Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

HealthifyMe

How do dictionaries work, and what are their typical applications in data analytics?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

How do dictionaries work, and what are their typical applications in data analytics?

Dictionaries in Python are a versatile and fundamental data structure that work as key-value pairs. Each key is associated with a value, and dictionaries allow you to store, retrieve, and manipulate data efficiently.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

How Dictionaries Work:

Key-Value Pairs: Dictionaries consist of key-value pairs. Keys are unique and immutable, which means they cannot be changed once assigned. Values can be of any data type and may be duplicated.

Efficient Lookup: Dictionaries use a hashing mechanism to provide fast and efficient access to values based on their keys. This makes dictionary lookups (retrieval of values by key) very quick, even for large datasets.

Dynamic: Dictionaries are dynamic, meaning you can add, modify, or remove key-value pairs during runtime.

Unordered: In Python versions before 3.7, dictionaries are unordered, meaning they don't preserve the order of elements. However, since Python 3.7, dictionaries maintain insertion

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

Applications of Data Analytics:

Data Aggregation: Dictionaries are commonly used for aggregating and summarizing data. For example, you can use a dictionary to count the occurrences of unique items in a dataset.

```
In [1]: data = ["apple", "banana", "apple", "orange", "banana"]
        item_counts = {}
        for item in data:
            if item in item_counts:
                item_counts[item] += 1
            else:
                item_counts[item] = 1
        print(item_counts)

        {'apple': 2, 'banana': 2, 'orange': 1}
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

Applications of Data Analytics:

Data Transformation: Dictionaries are helpful for transforming data. You can create lookup tables or mapping dictionaries to convert values from one representation to another.

```
In [4]: conversion = {"male": 0, "female": 1}
        gender_data = ["male", "female", "male", "female"]
        transformed_data = [conversion[gender] for gender in gender_data]

In [5]: transformed_data

Out[5]: [0, 1, 0, 1]
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

Applications of Data Analytics:

Data Grouping : Dictionaries are suitable for grouping and organizing data. You can group data based on specific attributes or criteria, creating a hierarchical structure.

```
In [6]: data = [
            {"name": "John", "age": 30},
            {"name": "Jane", "age": 25},
            {"name": "Bob", "age": 30},
        ]
        grouped_data = {}
        for entry in data:
            age = entry["age"]
            if age not in grouped_data:
                grouped_data[age] = []
            grouped_data[age].append(entry)

In [7]: grouped_data

Out[7]: {30: [{'name': 'John', 'age': 30}, {'name': 'Bob', 'age': 30}],
         25: [{'name': 'Jane', 'age': 25}]}
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 1

Data Transformation and Aggregation in Pandas: In data analysis with Pandas (a popular library for data analytics), DataFrames (tabular data structures) often use dictionaries for column renaming, data aggregation, and grouping operations.

```
In [8]: import pandas as pd

        data = pd.DataFrame({
            "Name": ["John", "Jane", "Bob"],
            "Age": [30, 25, 30]
        })

        # Renaming columns using a dictionary
        data.rename(columns={"Name": "Full Name", "Age": "Years Old"}, inplace=True)

        # Aggregating data using a dictionary
        aggregated_data = data.groupby("Years Old").size().reset_index(name="Count")
```

```
In [9]: aggregated_data
```

Out[9]:

	Years Old	Count
0	25	1
1	30	2

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

What is a Python module, and how do you import it into your code?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

What is a Python module, and how do you import it into your code?

In Python, a module is a file containing Python code, including variables, functions, and classes, that can be used in other Python scripts or programs.

Modules help organize code into reusable and manageable units, making it easier to maintain and scale projects. Modules can be part of the Python standard library, provided by Python itself, or created by users.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Here's how you import a Python module into your code:

Importing a Standard Library Module:

Standard library modules are included with Python. You can import them directly into your code using the import keyword.

```
In [1]: import math  # Import the math module

        # Use functions and constants from the math module
        print(math.sqrt(16))  # Calculate the square root
        print(math.pi)        # Access the value of pi
```

```
4.0
3.141592653589793
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Importing Specific Items from a Module:

You can import specific functions, variables, or classes from a module instead of importing the entire module. This is useful when you only need certain parts of a module.

```
In [2]: from math import sqrt, pi  # Import specific items from the math module

        # Use the imported items directly
        print(sqrt(16))
        print(pi)
```

```
4.0
3.141592653589793
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Using an Alias:

You can provide an alias or nickname for a module or its imported items using the as keyword. This can be helpful for shorter or more meaningful names.

