1.Define Artificial Intelligence (AI) and provide examples of its applications.

AI :- Artificial Intelligence (AI) refers to the theory and development of computer systems capable of performing tasks that traditionally required human intelligence, such as reasoning, decision-making, and problem-solving.

Applications:-

• Predictive analytics and DataOps for identifying trends.

• Recommendation engines for personalized suggestions.

• Sentiment analysis to categorize customer feedback.

• Customer support through chatbots.

• Security and risk management.

• Network optimization in telecommunications.

• Marketing management.

• Supply-chain planning and management.

2. Differentiate between supervised and unsupervised learning techniques in ML.

Supervised Learning

Unsupervised Learning

Input Data

Uses Known and Labelled Data as input.

Uses Unknown Data as input.

Computational Complexity

Less Computational Complexity

More Computational Complex

Real-Time

Uses off-line analysis

Uses Real-Time Analysis of Data

Number of Classes

The number of Classes is known

The number of Classes is not known

Accuracy of Results

Accurate and Reliable Results

Moderate Accurate and Reliable Results

Output data

The desired output is given.

The desired, output is not given.

Model

In supervised learning it is not possible to learn larger and more complex models than in unsupervised learning

In unsupervised learning it is possible to learn larger and more complex models than in supervised learning

Training data

In supervised learning training data is used to infer model

In unsupervised learning training data is not used.

Another name

Supervised learning is also called classification

Unsupervised learning is also called clustering.

Test of model

We can test our model.

We can not test our model.

Example

 Optical Character Recognition

Find a face in an image

3.What is Python? Discuss its main features and advantages.

Python: Python is a high-level, interpreted, and general-purpose dynamic programming language known for its focus on code readability. It offers a range of features and advantages that make it a popular choice among developers.

Features:

1. Expressive Language: Capable of performing complex tasks with minimal code, enhancing productivity.

2. Interpreted Language: Executes code line by line, aiding in easy debugging and portability.

3. Cross-platform Language: Runs on various platforms like Windows, Linux, UNIX, and Macintosh, ensuring portability.

4. Object-Oriented Language: Supports object-oriented concepts like inheritance, polymorphism, and encapsulation.

5. Extensible: Allows integration with languages like C/C++, enhancing its functionality.

6. Large Standard Library: Provides a vast range of libraries for different fields like machine learning and web development.

7. GUI Programming Support: Enables the development of desktop and web applications with libraries like PyQT5 and Tkinter.

8. Integrated: Easily integrates with languages like C, C++, and Java, simplifying code debugging.

9. Dynamic Memory Allocation: Automatically allocates memory to variables at runtime, eliminating the need for specifying data types

10. Easy to Learn and Use: Python's syntax is simple and resembles English,

making it easy to learn, especially for beginners.

Advantages:

1. Reduced Maintenance Cost: Simplifies application maintenance, reducing associated costs.

2. Wide Applicability: Widely used by engineers, scientists, and mathematicians due to its versatility.

3. Easy Memory Management: Offers efficient memory management capabilities.

4. Large Community: Active community support for rapid assistance and problem-solving.

5. Asynchronous Coding: Simplifies writing asynchronous code for efficient execution.

6. Integration with Other Languages: Allows integration with languages like C, C++, and Java for cross-platform development.

7. Portable: Runs on different platforms, enhancing its usability across systems.

8. Readability and Maintainability: Simple syntax and readability make code maintenance easier.

4.What are the advantages of using Python as a programming language for AI and ML?

The key advantages of using Python for AI and ML development are:

1. Simplicity and Readability: Python has a simple and intuitive syntax, making it easy to learn and understand, even for non-developers like data scientists. This simplicity enhances collaboration and productivity in AI/ML projects.

2. Extensive Library Ecosystem: Python offers a rich ecosystem of well-tested and widely-used libraries and frameworks for AI/ML, such as NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch. This reduces development time and effort.

3. Versatility and Platform Independence: Python is a versatile language that can be used for a wide range of AI/ML tasks, from data preprocessing to building complex models. It is also platform-independent, allowing for cross-platform development.

4. Strong Community and Support: Python has a large and active community that provides extensive documentation, tutorials, and support, making it easier for developers to find solutions to problems.

5. Faster Prototyping and Experimentation: Python's simplicity and extensive libraries enable faster development and testing of AI/ML prototypes, allowing for quicker iterations and experimentation.

6. Integration with Other Languages: Python can be easily integrated with other languages like C, C++, and Java, allowing developers to leverage the strengths of multiple languages in their AI/ML projects.

7. Suitability for a Wide Range of AI/ML Applications: Python's versatility makes it suitable for a wide range of AI/ML applications, from natural language processing and computer vision to predictive analytics and autonomous systems

In summary, Python's simplicity, extensive library support, versatility, and strong community make it a preferred choice for AI and ML development, enabling faster, more efficient, and more collaborative projects.

5. Discuss the importance of indentation in Python code.

The importance of indentation in Python code is as follows:

Defining Code Blocks

• In Python, indentation is not just for readability, but it is a fundamental part of the language's syntax.

• Indentation is used to define the scope of code blocks, such as functions, loops, conditionals, and classes.

