**ASSIGNMENT – 1**

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

## A. AI

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include [expert systems](https://www.techtarget.com/searchenterpriseai/definition/expert-system), [natural language processing](https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP), speech recognition and [machine vision](https://www.techtarget.com/searchenterpriseai/definition/machine-vision-computer-vision).

### Advantages of AI

The following are some advantages of AI.

* **Good at detail-oriented jobs.** AI has proven to be just as good, if not better than doctors at diagnosing certain cancers, including [breast cancer](https://www.nytimes.com/2023/03/05/technology/artificial-intelligence-breast-cancer-detection.html) and [melanoma](https://www.curemelanoma.org/blog/article/artificial-intelligence-and-melanoma-detection-closing-the-gaps).
* **Reduced time for data-heavy tasks.** AI is widely used in data-heavy industries, including banking and securities, pharma and insurance, to reduce the time it takes to analyze big data sets. Financial services, for example, routinely [use AI to process loan applications and detect fraud](https://www.techtarget.com/searchenterpriseai/feature/AI-in-banking-industry-brings-operational-improvements).

### Disadvantages of AI

The following are some disadvantages of AI.

* Expensive.
* Requires deep technical expertise.
* Limited supply of [qualified workers to build AI tools](https://www.techtarget.com/whatis/feature/How-to-become-an-artificial-intelligence-engineer).
* Reflects the biases of its training data, at scale.

**Examples**

### Chatbots.

Microsoft Bing.

Google Duplex and Hold For Me.

# **Application of AI**

Artificial Intelligence has various applications in today's society. It is becoming essential for today's time because it can solve complex problems with an efficient way in multiple industries, such as Healthcare, entertainment, finance, education, etc. AI is making our daily life more comfortable and fast.

Following are some sectors which have the application of Artificial Intelligence:

### **1. AI in Astronomy**

* Artificial Intelligence can be very useful to solve complex universe problems. AI technology can be helpful for understanding the universe such as how it works, origin, etc.

### **2. AI in Healthcare**

* In the last, five to ten years, AI becoming more advantageous for the healthcare industry and going to have a significant impact on this industry.
* Healthcare Industries are applying AI to make a better and faster diagnosis than humans. AI can help doctors with diagnoses and can inform when patients are worsening so that medical help can reach to the patient before hospitalization.

### **3. AI in Gaming**

* AI can be used for gaming purpose. The AI machines can play strategic games like chess, where the machine needs to think of a large number of possible places.

### **4. AI in Finance**

* AI and finance industries are the best matches for each other. The finance industry is implementing automation, chatbot, adaptive intelligence, algorithm trading, and machine learning into financial processes.

### **5. AI in Data Security**

* The security of data is crucial for every company and cyber-attacks are growing very rapidly in the digital world. AI can be used to make your data more safe and secure. Some examples such as AEG bot, AI2 Platform,are used to determine software bug and cyber-attacks in a better way.

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

## A ****Supervised learning****

Supervised learning is a type of [machine learning algorithm](https://www.geeksforgeeks.org/machine-learning-algorithms/) that learns from labeled data. Labeled data is data that has been tagged with a correct answer or classification.

Supervised learning, as the name indicates, has the presence of a supervisor as a teacher. Supervised learning is when we teach or train the machine using data that is well-labelled. Which means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the training data(set of training examples) and produces a correct outcome from labeled data.

For example, a labeled dataset of images of Elephant, Camel and Cow would have each image tagged with either “Elephant” , “Camel”or “Cow.”

* Supervised learning involves training a machine from labeled data.
* Labeled data consists of examples with the correct answer or classification.
* The machine learns the relationship between inputs (fruit images) and outputs (fruit labels).
* The trained machine can then make predictions on new, unlabeled data.

## Types of Supervised Learning

Supervised learning is classified into two categories of algorithms:

* [**Regression**:](https://www.geeksforgeeks.org/regression-classification-supervised-machine-learning/) A regression problem is when the output variable is a real value, such as “dollars” or “weight”.
* [**Classification**:](https://www.geeksforgeeks.org/getting-started-with-classification/) A classification problem is when the output variable is a category, such as “Red” or “blue” , “disease” or “no disease”.

Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer.

## ****Unsupervised learning****

Unsupervised learning is a type of machine learning that learns from unlabeled data. This means that the data does not have any pre-existing labels or categories. The goal of unsupervised learning is to discover patterns and relationships in the data without any explicit guidance.

Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance. Here the task of the machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data.

