**Date: 30/05/2024**

**Project-195   
Interview Questions**

**Python Questions**

1. How do you create a list in Python?
2. What is a dictionary in Python, and how do you access its values?
3. How do you write a **for** loop in Python? Provide an example.
4. What is the difference between **append()** and **extend()** methods in a Python list?
5. How do you define a function in Python?

**SQL Questions**

1. How do you select all columns from a table in SQL?
2. What is the purpose of the **WHERE** clause in SQL?
3. How do you insert a new record into a table in SQL?
4. What is a primary key in a database?
5. How do you update a record in a SQL table?

**Deep Learning Questions**

1. What is a neural network in simple terms?
2. What is the purpose of the activation function in a neural network?
3. What is the difference between supervised and unsupervised learning?
4. What is a dataset, and why is it important in machine learning?
5. What is the role of epochs in training a deep learning model?

PYTHON

1. In Python, a list is created by enclosing comma-separated elements within square brackets []. Lists can hold elements of various data types, such as integers, strings, floats, and even other lists. They support dynamic resizing and can be modified after creation. Examples include my\_list = [1, 2, 3] for a list of integers, or mixed\_list = [1, 'two', 3.0] for mixed data types. Lists can also be empty (empty\_list = []) or contain nested lists (nested\_list = [[1, 2], [3, 4]]). Lists are fundamental data structures in Python, commonly used for storing and manipulating collections of data.
2. In Python, a dictionary is a collection of key-value pairs enclosed within curly braces {}. Each key-value pair maps a unique key to its corresponding value. Dictionaries are mutable and unordered, meaning the order of elements is not preserved.

To access the values in a dictionary, you can use the keys as the index within square brackets [] or by using the get() method.

Here's an example:

my\_dict = {'name': 'John', 'age': 30, 'city': 'New York'}

# Accessing values using keys

print(my\_dict['name']) # Output: John

print(my\_dict['age']) # Output: 30

# Accessing values using get() method

print(my\_dict.get('city')) # Output: New York

If the key is not present in the dictionary, using square brackets to access its value will result in a KeyError, while using the get() method will return None by default or a specified default value if provided.

3) To write a for loop in Python, use the for keyword followed by a variable to iterate over, then the in keyword followed by the iterable object. Indent the code block to be executed within the loop.

for num in range(1, 5):

print(num)

4)

The append() method in Python adds a single element to the end of a list, modifying the original list. It takes a single argument, the element to be added.

The extend() method, on the other hand, adds multiple elements from an iterable (such as another list) to the end of the list. It modifies the original list and accepts an iterable as its argument.

5) To define a function in Python, use the def keyword followed by the function name and parentheses containing optional parameters. Then, include a colon : to begin the function body, indented below. Return values can be specified using the return keyword. Example:

def my\_function(parameter1, parameter2):

# Function body

return parameter1 + parameter2

SQL

* 1. To select all columns from a table in SQL, you can use the asterisk (\*) wildcard character after the SELECT keyword. Here's an example:

SELECT \* FROM your\_table;

This query retrieves all columns from the specified table (your\_table). It's a convenient way to fetch all columns without explicitly listing them.

* 1. The WHERE clause in SQL is used to filter rows returned by a SELECT, UPDATE, DELETE, or MERGE statement based on specified conditions. It allows you to specify one or more conditions that must be met for a row to be included in the result set. This helps narrow down the search and retrieve only the rows that meet the specified criteria, improving query precision and efficiency. The WHERE clause is essential for extracting relevant data from large datasets and for modifying data selectively in UPDATE and DELETE statements.

3)To insert a new record into a table in SQL, you can use the INSERT INTO statement followed by the table name and a list of column names within parentheses. Then, you specify the VALUES keyword followed by a list of values in the same order as the column names.

Here's an example:

INSERT INTO your\_table (column1, column2, column3)

VALUES (value1, value2, value3);

Replace your\_table with the name of your table, column1, column2, column3 with the column names, and value1, value2, value3 with the corresponding values for the new record. If you're inserting values into all columns, you can omit the column list after INSERT INTO.

4) In a database, a primary key is a unique identifier for each record (row) in a table. It uniquely identifies each record and ensures that no two records have the same key value. Primary keys are essential for data integrity and serve as the reference point for establishing relationships between tables in a relational database. Typically, primary keys are implemented using a single column, although composite primary keys consisting of multiple columns are also possible. Primary keys enforce entity integrity and provide a fast and efficient way to retrieve specific records from a table.

5) To update a record in a SQL table, you can use the UPDATE statement followed by the table name. Then, specify the SET keyword followed by the column names you want to update along with their new values. Finally, use the WHERE clause to specify the condition that identifies the record(s) to be updated.

