

DATA MANIPULATION WITH PYTHON

Introduction to Data Science

Author: Eng. Carlos Andrés Sierra, M.Sc.

carlos.andres.sierra.v@gmail.com

Lecturer
Computer Engineer
School of Engineering
Universidad Distrital Francisco José de Caldas

2024-II



Outline

1 Numerical Analysis with Numpy

2 Text Analysis and Regular Expressions

3 Data Manipulation with Pandas



Outline

1 Numerical Analysis with Numpy

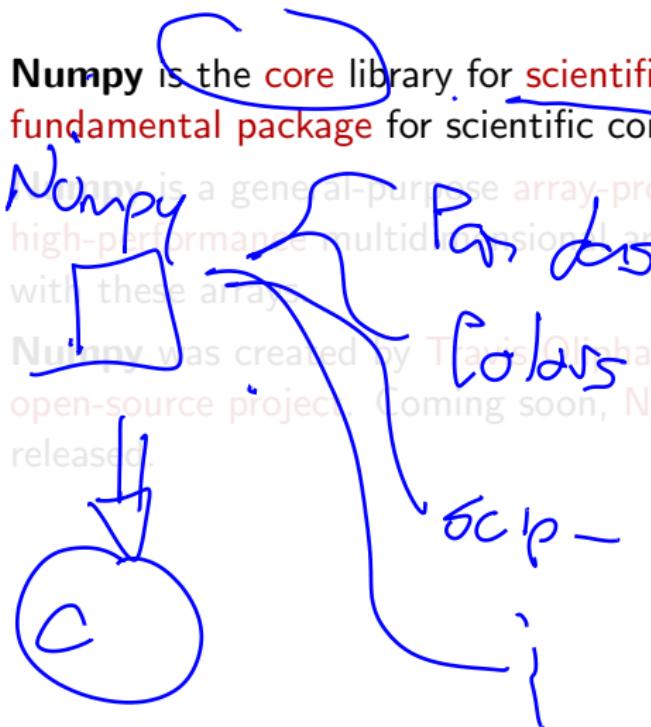
2 Text Analysis and Regular Expressions

3 Data Manipulation with Pandas



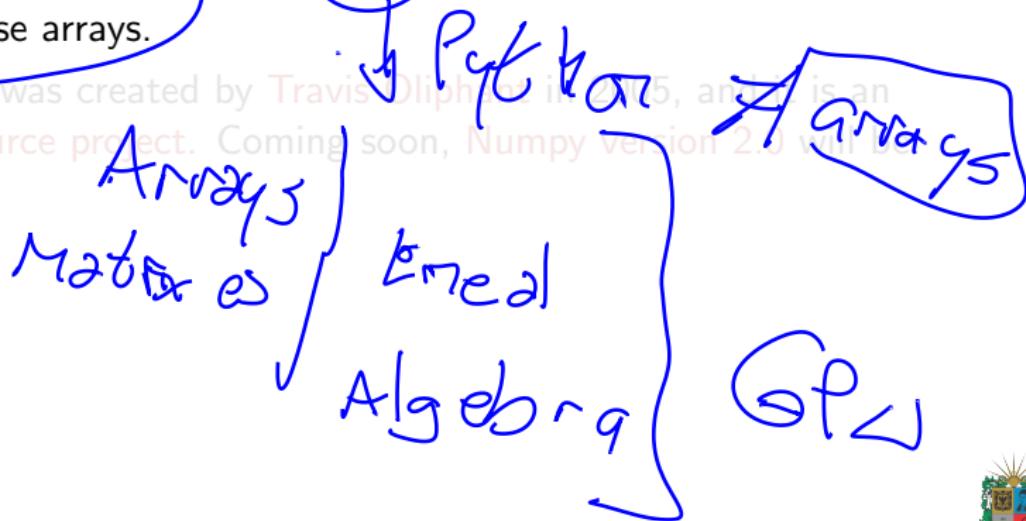
Numerical Python Library — Numpy

- Numpy is the **core** library for **scientific computing** in Python. It is the **fundamental package** for scientific computing with Python.
- Numpy is a general-purpose **array-processing** package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
- Numpy was created by **Travis Oliphant** in 2005, and it is an open-source project. Coming soon, Numpy version 2.0 will be released.