Unlike supervised learning, no teacher is provided that means no training will be given to the machine. Therefore the machine is restricted to find the hidden structure in unlabeled data by itself.

* Unsupervised learning allows the model to discover patterns and relationships in unlabeled data.
* Clustering algorithms group similar data points together based on their inherent characteristics.
* Feature extraction captures essential information from the data, enabling the model to make meaningful distinctions.
* Label association assigns categories to the clusters based on the extracted patterns and characteristics.

## Types of Unsupervised Learning

Unsupervised learning is classified into two categories of algorithms:

* **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
* **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

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

1. python

**Python** is a dynamic, high-level, free open source, and interpreted programming language. It supports object-oriented programming as well as procedural-oriented programming. In Python, we don’t need to declare the type of variable because it is a dynamically typed language.

For example,

x = 10 Here, x can be anything such as String, int, etc. In this article we will see what characteristics describe the python programming language

### **1. Free and Open Source**

Python language is freely available at the official website and you can download it from the given download link below click on the **Download Python** keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it, use it as well as share it.

### **2. Easy to code**

Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in the Python language and anybody can learn Python basics in a few hours or days. It is also a developer-friendly language.

### 3. Easy to Read

As you will see, learning Python is quite simple. As was already established, Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

### **4. Object-Oriented Language**

One of the key features of Python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, object encapsulation, etc.

### **5. GUI Programming Support**

Graphical User interfaces can be made using a module such as PyQt5, PyQt4,or Tk in Python. PyQt5 is the most popular option for creating graphical apps with Python.

### **6. High-Level Language**

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

### **7. Large Community Support**

Python has gained popularity over the years. Our questions are constantly answered by the enormous StackOverflow community. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.

### 8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to interpret Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

### **9. Python is a Portable language**

Python language is also a portable language. For example, if we have Python code for Windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

### **10. Python is an Integrated language**

Python is also an Integrated language because we can easily integrate Python with other languages like C, C++, etc.

### **11. Interpreted Language:**

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called **bytecode**.

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

A. Python is widely regarded as one of the best programming languages for artificial intelligence (AI) and machine learning (ML) due to several advantages it offers in these domains:

1. Rich Ecosystem of Libraries:

Python boasts a vast ecosystem of libraries and frameworks specifically designed for AI and ML, such as TensorFlow, PyTorch, scikit-learn, Keras, and NumPy. These libraries provide robust support for tasks like neural networks, deep learning, natural language processing (NLP),

computer vision, and more, enabling developers to leverage pre-built components and accelerate development.

2. Ease of Prototyping and Development:

Python's simple and concise syntax, combined with its high-level abstractions, facilitates rapid prototyping and development in AI and ML projects. Developers can quickly experiment with algorithms, models, and data without getting bogged down by low-level details, leading to faster iteration cycles and reduced time-to-market.

3. Community Support and Documentation:

Python has a large and active community of developers, researchers, and enthusiasts who contribute to its ecosystem by creating tutorials, documentation, and educational resources. This abundant community support makes it easier for newcomers to learn AI and ML concepts, troubleshoot issues, and collaborate on projects.

4. Flexibility and Versatility:

Python's flexibility and versatility make it suitable for a wide range of AI and ML tasks, from simple data analysis to complex deep learning projects. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming, allowing developers to choose the approach that best fits their project requirements.

5. Integration with Other Languages and Tools:

Python can be seamlessly integrated with other languages and tools commonly used in AI and ML, such as C/C++, Java, and MATLAB. This interoperability enables developers to leverage existing libraries and tools written in other languages, optimize performance-critical sections of code, and incorporate domain-specific functionalities into their Python-based projects.

6. Scalability and Performance Optimization:

While Python is not as performant as compiled languages like C++ or Java, it offers scalability through various optimization techniques. Developers can improve performance by leveraging multi-threading, multiprocessing, asynchronous programming, and using optimized libraries and frameworks like NumPy, TensorFlow, or PyTorch for computationally intensive tasks.

5. Discuss the importance of indentation in Python code.

## A. Indentation in Python

In the realm of Python programming, “indentation” signifies more than mere aesthetic preference; it's an integral element defining the language's syntax. The term “Indentation in Python” refers to the whitespaces at the start of a code line. Python utilizes indentation to ascertain the grouping of statements, and unlike other programming languages that use braces, parentheses, or keywords, Python uses this indentation approach to delimit the blocks of code for functions, loops, and conditionals.

