Programação Python

Aula 08: Series (pandas)

Prof. Eduardo Corrêa Gonçalves

29/03/2021

Sumário

Introdução (1/4)

- **O que é pandas?**
  - Python Data Analysis Library – biblioteca para ciência de dados mais utilizada no Python.
  - Oferece suporte para atividades de Data Wrangling:
    - seleção;
    - exploração;
    - integração;
    - limpeza; e
    - transformação de dados.
  - Acrescenta duas EDs ao Python:
    - Series
    - DataFrame.

Fonte: https://www.codeastar.com/data-wrangling/

Introdução (2/4)

- **O que é Series?**
  - ED composta por um vetor de dados e um vetor associado de rótulos (índice)

  - Mistura de lista com dicionário!
    - Como uma lista, armazena uma sequência ordenada de elementos.
    - Como um dicionário, permite que elemento seja acessado pelo rótulo.
  - Características:
    - Permite elementos **duplicados**.
    - **Mutável** – pode ser alterada.
    - **Iterável** – capaz de retornar seus elementos um por vez em um laço.
    - **Sequência** – elementos possuem ordem determinada.
    - **Eficiente** – vetor de dados é um array NumPy

Introdução (3/4)

- **Importação da Biblioteca pandas**
  - A ‘pandas’ **não** faz parte do Python padrão.
    - No entanto, está disponível no Google Colab
    - E em todas distribuições do Python voltadas para computação científica (ex.: WinPython e Anaconda).
  - Para utilizá-la, você precisa usar o comando **import** (além de importar a biblioteca, a renomeamos para “pd”):

```
import pandas as pd.
```

    - *Nem sempre vou mostrar esse import nos slides para poupar espaço.*
    - *Mas você precisa iseri-lo em seus programas.*

Introdução (4/4)

- Há **2 tipos** de Series:

  - **Series sem rótulos (rótulos são inteiros)**

[0]	[1]	[2]	[3]	[4]	
7.6	5.0	8.5	9.5	6.4	*s_notas*

  - **Series com rótulos**

[M02]	[M05]	[M13]	[M14]	[M19]	
Bob	Dayse	Bill	Cris	Jimi	*s_alunos*

[10/02/2021]	[11/02/2021]	[12/02/2021]	[13/02/2021]	[14/02/2021]	[15/02/2021]	
31	35	34	28	27	27	*s_temperaturas*

Criação de Series (1/2)

- **Criando Series**
  - Utiliza-se o construtor pd.Series()
  - Series sem rótulos: basta definir uma lista com os dados.
  - Series com rótulos: define-se lista com rótulos + lista com dados

```
import pandas as pd

# Series não rotulada
s_notas = pd.Series([7.6, 5.0, 8.5, 9.5, 6.4])

print(s_notas)


# Series rotulada
lst_matriculas = ['M02','M05','M13','M14','M19']
lst_nomes = ['Bob','Dayse','Bill','Cris','Jimi']
s_alunos = pd.Series(lst_nomes, index=lst_matriculas)

print(s_alunos)
```

```
0    7.6
1    5.0
2    8.5
3    9.5
4    6.4
dtype: float64


M02      Bob
M05    Dayse
M13     Bill
M14     Cris
M19     Jimi
dtype: object
```

Criação de Series (2/2)

- **Outras formas de criar Series**
  - Importando arquivo (veremos na próxima aula).
  - A partir de um dicionário.

```
import pandas as pd

dic_alunos = {'M02':'Bob','M05':'Dayse','M13':'Bill', 'M14':'Cris','M19':'Jimi'}

s_alunos = pd.Series(dic_alunos)

print(s_alunos)
```

Propriedades das Series (1/3)

- **dtype:** tipo dos elementos do vetor de dados.

dtype	Utilização	Tipo Python
int64	números inteiros	int
float64	números reais	float
bool	True / False	bool
object	texto	str
datetime64	data/hora	datetime

- **values:** vetor de dados.
- **index:** vetor de rótulos.
- **name:** nome do vetor de dados.
- **size:** tamanho da Series (quantidade de elementos).
- **index.name:** nome do vetor de rótulos.
- **index.dtype:** dtype do vetor de dados.