```
In [3]: import numpy as np  # Import the numpy module with an alias 'np'

        # Use the numpy module as 'np' in your code
        arr = np.array([1, 2, 3])
```

```
In [4]: arr

Out[4]: array([1, 2, 3])
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 2

Importing User-Defined Modules:

```
In [12]: #my_module.py
         def my_function():
             return "Hello from my_module!"
         my_variable = 42
```

```
In [ ]: import my_module

        print(my_module.my_function())
        print(my_module.my_variable)
```

```
In [ ]: from my_module import my_function, my_variable

        print(my_function())
        print(my_variable)
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

What is the scope of a variable in Python? Explain the difference between global and local variables within a function.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

What is the scope of a variable in Python? Explain the difference between global and local variables within a function.

In Python, the scope of a variable refers to where in the code that variable can be accessed or modified. Python has several levels of variable scope, primarily divided into two main categories: global and local scope.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

Global Scope:

- A variable defined outside of all functions or code blocks has global scope.
- It can be accessed and modified from anywhere in the code, both inside and outside functions.
- Global variables are usually declared at the top level of a script or module.
- They persist throughout the program's lifetime.

```
In [15]: global_variable = 10

         def my_function():
             print(global_variable)

         my_function()  # This will print 10

         10
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

Local Scope:

- A variable defined within a function has local scope.
- It can only be accessed and modified within that specific function.
- Local variables are created when the function is called and destroyed when the function exits.
- Variables with the same name can exist in different functions without conflicts, as they have independent local scopes.

```
In [16]: def my_function():
             local_variable = 5
             print(local_variable)

         my_function()  # This will print 5

5
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 3

Difference between Global and Local Variables:

Accessibility : Global variables are accessible from anywhere in the code, both inside and outside functions. Local variables are only accessible within the function in which they are defined.

Lifetime : Global variables persist throughout the program's execution.
Local variables have a shorter lifespan; they are created when the function is called and destroyed when the function exits.

Use Cases : Global variables are typically used for constants, configuration settings, or data that needs to be accessed globally.
Local variables are used for temporary storage within a function and help encapsulate data to avoid unintended side effects.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

Describe the purpose of the lambda function (anonymous function) in Python, and provide an example of how you might use it in data analysis.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

Describe the purpose of the lambda function (anonymous function) in Python, and provide an example of how you might use it in data analysis.

- A lambda function in Python is an anonymous, small, and inline function that is defined without a formal name. Lambda functions are also known as "anonymous functions" because they don't require a name like regular functions defined using the def keyword.

- Lambda functions are typically used for short, simple operations, and they are defined using the lambda keyword.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

The syntax of a lambda function is as follows:

```
In [17]: lambda arguments: expression

Out[17]: <function __main__.<lambda>(arguments)>
```

lambda: The keyword used to define a lambda function.

arguments: The input arguments or parameters that the lambda function takes.

expression: The single expression or operation that the lambda function performs. The result of this expression is returned by the lambda function.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

Purpose of Lambda Functions:

Short, Simple Operations: Lambda functions are handy when you need to define a small, one-off function for a brief operation without the need to create a named function.

Functional Programming: Lambda functions are often used in functional programming paradigms, where functions can be treated as first-class citizens, allowing functions to be passed as arguments to other functions.

Anonymous Callbacks: Lambda functions are commonly used as callbacks for functions that accept functions as arguments, such as map(), filter(), and sort().

Concise Code: Lambda functions can make code more concise and readable when used appropriately.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 4

Example of Using Lambda in Data Analysis:

Let's say you have a list of tuples, where each tuple represents a person's name and age. You want to sort the list based on the age of the individuals. You can use a lambda function as a key for the sorted() function to achieve this:

```
In [18]: people = [("Alice", 30), ("Bob", 25), ("Charlie", 35), ("David", 28)]

         # Sort the list of people by age using a lambda function as the key
         sorted_people = sorted(people, key=lambda person: person[1])

         print(sorted_people)
```

```
[('Bob', 25), ('David', 28), ('Alice', 30), ('Charlie', 35)]
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

How can you pass a variable number of arguments to a Python function, and what is the purpose of the *args and **kwargs syntax?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

How can you pass a variable number of arguments to a Python function, and what is the purpose of the *args and **kwargs syntax?

