**Python**

**Q1. Explain the difference between a generator and an iterator?**

* **Iterator**: An iterator is an object that implements the iterator protocol, which consists of the methods \_\_iter\_\_() and \_\_next\_\_(). Iterators can traverse through all the elements of a collection (like a list or a tuple) one at a time. Once all elements are accessed, an iterator raises a StopIteration exception.
* **Generator**: A generator is a special type of iterator defined using a function. Instead of returning a single value and terminating, a generator can yield multiple values, pausing its state between each yield and resuming from where it left off. This makes generators more memory-efficient, as they do not store their entire output in memory at once.

**Q2. Difference between list and tuple?**

* **List**:
  + Mutable (can be changed after creation).
  + Defined with square brackets: [].
  + Supports methods like .append(), .remove(), .extend(), etc.
* **Tuple**:
  + Immutable (cannot be changed after creation).
  + Defined with parentheses: ().
  + Does not support methods that modify the content, such as .append() or .remove().

**Q3. Can you create a custom class that behaves like a built-in Python type (e.g., list or dictionary)?**

Yes, you can create a custom class that behaves like a built-in Python type by implementing special methods. For example, here’s a simple custom list class:

class CustomList:

def \_\_init\_\_(self, items=None):

self.items = items if items is not None else []

def \_\_getitem\_\_(self, index):

return self.items[index]

def \_\_setitem\_\_(self, index, value):

self.items[index] = value

def \_\_len\_\_(self):

return len(self.items)

def append(self, value):

self.items.append(value)

**Q4. Explain the difference between a dictionary and a set in Python. Provide an example of each.**

* **Dictionary**: A dictionary is a collection of key-value pairs. Each key is unique and maps to a specific value. Dictionaries are defined using curly braces {}.

Example:

my\_dict = {'apple': 1, 'banana': 2, 'cherry': 3}

* **Set**: A set is an unordered collection of unique items. It does not allow duplicates and is also defined with curly braces.

Example:

my\_set = {1, 2, 3, 4, 5}

**Q5. Write a program to check whether the given number is even or not?**

def is\_even(number):

return number % 2 == 0

# Example usage

num = int(input("Enter a number: "))

if is\_even(num):

print(f"{num} is even.")

else:

print(f"{num} is odd.")

**SQL**

**Q1. Explain the difference between SQL and NoSQL databases.**

* **SQL Databases**: These are relational databases that use Structured Query Language (SQL) for defining and manipulating data. They are table-based and enforce a predefined schema, which ensures data integrity. Examples include MySQL, PostgreSQL, and Oracle.
* **NoSQL Databases**: These are non-relational databases that allow for a variety of data models (document, key-value, graph, etc.). NoSQL databases are schema-less and can handle unstructured data more flexibly. They are designed for scalability and performance with big data. Examples include MongoDB, Cassandra, and Redis.

**Q2. What are the basic SQL commands for interacting with a database?**

Basic SQL commands include:

* SELECT: Retrieve data from a database.
* INSERT: Add new records to a table.
* UPDATE: Modify existing records in a table.
* DELETE: Remove records from a table.
* CREATE TABLE: Create a new table.
* DROP TABLE: Delete a table.

**Q3. How do you retrieve all records from a table named Customers?**

SELECT \* FROM Customers;

**Q4. What are different aggregate functions in SQL?**

Common aggregate functions include:

* COUNT(): Returns the number of rows.
* SUM(): Returns the sum of a numeric column.
* AVG(): Returns the average value of a numeric column.
* MIN(): Returns the smallest value in a column.
* MAX(): Returns the largest value in a column.

**Q5. What is a foreign key in a database and how is it used?**

A foreign key is a field (or collection of fields) in one table that uniquely identifies a row of another table or the same table. It creates a relationship between the two tables. Foreign keys help ensure referential integrity within the database.

Example:

CREATE TABLE Orders (

OrderID int,

CustomerID int,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

**Data Science (DS)**

**Q1. Discuss issues like noise in text data, ambiguity, scalability, and extracting meaningful information from unstructured text.**

* **Noise in Text Data**: Noise refers to any irrelevant or extraneous information that does not contribute to the desired output. This can include typos, slang, or irrelevant context. Techniques such as preprocessing, filtering, and normalization help to reduce noise.
* **Ambiguity**: Ambiguity arises when a word or phrase can have multiple meanings, making it difficult to understand the intended context. Techniques such as context-based tokenization and disambiguation through machine learning help to resolve ambiguity.
* **Scalability**: Handling large volumes of text data can be challenging. Scalability issues can arise during processing and modeling if the algorithms and data structures used are not optimized for performance. Distributed computing and efficient data storage solutions are often employed.
* **Extracting Meaningful Information**: This involves processes like text classification, sentiment analysis, and topic modeling to derive insights from unstructured text data. Techniques like Natural Language Processing (NLP) are widely used for this purpose.

**Q2. Discuss techniques like Latent Dirichlet Allocation (LDA) and how they uncover latent topics within a corpus.**

Latent Dirichlet Allocation (LDA) is a generative probabilistic model used to identify topics present in a collection of documents. It assumes that documents are mixtures of topics, where a topic is defined as a distribution over words. LDA can reveal hidden thematic structures in large text corpora by inferring the distribution of topics in documents and the distribution of words in topics.

Steps in LDA include:

1. Preprocessing the text (tokenization, removal of stopwords).
2. Initializing random topic assignments for words in documents.
3. Iteratively refining topic assignments based on word co-occurrences.

**Q3. Discuss steps like lowercasing, removing stopwords, handling punctuation, and dealing with special characters.**

* **Lowercasing**: Converts all characters in the text to lowercase, which helps in standardizing the data and reducing redundancy (e.g., treating "Apple" and "apple" as the same).
* **Removing Stopwords**: Stopwords (common words like "the", "is", "in", etc.) are often removed to reduce noise, as they carry little meaning in text analysis.
* **Handling Punctuation**: Punctuation can interfere with textual analysis and should either be removed or handled appropriately, often by stripping it from tokens or replacing it with spaces.
* **Dealing with Special Characters**: Special characters (e.g., emojis, symbols) may need to be removed or translated into canonical forms to standardize the text for analysis.

**Q4. Discuss steps like lowercasing, removing stopwords, handling punctuation, and dealing with special characters**

**Ans:-**

* **Lowercasing**: Converts all characters in the text to lowercase, which helps in standardizing the data and reducing redundancy (e.g., treating "Apple" and "apple" as the same).
* **Removing Stopwords**: Stopwords (common words like "the", "is", "in", etc.) are often removed to reduce noise, as they carry little meaning in text analysis.
* **Handling Punctuation**: Punctuation can interfere with textual analysis and should either be removed or handled appropriately, often by stripping it from tokens or replacing it with spaces.
* **Dealing with Special Characters**: Special characters (e.g., emojis, symbols) may need to be removed or translated into canonical forms to standardize the text for analysis.