Numerical Python Library — Numpy

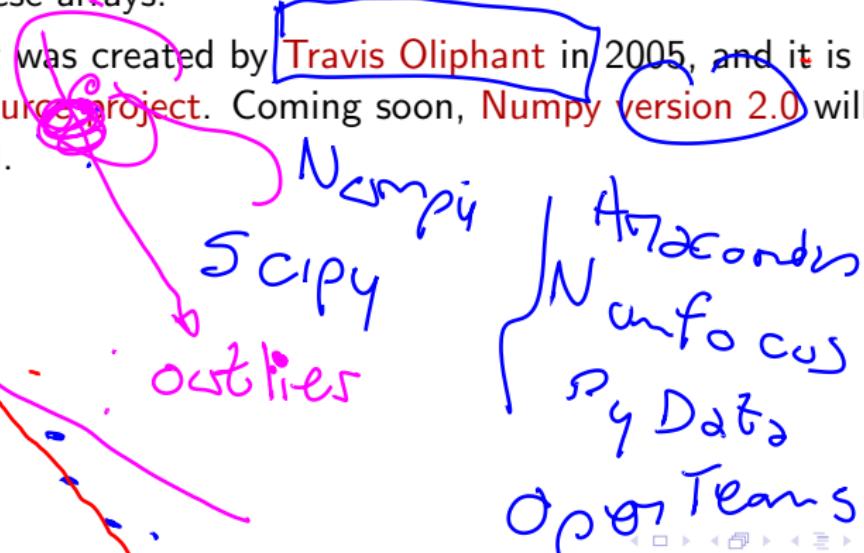
- **Numpy** is the **core** library for **scientific computing** in Python. It is the **fundamental package** for scientific computing with **Python**.
- **Numpy** is a general-purpose **array-processing** package. It provides a **high-performance** multidimensional array object, and tools for working with these arrays.
- Numpy was created by Travis Oliphant in 2005, and is an open-source project. Coming soon, Numpy version 2.0 will be released.



Numerical Python Library — Numpy

- **Numpy** is the **core** library for **scientific computing** in Python. It is the **fundamental package** for scientific computing with **Python**.
- **Numpy** is a general-purpose **array-processing** package. It provides a **high-performance** multidimensional array object, and tools for working with these arrays.
- **Numpy** was created by **Travis Oliphant** in 2005, and it is an **open-source project**. Coming soon, **Numpy version 2.0** will be released.

1 2 3
4 5 6
7 8 9



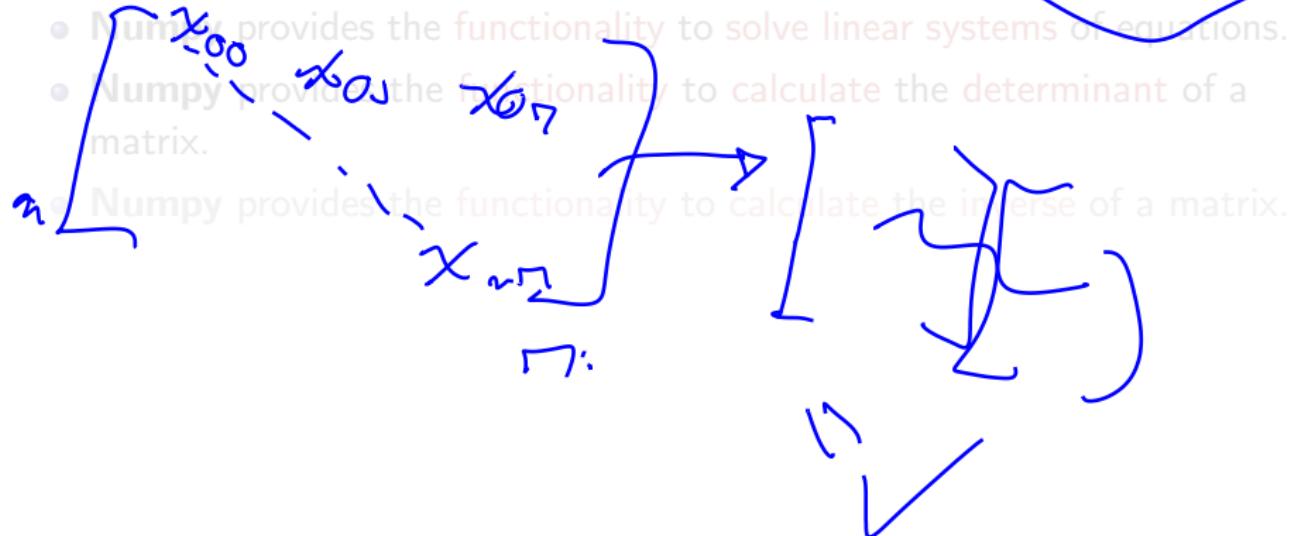
Lineal Algebra with Numpy

- **Numpy** provides a comprehensive set of **linear algebra** functions.
- Numpy provides the functionality to create and manipulate matrices.
- Numpy provides the functionality to solve linear systems of equations.
- Numpy provides the functionality to calculate the determinant of a matrix.
- Numpy provides the functionality to calculate the inverse of a matrix.



Lineal Algebra with Numpy

- **Numpy** provides a comprehensive set of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate matrices**.
- **Numpy** provides the **functionality** to solve linear systems of equations.
- **Numpy** provides the **functionality** to calculate the determinant of a matrix.
- **Numpy** provides the **functionality** to calculate the inverse of a matrix.