• Statements with the same level of indentation are considered part of the same code block.

• Incorrect indentation can lead to syntax errors and cause the code to fail to execute properly.

Improving Readability

• Proper indentation makes the structure of the code more visible and easier to understand.

• It helps to visually distinguish different blocks of code, making the code more readable and maintainable.

Reducing Errors

• Indentation helps to reduce common errors found in other languages, such as missing curly braces or keywords to define code blocks.

• Consistent indentation throughout the codebase ensures that the code follows the expected structure and reduces the chances of introducing bugs.

Easier Collaboration

• Proper indentation makes it easier for other developers to understand and work with the code, facilitating collaboration.

• It helps to establish a common coding style within a team or organization, improving code consistency

In summary, indentation is a crucial aspect of Python's syntax, as it defines the structure of the code and plays a vital role in improving readability, reducing errors, and enabling better collaboration among developers.

6. Define a variable in Python. Provide examples of valid variable names.

In Python, a variable is a named storage location used to store data values. Variables are essential for holding, manipulating, and referencing data throughout a program. They are created by assigning a value to a name, allowing developers to reuse and work with data efficiently.

Examples of Valid Variable Names in Python:

1. Valid Variable Names:

? Variable names must start with a letter or an underscore (\_).

? They can only contain letters, numbers, and underscores.

? Variable names cannot contain spaces or special characters.

? Variables are case-sensitive.

2. Examples:

? Valid:

? user\_age

? website

? password1

? answer

? weight

? message

? is\_authenticated

? fruits

? point

3. Naming Conventions:

? Variable names should be descriptive, not too short or too long.

? Use lowercase letters and underscores to separate words (snake\_case).

? Follow PEP8 naming conventions for consistency.

In Python, variables play a crucial role in storing and manipulating data. By following the naming rules and conventions, developers can create meaningful and readable variable names that enhance code clarity and maintainability.

7.Explain the difference between a keyword and an identifier in Python.

S. NO

KEYWORD

IDENTIFIER

1.

Keywords are predefined word that gets reserved for working program that have special meaning and cannot get used anywhere else.

Identifiers are the values used to define different programming items such as variables, integers, structures, unions and others and mostly have an alphabetic character.

2.

Specify the type/kind of entity.

Identify the name of a particular entity.

3.

It always starts with a lowercase letter.

First character can be a uppercase, lowercase letter or underscore.

4.

A keyword should be in lower case.

An identifier can be in upper case or lower case.

5.

A keyword contains only alphabetical characters.

An identifier can consist of alphabetical characters, digits and underscores.

6.

They help to identify a specific property that exists within a computer language.

They help to locate the name of the entity that gets defined along with a keyword.

7.

No special symbol, punctuation is used.

No punctuation or special symbol except ‘underscore’ is used.

8.

Examples of keywords are: int, char, if, while, do, class etc.

Examples of identifiers are: Test, count1, high\_speed, etc.

8. List the basic data types available in Python.

Python Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, Python data types are classes and variables are instances (objects) of these classes. The following are the standard or built-in data types in Python:

• Numeric

• Sequence Type

• Boolean

• Set

• Dictionary

• Binary Types

9. Describe the syntax for an if statement in Python.

In Python, the syntax for an if statement is structured as follows:

if condition: # Code block to execute if the condition is true

• The if keyword is used to start the conditional statement.

• Condition is a logical expression that evaluates to either True or False.

• The colon:signifies the start of the code block that will be executed if the condition is true.

• The code block under the ifstatement is indented to indicate that it belongs to the conditional block.

Example:

x = 10

if x > 5:

print("x is greater than 5")

In this example, if the value of x is greater than 5, the statement "x is greater than 5" will be printed.

Flow of Execution:

1. The condition after the if keyword is evaluated.

2. If the condition is True, the code block under the if statement is executed.

3. If the condition is False, the code block is skipped, and the program continues to the next block of code.

The syntax and structure of the if statement in Python are crucial for implementing conditional logic in programs, allowing developers to control the flow of execution based on specific conditions.

10. Explain the purpose of the elif statement in Python.

In Python, the elif statement is a crucial part of conditional statements used for decision-making. Here is the explanation of the purpose of the elif statement based on the provided sources:

1. Conditional Logic:

? The elif statement, short for "else if," allows for checking multiple conditions sequentially after an initial if statement.

? It is used when the initial if condition is not met, and you want to check for additional conditions.

? This sequential checking of conditions enables more complex decision-making in Python programs.

2. Series of Checks:

? if, elif, and else statements can be combined to create a series of checks where each condition is evaluated in order.

? The elif statement provides an alternative condition to check if the initial if condition is false.

3. Syntax:

? The syntax for using elif is elif condition:, followed by the code block to execute if the condition is true.

? It allows for a structured way to handle multiple possible outcomes based on different conditions.

4. Example:

? In scenarios where you need to evaluate multiple conditions and execute different code blocks based on those conditions, the elif statement is essential.

? It helps in creating more flexible and detailed decision-making processes within Python programs.

In summary, the elif statement in Python serves the purpose of providing an alternative condition to check if the initial if condition is false, allowing for sequential evaluation of multiple conditions and enabling more complex decision-making in Python programs.