Numpy

Programación II

Pandas

Programación II

Pandas es una herramienta de manipulación y análisis de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python.

La principal estructura de datos en Pandas es el DataFrame, que es similar a una tabla en una hoja de cálculo o una tabla en una base de datos relacional. Los DataFrames son estructuras de datos bidimensionales con filas y columnas etiquetadas, lo que los hace ideales para trabajar con datos tabulares.

(Pandas, 2023)

Pandas

Programación II

Visualización de datos: pandas proporciona funciones para crear gráficos y visualizaciones de datos utilizando bibliotecas de visualización de Python, como Matplotlib y Seaborn.

Pandas también proporciona una estructura de datos unidimensional llamada Series, que es similar a una columna en una hoja de cálculo o una tabla de base de datos.

(Pandas, 2023)

Pandas

Programación II

Algunas de las funciones más comunes que ofrece pandas son:

- *Lectura y escritura de archivos de datos (CSV, Excel, JSON, etc.)*
- *Manipulación de datos, como agregar, eliminar o modificar filas y columnas*
- *Selección y filtrado de datos*
- *Agrupamiento y agregación de datos (por ejemplo, calcular el promedio de una columna)*
- *Unión y combinación de datos de varias fuentes*
- *Manejo de valores faltantes y duplicados*
- *Transformación de datos (por ejemplo, escalando una columna numérica)*

Pandas

Programación II

- *Documentación Oficial*
  - *https://pandas.pydata.org/*
- *Guia de pandas*
  - *https://pandas.pydata.org/docs/user_guide/index.html*

Pandas (Instalación)

Programación II

Installation

Working with conda?

pandas is part of the Anaconda distribution and can be installed with Anaconda or Miniconda:

```
conda install pandas
```

Prefer pip?

pandas can be installed via pip from PyPI.

```
pip install pandas
```

Pandas (Creación de una Serie)

Programación II

```
>>> import pandas as pd
>>> s = pd.Series(['Matemáticas', 'Historia', 'Economía', 'Programación', 'Inglés'], dtype='string’)

>>> print(s)

0 Matemáticas
1 Historia
2 Economía
3 Programación
4 Inglés

dtype: string
```

Pandas (Creación de una Serie a partir de un diccionario)

Programación II

```
>>> import pandas as pd

>>> s = pd.Series({'Matemáticas': 6.0, 'Economía': 4.5, 'Programación': 8.5})

>>> print(s)

Matemáticas 6.0
Economía 4.5
Programación 8.5

dtype: float64
```

Pandas (Atributos de una serie)

Programación II

Existen varias propiedades o métodos para ver las características de una serie.

- *s.size : Devuelve el número de elementos de la serie s.*
- *s.index : Devuelve una lista con los nombres de las filas del DataFrame s.*
- *s.dtype : Devuelve el tipo de datos de los elementos de la serie s.*

```
>>> import pandas as pd

>>> s = pd.Series([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])

>>> s.size
10
>>> s.index
RangeIndex(start=0, stop=10, step=1)

>>> s.dtype

dtype('int64')
```

Pandas (Acceso por posicion)

Programación II

Acceso por posición

Se realiza de forma similar a como se accede a los elementos de un array.

- *s[i] : Devuelve el elemento que ocupa la posición i+1 en la serie s.*
- *s[posiciones]: Devuelve otra serie con los elementos que ocupan las posiciones de la lista posiciones.*

Acceso por índice

- *s[nombre] : Devuelve el elemento con el nombre nombre en el índice.*
- *s[nombres] : Devuelve otra serie con los elementos correspondientes a los nombres indicadas en la lista nombres en el índice.*

Pandas (Acceso por posición)

Programación II

```
>>> s[1:3]

Economía 4.5
Programación 8.5
dtype: float64

>>> s['Economía']
4.5

>>> s[['Programación', 'Matemáticas']]

Programación 8.5
Matemáticas 6.0
dtype: float64
```

Pandas (Resumen descriptivo de una serie)

Programación II

Las siguientes funciones permiten resumir varios aspectos de una serie:

- *s.count() : Devuelve el número de elementos que no son nulos ni NaN en la serie s.*

- *s.sum() : Devuelve la suma de los datos de la serie s cuando los datos son de un tipo numérico, o la concatenación de ellos cuando son del tipo cadena str.*

- *s.cumsum() : Devuelve una serie con la suma acumulada de los datos de la serie s cuando los datos son de un tipo numérico.*