Lineal Algebra with Numpy

- **Numpy** provides a comprehensive set of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate** matrices.
- **Numpy** provides the **functionality** to **solve linear systems** of equations.
- **Numpy** provides the **functionality** to calculate the determinant of a matrix.
- **Numpy** provides the **functionality** to calculate the inverse of a matrix.

$$\begin{aligned}
 & 2x + 3y = 5 \\
 & 3x + 7y = 12 \\
 \left[\begin{array}{cc|c} 2 & 3 & 5 \\ 3 & 7 & 12 \end{array} \right] \xrightarrow{\text{Row Operations}} \left[\begin{array}{cc|c} 1 & 3/2 & 5/2 \\ 0 & 1 & 3 \end{array} \right]
 \end{aligned}$$



Lineal Algebra with Numpy

- **Numpy** provides a comprehensive set of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate matrices**.
- **Numpy** provides the **functionality** to **solve linear systems** of equations.
- **Numpy** provides the **functionality** to **calculate the determinant** of a matrix.
- **Numpy** provides the **functionality** to calculate the inverse of a matrix.

$$\det = ad - bc$$



Lineal Algebra with Numpy

- **Numpy** provides a **comprehensive set of linear algebra functions**.
 - **Numpy** provides the **functionality to create and manipulate matrices**.
 - **Numpy** provides the **functionality to solve linear systems** of equations.
 - **Numpy** provides the **functionality to calculate the determinant** of a matrix.
 - **Numpy** provides the **functionality to calculate the inverse** of a matrix.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Inverse



Vectorization with Numpy

- **Vectorization** is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time.
- Vectorization is the process of replacing explicit loops with array expressions or matrix operations.
- The advantages of vectorization are speed and clarity. The disadvantages are memory and complexity.
- Numpy provides the functionality to vectorize operations on arrays.



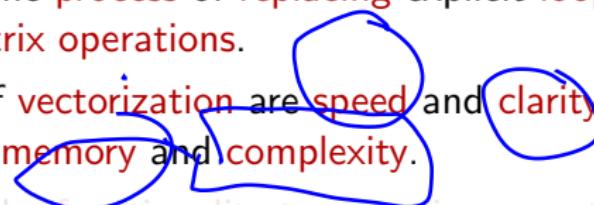
Vectorization with Numpy

- **Vectorization** is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time.
- **Vectorization** is the process of replacing explicit loops with array expressions or matrix operations.
- The advantages of vectorization are speed and clarity. The disadvantages are memory and complexity.
- Numpy provides the functionality to vectorize operations on arrays.



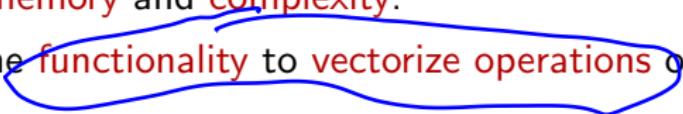
Vectorization with Numpy

- **Vectorization** is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time.
- **Vectorization** is the process of replacing explicit loops with array expressions or matrix operations.
- The advantages of vectorization are speed and clarity. The disadvantages are memory and complexity.
- Numpy provides the functionality to vectorize operations on arrays.



Vectorization with Numpy

- **Vectorization** is the process of converting an algorithm from operating on a single value at a time to operating on a set of values at one time.
- **Vectorization** is the process of replacing explicit loops with array expressions or matrix operations.
- The advantages of vectorization are speed and clarity. The disadvantages are memory and complexity.
- **Numpy** provides the functionality to vectorize operations on arrays.



Typical Operations with Numpy

- **Numpy** provides the **functionality** to **create** and **manipulate arrays**.
- **Numpy** provides the **functionality** to **perform element-wise operations** on **arrays**.
- **Numpy** provides the **functionality** to **perform matrix operations** on **arrays**.
- **Numpy** provides the **functionality** to **perform linear algebra operations** on **arrays**.
- **Numpy** provides the **functionality** to **perform statistical operations** on **arrays**.



Outline

1 Numerical Analysis with Numpy

2 Text Analysis and Regular Expresions

3 Data Manipulation with Pandas



Strings in Python

Definition

A **string** is a sequence of *characters*. In Python, a *string* is a sequence of *Unicode characters*; also, *strings* are *immutable*, *ordered*, *iterable*, *indexable*, and *slicable*.

The diagram shows the string "Hola Mundo" with red wavy lines indicating character positions from 0 to 10. A blue arrow points to the character 'o' at index 4, with the label "size = 11" and "len('Hola Mundo') = 11".

✓ Hola Mundo
0 1 2 3 4 5 6 7 8 9 10



Conditionals and Loops with Strings

A" == "q'

- **Strings** can be compared using conditional statements. *X*

- You could validate if a substring is contained in a string.

- You could iterate over the characters of a string \Rightarrow True

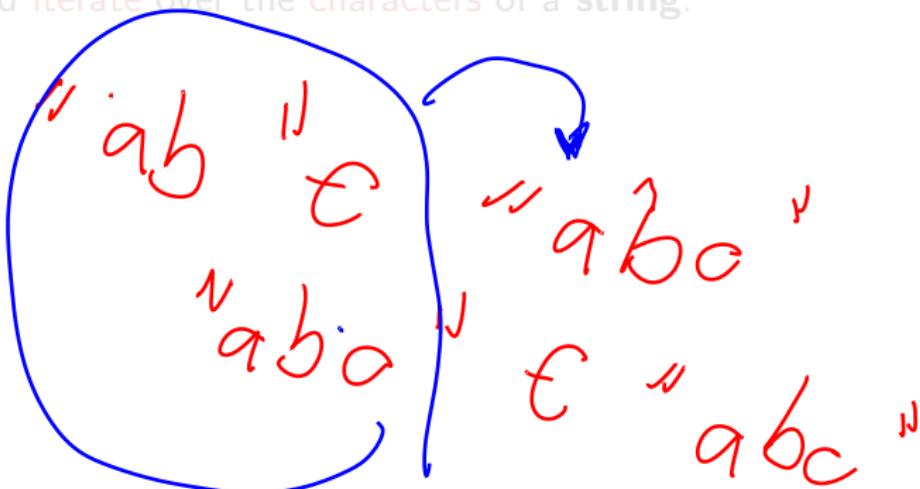
"ab" != "a c" \Rightarrow False
"ab" != "a c" \Rightarrow False
 ↳ Unicode

