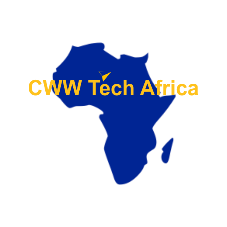
****

**CWW TECH AFRICA COHORT 4.0 PROGRAM**

**Week 2: Data Cleaning and Exploration**

**Lesson 1: Exploratory Data Analysis (EDA)**

**TOPIC ONE:**

**Data Visualization using Python**

**Introduction:**

Data visualization is a powerful technique in exploratory data analysis (EDA) that helps in understanding and communicating patterns, trends, and insights from data. Python provides various libraries and tools for creating informative and visually appealing visualizations. In this lesson, we will explore the basics of data visualization using Python.

**1. Matplotlib:**

Matplotlib is a widely used plotting library in Python. It provides a flexible and comprehensive set of functions for creating static, animated, and interactive visualizations. Some key concepts to understand in Matplotlib include:

- Creating basic plots: line plots, scatter plots, bar plots, histograms, etc.

- Customizing plot appearance: labels, titles, colors, markers, etc.

- Working with multiple subplots: creating a grid of plots

- Saving plots to files

**2. Seaborn:**

Seaborn is built on top of Matplotlib and offers a higher-level interface for creating aesthetic and informative statistical graphics. It provides a wide range of built-in themes and color palettes for creating visually appealing visualizations. Some key features of Seaborn include:

- Categorical plots: bar plots, count plots, box plots, etc.

- Statistical plots: distribution plots, regression plots, etc.

- Matrix plots: heatmap, clustermap, etc.

- Facet grids: creating multiple plots based on subsets of data

**3. Plotly**:

Plotly is a powerful library for creating interactive visualizations. It offers a wide range of chart types and provides interactive features like zooming, panning, and hovering. Plotly visualizations can be embedded in web applications or saved as standalone HTML files. Key features of Plotly include:

- Creating interactive line plots, scatter plots, bar plots, etc.

- Customizing visual appearance and adding annotations

- Creating 3D plots, contour plots, and surface plots

- Creating interactive dashboards using Dash framework

**4. Pandas:**

Pandas, in addition to its data manipulation capabilities, provides basic plotting functions based on Matplotlib. These functions allow for quick and convenient visualization of data stored in Pandas DataFrames. Key plotting functions in Pandas include:

- Line plots, bar plots, area plots

- Histograms, density plots, box plots

- Scatter plots, hexbin plots

**5. Other Libraries:**

There are several other libraries available in Python for data visualization, including:

- Plotnine: A Python implementation of the ggplot2 library in R.

- Bokeh: A library for interactive visualizations that target modern web browsers.

- Altair: A declarative statistical visualization library that creates interactive visualizations.

**Conclusion:**

Data visualization is an essential part of exploratory data analysis. Python provides a rich ecosystem of libraries and tools for creating informative and visually appealing visualizations. Matplotlib, Seaborn, Plotly, and Pandas are some of the popular libraries used for data visualization in Python. By leveraging these libraries, you can effectively communicate insights and patterns from your data.

**Lesson 2: Sorting and Filtering**

**TOPIC ONE:**

**Custom Sorting, Sorting in Ascending and Descending Orders**

**Introduction:**

Sorting and filtering are fundamental operations in data analysis that allow us to organize and extract meaningful information from datasets. In this lesson, we will focus on custom sorting and sorting data in ascending and descending orders using Python.

**1. Sorting Data in Ascending Order:**

Sorting data in ascending order arranges the values in a dataset from the smallest to the largest. This can be achieved using built-in functions or methods in Python. Some commonly used methods for sorting data in ascending order include:

- `sorted()`: This function returns a new sorted list from an iterable.

- `sort()`: This method sorts a list in place, meaning it modifies the original list directly.

- `DataFrame.sort\_values()`: This method in pandas sorts a DataFrame based on one or more columns.

**2. Sorting Data in Descending Order**:

Sorting data in descending order arranges the values in a dataset from the largest to the smallest. The same methods mentioned above can be used, but with an additional argument specifying the sorting order. For example:

- `sorted(iterable, reverse=True)`: This sorts the iterable in descending order.

- `sort(reverse=True)`: This sorts a list in descending order.