UPDATE your\_table

SET column1 = new\_value1, column2 = new\_value2, ...

WHERE condition;

Replace your\_table with the name of your table, column1, column2, etc., with the column names you want to update, new\_value1, new\_value2, etc., with their new values, and condition with the criteria to identify the record(s) to be updated.

UPDATE employees

SET salary = 50000

WHERE employee\_id = 123;

This query updates the salary column for the employee with employee\_id equal to 123 to 50000. Adjust the table name, column names, values, and condition as needed.

DEEP LEARNING

* 1. In simple terms, a neural network is a computational model inspired by the structure and function of the human brain. It consists of interconnected nodes, called neurons, organized into layers. Each neuron receives input signals, processes them, and produces an output signal. By adjusting the strengths of connections between neurons, known as weights, and applying activation functions, neural networks can learn to approximate complex relationships between input and output data. They are used for tasks such as pattern recognition, classification, regression, and more, making them a powerful tool in machine learning and artificial intelligence.
  2. The purpose of the activation function in a neural network is to introduce non-linearity into the output of each neuron. This non-linearity allows neural networks to model complex relationships in data, enabling them to approximate arbitrary functions. Without activation functions, the network would only be capable of learning linear transformations of the input data, limiting its ability to capture and represent more intricate patterns and relationships. Activation functions enable neural networks to learn and represent highly nonlinear and complex mappings, making them essential components of deep learning models.

3) Supervised and unsupervised learning are two main types of machine learning paradigms with distinct characteristics:

Supervised Learning:

In supervised learning, the algorithm learns from labeled data, where each training example consists of input features and their corresponding labels or target values.

The goal is to learn a mapping from inputs to outputs based on the labeled examples, enabling the algorithm to make predictions or decisions on new, unseen data.

Common tasks include classification (assigning inputs to discrete categories) and regression (predicting continuous values).

Examples: Email spam detection, image classification, predicting house prices.

Unsupervised Learning:

In unsupervised learning, the algorithm learns from unlabeled data, where training examples consist only of input features without any corresponding labels.

The goal is to discover hidden patterns, structures, or relationships within the data without explicit guidance from labeled examples.

Common tasks include clustering (grouping similar data points together) and dimensionality reduction (reducing the number of features while preserving important information).

Examples: Customer segmentation, anomaly detection, topic modeling.

In summary, supervised learning relies on labeled data to learn the relationship between inputs and outputs, while unsupervised learning discovers patterns and structures in unlabeled data without explicit guidance.

4) A dataset is a structured collection of data, typically organized in tabular format, containing examples used to train, validate, or test machine learning models. It consists of input features (independent variables) and corresponding labels or target values (dependent variables) for supervised learning tasks. In unsupervised learning, datasets may contain only input features without labels.

Datasets are crucial in machine learning for several reasons:

Training Models: Datasets serve as the foundation for training machine learning models by providing examples from which the model can learn patterns and relationships between inputs and outputs.

Evaluation and Validation: Datasets are used to evaluate and validate the performance of trained models on unseen data, ensuring their generalization ability and reliability.

Testing and Deployment: Datasets are also used for testing and deploying machine learning models in real-world applications, assessing their performance and effectiveness in practical scenarios.

Benchmarking: Datasets allow researchers and practitioners to compare the performance of different algorithms and techniques on standardized tasks, facilitating progress and advancements in the field of machine learning.

Overall, datasets play a central role in machine learning, providing the raw material for training, evaluating, and deploying models, and contributing to the advancement of the field through empirical experimentation and benchmarking.

5) In deep learning, an epoch refers to one complete pass through the entire training dataset during the training process. The training process involves iteratively updating the model's parameters (weights and biases) based on the training data to minimize the loss function and improve performance.

The role of epochs in training a deep learning model is as follows:

Iterative Optimization: Training a deep learning model typically requires multiple iterations or epochs to gradually improve its performance. Each epoch allows the model to adjust its parameters based on the training data, moving closer to the optimal solution.

Convergence: Deep learning models may not converge to their optimal solution after a single epoch. Multiple epochs allow the model to refine its parameters further, potentially improving performance and convergence.

Generalization: Training a model for too few epochs may result in underfitting, where the model fails to capture the underlying patterns in the data. Training for too many epochs, however, may lead to overfitting, where the model memorizes the training data without generalizing well to unseen data. The number of epochs is a hyperparameter that needs to be tuned to achieve the right balance between underfitting and overfitting.

Monitoring Progress: Monitoring the model's performance on a validation dataset after each epoch allows practitioners to track progress, detect overfitting, and make informed decisions about when to stop training.

In summary, epochs play a crucial role in training deep learning models by enabling iterative optimization, promoting convergence, and facilitating the balance between model complexity and generalization performance.