Propriedades das Series (2/3)

- **Exemplo – Explorando as Propriedades**

```
s_alunos = pd.Series({'M02':'Bob','M05':'Dayse','M13':'Bill', 'M14':'Cris','M19':'Jimi'})

# Atribui nomes para os vetores de dados e rótulos
s_alunos.name = "alunos"
s_alunos.index.name = "matrículas"

# Imprime a Series e suas propriedades
print(s_alunos)
print("\n")
print("- número de elementos: ", s_alunos.size)
print("- vetor de dados: ", s_alunos.values)
print("- vetor de rótulos: ", s_alunos.index)
print("- tipo de s_alunos: ", type(s_alunos))
print("- dtype da Series: ", s_alunos.dtype)
print("- dtype do vetor de rótulos: ", s_alunos.index.dtype)
```

Propriedades das Series (3/3)

- **Saída do programa...**

```
matrículas
M02      Bob
M05    Dayse
M13     Bill
M14     Cris
M19     Jimi
Name: alunos, dtype: object

- número de elementos:  5
- vetor de dados:  ['Bob' 'Dayse' 'Bill' 'Cris' 'Jimi']
- vetor de rótulos:  Index(['M02', 'M05', 'M13', 'M14', 'M19'],
dtype='object', name='matrículas')
- tipo de s_alunos:  <class 'pandas.core.series.Series'>
- dtype da Series:  object
- dtype do vetor de rótulos:  object
```

Propriedades e Operações (1/8)

- **Operações básicas**:
  - Indexar (por índice ou rótulo)
  - Modificar (por índice ou rótulo)
  - Fatiamento
  - Verificar se rótulo ou valor pertence à Series
  - Verificar se valor *x* pertence à Series
  - Iterar (percorrer todos os elementos *em ordem*)
  - Ordenar
  - Estatísticas básicas
  - Computação Vetorizada
  - Indexação booleana

  - *Outras operações serão mostradas na aula sobre DataFrames*

Propriedades e Operações (2/8)

[pai]	[mae]	[enteado]	[filho]
John	Yoko	Julian	Sean

s

- **Recuperando elementos pelo índice ou rótulo**

```
primeiro = s[0]              # 'John'
ultimo = s[3]                # 'Sean'
ultimo_tambem = s[-1]        #também retorna 'Sean'
a = s["pai"]                 # 'John'
b = s["mae"]                 # 'Yoko'
```

Propriedades e Operações (3/8)

[pai]	[mae]	[enteado]	[filho]
John	Yoko	Julian	Sean

s

- **Modificando a Series**

```
s[1] = "Yoko Ono"            # ["John", "Yoko Ono", "Julian", "Sean"]

s["pai"] = "John Lennon"   # ["John Lennon", "Yoko Ono", "Julian", "Sean"]

s[["filho", "enteado"]] = ["Sean Lennon", "Julian Lennon"]
                          # ["John Lennon", "Yoko Ono", "Julian Lennon", "Sean Lennon"]

# remove o elemento de rótulo “enteado”
s = s.drop ("enteado")

# insere o elemento de rótulo “amigo”
s ["amigo"] = "Paul"
```

Propriedades e Operações (4/8)

[M02]	[M05]	[M13]	[M14]	[M19]
Bob	Dayse	Bill	Cris	Jimi

s_alunos

- **Fatiamento (retorna uma *sub-series*)**

Exemplo	Resultado
s_alunos[0:2]	[M02:Bob, M05:Dayse]
s_alunos [2:4]	[M13:Bill, M14:Cris]
s_alunos [:2]	[M02:Bob, M05:Dayse]
s_alunos [2:]	[M13:Bill, M14:Cris, M19:Jimi]
s_alunos [-2:]	[M14:Cris, M19:Jimi]
s_alunos [1:5:2]	[M05:Dayse, M14:Cris]
s_alunos [[2, 0, 4]]	[M13:Bill, M02:Bob, M19:Jimi]
s_alunos [['M13', 'M02', 'M19']]	[M13:Bill, M02:Bob, M19:Jimi]

Propriedades e Operações (5/8)

[M02]	[M05]	[M13]	[M14]	[M19]
Bob	Dayse	Bill	Cris	Jimi

s_alunos

- **Verificando se rótulo existe na Series**

```
"M13" in s_alunos        # True
"M99" in s_alunos        # False
```

- **Verificando se valor está na Series**

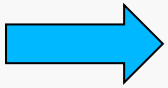
```
s_alunos.isin(['Bill']))     # [False, False, True, False, False]
```

  - *O método isin() retorna uma Series com dtype bool que terá o valor True para todas as posições associadas ao valor testado.*

Propriedades e Operações (6/8)

- **Iterando pelos valores**

```
for aluno in s_alunos:
    print(aluno)
```



```
Bob
Dayse
Bill
Cris
Jimi
```

