1. Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include expert systems, natural language processing, speech recognition and machine vision.

AI is incorporated into a variety of different types of technology. Here are seven examples.

Automation. When paired with AI technologies, automation tools can expand the volume and types of tasks performed. An example is robotic process automation (RPA), a type of software that automates repetitive, rules-based data processing tasks traditionally done by humans. When combined with machine learning and emerging AI tools, RPA can automate bigger portions of enterprise jobs, enabling RPA's tactical bots to pass along intelligence from AI and respond to process changes.

Machine learning. This is the science of getting a computer to act without programming. Deep learning is a subset of machine learning that, in very simple terms, can be thought of as the automation of predictive analytics. There are three types of machine learning algorithms:

- **Supervised learning.** Data sets are labeled so that patterns can be detected and used to label new data sets.
- **Unsupervised learning.** Data sets aren't labeled and are sorted according to similarities or differences.
- **Reinforcement learning.** Data sets aren't labeled but, after performing an action or several actions, the AI system is given feedback.

Machine vision. This technology gives a machine the ability to see. Machine vision captures and analyzes visual information using a camera, analog-to-digital conversion and digital signal processing. It is often compared to human eyesight, but machine vision isn't bound by biology and can be programmed to see through walls, for example. It is used in a range of applications from signature identification to medical image analysis. Computer vision, which is focused on machine-based image processing, is often conflated with machine vision.

Natural language processing (NLP). This is the processing of human language by a computer program. One of the older and best-known examples of NLP is spam detection, which looks at the subject line and text of an email and decides if it's junk. Current approaches to NLP are based on machine learning. NLP tasks include text translation, sentiment analysis and speech recognition.

Robotics. This field of engineering focuses on the design and manufacturing of robots. Robots are often used to perform tasks that are difficult for humans to perform or perform consistently. For example, robots are used in car production assembly lines or by NASA to move large objects in space. Researchers also use machine learning to build robots that can interact in social settings.

Self-driving cars. Autonomous vehicles use a combination of computer vision, image recognition and deep learning to build automated skills to pilot a vehicle while staying in a given lane and avoiding unexpected obstructions, such as pedestrians.

Text, image and audio generation. Generative AI techniques, which create various types of media from text prompts, are being applied extensively across businesses to create a seemingly limitless range of content types from photorealistic art to email responses and screenplays

2. Supervised vs unsupervised learning compared

The main difference between supervised vs unsupervised learning is the need for labelled training data. Supervised machine learning relies on labelled input and output training data, whereas unsupervised learning processes unlabelled or raw data. In supervised machine learning the model learns the relationship between the labelled input and output data.

The main differences of supervised vs unsupervised learning include:

- The need for labelled data in supervised machine learning.
- The problem the model is deployed to solve. Supervised machine learning is generally used to classify data or make predictions, whereas unsupervised learning is generally used to understand relationships within datasets.
- Supervised machine learning is much more resource-intensive because of the need for labelled data.
- In unsupervised machine learning it can be more difficult to reach adequate levels of explainability because of less human oversight.

3.Python is a computer programming language often used to build websites and software, automate tasks, and analyze data. Python is a general-purpose language, not specialized for any specific problems, and used to create various programmes.

Here are a few features of Python that make it a popular programming language in today's time.

1. Portable Language

It is a cross-platform language. It can run on Linux, macOS, and Windows. For example, you can run the Python code for Windows in Linux or macOS, too.

2. Standard Library

It offers various modules like operators, mathematical functions, libraries such as NumPy, Pandas, Tensorflow, etc., and packages paving the way for the developers to save time to avoid re-writing the codes from scratch. To provide more functionality and packages, they also provide Python Package Index.

3. High-Level Language

It is a high-level, general-purpose programming language. Unlike machine language like C, C++, It is a human-readable language. In other words, even a layman can understand the programs.

4. Easy to learn and use

It is easy to understand and easy to code, and anyone can learn Python within a few days. For example, a simple Python program to add two numbers is as follows:

```
a = 8
b = 9
print(a+b)
```

We have completed this program within three lines. Whereas in Java, C++ and C, it takes more lines. That is why Python is known as an easy and precise language.