In Python, you can pass a variable number of arguments to a function using *args and **kwargs syntax. These are special constructs that allow you to handle positional and keyword arguments of arbitrary length in a function.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

*args (Arbitrary Positional Arguments):

The *args syntax allows a function to accept an arbitrary number of positional arguments. These arguments are collected into a tuple, which you can then iterate through or use in various ways within the function.

```
In [21]: def example_function(arg1, *args):
             print(arg1)      # Print the first argument
             print(args)      # Print the tuple of additional positional arguments

         example_function(1, 2, 3, 4)
         # Output:
         # 1
         # (2, 3, 4)
```

```
1
(2, 3, 4)
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

**kwargs (Arbitrary Keyword Arguments):

The **kwargs syntax allows a function to accept an arbitrary number of keyword arguments. These arguments are collected into a dictionary, which you can then access and manipulate within the function.

In [22]:
```
def example_function(arg1, **kwargs):
    print(arg1)         # Print the first argument
    print(kwargs)       # Print the dictionary of additional keyword arguments

example_function(1, a=2, b=3, c=4)
# Output:
# 1
# {'a': 2, 'b': 3, 'c': 4}
```

```
1
{'a': 2, 'b': 3, 'c': 4}
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 5

Purposes of *args and **kwargs:

Flexibility: *args and **kwargs provide flexibility when defining functions, allowing them to accept varying numbers of arguments without explicitly specifying all argument names.

Wrapper Functions: They are often used when creating wrapper functions or decorators that need to accept arguments of unknown quantity and pass them along to another function.

Polymorphism: They support function overloading or polymorphism, where a single function can take different forms based on the arguments it receives.

Dynamic Functionality: They are useful in situations where you need to add new arguments to a function without breaking existing code that calls the function.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6

Explain the concept of encapsulation in OOP. How does it help in data analysis projects?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6

Explain the concept of encapsulation in OOP. How does it help in data analysis projects?

Encapsulation is one of the four fundamental principles of Object-Oriented Programming (OOP), along with abstraction, inheritance, and polymorphism. It refers to the bundling of data (attributes) and methods (functions) that operate on that data into a single unit called a class. Encapsulation restricts direct access to some of the object's components, which helps prevent the accidental modification of data, enforces data integrity, and promotes code organization and maintenance.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6

How Encapsulation Helps in Data Analysis Projects:

Data Integrity: In data analysis, maintaining data integrity is crucial. Encapsulation allows you to control how data is modified and accessed, reducing the risk of unintentional data corruption.

Abstraction: Encapsulation promotes abstraction by hiding complex implementation details. Data analysts can work with high-level interfaces (methods) without needing to understand the underlying complexity of data storage and manipulation.

Modularity: Encapsulation encourages code modularity. Data analysis projects often involve multiple data sources, transformations, and analyses. Classes can encapsulate related functionality, making the codebase more organized and maintainable.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 6

How Encapsulation Helps in Data Analysis Projects:

Code Reusability: Encapsulation facilitates code reusability. You can create classes with well-defined attributes and methods, making it easier to reuse those components in different parts of your analysis.

Security: While Python's encapsulation relies on conventions, it still helps improve security. Private and protected attributes and methods discourage unauthorized access and modification of data, enhancing data security.

encapsulation in OOP promotes data integrity, code organization, and abstraction, all of which are beneficial in data analysis projects. By encapsulating data and methods into classes, data analysts can work more efficiently, maintain code more easily, and ensure the reliability of their analyses.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 7

What is the difference between the sort() and sorted() functions when it comes to sorting lists?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 7

sort() Method:

In-place Sorting: The sort() method is an in-place sorting method, which means it modifies the original list and does not create a new one.

Usage: It is called on an existing list and sorts the elements of that list in ascending order by default.

Return Value: sort() returns None, so it cannot be used in expressions to assign a sorted list to a new variable.

```
In [1]: numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
        numbers.sort()

In [2]: numbers

Out[2]: [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 7

sorted() Function:

New List: The sorted() function returns a new sorted list based on the elements of the original list. It does not modify the original list.

Usage: It is called as a built-in function and can be applied to any iterable, not just lists.

Return Value: sorted() returns a new sorted list, which can be assigned to a variable.