"abC" < "aa"
X False



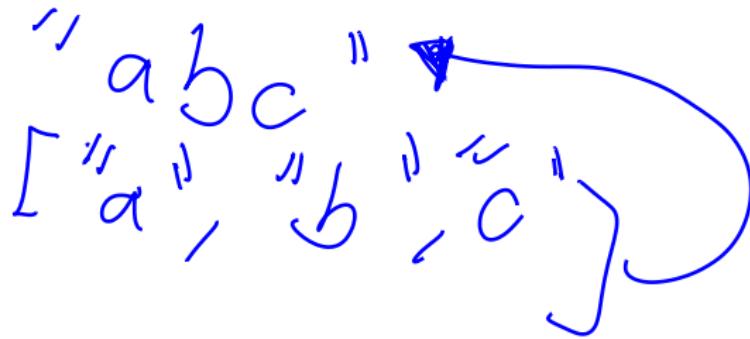
Conditionals and Loops with Strings

- **Strings** can be compared using conditional statements.
- You could validate if a **substring** is contained in a **string**.
- You could iterate over the characters of a **string**.



Conditionals and Loops with Strings

- **Strings** can be **compared** using **conditional statements**.
- You could validate if a **substring** is **contained** in a **string**.
- You could **iterate** over the **characters** of a **string**.



Python Built-in Functions for Strings

$$S * 4 = S + S + S + S$$

- You could concatenate strings using the `+` operator.

- You could repeat a string using the `*` operator.

- You could format a string using the `format()` method.

- You could split a string using the `split()` method.

- You could join a list of strings using the `join()` method.

- You could replace a substring in a string using the `replace()` method.

- You could find a substring in a string using the `find()` method.

- You could count the occurrences of a substring in a string using the `count()` method.

$$A + B = B + A$$



Python Built-in Functions for Strings

- You could **concatenate strings** using the `+` operator.
- You could **repeat** a string using the `*` operator.
- You could **format** a string using the `format()` method.
- You could **split** a string using the `split()` method.
- You could **join** a list of strings using the `join()` method.
- You could **replace** a substring in a string using the `replace()` method.
- You could **find** a substring in a string using the `find()` method.
- You could **count** the occurrences of a substring in a string using the `count()` method.



Python Built-in Functions for Strings

- You could concatenate strings using the `+` operator.
- You could repeat a string using the `*` operator.
- You could format a string using the `format()` method.
- You could split a string using the `split()` method.
- You could join a list of strings using the `join()` method.
- You could replace a substring in a string using the `replace()` method.
- You could find a substring in a string using the `find()` method.
- You could count the occurrences of a substring in a string using the `count()` method.

String --- {var}



Python Built-in Functions for Strings

- You could **concatenate strings** using the `+` operator.
- You could **repeat** a string using the `*` operator.
- You could **format** a string using the `format()` method.
- You could **split** a string using the `split()` method.
- You could **join** a list of strings using the `join()` method.
- You could **replace** a substring in a string using the `replace()` method.
- You could **find** a substring in a string using the `find()` method.
- You could **count** the occurrences of a substring in a string using the `count()` method.

`= "Aqa Bob C123".split(" ")`



Python Built-in Functions for Strings

- You could concatenate strings using the `+` operator.
- You could repeat a string using the `*` operator.
- You could format a string using the `format()` method.
- You could split a string using the `split()` method.
- You could join a list of strings using the `join()` method.
- You could replace a substring in a string using the `replace()` method.
- You could find a substring in a string using the `find()` method.
- You could count the occurrences of a substring in a string using the `count()` method.

JOIN (names)
`m[1] + " " + m[2] + " " ...`



Python Built-in Functions for Strings

- You could **concatenate strings** using the `+` operator.
- You could **repeat** a string using the `*` operator.
- You could **format** a string using the `format()` method.
- You could **split** a string using the `split()` method.
- You could **join** a list of strings using the `join()` method.
- You could **replace** a substring in a string using the `replace()` method.
- You could **find** a substring in a string using the `find()` method.
- You could **count** the occurrences of a substring in a string using the `count()` method.

abc

r ep bce ('a', 'A')



Python Built-in Functions for Strings

- You could **concatenate strings** using the `+` operator.
- You could **repeat** a string using the `*` operator.
- You could **format** a string using the `format()` method.
- You could **split** a string using the `split()` method.
- You could **join** a list of strings using the `join()` method.
- You could **replace** a substring in a string using the `replace()` method.
- You could **find** a substring in a string using the `find()` method.
- You could **count** the occurrences of a substring in a string using the `count()` method.

