**Assessment**

Name-Bharat Kumar

**Question 1. Mention some of the advantages of python?**

**Answer-**

1. **Simple to Use and Understand**
2. **Free and Open-Source**
3. **Productivity has Increased**
4. **Interpreted Language**
5. **Many in-built libraries**
6. **Portability**
7. **user friendly data structure**

**Question 2. What are local variables and global variables in Python?**

**Answer-**

Local Variables:

- Defined inside a function or block of code.

- Accessible only within that function or block.

- Lifespan is limited to the function or block's execution.

- Other functions or blocks cannot access them directly.

Global Variables:

- Defined at the top level of the program or declared as global within a function.

- Accessible throughout the entire program, including all functions and blocks.

- Lifespan lasts as long as the program is running.

- Functions can modify the value of a global variable using the 'global' keyword.

**Question 3. What is Lambda Functions in Python?**

**Answer-**

Lambda functions in Python are small, anonymous functions that are defined using the lambda keyword. They are also known as "inline functions" or "anonymous functions" because they don't require a formal name like regular functions defined using def. Lambda functions are typically used for simple and short operations, where a full function definition is not necessary.

**Question 4. What is a Negative Index in Python?**

**Answer -** Negative indexing in Python allows accessing elements in a sequence (like strings, lists, or tuples) from the end of the sequence. The negative index starts from -1 for the last element, -2 for the second-last element, and so on, moving backwards through the sequence.

For example, in a list [10, 20, 30, 40, 50], using positive indexing, we access elements like this: `list[0]` gives `10`, list[1] gives `20`, and so on.

With negative indexing, we can access the same elements from the end of the list: list[-1] gives `50` (last element), list[-2] gives `40` (second-last element), and so on.

**Question5. What is the difference between tuples and lists?**

**Answer-**  
Tuples and lists are both collection data types in Python. Tuples are created using parentheses ( ) while lists are created using square brackets [ ]. The main difference between them lies in their mutability. Tuples are immutable, meaning their elements cannot be changed after creation, whereas lists are mutable, allowing you to modify, add, or remove elements. Tuples are often used for fixed collections of data, such as coordinates or database records, where the data should remain constant. Lists, on the other hand, are preferred for dynamic collections of data that may change during program execution, like managing user inputs or maintaining a dynamic list of items. Understanding these differences helps in choosing the appropriate data structure based on the requirements of a Python program.

**Question6. What is a dynamically typed language?**

**Answer-**

A dynamically typed language is a programming language in which the data types of variables are determined at runtime. In dynamically typed languages like Python, you don't need to explicitly declare the data type of a variable; it is automatically assigned based on the value it holds.

**Question7. What are the data types available in python?Brief about it?**

**Answer-**

In Python, there are several built-in data types available to store and manipulate different kinds of data. Here's a brief overview of some common data types:

1. Numeric Types:

- int: Represents integer values like 1, 100, or -42.

- float: Represents floating-point numbers with decimal places, e.g., 3.14, 2.718.

2. String:

- str: Represents a sequence of characters, such as "Hello, World!" or 'Python'.

3. Boolean:

- bool: Represents Boolean values, which can be either True or False. Used for logical operations and flow control.

4. Sequence Types:

- list: Represents an ordered, mutable collection of elements. Defined using square brackets, e.g., [1, 2, 3].

- tuple: Similar to a list, but immutable. Defined using parentheses, e.g., (1, 2, 3).

5. Set Types:

- set: Represents an unordered, mutable collection of unique elements. Defined using curly braces, e.g., {1, 2, 3}.

6. Mapping Type:

- dict: Represents a collection of key-value pairs. Defined using curly braces with key-value pairs, e.g., {'name': 'John', 'age': 30}.

7. None Type:

- None: Represents a special data type that signifies the absence of a value or a null value.

.

**Question8 .Why we have to use functions in Python? Explain with an example?**

**Answer-**

Functions in Python are used to encapsulate a piece of code with specific functionality, making it easier to manage and reuse. They enhance code readability and promote good programming practices. By using functions, you can divide a large program into smaller, more manageable parts, reducing complexity and improving maintainability. Additionally, functions help avoid code duplication, enabling you to modify the behavior of your program in a centralized way. Overall, functions play a crucial role in structuring Python code and facilitating efficient development and maintenance processes.

#Example-

def square(num):

return num \* num

number = 5

result = square(number)

print("The square of", number, "is:", result)

Output:-The square of 5 is: 25

**Question 9. What is the difference between function and generators? Give an example?**

**Answer-**

Functions in Python are blocks of code that execute when called and return a value using the `return` statement. Generators, on the other hand, are a type of function that use the `yield` statement to produce a sequence of values lazily, one at a time, allowing efficient memory usage.

