BIG DATA

LABDATA

Introdução ao Big Data

Tema da Aula: **Big Data com Python**

Prof.: **Dino Magri**

Data: **25 de Julho de 2016**

Coordenação:

Prof. Dr. Adolpho Walter Pimazzi Canton

Profa. Dra. Alessandra de Ávila Montini

LABDATA

Coordenação:

Prof. Dr. Adolpho Walter Pimazzi Canton

Profa. Dra. Alessandra de Ávila Montini

- Contatos:
  - E-mail: professor.dinomagri@gmail.com
  - Twitter: https://twitter.com/prof_dinomagri
  - LinkedIn:http://www.linkedin.com/in/dinomagri
  - Site: http://www.dinomagri.com

Currículo

- **(2014-Presente)** – Professor no curso de Extensão, Pós e MBA na Fundação Instituto de Administração (FIA) – www.fia.com.br
- **(2013-Presente)** – Pesquisa e Desenvolvimento no Laboratório de Arquitetura e Redes de Computadores (LARC) na Universidade de São Paulo – www.larc.usp.br
- **(2013)** – Professor no MBA em Desenvolvimento de Inovações Tecnológicas para WEB na IMED Passo Fundo – RS – www.imed.edu.br
- **(2012)** – Bacharel em Ciência da Computação pela Universidade do Estado de Santa Catarina (UDESC) – www.cct.udesc.br
- **(2009/2010)** – Pesquisador e Desenvolvedor no Centro de Computação Gráfica – Guimarães – Portugal – www.ccg.pt
- **Lattes:** http://lattes.cnpq.br/5673884504184733

Material das aulas

- Material das aulas:
  - http://urls.dinomagri.com/posmba-turma2/
  - Senha: fia2016
- Faça o download do arquivo **2016-07-25-py-aula3.zip**
- Salve na Área de Trabalho (Desktop)
- Depois que finalizar o download, acesse a pasta Área de trabalho e descompacte o arquivo 2016-07-25-py-aula3.zip.

Conteúdo da Aula

- Objetivo
- Introdução
- Básico sobre Pandas
- Exemplo prático
- Exercícios

Conteúdo da Aula

- **Objetivo**
- Introdução
- Básico sobre Pandas
- Exemplo prático
- Exercícios

Objetivo

- O objetivo dessa aula é introduzir os conceitos básicos sobre as bibliotecas **NumPy** e **Pandas** que servem de base para a **manipulação** e **exploração** dos dados.

Conteúdo da Aula

- Objetivo
- **Introdução**
- Básico sobre Pandas
- Exemplo prático
- Exercícios

Introdução

- Pandas é uma biblioteca de código aberto para análise de dados em Python.
- Foi desenvolvido em 2008 por Wes McKinney.
  - Tem uma grande comunidade - http://pandas.pydata.org/community.html
  - Melhorias continuas - https://github.com/pydata/pandas/
  - Diversas funcionalidades
  - Iteração rápida
- Essa biblioteca com o tempo teve uma **ótima adoção**, se tornando a **biblioteca padrão para análise de dados** utilizando Python.

Introdução

- Por que utilizar Python e Pandas para Big Data?
  - Linguagem **interpretada** ao invés de compilada.
  - Tipagem dinâmica.
  - Capacidade modulares.
  - Extensível à outras linguagens de programação.
  - **Orientação à Objetos**.
  - Maioria dos paradigmas de programação: estrutural, OO, e funcional.
  - Além é claro, de ter **inúmeras bibliotecas Python**.

Introdução

- Por fim, Python tem um conjunto de bibliotecas que fornecem um completo kit para Ciência de Dados e análises. As principais:
  - `NumPy` – Array de proposito geral para computação numérica.
  - `SciPy` – Computação numérica.
  - `Matplotlib` - Gráficos
  - **`Pandas` – DataFrame e Series (tipos parecidos com arrays de 2D e 1D)**
  - `Scikit-learn` – Aprendizado de máquina.
  - `NLTK` – Processamento de linguagem natural.
  - `Statstool` – Análise estatística.

Introdução

- Principais características do Pandas
  - É possível **processar diversos conjuntos de dados em diferentes formatos** (Series temporais, dados tabulares heterogêneos, e matrizes)
  - Facilidade de **importar dados** de diversas fontes como CSV, DB/SQL.
  - Podemos lidar com **diversas operações** nesses conjuntos de dados: filtragem, agrupamento, reordenamento, remodelação, junção, fatiamento, entre outros.
  - Facilita trabalhar com **dados** que estão **faltando**.
  - Tem uma boa integração com outras bibliotecas Python, como a statsmodels, SciPy, e scikit-learn.