Find (~) → first occurrence



Python Built-in Functions for Strings

- You could **concatenate strings** using the `+` operator.
- You could **repeat** a string using the `*` operator.
- You could **format** a string using the `format()` method.
- You could **split** a string using the `split()` method.
- You could **join** a list of strings using the `join()` method.
- You could **replace** a substring in a string using the `replace()` method.
- You could **find** a substring in a string using the `find()` method.
- You could **count** the occurrences of a substring in a string using the `count()` method.

• `count('')`



Python Dates and Times

- Python provides the **datetime** module to work with dates and times.
- The **datetime** module provides the **datetime** class to work with dates and times.
"Y - m - d"
- To create a **datetime** object, you could use the **datetime()** constructor.
"H : M : S"
- The **datetime** class provides the **strptime()** method to format a **Timestamp** into a **datetime** object.
- The **datetime** class provides the **timedelta()** constructor to calculate the difference between two **datetime** objects.



Python Dates and Times

- Python provides the `datetime` module to work with dates and times.
 - The `datetime` module provides the `datetime` class to work with dates and times.
 - To create a `datetime` object, you could use the `datetime()` constructor.
 - The `datetime` class provides the `strftime()` method to format a `datetime` object.
 - The `datetime` class provides the `strptime()` method to parse a string into a `datetime` object.
 - The `datetime` class provides the `timedelta()` constructor to calculate the difference between two `datetime` objects.
- From *date*, *be* . Import *date*



Python Dates and Times

- Python provides the `datetime` module to work with dates and times.
- The `datetime` module provides the `datetime` class to work with dates and times.
- To create a `datetime` object, you could use the `datetime()` constructor.
- The `datetime` class provides the `strftime()` method to format a `datetime` object.
- The `datetime` class provides the `strptime()` method to parse a string into a `datetime` object.
- The `datetime` class provides the `timedelta()` constructor to calculate the difference between two `datetime` objects.



Python Dates and Times

- Python provides the `datetime` module to work with dates and times.
- The `datetime` module provides the `datetime` class to work with dates and times.
- To create a `datetime` object, you could use the `datetime()` constructor.
- The `datetime` class provides the `strftime()` method to format a `datetime` object.
- The `datetime` class provides the `strptime()` method to parse a string into a `datetime` object.
- The `datetime` class provides the `timedelta()` constructor to calculate the difference between two `datetime` objects.



Python Dates and Times

- Python provides the `datetime` module to work with dates and times.
- The `datetime` module provides the `datetime` class to work with dates and times.
- To create a `datetime` object, you could use the `datetime()` constructor.
- The `datetime` class provides the `strftime()` method to format a `datetime` object.
- The `datetime` class provides the `strptime()` method to parse a string into a `datetime` object.
- The `datetime` class provides the `timedelta()` constructor to calculate the difference between two `datetime` objects.



Python Dates and Times

- Python provides the `datetime` module to work with dates and times.
- The `datetime` module provides the `datetime` class to work with dates and times.
- To create a `datetime` object, you could use the `datetime()` constructor.
- The `datetime` class provides the `strftime()` method to format a `datetime` object.
- The `datetime` class provides the `strptime()` method to parse a string into a `datetime` object.
- The `datetime` class provides the `timedelta()` constructor to calculate the difference between two `datetime` objects.

26e + 1 day = 27e



Regular Expressions — ReLex

- A **regular expression** is a **sequence of characters** that **define a search pattern**.
- Regular expressions are used to search for patterns in strings.
- In Python, the `re` module provides the functionality to work with regular expressions.
- The `"[ab][0-9]*"` module provides the `compile()` function to compile a regular expression pattern.
- The `re` module provides the `search()` function to search for a pattern in a string.



Regular Expressions — ReGex

- A **regular expression** is a **sequence** of characters that **define** a search pattern.
- **Regular expressions** are used to **search** for patterns in strings.
- In Python, the **re** module provides the functionality to work with regular expressions.
- The **re** module provides the **compile()** function to compile a regular expression pattern.
- The **re** module provides the **search()** function to search for a pattern in a string.



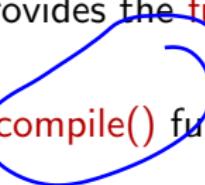
Regular Expressions — ReLex

- A **regular expression** is a **sequence** of characters that **define** a **search pattern**.
- **Regular expressions** are used to **search** for **patterns** in **strings**.
- In **Python**, the **re** module provides the **functionality** to **work** with **regular expressions**.
- The **re** module provides the **compile()** function to **compile** a **regular expression** pattern.
- The **re** module provides the **search()** function to **search** for a **pattern** in a **string**.



Regular Expressions — ReLex

- A **regular expression** is a **sequence of characters** that **define a search pattern**.
- **Regular expressions** are **used to search for patterns in strings**.
- In **Python**, the **re** module provides the **functionality to work with regular expressions**.
- The **re** module provides the **compile()** function to **compile a regular expression pattern**.
- The **re** module provides the **search()** function to **search for a pattern in a string**.



Regular Expressions — ReGex

Méjico star \Rightarrow *

$(gb)^*$

gb
caba
abab
abab

- A **regular expression** is a **sequence** of **characters** that **define** a **search pattern**.
- Regular expressions** are used to **search** for **patterns** in **strings**.
- In **Python**, the **re** module provides the **functionality** to **work** with **regular expressions**.
- The **re** module provides the **compile()** function to **compile** a **regular expression** pattern.
- The **re** module provides the **search()** function to **search** for a **pattern** in a **string**.



