**Q1. Why do we call Python as a general purpose and high-level programming language?**

Python is known as a general-purpose and high-level programming language due to numerous fundamental properties that make it versatile and user-friendly. Let's define these words:

General-purpose: A general-purpose programming language is made to be able to carry out a variety of jobs and solve a wide range of issues. Python is a general-purpose programming language that may be used for a wide range of tasks, including web development, data analysis, scientific computing, artificial intelligence, automation, scripting, and many other things. Because of its adaptability, developers can work on a variety of projects without having to learn a different language for each one.

High-level: Python is regarded as a high-level language since it abstracts a lot of the operating system and hardware's low-level elements. In other words, Python allows programmers to develop code without having to worry about it.

**Q2. Why is Python called a dynamically typed language?**

Python is referred to as a dynamically typed language because variables' data types are chosen during execution rather than when they are first declared. The data type of a variable does not need to be specified explicitly in dynamically typed programming languages like Python; Python instead determines the data type depending on the value that is assigned to the variable.

**Q3. List some pros and cons of Python programming language?**

Python is a popular and versatile programming language with many advantages, but also some disadvantages. Here is a list of pros and cons of Python.

**Pros:**

Easy to learn and read: Python has a clear and simple syntax similar to English, making it easy for beginners to learn and understand.

Large community and extensive libraries: Python has a large developer community and numerous libraries and packages available, making it easy to find solutions to various tasks without reinventing the wheel. can do.

Versatility and platform dependency: Python is a cross-platform language that allows you to write code on one platform and run it on another with minimal changes.

High-level language: As a high-level language, Python abstracts away many low-level details, simplifying development and saving time. Extensive data science and machine learning ecosystem: Python is widely used in data science and machine learning with libraries such as NumPy, Pandas, Scikit-Learn, TensorFlow, and PyTorch.

Web development: Python has several web frameworks such as Django and Flask that facilitate web application development.

Easy to read and maintain code: Python's clean and consistent syntax makes it easier to maintain and collaborate on large projects.

Rapid Prototyping: Python's ease of use and fast development cycles make it ideal for rapid prototyping and iterative development.

**Cons:**

Performance: Compared to low-level languages ​​like C and C++, Python can be slow due to its interpreted nature and dynamic typing. However, this can be mitigated by using external libraries written in low-level languages ​​for performance-critical tasks.

Global Interpreter Lock (GIL): Python's GIL allows only one thread to execute Python bytecode at a time, limiting performance gains from multi-core processors in CPU-bound applications.

Mobile development: Python is not the best language for mobile app development, but frameworks like Kivy and BeeWare aim to fill this gap.

Packaging and versioning: Python packaging and versioning can be difficult at times, leading to dependency conflicts and versioning issues.

Poorly suited for CPU-intensive tasks: Due to performance limitations, Python may not be well-suited for CPU-intensive tasks, especially when compared to languages ​​like C++ there is. Not ideal for memory-intensive applications: Python's memory consumption is relatively high, which can be a problem for memory-intensive applications.

Less secure web applications: Python web frameworks such as Django are secure, but Python itself may not be as secure as other languages ​​due to dynamic typing and the resulting vulnerabilities.

**Q4. In what all domains can we use Python?**

Python is a versatile and widely used programming language that can be used in many different fields. Its ease of use, readability, and extensive library make it popular for a variety of uses. Here are some of the domains in which Python is commonly used:

**Web development:** Python can be used to build web applications and websites using frameworks such as Django, Flask and Pyramid.

**Data Science:** Python is commonly used for data analysis, data visualization, and machine learning. Libraries such as NumPy, Pandas, Matplotlib, and Scikit-Learn are commonly used for these purposes. Artificial Intelligence and Machine Learning: Python is a popular choice for developing AI and ML applications due to libraries such as TensorFlow, Keras, PyTorch, and scikit-learn.

**Scientific computing:** Python is widely used in scientific research and engineering thanks to libraries such as SciPy, NumPy, and Pandas.

**Automation and scripting:** Python's simplicity and ease of use make it ideal for automation tasks and scripting.

**Game development:** Python can be used to develop 2D and simple 3D games using libraries such as Pygame.

**Desktop applications:** Python can be used in conjunction with frameworks such as Tkinter to create desktop applications with GUI interfaces. Mobile app development: Mobile applications for Android and iOS can be developed using frameworks such as Kivy and Python.