- **Iterando com base nos índices**

```
for k in range(s_alunos.size):
    print("elemento {} = {}:{} ".format(k, s_alunos.index[k], s_alunos[k])
```

```
elemento 0 = M02:Bob
elemento 1 = M05:Dayse
elemento 2 = M13:Bill
elemento 3 = M14:Cris
elemento 4 = M19:Jimi
```

Propriedades e Operações (7/8)

[0]	[1]	[2]	[3]	[4]	
7.6	5.0	8.5	9.5	6.4	*s_notas*

- **Ordenação e Estatísticas Básicas**

```
# ordena ascendente pelos valores
#(use ascending=False para descendente)
s_notas = s_notas.sort_values()
```

```
3    9.5
2    8.5
0    7.6
4    6.4
1    5.0
dtype: float64
```

```
#Estatísticas básicas
print("média da turma: ", s_notas.mean());
print("desvio padrão: ", s_notas.std());
print(s_notas.describe());
```

```
média:  7.4
desvio padrão:  1.76210
count    5.000000
mean     7.400000
std      1.762101
min      5.000000
25%      6.400000
50%      7.600000
75%      8.500000
max      9.500000
dtype: float64
```

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.describe.html

Propriedades e Operações (8/8)

- **Computação Vetorizada**
  - Operações são realizadas sobre cada elemento da Series sem que seja possível implementar um laço.
  - Qualquer método/função da NumPy pode ser aplicada à Series

```
import numpy as np
import pandas as pd
s1 = pd.Series([2, 4, 6]); s2 = pd.Series([1, 3, 5])

print(s1 * s2)
```

```
0     2
1    12
2    30
dtype: int64
```

```
print(s1 + s2)
```

```
0     3
1     7
2    11
dtype: int64
```

```
#raiz quadrada dos elementos de s1
#usando a NumPy
print(np.sqrt(s1))
```

```
0    1.414214
1    2.000000
2    2.449490
dtype: float64
```

Indexação Booleana (1/1)

[M02]	[M05]	[M13]	[M14]	[M19]	
Bob	Dayse	Bill	Cris	Jimi	*s_alunos*

- **O que é indexação Booleana?**
  - Técnica para recuperar dados de uma Series utilizando um vetor de booleanos como entrada.
  - **Exemplo**:

```
s = s_alunos[[False,True,True,False,True]]

print(s)
```

```
M05    Dayse
M13     Bill
M19     Jimi
dtype: object
```

Indexação Booleana (2/2)

[M02]	[M05]	[M13]	[M14]	[M19]	
Bob	Dayse	Bill	Cris	Jimi	*s_alunos*

[0]	[1]	[2]	[3]	[4]	
7.6	5.0	8.5	9.5	6.4	*s_notas*

- Nesse exemplo, obtemos os alunos aprovados em “s_alunos” com base nas notas com valor ≥ 7.0 em “s_notas”.

```
#gera series booleana com True
#nas posições de valor >= 7
i = (s_notas >= 7)
print(i)
```

```
0     True
1    False
2     True
3     True
4    False
dtype: bool
```

```
#indexa “s_alunos” usando os
valores de i
print(s_alunos[i.values])
```

```
M02     Bob
M13    Bill
M14    Cris
dtype: object
```

Resumo - Series

- **Pontos fortes**:
  - Flexível.
  - Oferece um enorme número de métodos.
  - Eficiente.
  - Base para a manipulação de DataFrames.

- **Pontos fracos**:
  - Não é uma ED nativa do Python.
  - Não tão simples quanto uma lista ou dicionário.

- **Quando usar?**:
  - Para *data wrangling* sobre coleção de dados ordenados ou coleção de pares chave-valor.
  - Quando você quiser trabalhar com Séries Temporais
    - Ex.: vetor de índices contém datas.

Tarefa

(1) – As Series “verde” e “azul” abaixo fornecem informações sobre a presença das cores “azul” e “verde” nas bandeiras de diferentes países.

[BR]	[FR]	[IT]	[UK]	
1	0	1	0	verde

[AR]	[BR]	[FR]	[IT]	[UK]	
1	1	1	0	1	azul

Faça um programa que:

- Crie as duas Series
- Determine quais países possuem ambas as cores em sua bandeira.

Referências

- Corrêa, E. (2020). “Pandas Python: Data Wrangling para Ciência de Dados”. Casa do Código.

- McKinney, W. (2017). “Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython”. 2ª ed., O’Reilly Media.

- Pandas (2021). https://pandas.pydata.org