Introdução

- Benefícios de utilizar pandas
  - Representação dos dados
  - Filtragem e fatiamento dos dados
  - Código limpo e conciso
  - **Bom desempenho**, especialmente quando faz computação numérica, afinal ele foi construído em cima da biblioteca NumPy.

Introdução

- Instalação
  - `pip install pandas`
    - `pytz` – biblioteca para calculo de `timezone`.
    - `numpy` – processamento de array numéricos.
    - `python-dateutil` – fornece extensão para o módulo de `datetime`.
    - `six` – fornece funções que permitem diminuir as diferenças entre as versões do Python 2 e 3.

Introdução

- Como vimos NumPy é um pacote para computação científica com Python. Tem as seguintes características:
  - Objeto para array de **N-dimensões**
  - Funções sofisticadas
  - Ferramentas para integrar código C/C++ e Fortran
  - Capacidades para **álgebra linear**, **números randômicos**, entre outros.

Introdução

- Vamos utilizar NumPy para:
  - Criar arrays de 1D e 2D
  - Produzir números aleatórios
  - Utilizar algumas funções matemáticas disponíveis.

Introdução

- Primeiro, precisamos importar a biblioteca

```
>>> import numpy as np
```

Introdução

- Agora vamos criar um Array de 1 dimensão.

```
>>> ar1 = np.array([0,1,2,3]) # Array 1D

>>> print(ar1)

>>> print(type(ar1))

[0 1 2 3]

<class 'numpy.ndarray'>
```

Introdução

- Agora vamos criar um Array de 2 dimensões.

```
>>> ar2 = np.array([0,1,2,3], [2,8,7]) # Array 1D

>>> print(ar2)

>>> print(type(ar2))

[[0 3 5]

 [2 8 7]]

<class 'numpy.ndarray'>
```

Introdução

- Para visualizar o formato do array

```
>>> print(ar2.shape)

(2, 3)
```

- Para visualizar o número de dimensões

```
>>> print(ar2.ndim)

2
```

Abra o arquivo "aula3-parte1-numpy.ipynb"

Conteúdo da Aula

- Objetivo
- Introdução
- **Básico sobre Pandas**
- Exemplo prático
- Exercícios

Pandas

- Conforme vimos Pandas foi construído em cima do NumPy e ele fornece diversas outras funcionalidades que não estão disponíveis no NumPy.
- **Permite criar estruturas de dados de fácil entendimento e que são rápidas**.
- Desta forma possibilita preencher a lacuna que existia entre Python e linguagem como R.

Pandas

- As 3 principais estruturas de dados no Pandas:
  - **Series**, **DataFrame** e Panel (não será abordado!)

- Para fazer uso dessas estruturas, primeiro precisamos importar a biblioteca.

```
>>> import pandas as pd
```

Pandas - Series

- **Series** é na verdade um array NumPy de 1 dimensão **com rótulos**.
- Podemos criar Series da seguinte maneira:

```
>>> s = pd.Series(dados)
```

- Onde, `dados` pode ser um dos itens abaixo:
  - Um `numpy.ndarray`
  - Um `dicionário`
  - Um `valor escalar`

Pandas - Series

- Além da criação, podemos realizar operações como **fatiamento** (slicing), **atribuições**, aplicar funções aritméticas e estatísticas, entre tantos outros.

Abra o arquivo "aula3-parte2-series.ipynb"

Pandas - DataFrame

- **DataFrame** é um array 2D com rótulos nas colunas e nas linhas.
- Tem as seguintes características:
  - Conceitualmente é semelhante a uma tabela ou planilha de dados.
  - Colunas podem ser de diferentes tipos: `float64, int, bool`.
  - Uma coluna do DataFrame é uma Series.
  - Podemos pensar que é um dicionário de Series, onde as colunas e linhas são indexadas, denota `index` no caso da linhas e `columns` no caso de colunas.
  - Seu **tamanho** é **mutável**: colunas podem ser inseridas e deletadas

Pandas - DataFrame

- **DataFrame** é um array 2D com rótulos nas colunas e nas linhas.
- Cada eixo do DataFrame (e também da Series) tem índices, seja o definido ou padrão.
- Os índices são necessários para acesso rápido aos dados.

Pandas - DataFrame

- Podemos criar DataFrame da seguinte maneira:

```
>>> df = pd.DataFrame(dados)
```

- Onde, `dados` pode ser:
  - Dicionário de ndarrays de 1D, listas, dicionários, ou Series
  - Array 2D do NumPy
  - Estruturado
  - Series
  - Outra estrutura DataFrame

Pandas - DataFrame

- Podemos realizar inúmeras operações, como **seleção**, **atribuição**, **remoção**, alinhamento, aplicar funções aritméticas e estatísticas entre outros.