**Network programming:** Python's standard library includes modules for network programming, which are commonly used when building network applications.

**Cybersecurity:** Python is used in cybersecurity for tasks such as penetration testing, security analysis, and automation.

**Internet of Things (IoT):** Python is widely used for IoT projects due to its simplicity and support for a wide variety of microcontrollers and devices.

**Big data:** Python is used for big data processing and analysis and integrates well with frameworks such as Apache Spark and Hadoop. Cloud Computing: Python is widely used for cloud-related tasks, from managing cloud infrastructure to developing cloud-based applications.

Natural Language Processing (NLP): Python is commonly used in NLP applications, and

libraries like NLTK and spaCy provide tools for processing human language data.

**DevOps:** Python is commonly used to automate deployment, monitoring, and infrastructure management tasks.

These are just a few of the many areas where Python is used. Due to its versatility and large developer community, its applications continue to expand in various fields.

**Q5. What are variable and how can we declare them?**

In Python, variables are names that refer to values ​​stored in memory. It acts as a container for data such as numbers, strings, lists, and other objects. Variables are fundamental to programming because they allow us to store and manipulate data and facilitate manipulation of information in our code.

To declare (or create) a variable in Python, simply assign a value to its name using the assignment operator (=). The general syntax for declaring variables is:

variable name = value

Variable Name: This is the name you chose for your variable. Must follow Python identifier naming conventions. Valid variable names can contain letters (both uppercase and lowercase), numbers, and underscores (\_). It cannot start with a number and cannot be a reserved word (eg if, else, while, etc.).

=: The assignment operator is used to assign a value to a variable.

Value: This is the actual data to store in the variable. It can be any valid Python expression, including literals (numbers, strings, etc.) and the results of expressions.

**Q6. How can we take an input from the user in Python?**

In Python, you can receive input from the user using the built-in input() function. The input() function allows you to prompt the user for text or a value and wait for user input. When the user presses the Enter key, the input is read as a string and can be saved in a variable for further processing.

The basic syntax for using the input() function is:

user\_input = input("Please enter your input:")

In this example, the input() function displays the message "Please enter your input:" to the user and the user's input is saved as a string in his user\_input variable.

**Q7. What is the default datatype of the value that has been taken as an input using input() function?**

**input()** function to take user input,

The data type of the value entered by the user is always a string **(str).**

This means that even if the user enters a numerical value, it will be treated as a string until you explicitly convert it to the desired data type (e.g., integer or float) using functions like int() or float().

**Q8. What is type casting?**

Type casting in Python refers to the process of converting a value from one data type to another. This is often necessary when you want to perform operations or comparisons involving values of different data types.

**Q9. Can we take more than one input from the user using single input() function? If yes, how? If no, why?**

In Python, the **input()** function is designed to take a single line of input from the user, meaning it collects a single string of text entered by the user, typically terminated by pressing the "Enter" or "Return" key. It doesn't directly support taking multiple inputs in a single call. However, you can take multiple inputs from the user by using multiple **input()** statements sequentially.

**Q10. What are keywords?**

**Keywords** in Python are reserved words that have predefined meanings and cannot be used as identifiers (such as variable names, function names, or class names) in your Python programs. These keywords are an integral part of the Python language and serve specific purposes in the language's syntax and semantics. You cannot redefine or override the behavior of these keywords.

**Q11.Can we use keywords as a variable? Support your answer with reason.**

No, you cannot use keywords as variable names in Python. Keywords are reserved words that have specific meanings and purposes in the Python language. They are not allowed to be used as identifiers (such as variable names, function names, or class names) because doing so would lead to ambiguity and confusion in the code.

**Q12. What is indentation? What's the use of indentaion in Python?**

Indentation in Python refers to the whitespace (typically spaces or tabs) at the beginning of a line of code that determines the structure and grouping of statements in the program. Indentation is not just a matter of style or formatting; it is a fundamental aspect of Python's syntax and is used to define blocks of code.

**Q13. How can we throw some output in Python?**

In Python, you can display output to the console or terminal using the **print()** function. The print() function is a built-in function that allows you to output text, variables, or expressions to the standard output.

