Python Master Interview Sheet

Q.1)Which are the different python modules used in data analytics?

There are various modules which help in data manipulation such as data cleaning, understanding data distribution, analysing data, formatting data, visualizing data:

1. **NumPy**: Used for **scientific computation**.
2. **Pandas:** It is primarily used for **data analysis, data manipulation, and data cleaning**.
3. **Matplotlib:** extensive library for **creating fixed, interactive, and animated Python visualizations.**
4. **Seaborn:** popular Matplotlib-based Python **data visualization framework.**
5. **Scikit-Learn:** Used for applying machine learning algorithms such as classification, clustering, regression.
6. **Statsmodels:** Statsmodels provides classes and functions that allow users to estimate various statistical models, conduct statistical tests, and do statistical data exploration.

Q.2) Your client operates a leading online retail e-commerce site, and you are tasked with analysing a large dataset from their operations. You need to create a Python script for data manipulation and exploration. After loading the dataset into your script, you observe that there are multiple null values present in the dataset. How would you handle this scenario?

1. First step would be to analyse the missing values in the dataset such as, identify the missing values using “df.isnull().sum()”.
2. If the columns containing missing values are not so complex and their existence or non existence does not affect the analysis then it is easier to remove the null values using Pandas function “df.dropna()”.
3. Else if the null values can be guessed then there are several methods with which we can fill the null values in the dataset using the df.fillna() function.
4. But all this should be done with the permission of the client as the dataset is being modified here.

Q.3) How to perform merge operation between two pandas DataFrames?

A Pandas is a powerful library in Python used for data manipulation.

Merge is nothing but an operation to combine or joining two tables based on a common column so that you can see the whole information at a single place.

For performing this operation pandas function “merge()” can be used.

For example: -

```
import pandas as pd

#Sample Dataset

df_food = pd.DataFrame({'Name': ['Alice', 'Bob', 'Dave'], 'Favorite Food': ['Pizza', 'Pasta', 'Burgers']})

df_drink = pd.DataFrame({'Name': ['Alice', 'Bob', 'Claire'], 'Favorite Drink': ['Water', 'Soda', 'Wine']})

# Merging

df_merged = pd.merge(df_food, df_drink, on='Name', how='inner')
```

Q.4) How do you identify and remove duplicate rows from a DataFrame?

To identify duplicate rows in a DataFrame, use the df.duplicated() method, which returns a boolean Series indicating duplicate rows. To remove duplicate rows, use df.drop_duplicates().

Therefore functions: -

1. Df.duplicated()
2. Df.drop_duplicates().

Q.5) Explain the difference between loc and iloc in pandas with examples.

Indexing Method	Description	Example

loc	Label-based indexing. Allows access to rows and columns by their labels.	df.loc['a', 'A'] Accesses the row with label 'a' and column 'A'.
Iloc	Integer-location based indexing. Allows access to rows and columns by their integer positions.	df.iloc[0, 0] Accesses the first row and first column (both indices start at 0).

Q.6) How would you split your dataset using Python?

To split a dataset into training and testing sets, you can use the train_test_split function from the sklearn.model_selection module. This function randomly splits the data into specified proportions for training and testing.