Outline

1 Numerical Analysis with Numpy

2 Text Analysis and Regular Expressions

3 Data Manipulation with Pandas



Introduction to Pandas

- Pandas is a fast, powerful, flexible, and easy-to-use open-source data manipulation and data analysis library built on top of the Python programming language.
- Pandas is a high-level data manipulation tool developed by Wes McKinney in 2008.
- Pandas is a fast and efficient data manipulation tool that is built on top of NumPy.
- Pandas is one of the most popular and widely-used data manipulation libraries in the world.



Introduction to Pandas

- Pandas is a fast, powerful, flexible, and easy-to-use open-source data manipulation and data analysis library built on top of the Python programming language.
- Pandas is a high-level data manipulation tool developed by Wes McKinney in 2008.
- Pandas is a fast and efficient data manipulation tool that is built on top of NumPy.
- Pandas is one of the most popular and widely-used data manipulation libraries in the world.



Introduction to Pandas

- Pandas is a fast, powerful, flexible, and easy-to-use open-source data manipulation and data analysis library built on top of the Python programming language.
- Pandas is a high-level data manipulation tool developed by Wes McKinney in 2008.
- Pandas is a fast and efficient data manipulation tool that is built on top of NumPy.
- Pandas is one of the most popular and widely-used data manipulation libraries in the world.

Pandas ~ built on top of numpy ~~



Introduction to Pandas

- **Pandas** is a fast, powerful, flexible, and easy-to-use open-source data manipulation and data analysis library built on top of the Python programming language.
- **Pandas** is a high-level data manipulation tool developed by Wes McKinney in 2008.
- **Pandas** is a fast and efficient data manipulation tool that is built on top of NumPy.
- **Pandas** is one of the most popular and widely-used data manipulation libraries in the world.

Polaris



The “Series” Data Structure

- A **Series** is a one-dimensional array-like object that contains a sequence of values and an associated array of data labels, called the index.
- The index of a Series is an array of labels that correspond to the values in the Series. The index of a Series is an optional parameter that defaults to a sequence of integers starting at zero.
- The Series object is a core data structure in Pandas.



The “Series” Data Structure

- A **Series** is a one-dimensional array-like object that contains a sequence of values and an associated array of data labels, called the index.
- The index of a **Series** is an array of labels that correspond to the values in the **Series**. The index of a **Series** is an optional parameter that defaults to a sequence of integers starting at zero.
- The **Series** object is a core data structure in Pandas.



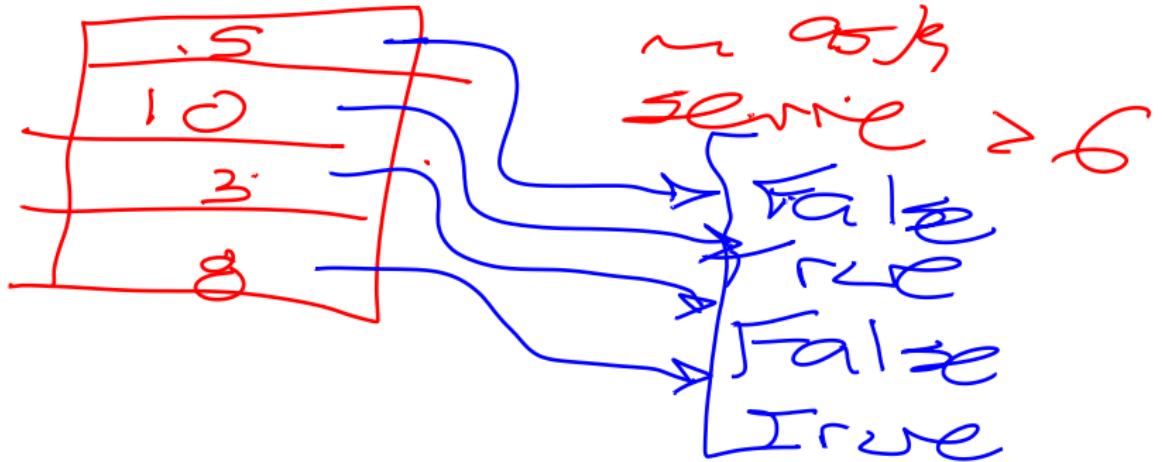
The “Series” Data Structure

- A **Series** is a one-dimensional array-like object that contains a sequence of values and an associated array of data labels, called the index.
- The index of a **Series** is an array of labels that correspond to the values in the **Series**. The index of a **Series** is an optional parameter that defaults to a sequence of integers starting at zero.
- The **Series** object is a core data structure in **Pandas**.