```
In [4]: numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
        sorted_numbers = sorted(numbers)

In [5]: sorted_numbers

Out[5]: [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 7

```
In [7]: numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
        numbers.sort()
        print(numbers)

        [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
```

```
In [8]: numbers = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
        sorted_numbers = sorted(numbers)
        print(sorted_numbers)
        print(numbers)

        [1, 1, 2, 3, 3, 4, 5, 5, 5, 6, 9]
        [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5]
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 8

How can you use Python lists to store and manipulate data in a data analysis project?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 8

How can you use Python lists to store and manipulate data in a data analysis project?

Python lists can be a valuable tool for storing and manipulating data in a data analysis project, especially for smaller datasets or when you need a flexible and versatile data structure.

1. Data Storage:

Storing Raw Data: Python lists can be used to store raw data from various sources, including CSV files, databases, web scraping, or APIs. You can create lists to hold data records, rows, or observations.

Data Preprocessing: Lists are useful for storing intermediate data during preprocessing steps, such as data cleaning, transformation, and feature engineering.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 8

How can you use Python lists to store and manipulate data in a data analysis project?

2. Data Exploration:

Accessing Data: Lists provide easy access to data using indexing and slicing. You can quickly retrieve specific data points or subsets for exploration.

Summary Statistics: Lists can hold numerical data for calculating summary statistics like mean, median, standard deviation, and more. You can write custom functions or use libraries like NumPy for these calculations.

3. Data Visualization: Python libraries like Matplotlib and Seaborn can work directly with lists to create various types of data visualizations, including bar charts, histograms, scatter plots, and more.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 8

How can you use Python lists to store and manipulate data in a data analysis project?

4. Data Transformation:

Filtering Data: Lists can be used to filter data based on specific conditions. You can iterate through a list and apply filtering criteria to extract relevant data.

Aggregating Data: Lists are handy for aggregating data by grouping and summarizing. You can create sublists or dictionaries to organize data based on categories or attributes.

5. Data Export:

Exporting Data: Once you've processed and analyzed data using lists, you can export the results back to files, databases, or other storage formats for further use or sharing

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 8

How can you use Python lists to store and manipulate data in a data analysis project?

6. Flexibility:

Mixed Data Types: Lists can hold mixed data types (e.g., integers, strings, and floats) within the same list, making them flexible for handling diverse datasets.

Dynamic Size: Lists can dynamically grow or shrink in size as you add or remove elements, allowing you to adapt to changing data requirements.

7. Iteration:

Iterating Over Data: Lists are iterable, making it easy to loop through data and apply operations or calculations to each element.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

What is Jupyter Notebook, and why is it commonly used in data analysis projects?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

What is Jupyter Notebook, and why is it commonly used in data analysis projects?

Jupyter Notebook is an open-source, web-based interactive computing environment that is widely used in data analysis and scientific computing projects. It allows users to create and share documents that contain live code, equations, visualizations, and narrative text

Interactive Computing: Jupyter Notebook provides an interactive environment where users can write and execute code in a step-by-step manner. This interactivity is particularly valuable in data analysis, as it allows for quick experimentation and iterative development.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

Support for Multiple Languages: While Jupyter Notebook is often associated with Python, it supports multiple programming languages, including R, Julia, and more. This flexibility enables data analysts to use the language that best suits their needs.

Rich Documentation: Jupyter Notebooks allow users to combine code cells with Markdown text cells. This makes it easy to provide rich documentation, explanations, and commentary alongside code, enhancing the clarity and reproducibility of data analysis workflows.

Data Visualization: Jupyter Notebook supports the integration of data visualization libraries like Matplotlib, Seaborn, and Plotly. Analysts can create interactive charts and graphs directly within the notebook, facilitating data exploration and presentation.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

Integration with Data Science Ecosystem: Jupyter Notebook seamlessly integrates with popular data science libraries and tools, such as NumPy, Pandas, SciPy, Scikit-Learn, and TensorFlow, making it a powerful tool for end-to-end data analysis and machine learning workflows.

In-Browser Execution: Users can run Jupyter Notebook in a web browser, which eliminates the need for complex software installations and configurations. This accessibility makes it easy for individuals and teams to collaborate on data analysis projects.

Code Reusability: Jupyter Notebooks allow users to save and share their work. This feature is valuable for sharing analysis results, collaborating with colleagues, and reproducing research findings.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 10

What are regular expressions, and why are they useful in data analysis?

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

Regular expressions, often abbreviated as regex or regexp, are powerful and flexible patterns used to match and manipulate text in strings. They are a fundamental tool in text processing, and they find extensive use in data analysis for the following reasons:

1. Pattern Matching: Regular expressions allow you to specify complex patterns to search for and match within text data. This is extremely useful when you need to extract specific information from unstructured text, such as emails, log files, or web pages.

2. Text Cleaning: In data analysis, it's common to encounter messy or inconsistent text data. Regular expressions can be used to clean and preprocess text by removing unwanted characters, spaces, or formatting.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