Example

```
from sklearn.model_selection import train_test_split

# Assuming X is the feature set and y is the target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

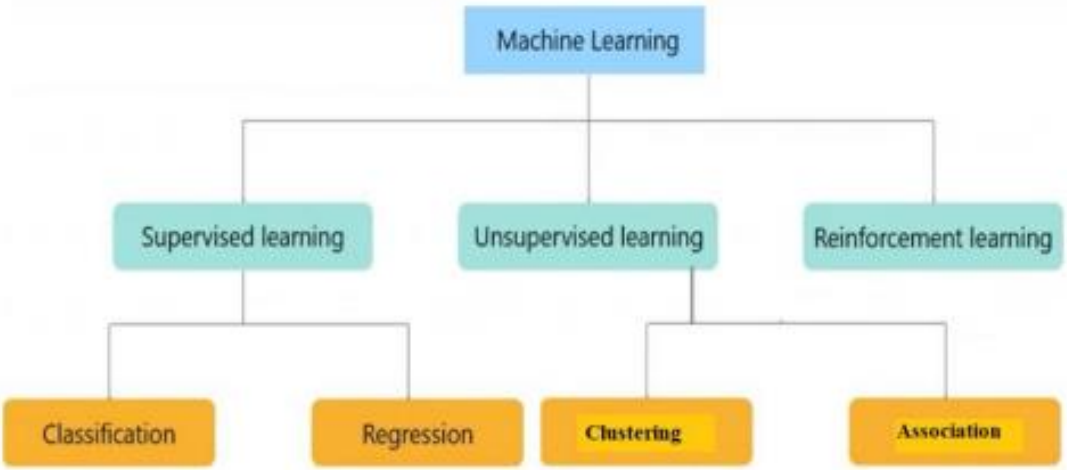
Q.7) What is Data abstraction?

➔ The hiding of the unnecessary code details from the user or hiding the sensitive parts of the code is where data abstraction comes into the picture.

➔ For example, when we create a variable with double underscore __variable_name then that variable behaves like a private variable.

Q.8) Do you have any idea on what is machine learning? Can you explain us the different types of machine learning?

Machine Learning is an application of Artificial intelligence. It is a concept to which allows machines to learn from the data and experience and take decisions with or minimal human support.



Q.9) Your client is having the data in an SQL database. Your job is to integrate the dataset into Python and analyse the dataset. What will be the steps you will perform in order to do that?

For integrating Python with SQL we have a library in python named as sqlite3

Consider the following example with which we can retrieve the data from sql into our python script.