- `DataFrame.sort\_values(ascending=False)`: This sorts a DataFrame in descending order.

**3. Custom Sorting:**

Custom sorting allows you to define your own criteria for sorting data. This is useful when the default sorting order doesn't meet your specific requirements. Python provides options to implement custom sorting using functions or lambda expressions. The key parameter in the sorting functions allows you to specify a custom sorting key based on a specific attribute or criteria.

- `sorted(iterable, key=function)`: This sorts the iterable based on the result of the function applied to each element.

- `sort(key=function)`: This sorts a list based on the result of the function applied to each element.

- `DataFrame.sort\_values(by=column, key=function)`: This sorts a DataFrame based on a column using a custom sorting key.

By defining your own sorting key function, you can specify complex sorting criteria such as sorting by multiple columns or sorting based on custom calculations.

**Conclusion:**

Sorting data is an essential operation in data analysis, allowing you to organize and understand your dataset better. Python provides various methods and functions for sorting data in ascending and descending orders. Additionally, custom sorting enables you to define your own sorting criteria to meet specific requirements. By mastering these techniques, you can effectively sort and arrange data to gain valuable insights from your datasets.

**TOPIC 2:**

**Advanced Filters**

**Introduction:**

In addition to basic filtering operations, advanced filtering techniques allow us to extract specific subsets of data based on complex conditions or criteria. These techniques provide more flexibility and control over the filtering process. In this lesson, we will explore advanced filtering methods and their application in Python.

**1. Boolean Indexing:**

Boolean indexing is a powerful technique in filtering that involves using boolean conditions to select specific rows or columns from a dataset. With boolean indexing, you can create complex filtering conditions using logical operators (e.g., AND, OR, NOT) and comparison operators (e.g., >, <, ==) to filter the data. In Python, boolean indexing can be performed using various libraries, such as NumPy and Pandas.

**2. Filtering with Multiple Conditions:**

When filtering data, you may need to apply multiple conditions simultaneously. Python provides logical operators such as `&` (AND), `|` (OR), and `~` (NOT) to combine multiple conditions. By using these operators, you can construct complex filtering conditions and extract subsets of data that satisfy all or any of the specified conditions.

**3. String Filters:**

In some cases, you may need to filter data based on specific string patterns or substrings. Python offers various methods and functions for string filtering, such as:

- `str.contains()`: This method in Pandas allows you to filter rows based on whether a string column contains a specific substring.

- Regular expressions (regex): Regular expressions provide a powerful way to search and filter strings based on patterns. Python's `re` module provides functions for working with regular expressions.

**4. Date and Time Filters:**

Filtering data based on date and time is a common requirement in data analysis. Python provides libraries like Pandas and NumPy that offer functions and methods for working with dates and times. Some useful techniques for date and time filtering include:

- Filtering by specific dates or time ranges

- Extracting specific components of dates and times (e.g., year, month, day, hour)

- Filtering based on day of the week or time of the day

**5. Advanced Numeric Filters:**

For numeric data, advanced filtering techniques can involve applying complex mathematical or statistical conditions. Some examples include:

- Filtering based on ranges (e.g., values between a minimum and maximum threshold)

- Filtering outliers using statistical measures such as mean and standard deviation

- Filtering based on mathematical operations or calculations

**Conclusion:**

Advanced filtering techniques provide greater flexibility and control in extracting specific subsets of data based on complex conditions or criteria. Boolean indexing, filtering with multiple conditions, string filtering, date and time filtering, and advanced numeric filtering are some of the methods used to perform advanced filtering in Python. By leveraging these techniques, you can efficiently filter and extract relevant information from your datasets in data analysis tasks.

**Lesson 3: Cleaning Functions**

**TOPIC ONE**:

**Mid, Right, and Left Functions**

**Introduction:**

In data cleaning, it is often necessary to manipulate text or extract specific portions of a string. Functions like MID, RIGHT, and LEFT are commonly used for these purposes. In this lesson, we will explore these functions and their application in data cleaning tasks.

