Engenharia de Software
Dynamic Programming
Aula 10 – Função apply e junção de dataframes

Prof. Dr. Francisco Elânio

Função apply

A função Pandas DataFrame apply() é usada para aplicar uma função ao longo de um eixo do DataFrame.

Link para documentação

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.apply.html

Função apply

Criando dataframe

```
import pandas as pd
df = pd.DataFrame([[4, 9 , 2]] * 4, columns=['A', 'B', 'C’])
```

Raiz quadrada

```
df.apply(np.sqrt)
```

Soma de linhas

```
df.apply(np.sum, axis=0)
```

Soma de colunas

```
df.apply(np.sum, axis=1)
```

Função apply para operações matemáticas

Aplicando função apply e lambda

```
df['D'] = df['A'].apply(lambda x: x * 2)
```

Função apply e lambda para somar colunas e linhas

```
result = df.apply(lambda row: row['A'] + row['B'], axis=1)
```

Função apply para calcular média

Calculando a média das colunas

```
df = pd.DataFrame([[4, 9 ]] * 3, columns=['A', 'B'])

resultado = df.apply(lambda col: col.mean())
```

Calculando a média de uma coluna específica

```
resultado = df['A'].apply(lambda x: x).mean()
resultado = df[‘B'].apply(lambda x: x).mean()
```

Função apply e lambda para normalização

Aplicar função apply e lambda para normalizar os dados das colunas entre 0 e 1 (min-max)

```
df = pd.DataFrame({
    'A': [6, 5, 8, 11],
    'B': [23, 20, 25, 30]
})

df_normalized = df.apply(lambda col: (col - col.min()) / (col.max() - col.min()))
```

Função apply e lambda para variáveis não numéricas

Aplicar função apply e lambda para normalizar os dados das colunas entre 0 e 1 (min-max)

```
df = pd.DataFrame({
    'Nome': ['João', 'Maria', 'Pedro', 'Ana']
})

df['Nome'] = df['Nome'].apply(lambda x: x + ' Silva')
```

Função apply e lambda para variáveis não numéricas

Contando o número de caracteres em cada valor da coluna 'Texto

```
df = pd.DataFrame({
    'Texto': ['Olá', 'Mundo', 'Python', 'Pandas']
})

df['Contagem de Caracteres'] = df['Texto'].apply(lambda texto:
len(texto))
```

Função apply e lambda para variáveis não numéricas

Verificar se cada linha está abaixo ou acima da média

```
df_numerico = pd.DataFrame({
   'A': [5, 7, 8, 9, 12]
})

df_numerico['verificao'] = df_numerico['A'].apply(lambda x: 'Maior'
if x > df_numerico['A'].mean() else 'Menor')
df_numerico
```

Função apply e lambda para variáveis não numéricas

Condição para verificar o sinal dos valores

```
df = pd.DataFrame({
    'A': [1, -2, 3, 0, -5]
})

df['Sinal'] = df['A'].apply(lambda x: 'Positivo' if x > 0 else 'Negativo' if x < 0 else 'Zero')
```

Função apply e lambda para variáveis não numéricas

Condição para verificar se os valores estão dentro do intervalo

```
df = pd.DataFrame({
    'Value': [10, 20, 30, 40, 50]
})

lower_bound = 20
upper_bound = 40

df['Within_Range'] = df['Value'].apply(lambda x: 'Yes' if lower_bound <= x
<= upper_bound else 'No')
```

Função apply e lambda para variáveis não numéricas

Condição para verificar o comprimento das strings

```
df = pd.DataFrame({
    'Texto': [‘maça', 'banana', ‘laranja’, ‘uva', 'kiwi']
})

df['Length'] = df['Texto'].apply(lambda x: 'Long' if len(x) > 5 else 'Short')
```

Função apply e lambda para variáveis não numéricas

Condição para verificar o comprimento das strings

```
df = pd.DataFrame({
    'Text': ['apple', 'banana', 'orange', 'grape', 'kiwi']
})

target_letter = 'a'

df['Contains'] = df['Text'].apply(lambda x: 'Yes' if target_letter in x
else 'No')
```

Exemplos/Exercícios
Cálculo de Lucro

```
import pandas as pd

# Criando um DataFrame de exemplo
data = {'produto': ['A', 'B', 'C', 'D'],
        'preco_venda': [100, 150, 200, 120],
        'custo_produto': [50, 80, 100, 60]}
df = pd.DataFrame(data)

# Usando apply e lambda para calcular o lucro
df['lucro'] = df.apply(lambda row: row['preco_venda'] - row['custo_produto'],
axis=1)
```