```
In [ ]: import re
```

```
In [11]: text = "Chennai is a beautiful city. It's the capital of the state of Tamil Nadu.
Chennai has an area close to 430 kilometer squares.
Well chennai is not as large as mumbai which has an area of 603.4 kilometer squares.
By road, Chennai is about 1500 kilometers away from Mumbai. Whereas, it is about 2200
kilometers away from Delhi, the capital of India."
```

```
In [12]: cities_record = 'Chennai'
re.findall(cities_record, text)

Out[12]: ['Chennai', 'Chennai', 'Chennai']
```

```
In [13]: cities_record = 'Chennai'
re.findall(cities_record, text, flags=re.IGNORECASE)

Out[13]: ['Chennai', 'Chennai', 'chennai', 'Chennai']
```

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

3. Data Extraction: Regular expressions enable you to extract specific pieces of information from text, such as email addresses, phone numbers, dates, URLs, and more. This is valuable for parsing structured data from unstructured sources.

4. Text Analysis: For sentiment analysis, named entity recognition, and other text-based analysis tasks, regular expressions can help identify and classify text patterns, making it easier to analyze and interpret textual data.

5. Data Validation: Regular expressions are valuable for data validation and verification. You can use them to check if user input matches a specified format (e.g., validating email addresses, credit card numbers, or postal codes).

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

6. Data Transformation: In data preparation, regular expressions can transform text data into a structured format. For instance, you can reformat dates, convert units, or standardize text representations.

7. Search and Replace: Regular expressions can be used to find and replace specific text patterns in a document or dataset, which is handy for bulk text editing or correcting errors.

8. Efficient Data Parsing: When dealing with large text files, regular expressions provide an efficient way to parse and extract relevant information without the need for manual scanning or looping.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Interview Question - 9