- *s.value_counts() : Devuelve una serie con la frecuencia (número de repeticiones) de cada valor de la serie s.*

Pandas (Resumen descriptivo de una serie)

Programación II

- *s.min() : Devuelve el menor de los datos de la serie s.*
- *s.max() : Devuelve el mayor de los datos de la serie s.*
- *s.mean() : Devuelve la media de los datos de la serie s cuando los datos son de un tipo numérico.*
- *s.var() : Devuelve la varianza de los datos de la serie s cuando los datos son de un tipo numérico.*
- *s.std() : Devuelve la desviación típica de los datos de la serie s cuando los datos son de un tipo numérico.*
- *s.describe(): Devuelve una serie con un resumen descriptivo que incluye el número de datos, su suma, el mínimo, el máximo, la media, la desviación típica y los cuartiles.*

Pandas (Aplicar operaciones a una serie)

Programación II

*Los operadores binarios (+, *, /, etc.) pueden utilizarse con una serie, y devuelven otra serie con el resultado de aplicar la operación a cada elemento de la serie.*

```
>>> import pandas as pd

s = pd.Series([1, 2, 3, 4])

>>> s * 2 0 2 1 4 2 6 3 8

dtype: int64
```

```
>>> s % 2

0 1
1 0
2 1
3 0

dtype: int64
```

```
>>> s = pd.Series(['a', 'b', 'c'])
>>> s * 5

0 aaaaa
1 bbbbb
2 ccccc

dtype: object
```

Pandas (Aplicar funciones a una serie)

Programación II

También es posible aplicar una función a cada elemento de la serie mediante el siguiente método:

• *s.apply(f) : Devuelve una serie con el resultado de aplicar la función f a cada uno de los elementos de la serie s*

```
>>> import pandas as pd
>>> from math import log

>>> s = pd.Series([1, 2, 3, 4])
>>> s.apply(log)

0 0.000000
1 0.693147
2 1.098612
3 1.386294

dtype: float64
```

```
>>> s = pd.Series(['a', 'b', 'c'])
>>> s.apply(str.upper)

0 A
1 B
2 C

dtype: object
```

Pandas (Filtrar una serie)

Programación II

Para filtrar una serie y quedarse con los valores que cumplen una determinada condición se utiliza el siguiente método:

• *s[condicion] : Devuelve una serie con los elementos de la serie s que se corresponden con el valor True de la lista booleana condicion. condicion debe ser una lista de valores booleanos de la misma longitud que la serie*

```
>>> import pandas as pd

>>> s = pd.Series({'Matemáticas': 6.0, 'Economía': 4.5, 'Programación': 8.5})

>>> print(s[s > 5])

Matemáticas 6.0
Programación 8.5

dtype: float64
```

Pandas (Ordenar una serie)

Programación II

Para ordenar una serie se utilizan los siguientes métodos:

- *s.sort_values(ascending=booleano) : Devuelve la serie que resulta de ordenar los valores la serie s. Si argumento del parámetro ascending es True el orden es creciente y si es False decreciente.*

- *df.sort_index(ascending=booleano) : Devuelve la serie que resulta de ordenar el índice de la serie s. Si el argumento del parámetro ascending es True el orden es creciente y si es False decreciente*

Pandas (Ordenar una serie)

Programación II

```
>>> import pandas as pd

>>> s = pd.Series({'Matemáticas': 6.0, 'Economía': 4.5, 'Programación': 8.5})
>>> print(s.sort_values())

Economía 4.5
Matemáticas 6.0
Programación 8.5
dtype: float64

>>> print(s.sort_index(ascending = False))

Programación 8.5
Matemáticas 6.0
Economía 4.5 dtype: float64
```

Pandas (Eliminar datos desconocidos)

Programación II

Los datos desconocidos representan en Pandas por NaN y los nulos por None. Tanto unos como otros suelen ser un problema a la hora de realizar algunos análisis de datos, por lo que es habitual eliminarlos. Para eliminarlos de una serie se utiliza el siguiente método:

• *s.dropna() : Elimina los datos desconocidos o nulos de la serie s.*

```
>>> import pandas as pd
>>> import numpy as np

>>> s = pd.Series(['a', 'b', None, 'c', np.NaN, 'd'])
>>> s

0 a
1 b
2 None
3 c
4 NaN
5 d

 dtype: object
```

```
>>> s.dropna()

0 a
1 b
3 c
5 d

dtype: object
```