Exemplos/Exercícios
Análise de margem de Lucro

```
df['margem_lucro'] = df.apply(lambda row: (row['lucro'] / row['preco_venda']) * 100, axis=1)
```

Identificação de Produtos Rentáveis

```
margem_minima = 30
```

```
df['rentabilidade'] = df['margem_lucro'].apply(lambda margin: 'Rentável' if margin >= margem_minima else 'Não Rentável')
```

Exemplos/Exercícios
Análise de vendas por produto

```
df['categoria'] = ['Eletrônicos', 'Roupas', 'Eletrônicos', 'Alimentos']

vendas_por_categoria = df['categoria'].apply(lambda categoria:
df[df['categoria'] == categoria]['preco_venda'].sum())
```

Agrupamento dados por categoria
Groupby por média

```
import pandas as pd

data = {
    'Categoria': ['A', 'B', 'A', 'B', 'A', 'B'],
    'Valor': [10, 20, 30, 40, 50, 60]
}
df = pd.DataFrame(data)

grouped = df.groupby('Categoria').mean()
```

Agrupamento dados por categoria
Groupby – min, max e median

```
Grouped_min = df.groupby('Categoria’).min()

Grouped_max = df.groupby('Categoria’).max()

Grouped_mediana = df.groupby('Categoria').median()
```

Agrupamento dados por categoria
Groupby – duas colunas

```
import pandas as pd
data = {
    'Categoria': ['A', 'B', 'A', 'B', 'A', 'B'],
    'Grupo': ['X', 'X', 'Y', 'Y', 'X', 'Y'],
    'Valor': [10, 20, 30, 40, 50, 60]
}
df = pd.DataFrame(data)

grouped = df.groupby(['Categoria', 'Grupo']).mean()
```

Trabalhando com dataframes
Merge

Juntando dados do dataframe 1 com o dataframe 2.
Juntar dados do dataframe 1 com o dataframe 3.
Juntar dados do dataframe 1 com o dataframe 4.

Responder as seguintes questões:

-Crie um filtro para mostrar o tipo de vaga para cientista de dados.
- Mostre qual a pessoa com menor e maior idade para a vaga cientista de dados.
- Calcule a média de idade para a vaga cientista de dados.
- Compare a faixa de idade média entre as três profissões.
- Qual profissão apresenta pessoas com menor e maior faixa etária?

Datasets relacionado a acesso a vagas por idade

Dataset 1

Empresa	Tipo de Vaga	Salário oferecido
A	Cientista de dados	10000
A	Engenheiro de dados	9500
A	Data Analytics	7300
B	Cientista de dados	15000
B	Engenheiro de dados	10700
B	Data Analytics	7000
C	Cientista de dados	11500
C	Engenheiro de dados	13000
C	Data Analytics	8500

Dataset 2

Empresa	Idade	Tipo Vaga
A	30	Cientista de dados
A	32	Engenheiro de dados
A	25	Data Analytics
A	26	Cientista de dados
A	40	Engenheiro de dados
A	38	Data Analytics
A	20	Cientista de dados
A	29	Engenheiro de dados
A	36	Data Analytics

Dataset 3

Empresa	Idade	Tipo Vaga
B	31	Cientista de dados
B	33	Engenheiro de dados
B	27	Data Analytics
B	29	Cientista de dados
B	35	Engenheiro de dados
B	37	Data Analytics
B	21	Cientista de dados
B	22	Engenheiro de dados
B	28	Data Analytics

Dataset 4

Empresa	Idade	Tipo Vaga
C	41	Cientista de dados
C	38	Engenheiro de dados
C	26	Data Analytics
C	28	Cientista de dados
C	36	Engenheiro de dados
C	33	Data Analytics
C	28	Cientista de dados
C	24	Engenheiro de dados
C	23	Data Analytics

“O prazer mais nobre é o júbilo de compreender”
(Leonardo Da Vinci)

Referências

- John Paul Mueller / Luca Massaron, Algoritmos para leigos, Editora Alta books, 2018 - 1ª edição.
- Additya Y. Bhargava, Entendendo Algoritmos - um guia ilustrado para programadores e outros curiosos, Editora Novatec, 2018, 1ª edição.
- José Augusto N. G. Manzano / Jayr Figueiredo de Oliveira, Algoritmos: Lógica Para Desenvolvimento de Programação de Computadores, 29ª edição, 2019.
- Thomas H. Cormen / Charles E. Leiseson / Ronald L. Rivest / Clifford Stein, Algorítmos - Teoria e Prática, Editora Elsevier, 2012, 3ª edição.