9. Web Scraping: In web scraping and data collection from websites, regular expressions help locate and extract data from HTML or XML documents by specifying patterns that match desired content.

10. Natural Language Processing (NLP): In NLP tasks, regular expressions can be used to tokenize text into words or sentences, remove stopwords, or identify specific linguistic patterns.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Question - 1

1) What is the purpose of the __init__ method in a Python class?

A. It is a constructor method called when an object is created from the class.

B. It is used to initialize all instance variables.

C. It is used to destroy objects.

D. It is used to define class variables.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Question - 2

2) What is the primary purpose of the groupby() function in the Pandas library?

A. Sorting data

B. Filtering data

C. Aggregating data

D. Merging data

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Question - 3

3) Which Python library provides tools for working with dates and times, making it useful for time-series analysis?

A. Pandas

B. NumPy

C. DateTime

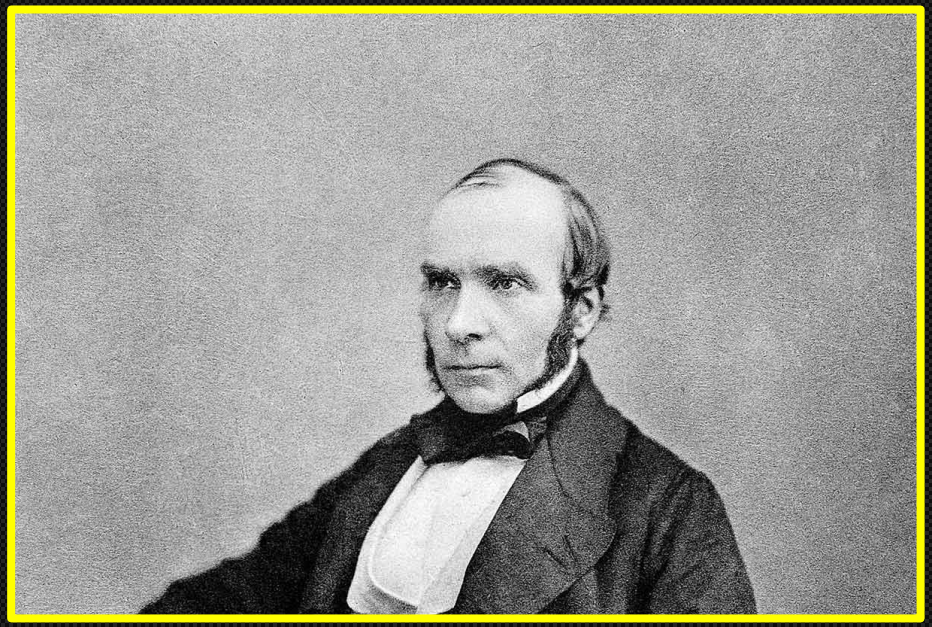
D. Calendar

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds



India's Most Affordable Pay After Placement Data Analytics Course

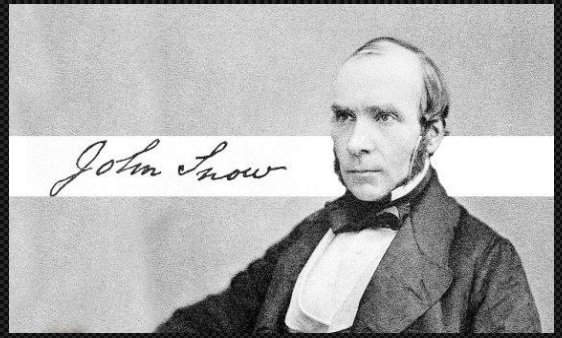
+91-7880-113-112 Contact or Fill the Form in the Description

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds



1) Dr. John Snow (1813-1858) was a British physician who is considered one of the founding figures of modern epidemiology.

2) He is best known for his pioneering work in identifying the source of a cholera outbreak in London in 1854. His investigation of the outbreak is often cited as a seminal moment in the history of public health and data analysis.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds

The 1832-33 cholera epidemic claimed 4,000 to 7,000 victims in London.