```
import pandas as pd
import matplotlib.pyplot as plt
import sqlite3

# Connect to your database
conn = sqlite3.connect('mystery_database.db')

# Run the SQL query and load into a DataFrame
df = pd.read_sql_query("YOUR SQL QUERY HERE", conn)

# Plotting
df.plot(kind='bar', x='Month', y='CasesSolved')
plt.show()
```

Example of SQL Query: -

```sql
SELECT Month, COUNT(*) as CasesSolved
FROM CaseFiles
WHERE Status = 'Solved'
GROUP BY Month;
```

Q.10) What are outliers in a dataset and how will you analyze the outliers?

Outliers are data points that differ significantly from other observations in a dataset. They can result from variability in the data, measurement errors, or other factors and can affect statistical analyses.

There are various methods with which we can detect the outliers in the dataset: -

1. **Box Plot:** Visualize the distribution of data and identify outliers.
2. **Scatter Plot:** Detect outliers in two-dimensional data.
3. **Z-score:** Calculate the Z-score to identify outliers (values with Z-scores > 3 or < -3 are considered outliers).
4. **IQR (Interquartile Range):** Identify outliers using the IQR method (values below Q1 - 1.5*IQR or above Q3 + 1.5*IQR are considered outliers).

Q.11) Since you are familiar with python let me ask you a question related to object oriented programming. **What are classes and objects?**

➔ Class is a design or blueprint of an object. Example, I use Samsung Galaxy A31 smartphone, now there are 100's and 1000's of A31 phone out there but each A31 phone has been designed in the same way or using the same procedure. That design behaviour of an object is known as class. Therefore, class is a blueprint of an object or how an object should or can behave. Whereas object is an instance of class.

Q.12) **What is Inheritance? What are different types of Inheritance?**

➔ **Single Inheritance:**
Single-level inheritance enables a derived class to inherit characteristics from a single-parent class.

➔ **Multilevel Inheritance:**
Multi-level inheritance enables a derived class to inherit properties from an immediate parent class which in turn inherits properties from his parent class.

➔ **Hierarchical Inheritance:**
Hierarchical level inheritance enables more than one derived class to inherit properties from a parent class.

➔ **Multiple Inheritance:**
Multiple level inheritance enables one derived class to inherit properties from more than one base class.

Q.13) Explain me **Polymorphism?**

➔ The word polymorphism can be understood as poly which means many and morphism means “forms”. That is objects will have multiple forms. The ways to implement polymorphism is Duck Typing, Operator Overloading, Method overloading, Method overriding.

Q.14) **What is Abstract class and methods in python?**

➔ When a method with no body is created then the method is known as abstract method and the class which contains the abstract method is known as abstract classes. We cannot create object of the abstract classes.
➔ Python does not support the concept of abstract class, but we can apply the abstract class by **importing ABC, abstractmethod from the abc module**.
➔ To **define an abstract method**, we need to use the decorator **@abstractmethod** before the declaration statement of the method.
➔ **An abstract class will at least have 1 abstract method. A class can inherit an abstract class only when it has defined all the abstract methods of the abstract class.**

Q.15) **Describe a Typical Data Analysis Process**

- **Data Collection:** Gather relevant data from various sources. In Python, you can use the requests library to retrieve data from web servers by sending HTTP GET and POST You can also use web scraping using libraries like BeautifulSoup or Selenium.
- **Data Cleaning:** Remove errors, duplicates, and inconsistencies from the dataset. If you need to perform complex operations as you clean your data, you can use a Python data-cleaning library like NumPy or pandas.
- **Exploratory Data Analysis (EDA)**: Explore the dataset to understand its structure, patterns, and relationships. Python libraries such as pandas, NumPy, and Scikit-learn are excellent tools for performing EDA on a dataset.

- **Modeling:** Apply statistical techniques or machine learning algorithms to analyze the data. Python libraries like Scikit-learn, TensorFlow, and PyTorch provide a set of tools and algorithms for data modeling and machine learning tasks.
- **Visualization:** Present findings visually through charts, graphs, or dashboards. You can use Python data visualization libraries like Matplotlib or Seaborn to create plots and charts.

Q.16) **Explain The Concept of List Comprehension in Python**

List comprehension is a concise and powerful way to create lists in Python. It provides a compact syntax for generating lists and applying operations or conditions to each element.

General syntax:

```
new_list = [expression for item in iterable if condition]
```

Example:

The following list comprehension squares each element of a given list, excluding the number 1:

Code

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = [x**2 for x in numbers if x > 1]
print(squared_numbers) #Prints 4, 9, 16, 25
```

Q.17) What is a histogram, and how is it used in data analysis? Using Python libraries how would your create a histogram?

1. A histogram is a graphical representation of the distribution of numerical data. It consists of a series of bars, where each bar represents a range of values and the height of the bar represents the frequency of values within that range. Histograms are commonly used to visualize the frequency distribution of a dataset.
2. You can create a histogram using the `plt.hist()` function in Matplotlib. For example:

```
import matplotlib.pyplot as plt
plt.hist(data, bins=10)
```

Q.18) What is the purpose of data normalization in data analysis?

1. The purpose of data normalization is to rescale the values of numerical features to a common scale without distorting differences in the ranges of values. It is particularly useful in machine learning algorithms that require input features to be on a similar scale to prevent certain features from dominating others.
2. You can perform data normalization using the MinMaxScaler, StandardScaler, or RobustScaler classes in scikit -learn. For example:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)
```

Q.19) How do you perform data aggregation using Pandas?

You can perform data aggregation using the groupby() method in Pandas to group data based on one or more columns and then apply an aggregation function to compute summary statistics for each group. For example:

grouped = df.groupby('Name').mean()

Q.20) What is the purpose of the map() function in Python and its relevance in data analysis?
The map() function applies a given function to each item of an iterable and returns a list of the results. In data analysis, it's useful for applying functions element-wise to data structures like lists or Pandas Series.

Q.21) How do you calculate and interpret the skewness and kurtosis of a dataset

Skewness:

- Skewness measures the asymmetry of the data distribution. Positive skewness indicates a distribution with a long right tail, while negative skewness indicates a long left tail.
- Use pandas or scipy to calculate skewness:

```
from scipy.stats import skew
skewness = skew(df['column_name'])
```

```
print(f"Skewness: {skewness}")
```

Kurtosis:

- Kurtosis measures the "tailedness" of the data distribution. High kurtosis indicates heavy tails, while low kurtosis indicates light tails. Normal distribution has a kurtosis of 3 (excess kurtosis is 0).
- Use pandas or scipy to calculate kurtosis:

```
from scipy.stats import kurtosis

kurt = kurtosis(df['column_name'])

print(f"Kurtosis: {kurt}")
```

Interpreting Skewness and Kurtosis:

1. **Skewness:**
    - **Skewness ≈ 0:** The distribution is approximately symmetric.
    - **Skewness > 0:** The distribution is positively skewed (right-tailed).
    - **Skewness < 0:** The distribution is negatively skewed (left-tailed).
2. **Kurtosis:**
    - **Kurtosis < 3:** The distribution has light tails (platykurtic).
    - **Kurtosis = 3:** The distribution is normal (mesokurtic).
    - **Kurtosis > 3:** The distribution has heavy tails (leptokurtic).

Q.22) Can you explain the journey of a data analytics project along with an example?

Journey of a Data Analytics Project:

1. **Define the Objective:**
    - Determine the business problem or question to be addressed.
    - **Example:** An e-commerce company wants to reduce customer churn.
2. **Data Collection:**

- Gather relevant data from various sources such as databases, APIs, and external files.
- **Example:** Collect customer transaction data, website activity logs, and customer service interactions.

3. **Data Cleaning and Preparation:**
   - Handle missing values, remove duplicates, and correct errors.
   - Transform data into a suitable format for analysis.
   - **Example:** Remove duplicates, fill missing values with mean/mode, and standardize date formats.
4. **Exploratory Data Analysis (EDA):**
   - Analyze data distributions, identify patterns, and detect anomalies.
   - Visualize data using plots and charts.
   - **Example:** Use histograms to understand purchase frequency, scatter plots to explore correlations, and box plots to identify outliers.
5. **Feature Engineering:**
   - Create new features or transform existing ones to improve model performance.
   - **Example:** Create features like average purchase value, customer tenure, and frequency of site visits.
6. **Model Building:**
   - Select appropriate machine learning algorithms.
   - Train models using training data and evaluate them on validation data.
   - **Example:** Use logistic regression, decision trees, or random forests to predict customer churn.
7. **Model Evaluation:**
   - Assess model performance using metrics like accuracy, precision, recall, and F1-score.
   - Compare different models and select the best one.
   - **Example:** Evaluate models using a confusion matrix and ROC curve, and choose the model with the highest F1-score.
8. **Deployment:**

- Implement the chosen model in a production environment.
- Ensure the model can make predictions in real-time or batch mode.
- **Example:** Deploy the churn prediction model as a REST API or integrate it into the company's CRM system.

9. **Monitoring and Maintenance:**
   - Continuously monitor model performance and update it as needed.
   - Handle data drift and retrain the model with new data.
   - **Example:** Set up alerts for model performance degradation and periodically retrain the model with recent data.

Example: Customer Churn Prediction for an E-Commerce Company:

1. **Objective:** Reduce customer churn by identifying at-risk customers.
2. **Data Collection:** Gather data on customer purchases, website visits, and interactions with customer service.
3. **Data Cleaning:** Handle missing values, remove duplicates, and correct data formats.
4. **EDA:** Analyze purchase patterns, website activity, and customer service interactions to identify trends.
5. **Feature Engineering:** Create features like average purchase value, customer tenure, and frequency of visits.
6. **Model Building:** Train machine learning models (e.g., logistic regression, decision trees) to predict churn.
7. **Model Evaluation:** Use precision, recall, and F1-score to evaluate models and select the best one.
8. **Deployment:** Deploy the model as an API integrated with the CRM system.
9. **Monitoring:** Continuously monitor model performance.

Q.23) Your client is using Microsoft Power BI as their visualization tool, there is a requirement to create custom visualizations with the help of programming. How to use python in Power bi for generating custom visuals?

To use Python in Power BI for custom visuals:

1. Install Python and required libraries.
2. Enable Python scripting in Power BI settings.
3. Import your data into Power BI.
4. Drag fields into the Python visual and write your script to generate custom visuals.
5. Run the script to display the visual in your Power BI report.

Libraries like seaborn, matplotlib.pyplot can be used in the generation of interactive visualizations.

Q.24) How do you handle datetime data in Pandas?

- Datetime data in Pandas can be handled using the to_datetime() function to convert strings or integers to datetime objects, and the dt accessor can be used to extract specific components like year, month, day, etc.

Q.25) How do you calculate descriptive statistics for a DataFrame in Pandas?

The describe() method in Pandas provides a summary of descriptive statistics for each numerical column in a DataFrame. Here's how it works:

1. **Count:** It counts the number of non-null values in each column.
2. **Mean:** It calculates the arithmetic mean (average) of each column.
3. **Standard Deviation:** It measures the dispersion of values around the mean. A higher standard deviation indicates more spread out values.
4. **Minimum:** It shows the minimum value in each column.
5. **25th Percentile (Q1):** It represents the value below which 25% of the data falls.
6. **Median (50th Percentile or Q2):** It is the middle value of the dataset when arranged in ascending order. Also known as the 50th percentile or the second quartile.
7. **75th Percentile (Q3):** It represents the value below which 75% of the data falls.
8. **Maximum:** It shows the maximum value in each column

Q.26) What is the purpose of data filtering in data analysis?

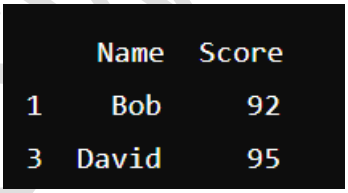
- Purpose of filtering the data
    - Data filtering is essential in data analysis to focus on specific subsets of data that meet predefined criteria or conditions. It helps in isolating relevant information, reducing noise, and facilitating targeted analysis or visualization. By filtering data, analysts can extract insights more efficiently and make informed decisions based on the selected subset of data.
- **How to Filter Data in a DataFrame using Pandas:**
    - In Pandas, data filtering is commonly performed using boolean indexing. Boolean indexing involves creating a boolean mask based on specified conditions and using it to filter rows or columns in the DataFrame.
- Example:

```
import pandas as pd

# Example DataFrame
df = pd.DataFrame({'Name': ['Alice', 'Bob', 'Charlie', 'David'], 'Score': [85, 92, 88, 95]})

# Filter rows where 'Score' is greater than 90
filtered_df = df[df['Score'] > 90]
print(filtered_df)
```

- Output



```
    Name  Score
1    Bob     92
3  David     95
```

Q.27) What is the difference between a dataframe and a series in pandas?

Feature	DataFrame	Series
Definition	Two-dimensional labeled data structure	One-dimensional labeled array
Dimensions	Rows and columns	Single dimension
Creation	Created by passing a dictionary of arrays/lists	Created from a single array/list or dictionary
Accessing Data	Access data using column and row labels or indices	Access data using index labels or integer indices
Examples	Represented as tables in a database	Represented as arrays or vectors

Q.28) How do you customize the appearance of a plot in Matplotlib?

You can customize the appearance of a plot in Matplotlib by setting various attributes such as title, labels, colors, line styles, markers, and axis limits using corresponding functions like plt.title(), plt.xlabel(), plt.ylabel(), plt.color(), plt.linestyle(), plt.marker(), plt.xlim(), and plt.ylim().

Q.29) How do you handle categorical data in Pandas?
Categorical data in Pandas can be handled using the astype('category') method to convert columns to categorical data type or by using the Categorical() constructor. It helps in efficient memory usage and enables faster operations.

Q.30) How do you handle imbalanced datasets in Pandas?

Imbalanced datasets in Pandas can be handled using techniques like resampling (oversampling minority class or undersampling majority class), using class weights in machine learning models, or using algorithms specifically designed for imbalanced datasets.

Q.31) If there is a Python list given and you are assigned to find the first duplicate number in the list, how will you do it with/without using predefined functions?

Logic:

- **With predefined functions:** Use a set to track seen numbers. Iterate through the list, and for each number, check if it is in the set. If it is, return it as the first duplicate. If not, add it to the set.