5. Dynamic Language

Declaring the type of a variable is not needed. For example, let us declare an integer number 7 for a variable a. Rather than declare it as:

int a = 7 (this is necessary for statically-typed language like C)
We declare it as
a = 7

But, the programmers have to be careful regarding runtime errors.

6. Extensible Language

Code can be used to compile in C or C++ language so that it can be utilized for our Python code. This is achieved because it converts the program to byte code.

7. Interpreted Language

Line-by-line execution of source code, converted into byte code; thus, compiling the code is not necessary, making it easy to debug if required. Python has always been a favorite programming language for programmers who start their careers in the field of Information Technology. This may be because of the advantages of Python and that it is easy to learn and apply. Here are the top five applications of Python.

1. Web development
2. Data Analysis
3. Artificial Intelligence, Machine Learning, Deep Learning,
4. Data Science
5. Software Development

4. is a popular choice for artificial intelligence (AI) and machine learning (ML) projects due to several compelling advantages:

Huge Number of Libraries and Frameworks:

Python has an extensive ecosystem of libraries and frameworks specifically designed for AI and ML. These pre-built tools significantly reduce the complexity of writing code.

Libraries like Scikit-learn, spaCy, Natural Language Toolkit (NLTK), NumPy, Pandas, and Seaborn provide implementations of various ML algorithms and data manipulation capabilities.

Popular deep learning libraries such as TensorFlow, PyTorch, and Keras are also widely used in the AI community1.

Easy Syntax and Readability:

Python’s syntax is straightforward and resembles everyday English, making it easy for developers to learn and understand.

Unlike languages that use brackets, Python relies on indentation for code structure, which reduces complexity.

No Need to Recompile Source Code:

Developers can make changes to Python code without the need for recompilation. This flexibility speeds up development and debugging.

Platform Independence:

Python code runs seamlessly on different platforms, including Windows, Mac, UNIX, and Linux.

Great Community Support:

Python is an open-source language with a large global community. Developers at all levels actively contribute and provide support.

The community's willingness to help is especially valuable during debugging and troubleshooting.

Simplicity and Readability:

Python's design emphasizes readability, allowing developers to focus on problem-solving aspects of AI and ML applications.

5. Indentation in Python is a fundamental concept that plays a crucial role in the structure and readability of your code. Let's explore why indentation matters:

Code Blocks and Scope:

In Python, indentation is used to define code blocks (such as loops, conditionals, and functions). Unlike other programming languages that use braces {} or keywords like begin and end, Python relies on consistent indentation.

Proper indentation defines the scope of a block. The statements indented at the same level belong to the same block. For example:

Python

```
if x > 10:
    print("x is greater than 10")
    print("This is still inside the if block")
else:
    print("x is not greater than 10")
```

AI-generated code. Review and use carefully. More info on FAQ.

Readability and Aesthetics:

Well-indented code is easier to read and understand. It visually separates different levels of nesting.

Consistent indentation makes your code look clean and professional.

Compare the following examples:

Python

```
# Poor indentation
```

```
def calculate_total(price, quantity):

total = price * quantity

return total

# Proper indentation

def calculate_total(price, quantity):

    total = price * quantity

    return total
```

AI-generated code. Review and use carefully. More info on FAQ.

Avoiding Errors:

Incorrect indentation can lead to syntax errors. Python relies on consistent indentation to determine the structure of your code.

Forgetting to indent or using inconsistent indentation can result in unexpected behavior.

Example:

Python

```
# Incorrect indentation

if x > 5:

print("x is greater than 5")  # IndentationError

# Correct indentation

if x > 5:

    print("x is greater than 5")
```

6.variable is a name given to a memory location that stores a value. Unlike some other programming languages, Python is not “statically typed,” which means we don’t need to declare variables before using them or specify their type explicitly. Instead, a variable is created the moment we assign a value to it. Let’s explore some examples:

Assigning an Integer Value:

Python

```
age = 30

print(age)  # Output: 30
```

Assigning a Floating-Point Value:

Python

```
salary = 1456.8

print(salary)  # Output: 1456.8
```

AI-generated code. Review and use carefully. More info on FAQ.