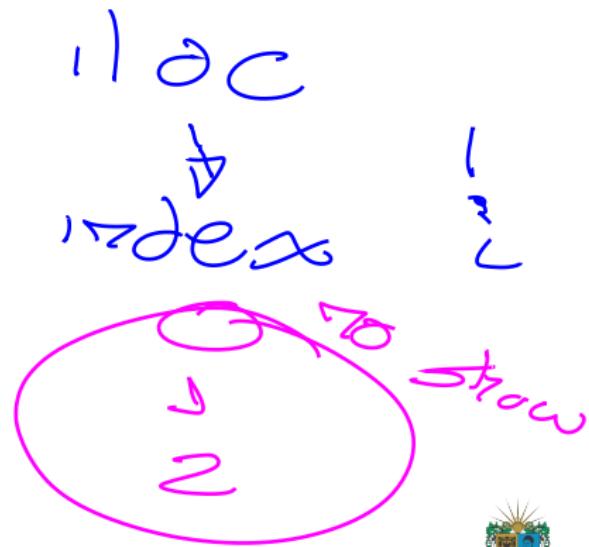
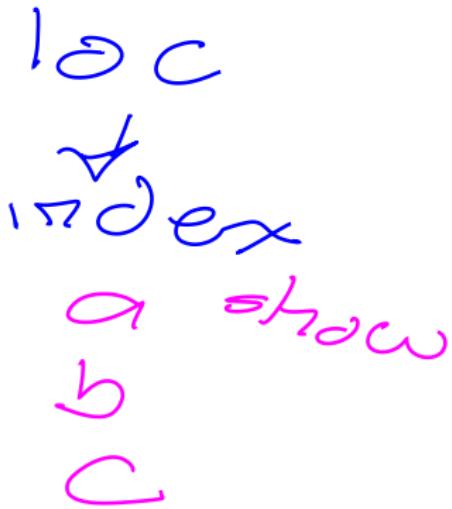
Querying a Series

- You could ~~query~~ a Series using indexing (boolean or fancy).
- You could ~~query~~ a Series using loc and iloc indexers.



Querying a Series

- You could **query** a **Series** using **indexing** (**boolean** or **fancy**).
- You could **query** a **Series** using **loc** and **iloc** indexers.



The “DataFrame” Data Structure

- A **DataFrame** is a two-dimensional labeled data structure with columns of potentially different types.
- A **DataFrame** is a tabular data structure that is similar to a spreadsheet or a **SQL** table.
- A **DataFrame** is a core data structure in **Pandas**. It is a two-dimensional size-mutable data structure with labeled axes (rows and columns).
- A **DataFrame** is a container for **Series** objects.



The “DataFrame” Data Structure

- A **DataFrame** is a two-dimensional labeled data structure with columns of potentially different types.
- A **DataFrame** is a ~~tabular data structure~~ that is **similar** to a ~~spreadsheet or a SQL table~~.
- A **DataFrame** is a **core data structure** in **Pandas**. It is a two-dimensional size-mutable data structure with **labeled axes** (rows and columns).
- A **DataFrame** is a **container** for **Series** objects.



The “DataFrame” Data Structure

- A **DataFrame** is a two-dimensional labeled data structure with columns of potentially different types.
- A **DataFrame** is a tabular data structure that is similar to a spreadsheet or a **SQL table**.
- A **DataFrame** is a core data structure in **Pandas**. It is a two-dimensional size mutable data structure with labeled axes (rows and columns).
- A **DataFrame** is a container for **Series** objects.

Value

Index



The “DataFrame” Data Structure

- A **DataFrame** is a two-dimensional labeled data structure with columns of potentially different types.
- A **DataFrame** is a tabular data structure that is **similar** to a spreadsheet or a **SQL** table.
- A **DataFrame** is a **core data structure** in **Pandas**. It is a two-dimensional size-mutable data structure with **labeled axes** (rows and columns).
- A **DataFrame** is a **container** for **Series** objects.



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV file**.
- You could **load** a **DataFrame** from a **JSON file**.
- You could **load** a **DataFrame** from a **SQL database**.



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV** file.
- You could **load** a **DataFrame** from a **JSON** file.
- You could **load** a **DataFrame** from a **SQL database**.
(read_csv('path'))



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV** file.
- You could **load** a **DataFrame** from a **JSON** file.
- You could **load** a **DataFrame** from a **SQL database**.

• **read-json ('path')**



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV** file.
- You could **load** a **DataFrame** from a **JSON** file.
- You could **load** a **DataFrame** from a **SQL database**.

Pandas \Rightarrow Psycopgi
Query \Rightarrow Select
Excel \Rightarrow Pyxls
read_excel



Date Time Handling in Pandas

- You could convert a **string** to a **datetime** object using the **to_datetime()** method.
- You could **convert** a **datetime** object to a **string** using the **strftime()** method.
- You could **convert** a **datetime** object to a **timestamp** using the **timestamp()** method.



DateTime Handling in Pandas

- You could convert a **string** to a **datetime** object using the `to_datetime()` method.
- You could convert a **datetime** object to a **string** using the `strftime()` method.
- You could convert a **datetime** object to a **timestamp** using the `timestamp()` method.



DateTime Handling in Pandas

- You could convert a **string** to a **datetime** object using the **to_datetime()** method.
- You could convert a **datetime** object to a **string** using the **strftime()** method
- You could convert a **datetime** object to a **timestamp** using the **timestamp()** method.

EPOCH time
seconds from
Jan 1 1970



Querying a DataFrame

- You could **query** a **DataFrame** using **indexing (boolean or fancy)**.
- You could **query** a **DataFrame** using **loc and iloc indexers**. *(overwritten)*
- You could **query** a **DataFrame** using **the Query method**.