```python
def find_first_duplicate_with_set(nums):
    seen = set()
    for num in nums:
        if num in seen:
            return num
        seen.add(num)
    return None

# Example usage
nums = [1, 2, 3, 4, 2, 5]
print(find_first_duplicate_with_set(nums))  # Output: 2
```

- **Without predefined functions:** Use a nested loop where the outer loop picks elements one by one, and the inner loop checks if the picked element appears later in the list.

```python
def find_first_duplicate_without_set(nums):
    for i in range(len(nums)):
        for j in range(i + 1, len(nums)):
            if nums[i] == nums[j]:
                return nums[i]
    return None

# Example usage
nums = [1, 2, 3, 4, 2, 5]
print(find_first_duplicate_without_set(nums))  # Output: 2
```

Q.32) **How would you reverse a string in Python with/without using predefined functions?**

Logic:

- **With predefined functions:** Use Python's slicing feature to reverse the string (string[::-1]) or use the reversed() function and join the result.

```python
def reverse_string_with_slicing(s):
    return s[::-1]

# Example usage
s = "hello"
print(reverse_string_with_slicing(s))  # Output: "olleh"
```

- **Without predefined functions:** Iterate through the string from the end to the beginning and construct a new string character by character.

```python
def reverse_string_without_slicing(s):
    reversed_s = ""
    for char in s:
        reversed_s = char + reversed_s
    return reversed_s

# Example usage
s = "hello"
print(reverse_string_without_slicing(s))  # Output: "olleh"
```

Q.33) **Given a list of integers, how would you find the largest and smallest numbers in the list with/without using predefined functions?**

Logic:

- **With predefined functions:** Use the max() and min() functions to find the largest and smallest numbers.

```python
def find_largest_and_smallest_with_functions(nums):
    return max(nums), min(nums)

# Example usage
nums = [1, 2, 3, 4, 5]
print(find_largest_and_smallest_with_functions(nums))  # Output: (5, 1)
```

- **Without predefined functions:** Initialize two variables to store the largest and smallest numbers. Iterate through the list and update these variables accordingly.

```python
def find_largest_and_smallest_without_functions(nums):
    largest = nums[0]
    smallest = nums[0]
    for num in nums:
        if num > largest:
            largest = num
        if num < smallest:
            smallest = num
    return largest, smallest

# Example usage
nums = [1, 2, 3, 4, 5]
print(find_largest_and_smallest_without_functions(nums))  # Output: (5, 1)
```

Q.34) **How would you check if a string is a palindrome with/without using predefined functions?**

Logic:

- **With predefined functions:** Compare the string with its reversed version (string == string[::-1]) to check if it is a palindrome.

```python
def is_palindrome_with_slicing(s):
    return s == s[::-1]

# Example usage
s = "racecar"
print(is_palindrome_with_slicing(s))  # Output: True
```

- **Without predefined functions:** Iterate through the string from both ends towards the center, comparing corresponding characters. If all corresponding characters match, the string is a palindrome.