1. MID Function:

The MID function is used to extract a substring from a given string, starting at a specified position and with a specified length. In most programming languages, including Python, the equivalent function is called `slice`. The syntax for the MID function is as follows:

- MID(string, start\_position, length)

For example, if you have a string "Hello, World!" and want to extract the substring "World," you would use the MID function as follows:

- MID("Hello, World!", 8, 5)

**2. RIGHT Function:**

The RIGHT function is used to extract a specified number of characters from the right side of a string. This can be useful when you want to extract a portion of a string that appears at the end. The syntax for the RIGHT function is as follows:

- RIGHT(string, length)

For example, if you have a string "Data Analysis" and want to extract the last 7 characters "Analysis," you would use the RIGHT function as follows:

- RIGHT("Data Analysis", 7)

**3. LEFT Function:**

The LEFT function is used to extract a specified number of characters from the left side of a string. This can be useful when you want to extract a portion of a string that appears at the beginning. The syntax for the LEFT function is as follows:

- LEFT(string, length)

For example, if you have a string "Canada is amazing" and want to extract the first 6 characters "Canada," you would use the LEFT function as follows:

- LEFT("Canada is amazing", 6)

These functions are commonly used in data cleaning tasks, such as extracting specific parts of text, removing unwanted characters, or creating new columns based on existing strings.

Conclusion:

The MID, RIGHT, and LEFT functions are valuable tools for manipulating and extracting substrings from text data. They are commonly used in data cleaning tasks to extract specific portions of strings or manipulate text based on position and length. By mastering these functions, you can efficiently clean and transform text data in various data analysis projects.

**Lesson 3: Cleaning Functions**

**TOPIC TWO:**

**IF Error, IFNA, ISNA, ISTEXT, ISNUMBER, ISODD, ISREF, ISTEXT Functions**

**Introduction:**

In data cleaning and analysis, it is common to perform conditional checks on data to handle errors, missing values, or to categorize data based on specific criteria. Functions like IF Error, IFNA, ISNA, ISTEXT, ISNUMBER, ISODD, ISREF, and ISTEXT are useful for these tasks. In this lesson, we will explore these functions and their applications in data cleaning and analysis.

**1. IF Error Function:**

The IF Error function is used to handle errors that may occur during calculations or data transformations. It allows you to specify an alternative value or action if an error occurs. The syntax for the IF Error function is as follows:

- IFERROR(value, value\_if\_error)

For example, if you have a division operation and want to handle the potential division by zero error, you can use the IF Error function as follows:

- IFERROR(dividend / divisor, "Error: Division by zero")

**2. IFNA Function:**

The IFNA function is used to handle #N/A (not available) errors that may occur in data. It allows you to specify an alternative value or action if the value is #N/A. The syntax for the IFNA function is as follows:

- IFNA(value, value\_if\_na)

For example, if you have a lookup operation and want to handle the case where the value is not found, you can use the IFNA function as follows:

- IFNA(VLOOKUP(lookup\_value, lookup\_range, column\_index, False), "Not Found")

**3. ISNA Function:**

The ISNA function is used to check if a value is #N/A. It returns True if the value is #N/A; otherwise, it returns False. The syntax for the ISNA function is as follows:

- ISNA(value)

**4. ISTEXT Function:**

The ISTEXT function is used to check if a value is text. It returns True if the value is text; otherwise, it returns False. The syntax for the ISTEXT function is as follows:

- ISTEXT(value)

**5. ISNUMBER Function:**

The ISNUMBER function is used to check if a value is a number. It returns True if the value is a number; otherwise, it returns False. The syntax for the ISNUMBER function is as follows:

- ISNUMBER(value)

**6. ISODD Function:**

The ISODD function is used to check if a value is an odd number. It returns True if the value is odd; otherwise, it returns False. The syntax for the ISODD function is as follows:

- ISODD(value)

**7. ISREF Function:**

The ISREF function is used to check if a value is a reference. It returns True if the value is a reference; otherwise, it returns False. The syntax for the ISREF function is as follows:

- ISREF(value)

These functions are commonly used in data cleaning and analysis tasks to handle errors, check data types, categorize data, or perform conditional checks based on specific criteria.

**Conclusion:**

Functions like IF Error, IFNA, ISNA, ISTEXT, ISNUMBER, ISODD, ISREF, and ISTEXT are valuable tools in data cleaning and analysis. They help handle errors, identify missing values, check data types, and perform conditional checks on data. By utilizing these functions effectively, you can clean and analyze data more efficiently and make informed decisions based on specific criteria or conditions.