Example of a function:

def square(num):

return num \* num

result = square(5) # Function call

print(result) # Output: 25

Example of a generator:

def square\_generator(nums):

for num in nums:

yield num \* num

numbers = [1, 2, 3, 4, 5]

generator = square\_generator(numbers) # Generator creation

for result in generator:

print(result) # Output: 1, 4, 9, 16, 25 (produced one at a time)

**Question 10. Explain briefly about conditional statements with an example?**

**Answer**-

Conditional statements are also called decision-making statements. We use those statements when we want to execute a block of code when the given condition is true or false.

Type of condition statement in Python:

## If statement: It is most usually used as a conditional statement. Eg.

## a = 10

## b = 20

## if a<b:

## print("a is less than b")

## If else statement: It is a conditional statement. The statement itself says that if a given condition is true or false. True means executing the “if” statement to the output. False means executing the “else” statement to the output. Eg.

## a = 10

## b = 20

## if a==b:

## print("a and b are equal")

## else:

## print("a and b are not equal")

## Elif statement: It is a shortcut for else if condition statements. In Python, one or more conditions are used in the elif statement. Eg.

## a = 10

## b = 10

## if a < b:

## print("a is greater than b")

## elif a == b:

## print("a and b are equal")

## else:

## print("b is greater than a")

## Nested if statement: In Python using "if" statements inside other if statements it is called a nested if statement. Eg.

## a= 1001

## if a> 100:

## print("Above 100")

## if a > 1000:

## print("and also above 1000")

## Nested if else statement: In Python using one “if else” statement inside other if else statements it is called a nested if else statement. Eg. a=int(input("enter the a value"))#user give a value

## if a> 100:

## print("Above 100")

## if a > 1000:

## print("and also above 1000")

## else:

## print("and also below 1000")

## else:

## print("below 100")

**Question 11. why we have to use exceptional handling in python?What are the keyword used to handle the error , give one example?**

**Answer-**

Exception handling in Python is used to handle errors and exceptions that may occur during the execution of a program. When an error occurs, it can cause the program to terminate abruptly. Exception handling allows us to gracefully deal with such errors, preventing the program from crashing and providing a way to handle unexpected situations.

The keywords used for exception handling in Python are `try`, `except`, `else`, and `finally`.

#Example-

try:

num1 = int(input("Enter a number: "))

num2 = int(input("Enter another number: "))

result = num1 / num2

print("Result:", result)

except ValueError:

print("Invalid input. Please enter valid numbers.")

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

else:

print("Division successful.")

finally:

print("This will always be executed, regardless of exceptions.")

Output-

Enter a number: 3

Enter another number: 2

Result: 1.5

Division successful.

This will always be executed, regardless of exceptions.

**Question12. Define class and objects with an example?**

**Answer**-

In object-oriented programming, a class is a blueprint or template that defines the structure and behavior of objects. It is a user-defined data type that encapsulates data and functions (also known as methods) that operate on that data. An object, on the other hand, is an instance of a class, representing a specific entity in the program's domain.

#Example:

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def greet(self):

return f"Hello, my name is {self.name} and I am {self.age} years old."

# Create an object of the class 'Person'

person1 = Person("John", 30)

print(person1.name)

print(person1.greet())

Output-

John Hello, my name is John and I am 30 years old.

**Question13. write a program to swap the two numbers in python?**

**Answer**-

def swap\_numbers(a, b):

    temp = a

    a = b

    b = temp

    return a, b

num1 = float(input("Enter the first number: "))

num2 = float(input("Enter the second number: "))

num1, num2 = swap\_numbers(num1, num2)

print("After swapping:")

print("First number:", num1)

print("Second number:", num2)

**Question14. Write a palindrome program in python**

**Answer-**

def is\_palindrome(word):

    cleaned\_word = word.replace(" ", "").lower()

    return cleaned\_word == cleaned\_word[::-1]

word = input("Enter a word: ")

if is\_palindrome(word):

    print("Yes, it's a palindrome!")

else:

    print("No, it's not a palindrome.")

**Question15. What is the use of Numpy array over than list?**

**Answer**-

NumPy arrays offer better performance, memory efficiency, and support for multidimensional data compared to Python lists. They enable efficient computation, broadcasting, and element-wise operations, making them essential for numerical and scientific computing tasks.

**Question16. Explain brief about reshape in numpy with an example?**

**Answer**-

In NumPy, the `reshape` function is used to change the shape or dimensions of an array without modifying its data. It returns a new array with the specified shape, and the total number of elements in the reshaped array remains the same as the original array.