Querying a DataFrame

- You could `query` a **DataFrame** using `indexing` (`boolean` or `fancy`).
- You could `query` a **DataFrame** using `loc` and `iloc` indexers.
- You could `query` a **DataFrame** using `query` method.



Querying a DataFrame

- You could `query` a **DataFrame** using `indexing` (`boolean` or `fancy`).
- You could `query` a **DataFrame** using `loc` and `iloc` indexers.
- You could `query` a **DataFrame** using `query` method.



Missing Values in a DataFrame

- You could **detect** missing values in a **DataFrame**. The `isnull()` method returns a **Boolean DataFrame** indicating the **presence** of missing values.
- You could **fill** missing values in a **DataFrame**. The `fillna()` method returns a **DataFrame** with missing values filled.
- You could **drop** missing values in a **DataFrame**. The `dropna()` method returns a **DataFrame** with missing values dropped.



Missing Values in a DataFrame

- You could **detect** missing values in a **DataFrame**. The `isnull()` method returns a **Boolean DataFrame** indicating the **presence** of missing values.
- You could **fill** missing values in a **DataFrame**. The `fillna()` method returns a **DataFrame** with missing values filled.
- You could **drop** missing values in a **DataFrame**. The `dropna()` method returns a **DataFrame** with missing values dropped.



Missing Values in a DataFrame

- You could **detect** missing values in a **DataFrame**. The `isnull()` method returns a **Boolean DataFrame** indicating the **presence** of missing values.
- You could **fill** missing values in a **DataFrame**. The `fillna()` method returns a **DataFrame** with missing values filled.
- You could **drop** missing values in a **DataFrame**. The `dropna()` method returns a **DataFrame** with missing values dropped.



Merging DataFrames

- You could **merge** two **DataFrames** using the **merge()** method.
- You could **concatenate** two **DataFrames** using the **concat()** method.
- You could **join** two **DataFrames** using the **join()** method.



Merging DataFrames

- You could **merge** two **DataFrames** using the **merge()** method.
- You could **concatenate** two **DataFrames** using the **concat()** method.
- You could **join** two **DataFrames** using the **join()** method.



Merging DataFrames

- You could **merge** two **DataFrames** using the **merge()** method.
- You could **concatenate** two **DataFrames** using the **concat()** method.
- You could **join** two **DataFrames** using the **join()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the `groupby()` method.
- You could **aggregate** a **DataFrame** using the `agg()` method.
- You could **transform** a **DataFrame** using the `transform()` method.
- You could **filter** a **DataFrame** using the `filter()` method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



Scales DataFrame

- You could **pivot** a **DataFrame** using the **pivot()** method.
- You could **pivot table** a **DataFrame** using the **pivot_table()** method.



Scales DataFrame

- You could **pivot** a **DataFrame** using the `pivot()` method.
- You could **pivot table** a **DataFrame** using the `pivot_table()` method.



Pandas Idioms

- You could **apply functions** to **DataFrames**.
- You could **chain methods** in **Pandas**.



Pandas Idioms

- You could **apply functions** to **DataFrames**.
- You could **chain methods** in **Pandas**.



Basic Statistical Testing

You could perform statistical tests as:

- t-tests using the `ttest_ind()` method from the `scipy` library.
- ANOVA using the `f_oneway()` method from the `scipy` library.
- Chi-square using the `chi2_contingency()` method from the `scipy` library.
- Correlation using the `corr()` method from the `pandas` library.



Basic Statistical Testing

You could perform statistical tests as:

- t-tests using the `ttest_ind()` method from the `scipy` library.
- ANOVA using the `f_oneway()` method from the `scipy` library.
- Chi-square using the `chi2_contingency()` method from the `scipy` library.
- Correlation using the `corr()` method from the `pandas` library.



Basic Statistical Testing

You could perform statistical tests as:

- t-tests using the `ttest_ind()` method from the `scipy` library.
- ANOVA using the `f_oneway()` method from the `scipy` library.
- Chi-square using the `chi2_contingency()` method from the `scipy` library.
- Correlation using the `corr()` method from the `pandas` library.



Basic Statistical Testing

You could perform statistical tests as:

- t-tests using the `ttest_ind()` method from the `scipy` library.
- ANOVA using the `f_oneway()` method from the `scipy` library.
- Chi-square using the `chi2_contingency()` method from the `scipy` library.
- Correlation using the `corr()` method from the `pandas` library.



p-hacking and p-value

- **p-hacking** is the practice of manipulating the data and analysis of statistical tests to produce significant results.
- The p-value is the probability of obtaining an effect at least as extreme as the one in your sample data, assuming the null hypothesis is true.



p-hacking and p-value

- **p-hacking** is the practice of manipulating the data and analysis of statistical tests to produce significant results.
- The **p-value** is the probability of obtaining an effect at least as extreme as the one in your sample data, assuming the null hypothesis is true.



Goodhart's Law

Definition

When a measure becomes a target, it ceases to be a good measure.



Outline

- 1 Numerical Analysis with Numpy
- 2 Text Analysis and Regular Expressions
- 3 Data Manipulation with Pandas



Thanks!

Questions?



Repo: <https://github.com/EngAndres/ud-public/tree/main/courses/data-science-introduction>