```python
def is_palindrome_without_slicing(s):
    length = len(s)
    for i in range(length // 2):
        if s[i] != s[length - 1 - i]:
            return False
    return True

# Example usage
s = "racecar"
print(is_palindrome_without_slicing(s))  # Output: True
```

Q.35) How would you merge two sorted lists into one sorted list with/without using predefined functions?

Logic:

- **With predefined functions:** Use the sorted() function to merge and sort the lists or use heapq.merge() for an efficient merge of sorted lists.

```python
def merge_sorted_lists_with_sorted(list1, list2):
    return sorted(list1 + list2)

# Example usage
list1 = [1, 3, 5]
list2 = [2, 4, 6]
print(merge_sorted_lists_with_sorted(list1, list2))  # Output: [1, 2, 3, 4, 5, 6]
```

- **Without predefined functions:** Use two pointers, one for each list. Compare the elements pointed to by the pointers, append the smaller element to the result list, and move the pointer for that list. Continue until all elements are merged.

```python
def merge_sorted_lists_without_sorted(list1, list2):
    merged_list = []
    i, j = 0, 0
    while i < len(list1) and j < len(list2):
        if list1[i] < list2[j]:
            merged_list.append(list1[i])
            i += 1
        else:
            merged_list.append(list2[j])
            j += 1
    while i < len(list1):
        merged_list.append(list1[i])
        i += 1
    while j < len(list2):
        merged_list.append(list2[j])
        j += 1
    return merged_list

# Example usage
list1 = [1, 3, 5]
list2 = [2, 4, 6]
print(merge_sorted_lists_without_sorted(list1, list2))  # Output: [1, 2, 3, 4, 5, 6]
```

Q.36) You are given two lists of integers, list1 and list2. Write a function in Python that performs the following operations:

1. Append all unique elements from list2 that are not already in list1.
2. Sort the resulting list in ascending order.
3. If the resulting list's length is odd, append the sum of the largest and smallest elements to the list.
4. If the resulting list's length is even, append the average of the first and last elements to the list.

```
# Given lists
list1 = [1, 3, 5, 7]
list2 = [2, 3, 6, 8, 7]

# Expected result after the operations:
# Step 1: [1, 3, 5, 7, 2, 6, 8]
# Step 2: [1, 2, 3, 5, 6, 7, 8]
# Step 3: Length is odd (7 elements), append 1 + 8 = 9
# Final result: [1, 2, 3, 5, 6, 7, 8, 9]
```

```
def complex_list_operation(list1, list2):
    # Step 1: Append unique elements from list2 to list1
    for num in list2:
        if num not in list1:
            list1.append(num)

    # Step 2: Sort the list
    list1.sort()

    # Step 3: Check the length of the list and append accordingly
    if len(list1) % 2 == 1:  # If length is odd
        largest = list1[-1]
        smallest = list1[0]
        list1.append(largest + smallest)
    else:  # If length is even
        average = (list1[0] + list1[-1]) / 2
        list1.append(average)

    return list1

# Example usage
list1 = [1, 3, 5, 7]
list2 = [2, 3, 6, 8, 7]
result = complex_list_operation(list1, list2)
print(result)  # Output: [1, 2, 3, 5, 6, 7, 8, 9]
```

Q.37) Write a Python function to find all Armstrong numbers within a given range [start, end]. An Armstrong number (also known as a narcissistic number) is a number that is equal to the sum of its own digits each raised to the power of the number of digits. Additionally, return the count of such Armstrong numbers found within the range.

Example:

- For the range [100, 500], the Armstrong numbers are [153, 370, 371, 407].

- The function should return both the list of Armstrong numbers and the count.

Solution:

```python
def is_armstrong(number):
    # Convert the number to string to easily iterate over digits
    num_str = str(number)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    return number == sum_of_powers

def find_armstrong_numbers(start, end):
    armstrong_numbers = []

    for num in range(start, end + 1):
        if is_armstrong(num):
            armstrong_numbers.append(num)

    count = len(armstrong_numbers)
    return armstrong_numbers, count

# Example usage
start = 100
end = 500
armstrong_numbers, count = find_armstrong_numbers(start, end)
print(f"Armstrong numbers between {start} and {end}: {armstrong_numbers}")
print(f"Count of Armstrong numbers: {count}")