Abra o arquivo "aula3-parte3-dataframe.ipynb"

Outras operações

- Conforme vimos o fatiamento e a indexação podem ser um pouco confusos.
  - Por exemplo, se uma Series tem um índice explicito de inteiros, uma operação como `s1[1]` irá utilizar o índice **explícito**, enquanto que uma operação de fatiamento como `s1[1:3]` irá utilizar o índice **implícito**, no mesmo estilo de Python. Exemplo:

```
>>> s1 = pd.Series(['a', 'b', 'c'], index=[1,3,5])
1    a
3    b
5    c
dtype: object
```

Outras operações

- Utilizando o índice **explicito** quando se está indexando irá produzir o resultado:

```
>>> s1[1]
'a'
```

- Utilizando o índice **implícito** quando se está fatiando irá produzir o resultado:

```
>>> s1[1]
3    b
5    c
dtype: object
```

Outras operações

- Como podemos perceber isso pode ser um pouco confuso no caso de índices de números inteiros.
- Por isso, Pandas fornece alguns **indexadores especiais** que explicitamente contém esquemas de acesso aos índices.
- Eles não são métodos funcionais e sim **atributos que expõe uma interface de fatiamento particular para o dados**.
- São eles: `loc`, `iloc`

Outras operações

- O atributo `loc` permite indexar e fatiar sempre utilizando o índice explícito.

```
>>> s1
1    a
3    b
5    c
>>> s1.loc[1]
'a'
>>> s1.loc[1:3]
1    a
3    b
```

Outras operações

- O atributo `loc` permite indexar e fatiar sempre utilizando o índice implícito.

```
>>> s1
1    a
3    b
5    c
>>> s1.iloc[1]
'b'
>>> s1.iloc[1:3]
3    b
5    c
```

Abra o arquivo "aula3-parte3-dataframe.ipynb"

Conteúdo da Aula

- Objetivo
- Introdução
- Básico sobre Pandas
- **Exemplo prático**
- Exercícios

Exemplo prático

- Iremos utilizar o arquivo `capitais.csv` que é um arquivo que tem todas as capitais do Brasil, bem como a população e a área de cada capital (km2).
- O Pandas disponibiliza diversos métodos para carregar diferentes tipos de dados, segue alguns deles:
  - `pd.read_csv('caminho-ate-arquivo.csv', sep=';')`
  - `pd.read_excel('caminho-ate-arquivo.xlsx', 'Sheet1')`
  - `sql.read_frame(query, connection)` – necessita do módulo pandas.io

Abra o arquivo "aula3-parte4-exemplo-pratico.ipynb"

Conteúdo da Aula

- Objetivo
- Introdução
- Básico sobre Pandas
- Exemplo prático
- **Exercícios**

Exercícios

Utilizando o dataframe `capitais` criado anteriormente, faça os seguintes exercícios:

- **Exercício 1** - Selecione todas as capitais que tenham área maior que 400 km2. Quantas foram?
- **Exercício 2** - Selecione as capitais que tenham população maior que 2 milhões.

Exercícios

- **Exercício 3** - Crie uma função que retorne uma lista contendo somente as capitais que começam com uma determinada letra. A função deve receber dois parâmetros:
  - O primeiro parâmetro será uma lista com **todas as capitais**.
  - O segundo será uma **letra**.
- Para testar a função, selecione as capitais que começam com as letras **B** e **Z**. Lembre-se de tratar possíveis erros.

```
>>> def capitais_com_letra(todas_capitais, letra):
```

Exercícios

- **Exercício 4** - Utilizando a função criada no exercício 3, calcule o total da população para as capitais que começam com **S**. Por fim, imprima a seguinte frase:

```
As capitais X, Y e Z tem W pessoas.
```

Referências Bibliográficas

- **Mastering pandas** – Femi Anthony – Packt Publishing, 2015.

- **Python for Data Analysis** – Wes McKinney – USA: O'Reilly, 2013.

- Tutoriais disponíveis no site oficial do Pandas - http://pandas.pydata.org/pandas-docs/version/0.18.0/tutorials.html

- Livro de receitas disponíveis no site oficial do Pandas - http://pandas.pydata.org/pandas-docs/version/0.18.0/cookbook.html

Referências Bibliográficas

- **Python for kids – A playful Introduction to programming** – Jason R. Briggs – San Francisco – CA: No Starch Press, 2013.

- **Python Cookbook** – David Beazley & Brian K. Jones – O'Reilly, 3th Edition, 2013.

- As referências de links utilizados podem ser visualizados em http://urls.dinomagri.com/refs