September 1853 that it was officially announced by the British authorities that a cholera epidemic was claiming victims not only in London but also in other parts of the country.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds

In 1854, London experienced a severe cholera epidemic, and the prevailing belief was that the disease was spread through "miasma" or foul air.

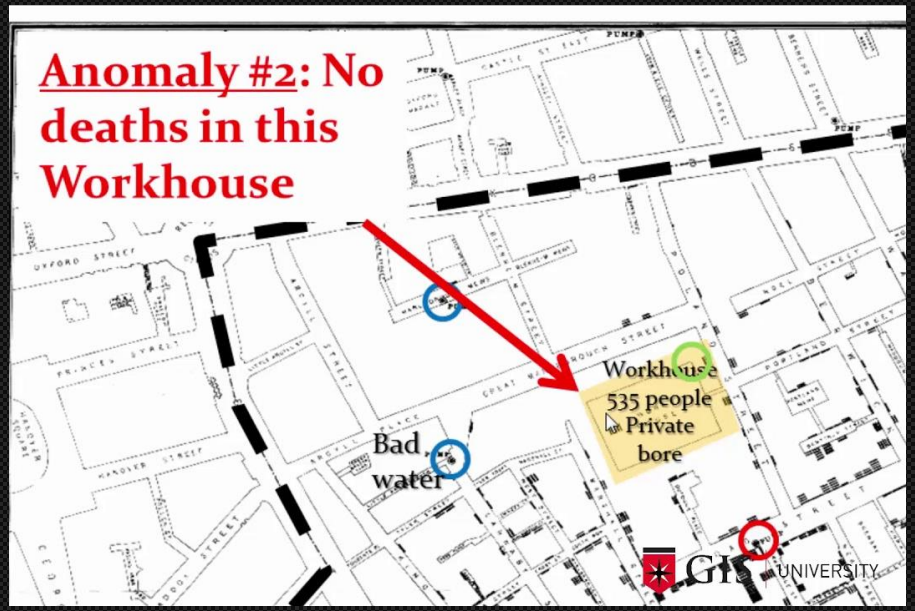
However, Dr. John Snow had a different hypothesis. He believed that contaminated water was the source of the outbreak.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds



To prove his theory, Dr. Snow collected data meticulously.

He plotted cases of cholera on a map of the affected area and noted the locations of water pumps.

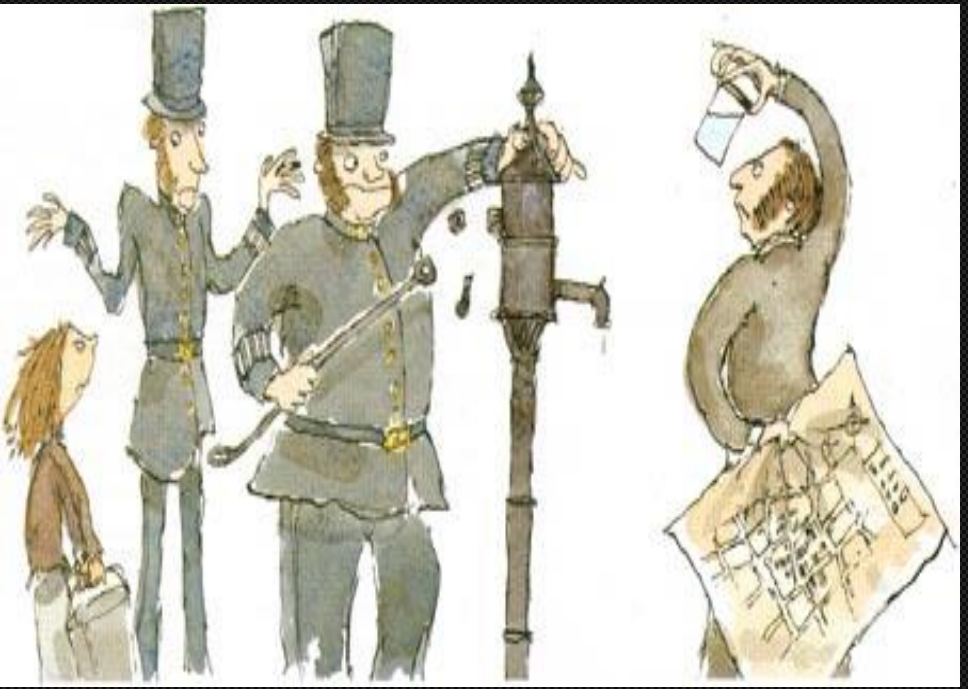
He found a significant cluster of cases around the Broad Street pump.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds



To confirm his hypothesis, he interviewed residents and discovered that the pump was indeed the primary source of contaminated water.

Dr. Snow's groundbreaking work, supported by compelling data analysis and mapping, led him to convince local authorities to remove the pump's handle, preventing further use.

The epidemic began to subside, and this event marked the first time that data analysis was used to identify the source of a disease outbreak.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds

John Snow's story not only highlights the importance of data analysis in solving real-world problems but also underscores the critical role of data-driven decision-making in public health.

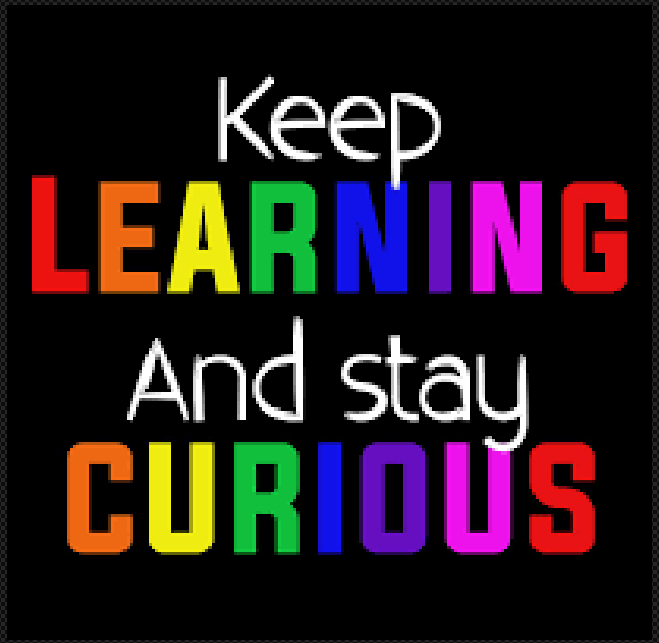
His work laid the foundation for modern epidemiology and public health practices, emphasizing the power of data in saving lives and preventing the spread of diseases.

Day - 1

10 Days Python Data Analytics Interview Class

Industries Helping Hands

Curious Data Minds



India's Most Affordable Pay After Placement Data Analytics Course

+91-7880-113-112 Contact or Fill the Form in the Description