A small indentation error can lead to syntax problems and incorrect code execution. As such, understanding and mastering Python indentation is a core skill that every Python programmer must acquire. It streamlines code readability and organization, fostering a smooth coding experience.

why indentation is important in Python:

1. Readability:

Python emphasizes readability and clean code, and indentation greatly contributes to this. By using consistent indentation, developers can easily understand the structure of the code and visually identify blocks of code, such as loops, conditionals, and function definitions. This makes code easier to read, maintain, and debug, especially for large and complex projects.

2. Enforcement of Code Structure:

In Python, indentation is not just a matter of style; it is enforced by the interpreter as part of the language syntax. Incorrect indentation can lead to syntax errors or, even worse, alter the logical structure of the code, resulting in unintended behavior. Therefore, proper indentation is essential for ensuring the correct interpretation of Python code.

3.Clarity of Nested Blocks:

Indentation helps to clearly define nested blocks of code. In Python, indentation levels indicate the hierarchy of code blocks, making it easy to understand which statements are part of a loop, conditional, function, or class definition. This clarity reduces ambiguity and enhances code comprehension, even for complex nested structures.

4. Consistency:

Python's indentation rules encourage consistent coding practices across different projects and among team members. By adhering to a standardized indentation style, developers can maintain a uniform codebase, promote collaboration, and reduce the likelihood of errors introduced by inconsistent formatting.

5. Expressiveness and Concision:

Python's reliance on indentation allows for more expressive and concise code compared to languages that use explicit block delimiters. Instead of cluttering the code with curly braces or keywords, Python code relies on indentation to convey structure, resulting in cleaner and more compact code that focuses on the logic rather than the syntax.

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

## A. Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:

* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* A variable name cannot be any of the Python keywords.

### **Example**

Legal variable names:

myvar = "krish"  
my\_var = "krish"  
\_my\_var = "krish"  
myVar = "krish"  
MYVAR = "krish"  
myvar2 = "krish"

### **Example**

Illegal variable names:

2myvar = "krish"  
my-var = "krish"  
my var = "krish"

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

## A. Keywords in Python

**Python Keywords** are some predefined and reserved words in Python that have special meanings. Keywords are used to define the syntax of the coding. The keyword cannot be used as an identifier, function, or variable name. All the keywords in Python are written in lowercase except True and False. There are 35 keywords in Python 3.11.

In Python, there is an inbuilt keyword module that provides an iskeyword() function that can be used to check whether a given string is a valid keyword or not. Furthermore, we can check the name of the keywords in Python by using the kwlist attribute of the keyword module.

## Rules for Keywords in Python

* Python keywords cannot be used as identifiers.
* All the keywords in Python should be in lowercase except True and False.

## List of Python Keywords

| **Keywords** | **Description** |
| --- | --- |
| and | This is a logical operator which returns true if both the operands are true else returns false. |
| or | This is also a logical operator which returns true if anyone operand is true else returns false. |
| not | This is again a logical operator it returns True if the operand is false else returns false. |
| if | This is used to make a conditional statement. |
| elif | Elif is a condition statement used with an if statement. The elif statement is executed if the previous conditions were not true. |

Code:

# code

**import** keyword

print(keyword.kwlist)

**Output**

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',...]

## Identifiers in Python

**Identifier**is a user-defined name given to a variable, function, class, module, etc. The identifier is a combination of character digits and an underscore. They are case-sensitive i.e., ‘num’ and ‘Num’ and ‘NUM’ are three different identifiers in python. It is a good programming practice to give meaningful names to identifiers to make the code understandable.

We can also use the Python string is identifier () method to check whether a string is a valid identifier or not.

## Rules for Naming Python Identifiers

* It cannot be a reserved python keyword.
* It should not contain white space.
* It can be a combination of A-Z, a-z, 0-9, or underscore.
* It should start with an alphabet character or an underscore ( \_ ).
* It should not contain any special character other than an underscore ( \_ ).

## Examples of Python Identifiers

***Valid identifiers:***

* *var1*
* *\_var1*
* *\_1\_var*
* *var\_1*

***Invalid Identifiers***

* *!var1*
* *1var*
* *1\_var*
* *var#1*
* *var 1*

8.List the basic data types available in Python.