#Example-

import numpy as np

arr = np.arange(1, 13)

reshaped\_arr = arr.reshape(3, 4)

print("Original array:")

print(arr)

print("Reshaped array:")

print(reshaped\_arr)

**Question17. what is the use of pandas explain in brief?**

**Answer-**

Pandas is a Python library used for data manipulation, analysis, and preparation. It provides data structures like DataFrame and Series for handling structured and labeled data effectively. Key uses of Pandas include data cleaning, preparation, analysis, visualization, merging, and time series data handling. It simplifies data manipulation tasks and is an essential tool for data scientists and analysts to work with structured data efficiently in Python.

**Question18. what do you mean by supervised machine learning and unsupervised machine learning ? Explain in brief ?**

**Answer**-

Supervised Machine Learning:

Supervised machine learning uses labeled data to train the algorithm and make predictions on new data. The algorithm learns a mapping function from input to output based on provided examples. It is used for tasks like classification and regression.

**Example:** Predicting whether an email is spam or not based on labeled emails (spam or not spam).

Unsupervised Machine Learning:

Unsupervised machine learning deals with unlabeled data and aims to find patterns or structures within the data. It clusters data points based on similarities or identifies hidden relationships in the data.

**Example:** Clustering customers into different groups based on their buying behavior without prior knowledge of customer categories.

In summary, supervised learning uses labeled data to make predictions, while unsupervised learning explores patterns in unlabeled data to discover hidden structures and relationships.

**Question19. Defferenciate between Decision tree and Randam forest?**

**Answer-**

Decision Tree:

* Single tree structure.
* Prone to overfitting, especially on complex datasets.
* Uses the entire training data to build the tree.
* Selects the best split at each node based on feature importance measures (e.g., Gini impurity).

Random Forest:

* Ensemble of multiple decision trees.
* Reduces overfitting by combining predictions from multiple trees.
* Uses random subsets of data (bagging) to build each tree.
* Ranks feature importance by averaging over all trees in the forest.

**Question20. what do you mean by forward propagation and backward propagation in ANN?**

**Answer-**

Forward Propagation: In Artificial Neural Networks (ANN), forward propagation refers to the process of passing input data through the network's layers to compute the predicted output. It involves multiplying the input by weights, applying activation functions, and passing the result to the next layer.

Backward Propagation: Backward propagation is the process of updating the network's weights during training to minimize the prediction error. It calculates the gradient of the loss function with respect to the weights and adjusts the weights in the opposite direction of the gradient using optimization algorithms like gradient descent. This helps the network learn and improve its predictions over time.

**Question21. Explain the working flow of ANN and CNN?**

**Answer-**

Artificial Neural Network (ANN) Working Flow:

1. Input data is passed through the input layer.

2. Data is multiplied by weights and passed through activation functions in hidden layers.

3. Output layer generates predictions.

4. Predictions are compared to the actual target values using a loss function.

5. Backward propagation updates weights to minimize the loss.

6. Repeat steps 1-5 iteratively to improve predictions through training.

Convolutional Neural Network (CNN) Working Flow:

1. Input data (usually images) is passed through the input layer.

2. Convolutional layers apply filters to extract features from the input data.

3. Activation functions introduce non-linearity in the network.

4. Pooling layers downsample the data to reduce computation and extract dominant features.

5. Fully connected layers combine features and generate predictions.

6. Predictions are compared to the actual target values using a loss function.

7. Backward propagation updates weights to minimize the loss.

8. Repeat steps 1-7 iteratively to improve predictions through training.

**Question22. what is the use of activation function and optimizers?**

**Answer-**

Activation Function:

Activation functions introduce non-linearity to the neural network, enabling it to learn complex patterns and make more accurate predictions. They decide whether a neuron should be activated or not based on the input it receives. Common activation functions include ReLU, Sigmoid, and Tanh.

Optimizers:

Optimizers are algorithms used to update the neural network's weights during training. They minimize the prediction error by adjusting weights in the right direction. Popular optimizers include Gradient Descent, Adam, and RMSprop. They help the network converge faster and improve its performance.

**Question23. What is the difference between R square and adjusted Rsquare?**

**Answer-**

R-Squared (R²): R-squared is a statistical metric that represents the proportion of the variance in the dependent variable (target) that is explained by the independent variables (features) in a regression model. It ranges from 0 to 1, with 1 indicating a perfect fit of the model to the data.

Adjusted R-Squared: Adjusted R-squared is a modification of R-squared that takes into account the number of independent variables in the model and adjusts R-squared accordingly. It penalizes the inclusion of irrelevant variables and rewards the inclusion of relevant ones, making it a more reliable metric for model evaluation, especially in multiple regression scenarios. It also ranges from 0 to 1, with 1 indicating a better fit of the model to the data.

**Question24. What do you mean by LSTM in RNN?**

**Answer-**

LSTM stands for Long Short-Term Memory, which is a type of recurrent neural network (RNN) architecture. LSTM is designed to address the vanishing and exploding gradient problems in traditional RNNs, making it more effective in handling long sequences and retaining important information for long periods. It is widely used in natural language processing, speech recognition, and other sequential data tasks.