Assigning a String Value:

Python

name = "John"

print(name) # Output: John

7.Keywords:

Keywords are predefined reserved words in Python that have special meanings and cannot be used as identifiers.

They are an essential part of the language syntax and serve specific purposes.

Examples of Python keywords include if, else, for, while, def, return, True, False, and many more.

Keywords are case-sensitive, meaning that if and IF are treated differently.

You cannot use a keyword as a variable name, function name, or any other identifier.

All Python keywords are written in lowercase except for True and False.

There are 35 keywords in Python 3.11.

You can check whether a given string is a valid keyword using the iskeyword() function from the keyword module1.

Identifiers:

Identifiers are user-defined names used to identify various programming entities, such as variables, functions, classes, modules, and other objects in Python code.

Unlike keywords, identifiers are not predefined; you create them based on your needs.

Rules for identifiers:

They must start with a letter (a-z, A-Z) or an underscore (_).

The subsequent characters can be letters, digits (0-9), or underscores.

Identifiers are case-sensitive (myVar and myvar are different identifiers).

Avoid using Python keywords as identifiers.

Examples of valid identifiers: my_variable, user_input, total_sum, calculate_area.

Examples of invalid identifiers: if, for, while, True, False (since they are keywords).

8. Python provides several built-in data types that allow you to represent different kinds of data. Let's explore them:

Integers (int):

Used for whole numbers (positive, negative, or zero).

Example: 42, -10, 0

Floating-Point Numbers (float):

Represent decimal numbers.

Example: 3.14, -0.5, 2.718

Complex Numbers (complex):

Consist of a real part and an imaginary part.

Written as a + bj, where a and b are real numbers.

Example: 1 + 2j, -3.5 + 4j

Strings (str):

Used for textual data (sequences of characters).

Enclosed in single (') or double (") quotes.

Example: "Hello, World!", 'Python'

Booleans (bool):

Represent truth values (True or False).

Used for logical operations and control flow.

Example: True, False

9. if statement in Python is a fundamental control structure that allows you to execute a block of code only when a specific condition is met. Let's dive into the details:

Basic Syntax of the if Statement:

The basic syntax of the if statement is as follows:

Python

```
if condition:
    # Body of the if statement
```

Here's what each part means:

condition: A boolean expression that evaluates to either True or False.

Body of the if statement: The indented code block that executes if the condition is True.

For example:

Python

```
number = 10
if number > 0:
    print('Number is positive')
print('This statement always executes')
```

AI-generated code. Review and use carefully. More info on FAQ.

Output:

```
Number is positive

This statement always executes
```

In the above example, since number is greater than 0, the condition evaluates to True, and the body of the if statement executes. The subsequent print statement also runs regardless of the condition.

Adding an else Clause:

You can include an optional else clause to handle the case when the condition evaluates to False.

The syntax for an if...else statement is:

Python

```python
if condition:

   # Body of the if statement

else:

   # Body of the else statement
```

AI-generated code. Review and use carefully. More info on FAQ.

If the condition is True, the if block executes; otherwise, the else block executes.

Example:

Python

```python
number = 10

if number > 0:

   print('Positive number')

else:

   print('Negative number')

print('This statement always executes')
```

Output:

```
Positive number

This statement always executes
```

10. elif Statement:

When you have multiple conditions to check, you can use the elif statement.

Python evaluates the if condition first. If it’s False, it moves on to the elif conditions.

The first elif block with a True condition will be executed.

Example:

Python

```
price = 50

quantity = 5

if price * quantity < 500:

   print("price*quantity is less than 500")

   print("price =", price)

   print("quantity =", quantity)
```

Output:

```
price*quantity is less than 500

price = 50

quantity = 5
```

else Statement (Optional):

You can also include an else block after the if and elif conditions.

The code inside the else block executes if none of the previous conditions are True.

Example:

Python

```
a = 33

b = 33

if b > a:

   print("b is greater than a")

elif a == b:

   print("a and b are equal")
```

Output: “a and b are equal”