# A. Python Data Types

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**

Here are the basic data types available in Python:

1. Integer (`int`):

Integers represent whole numbers without any decimal point. They can be positive, negative, or zero. Example: `x = 10`

2. Float (`float`):

Floats represent real numbers with a decimal point. They can also be written in scientific notation. Example: `y = 3.14`

3. String (`str`):

Strings represent sequences of characters enclosed within single (' '), double (" "), or triple (''' ''' or """ """) quotes. Example: `name = 'John'`

4. Boolean (`bool`):

Booleans represent the two truth values `True` and `False`. They are used to represent logical states. Example: `is\_active = True`

5. List (`list`):

Lists are ordered collections of items, which can be of different data types. They are mutable, meaning their elements can be changed after creation. Lists are defined by square brackets `[ ]`. Example: `my\_list = [1, 2, 3, 'a', 'b', 'c']`

6. Tuple (`tuple`):

Tuples are similar to lists but are immutable, meaning their elements cannot be changed after creation. Tuples are defined by parentheses `( )`. Example: `my\_tuple = (1, 2, 3, 'a', 'b', 'c')`

7. Dictionary (`dict`):

Dictionaries are unordered collections of key-value pairs. Each key-value pair maps the key to its corresponding value. Dictionaries are defined by curly braces `{ }`, with each key-value pair separated by a colon `:`. Example: `my\_dict = {'name': 'John', 'age': 30, 'city': 'New York'}`

8. Set (`set`):

Sets are unordered collections of unique elements. They are useful for storing and performing operations on unique items. Sets are defined by curly braces `{ }`. Example: `my\_set = {1, 2, 3, 4, 5}`

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

A. In Python, an `if` statement is used to conditionally execute a block of code based on a specified condition. The syntax for an `if` statement in Python is as follows:

if condition:

# Indented block of code to be executed if the condition is true

    statement1

    statement2

    ...

Here's a breakdown of the syntax components:

- `if`: The keyword that introduces the `if` statement, indicating that a condition is to be evaluated.

- `condition`: A Boolean expression that evaluates to either `True` or `False`. If the condition evaluates to `True`, the block of code following the `if` statement is executed. If the condition evaluates to `False`, the block of code is skipped.

- `:` (colon): A colon is used to signify the end of the `if` statement's condition and the beginning of the indented block of code that should be executed if the condition is true.

- Indented block of code: The block of code that will be executed if the condition specified in the `if` statement evaluates to `True`. This block of code must be indented consistently by a fixed number of spaces (typically four spaces), as indentation is significant in Python.

Here's an example of an `if` statement in Python:

x = 10

if x > 5:

    print("x is greater than 5")

In this example, the condition `x > 5` is evaluated. Since the value of `x` is `10`, which is greater than `5`, the condition evaluates to `True`. Therefore, the indented block of code (containing the `print` statement) is executed, and the output will be:

x is greater than 5

If the condition had been `x < 5`, the condition would evaluate to `False`, and the indented block of code would be skipped.

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

## A. elif Condition

Python elif (short for else if) is used to execute a continuous chain of conditional logic ladder.

In elif, there are multiple conditions and the corresponding statement(s) as a ladder. Only one of the blocks gets executed when the corresponding boolean expression evaluates to true.

## Syntax of elif

The syntax of python elif statement is as shown below.

if boolean\_expression\_1:

statement(s)

elif boolean\_expression\_2:

statement(s)

elif boolean\_expression\_3:

statement(s)

else

statement(s)

note: You can have as many elif statements as required.

Here's a breakdown of the purpose and usage of the `elif` statement:

1. Alternative Conditions: The `elif` statement allows you to check additional conditions after the initial `if` condition. If the `if` condition evaluates to `False`, Python evaluates the `elif` condition. If the `elif` condition is `True`, the corresponding block of code is executed. If the `elif` condition is `False`, Python
2. continues to the next `elif` statement or the `else` statement if no further `elif` conditions exist.

2. Chained Conditions: `elif` statements can be chained together to test multiple conditions in sequence. This allows you to handle complex scenarios where different actions need to be taken based on various conditions.

3. Mutual Exclusivity: Each `elif` statement is evaluated only if the preceding `if` and `elif` conditions are `False`. This ensures that the conditions are mutually exclusive, and only one block of code is executed, even if multiple conditions are true.

4. Default Case: Optionally, you can include an `else` statement at the end of the `if-elif-else` chain to provide a default case to be executed if none of the preceding conditions are true.

Here's an example of using `if`, `elif`, and `else` statements in Python:

x = 20

if x > 20:

    print("x is greater than 20")

elif x < 20:

    print("x is less than 20")

else:

    print("x is equal to 20")

In this example, the `elif` statement checks whether `x` is less than `20` if the initial condition `x > 20` is false. If `x` is indeed less than `20`, the corresponding message is printed. If neither the `if` condition nor the `elif` condition is true, the `else` block is executed, printing "x is equal to  "20".