**Question25. What is the use of image processing?**

**Answer-**

Image processing plays a crucial role in various fields due to its versatility. It is extensively used in medical imaging to aid in diagnoses, monitoring, and treatment planning. In computer vision, image processing enables machines to interpret and understand the visual world, leading to advancements in autonomous vehicles, facial recognition, and surveillance systems. Additionally, image processing techniques are essential in object detection, enabling applications like augmented reality, robotics, and quality control in manufacturing. Image compression, another significant application, allows efficient storage and transmission of images in digital media, improving bandwidth utilization and user experience. Overall, image processing's wide-ranging applications make it an indispensable tool in modern technology and research.

**Question26. What are the steps involved in NLP?**

**Answer-**

1. Text Preprocessing: The initial step involves cleaning and formatting the raw text data to remove irrelevant information and ensure consistency in the data.

2. Stopword Removal: Common words like "the," "is," and "a" (stopwords) are eliminated from the text since they do not contribute much to the overall meaning.

3. Stemming/Lemmatization: Words are reduced to their base or root form (stemming) or transformed to their dictionary form (lemmatization) to treat variations of words as a single entity.

4. Feature Extraction: Text data is transformed into numerical features using techniques like Bag-of-Words or TF-IDF, enabling machine learning algorithms to process and analyze the data.

5. Model Building: Machine learning or deep learning algorithms are applied to build models for specific NLP tasks, such as sentiment analysis, text classification, or named entity recognition.

6. Model Evaluation: The performance of the NLP model is evaluated using various metrics to determine its effectiveness and accuracy.

7. Model Deployment: The trained NLP model is integrated into applications or systems to process new text data and provide meaningful insights.

8. Continuous Improvement: NLP models may require fine-tuning and updates as they encounter new data or real-world scenarios, necessitating continuous improvement to maintain optimal performance.

**Question27. What do you mean by bias and variance?**

**Answer**-

1.Bias:

   - Bias refers to the error introduced by approximating a real-world problem with a simplified model. It represents the model's tendency to consistently underfit or overfit the training data.

   - A high bias means the model is too simple and fails to capture the underlying patterns in the data, resulting in poor performance on both the training and test datasets (low accuracy).

   - Underfitting occurs when the model is too biased and cannot learn from the data, leading to low training and test accuracy.

2.Variance:

   - Variance refers to the sensitivity of the model to fluctuations in the training data. It measures how much the model's predictions vary when trained on different subsets of the data.

   - A high variance means the model is too complex and overfits the training data, performing well on the training dataset but poorly on new, unseen data (high training accuracy but low test accuracy).

   - Overfitting occurs when the model is too sensitive to the training data and captures noise and random fluctuations, resulting in poor generalization to new data.

**Question28.What are the steps involved in machine learning to create model?**

**Answer-**

1.Data Collection and Preparation:

   - The first step is to gather and collect the relevant data for the machine learning task. This data might come from various sources like databases, APIs, or files. The data is then preprocessed and cleaned to handle missing values, remove noise, and perform feature engineering.

2.Data Splitting:

   - The data is split into two or more sets: a training set, a validation set, and a test set. The training set is used to train the model, the validation set is used to tune hyperparameters and avoid overfitting, and the test set is used to evaluate the final model's performance.

3.Feature Selection and Engineering:

   - This step involves selecting the most relevant features from the dataset and creating new features that might improve the model's performance. Feature engineering can include scaling, normalization, one-hot encoding, and other transformations.

4.Model Selection:

   - Choose the appropriate machine learning algorithm or model for the task at hand. The selection depends on the problem type (classification, regression, clustering, etc.), data size, interpretability, and other factors.

5.Model Training:

   - Train the selected model using the training dataset. The model learns the relationships between the input features and the target output during this stage.

6.Hyperparameter Tuning:

   - Fine-tune the hyperparameters of the model to optimize its performance. Techniques like cross-validation and grid search help identify the best combination of hyperparameters.

7.Model Evaluation:

   - Evaluate the model's performance using the validation set. Common evaluation metrics depend on the problem type, such as accuracy, mean squared error (MSE), or F1-score.

8.Model Optimization:

   - Based on the evaluation results, further optimize the model if necessary. This may involve adjusting hyperparameters or performing additional feature engineering.

9.Model Testing:

   - Once the model is optimized, evaluate its performance on the test set to get an unbiased estimate of how well it will perform on new, unseen data.

10.Model Deployment:

   - If the model meets the desired performance, deploy it in the production environment to make predictions on new incoming data.

11.Model Monitoring and Maintenance:

   - Continuously monitor the model's performance in the real-world setting and update it as needed to maintain its accuracy and effectiveness.

------------------------------- ----------END-----------------------------------------------