eff> [↑](#footnote-ref-2)
3. A biblioteca PyPlot pode ser vista no link: <https://matplotlib.org/api/pyplot_api.html> [↑](#footnote-ref-3)