# Output:
# Armstrong numbers between 100 and 500: [153, 370, 371, 407]
# Count of Armstrong numbers: 4
```

Explanation:

1. **is_armstrong(number):**
    - This helper function checks if a given number is an Armstrong number.
    - It converts the number to a string to iterate over its digits.
    - It calculates the sum of each digit raised to the power of the number of digits.
    - It returns True if the number equals this sum, otherwise False.
2. **find_armstrong_numbers(start, end):**
    - This function iterates over the range from start to end (inclusive).
    - For each number in the range, it checks if it is an Armstrong number using the is_armstrong function.
    - If it is, the number is added to the armstrong_numbers list.
    - The function returns the list of Armstrong numbers and their count.

Q.38) **Write a Python function to check whether a given list of integers represents a Fibonacci series. A Fibonacci series is a sequence of numbers where each number is the sum of the two preceding ones, usually starting with 0 and 1.**

Example:

- For the list [0, 1, 1, 2, 3, 5, 8], the function should return True.
- For the list [0, 1, 1, 2, 4, 6, 10], the function should return False.

```python
def is_fibonacci_series(lst):
    if len(lst) < 2:
        return False

    if lst[0] != 0 or lst[1] != 1:
        return False

    for i in range(2, len(lst)):
        if lst[i] != lst[i-1] + lst[i-2]:
            return False

    return True

# Example usage
list1 = [0, 1, 1, 2, 3, 5, 8]
list2 = [0, 1, 1, 2, 4, 6, 10]

print(is_fibonacci_series(list1))  # Output: True
print(is_fibonacci_series(list2))  # Output: False
```

Explanation:

1. **Initial Checks:**
    - The function starts by checking if the length of the list is less than 2. If it is, the function returns False because a Fibonacci series must have at least two numbers.
    - It then checks if the first two numbers in the list are 0 and 1, respectively. If not, it returns False.
2. **Iterative Check:**
    - The function iterates through the list starting from the third element.
    - For each element, it checks if the current element is equal to the sum of the two preceding elements. If any element does not satisfy this condition, the function returns False.
3. **Return True:**
    - If all elements pass the checks, the function returns True, indicating that the list represents a Fibonacci series.

Q.39) With the rise of Artificial Intelligence tools such as ChatGPT, Google Gemini, etc., which have the capability to write and generate code based on the provided requirements, what do you think would be its impact on your job? How will you use it in your work?

AI tools like ChatGPT and Google Gemini will increase efficiency by automating repetitive coding tasks, allowing developers to focus on complex problems. They enhance learning by providing examples and explanations. Collaboration improves with AI-generated documentation and clear communication of technical concepts. While AI can handle routine tasks, skilled developers remain essential for creative and critical thinking. I'll use AI to streamline coding, prototype quickly, and stay updated with new technologies.

Q.40) Describe a challenging data science problem you have encountered and how you approached solving it. What were the key steps and methodologies you used?

In one challenging data science project, I had to predict customer churn for a subscription-based service.

1. **Data Collection and Preprocessing:** I collected and cleaned data from various sources, handling missing values and outliers.
2. **Exploratory Data Analysis (EDA):** I performed EDA to understand the data distribution, identify patterns, and select relevant features.
3. **Feature Engineering:** I created new features based on domain knowledge and interaction terms that might affect churn.
4. **Model Selection and Training:** I experimented with various models (logistic regression, random forests, and XGBoost), tuning hyperparameters using cross-validation.
5. **Evaluation and Deployment:** I evaluated model performance using metrics like accuracy, precision, recall, and AUC-ROC. The final model was deployed to predict churn and inform retention strategies.