**Q14. What are operators in Python?**

Operators in Python are special symbols or keywords that are used to perform operations on operands. Operands are the values or variables on which these operations are applied. Python provides a wide range of operators to perform various tasks, such as arithmetic calculations, comparisons, logical operations, assignment, and more.

**Q15. What is difference between / and // operators?**

In Python, the / operator and the // operator are both used for division, but they have different behaviors based on the types of operands and the results they produce:

1. **/ Operator (True Division):**

The / operator performs true division or floating-point division.

It returns a floating-point result, even if the operands are integers.

If both operands are integers and the division is not exact, it returns a floating-point number with decimal places.

It is used when you want a precise result with decimal places.

Example:

result = 7 / 2 # Result is 3.5

1. **// Operator (Floor Division):**

The // operator performs floor division, which means it returns the largest integer that is less than or equal to the result.

It returns an integer result, even if the operands are floating-point numbers.

If both operands are integers, it performs integer division, discarding the decimal part and returning an integer.

It is used when you want to obtain the quotient as an integer, effectively rounding down to the nearest whole number.

Example:

result = 7 // 2 # Result is 3

**Q16. Write a code that gives following as an output.**

**iNeuroniNeuroniNeuroniNeuron**

text= "iNeuro"

output = text \* 4

print(output)

**Q17. Write a code to take a number as an input from the user and check if the number is odd or even.**

user\_input = input("Enter a number: ")

try:

number = int(user\_input)

if number % 2 == 0:

print(f"{number} is even.")

else:

print(f"{number} is odd.")

except ValueError:

print("Invalid input. Please enter a valid number.")

**Q18. What are boolean operator?**

Boolean operators in Python are used to perform logical operations on Boolean values (i.e., True or False).

**Q19. What will the output of the following?**

**```**

**1 or 0**

**0 and 0**

**True and False and True**

**1 or 0 or 0**

**Ans:**

1

0

False

1

**Q20. What are conditional statements in Python?**

Conditional statements in Python are used to control the flow of a program based on specific conditions or criteria. They allow you to make decisions in your code by executing certain blocks of code if a condition is true and potentially different blocks of code if the condition is false. Conditional statements are essential for creating programs that can respond dynamically to different situations.

Python provides several types of conditional statements, with the most common ones being:

1. **if Statements:**
2. **if-else Statements:**
3. **if-elif-else Statements:**
4. **Nested Conditional Statements:**

**Q21. What is use of 'if', 'elif' and 'else' keywords?**

The **if, elif** (short for **"else if"**), and else keywords are used in Python to create conditional statements, allowing you to control the flow of your program based on specific conditions.

These keywords are essential for making decisions in your code and executing different blocks of code depending on whether conditions are true or false. Here's a breakdown of their use:

**Q22. Write a code to take the age of person as an input and if age >= 18 display "I can vote". If age is < 18 display "I can't vote".**

try:

age = int(input("Enter your age: "))

if age >= 18:

print("I can vote.")

else:

print("I can't vote.")

except ValueError:

print("Invalid input. Please enter a valid age as a number.")

**Q23. Write a code that displays the sum of all the even numbers from the given list.**

**numbers = [12, 75, 150, 180, 145, 525, 50]**

numbers = [12, 75, 150, 180, 145, 525, 50]

even\_sum = 0

for num in numbers:

if num % 2 == 0: # Check if the number is even

even\_sum += num # Add the even number to the sum

print("Sum of even numbers:", even\_sum)

**Q24. Write a code to take 3 numbers as an input from the user and display the greatest no as output.**

try:

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

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

num3 = float(input("Enter the third number: "))

# Compare the numbers to find the greatest one

if num1 >= num2 and num1 >= num3:

greatest = num1

elif num2 >= num1 and num2 >= num3:

greatest = num2

else:

greatest = num3

print("The greatest number is:", greatest)

except ValueError:

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

**Q25. Write a program to display only those numbers from a list that satisfy the following conditions**

- The number must be divisible by five

- If the number is greater than 150, then skip it and move to the next number

- If the number is greater than 500, then stop the loop

numbers = [12, 75, 150, 180, 145, 525, 50]

**Ans :**

numbers = [12, 75, 150, 180, 145, 525, 50]

for num in numbers:

if num > 500:

break

if num > 150:

continue

if num % 5 == 0:

print(num)